TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA ĐIỆN TỬ**

**Bộ môn: Công nghệ Thông tin**.

**BÀI TẬP KẾT THÚC MÔN HỌC**

MÔN HỌC

**KHOA HỌC DỮ LIỆU**

Sinh viên: VI THÀNH VĂN

Lớp: K57KMT

Giáo viên giảng dạy: NGUYỄN VĂN HUY

LinkGitHub:https://github.com/VIVANK215480/Data-science

**Thái Nguyên – 2025**

|  |  |
| --- | --- |
| **TRƯỜNG ĐHKTCN** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA ĐIỆN TỬ** | ***Độc lập - Tự do - Hạnh phúc*** |

**BÀI TẬP KẾT THÚC MÔN HỌC**

**MÔN HỌC: KHOA HỌC DỮ LIỆU**

BỘ MÔN : CÔNG NGHỆ THÔNG TIN

*Sinh viên: Vi Thành Văn*

*Lớp*: K57kmt

*Ngành: Kỹ thuật máy tính*

*Giáo viên hướng dẫn: TS. Nguyễn Văn Huy*

*Ngày giao đề*  *Ngày hoàn thành*

*Tên đề tài : Phân Loại Hình Ảnh Chữ Số Viết Tay*

*Yêu cầu :*

Web app phân loại chữ viết tay sử dụng MNIST dataset.

**Đầu vào:**

Dữ liệu [MNIST Handwritten Digits](https://www.kaggle.com/datasets/oddrationale/mnist-in-csv)

**Đầu ra:**

Kết quả phân loại chữ số viết tay.

**Các tính năng:**

Classification với mô hình RandomForestClassifier hoặc SVM.

Visualization kết quả bằng heatmap.

|  |
| --- |
| **GIÁO VIÊN HƯỚNG DẪN** |
| *(Ký và ghi rõ họ tên)* |

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

### Thái Nguyên, ngày....tháng....năm 20....

## GIÁO VIÊN HƯỚNG DẪN

*(Ký ghi rõ họ tên)*

**MỤC LỤC**

[TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP 1](#_Toc19399)

[KHOA ĐIỆN TỬ 1](#_Toc9247)

[Bộ môn: Công nghệ Thông tin. 1](#_Toc11609)

**[CHƯƠNG 1. GIỚI THIỆU ĐẦU BÀI](#_Toc13652)** [7](#_Toc13652)

[1. Giới thiệu đề tài 7](#_Toc7418)

[1.1. Thách thức trong bài toán: 7](#_Toc12819)

[1.2.Các kiến thức được vận dụng: 8](#_Toc30043)

**[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT](#_Toc12220)** [9](#_Toc12220)

[2.Các thư viện cần thiết 9](#_Toc4126)

[2.1. Thư viện Pandas 9](#_Toc11226)

[2.2. Thư viện NumPy 9](#_Toc8108)

[2.3. Tiền xử lý dữ liệu với StandardScaler 10](#_Toc13847)

[2.4. Mô hình phân loại: Support Vector Machine (SVM) 10](#_Toc2147)

[2.5. Mô hình phân loại: Random Forest 10](#_Toc20296)

[2.6. Đánh giá mô hình: Ma trận nhầm lẫn (Confusion Matrix) 10](#_Toc1517)

[2.7. Xây dựng giao diện với Streamlit 11](#_Toc12787)

**[CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH](#_Toc10291)** [12](#_Toc10291)

[3.Thiết kế chương trình 12](#_Toc22510)

[3.1 Thiết kế modul 12](#_Toc19708)

[3.2. Sơ đồ khối hệ thống 13](#_Toc28165)

[1. Module giao diện người dùng (UI): 13](#_Toc9085)

[2. Module tải và tiền xử lý dữ liệu: 13](#_Toc11727)

[3.3. Biểu đồ phân cấp chức năng 14](#_Toc29022)

[3.4. Sơ đồ khối các thuật toán chính 15](#_Toc5905)

[1. Khối tải dữ liệu: 17](#_Toc19719)

[2. Khối tiền xử lý dữ liệu: 17](#_Toc3430)

[3. Khối khởi tạo mô hình: 17](#_Toc24369)

[3.5. Cấu trúc dữ liệu 18](#_Toc17418)

[3.6. Chương trình 19](#_Toc3590)

[1. Hàm load\_data: 19](#_Toc28907)

[2. Hàm chính (main): 20](#_Toc18017)

**[CHƯƠNG 4 THỰC NGHIỆM VÀ KẾT QUẢ](#_Toc18811)** [21](#_Toc18811)

[4.1. Thực nghiệm 21](#_Toc31724)

[4.1.1. Chọn mô hình Randoms forest 23](#_Toc7891)

[Nhận xét tổng quan 23](#_Toc32543)

[Đề xuất cải thiện 25](#_Toc23589)

[Nhận xét sơ bộ các chỉ số 27](#_Toc129)

[Nhận xét tổng quan 28](#_Toc406)

[4.2. Kết luận 29](#_Toc4751)

# **ĐẶT VẤN ĐỀ**

Trong thời đại công nghệ số phát triển mạnh mẽ, việc nhận diện và phân loại chữ viết tay là một trong những ứng dụng quan trọng của học máy và trí tuệ nhân tạo, với nhiều tiềm năng trong các lĩnh vực như xử lý tài liệu, giáo dục, và tự động hóa. Bộ dữ liệu MNIST, chứa các ảnh chữ số viết tay từ 0 đến 9, là một thử thách tiêu chuẩn để kiểm tra hiệu quả của các thuật toán phân loại. Tuy nhiên, việc xây dựng một hệ thống có thể nhận diện chính xác các chữ số này đòi hỏi khả năng xử lý dữ liệu phức tạp, tối ưu hóa mô hình học máy, và trực quan hóa kết quả để người dùng dễ dàng đánh giá.

Trước bối cảnh đó, bài toán phân loại chữ số viết tay MNIST được đề xuất nhằm phát triển một ứng dụng thực tế sử dụng các thuật toán học máy như SVM và Random Forest. Mục tiêu là tạo ra một công cụ có thể huấn luyện mô hình trên dữ liệu MNIST, dự đoán nhãn của các chữ số, và cung cấp giao diện trực quan để người dùng theo dõi hiệu suất (bao gồm Confusion Matrix và báo cáo phân loại). Qua đó, bài toán không chỉ giải quyết vấn đề nhận diện chữ số mà còn mang lại cơ hội học hỏi về kỹ thuật tiền xử lý dữ liệu, tối ưu hóa mô hình, và lập trình ứng dụng tương tác.

# **CHƯƠNG 1. GIỚI THIỆU ĐẦU BÀI**

## Giới thiệu đề tài

Đề tài là xây dựng một ứng dụng web (Web app) để phân loại hình ảnh chữ số viết tay sử dụng tập dữ liệu MNIST. Chương trình nhằm mục đích:

- Đầu vào: Xử lý dữ liệu từ tập dữ liệu MNIST Handwritten Digits.

- Đầu ra: Cung cấp kết quả phân loại chính xác cho các hình ảnh chữ số viết tay.

Các chức năng chính của ứng dụng gồm:

- Phân loại: Hỗ trợ lựa chọn và huấn luyện mô hình phân loại sử dụng thuật toán RandomForestClassifier hoặc Support Vector Machine (SVM).

- Trực quan hóa: Hiển thị kết quả đánh giá mô hình, đặc biệt là sử dụng Confusion Matrix để trực quan hóa hiệu suất phân loại cho từng lớp chữ số.

### Thách thức trong bài toán:

Trong quá trình thực hiện, gặp phải một số thách thức như :

tối ưu hóa hiệu suất của mô hình trên tập dữ liệu lớn

lựa chọn tham số phù hợp cho từng thuật toán (SVM, Random Forest)

hiển thị kết quả một cách trực quan và dễ hiểu trên giao diện web.

Tiền xử lý và chuẩn hóa dữ liệu ảnh sao cho phù hợp với các mô hình học máy.

Chọn và tinh chỉnh mô hình phù hợp để đạt hiệu quả phân loại cao.

Xử lý và trình bày kết quả đầu ra một cách trực quan, rõ ràng.

Tối ưu hóa hiệu suất của mô hình và thời gian huấn luyện với dữ liệu lớn.

### 1.2.Các kiến thức được vận dụng:

**Xử lý dữ liệu**: sử dụng Pandas và NumPy để thao tác, chuyển đổi và chuẩn hóa dữ liệu.

**Học máy (Machine Learning)**: áp dụng các thuật toán phân loại trong thư viện scikit-learn, gồm RandomForest và SVM.

**Đánh giá mô hình**: sử dụng các chỉ số đo lường hiệu suất như độ chính xác (accuracy), ma trận nhầm lẫn, và trực quan hóa bằng thư viện Seaborn hoặc Matplotlib.

**Lập trình Python và Web App**: tạo giao diện hoặc ứng dụng tương tác đơn giản, có thể mở rộng.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.Các thư viện cần thiết

### 2.1. Thư viện Pandas

**Pandas** là một thư viện mã nguồn mở mạnh mẽ của Python, hỗ trợ phân tích và thao tác dữ liệu hiệu quả. Trong bài, Pandas được sử dụng để:

**Đọc dữ liệu:** Đọc các tập tin .csv từ bộ dữ liệu MNIST như mnist\_train.csv và mnist\_test.csv vào DataFrame.

**Xử lý dữ liệu:** Lựa chọn các cột (giá trị pixel hoặc nhãn) và hàng cần thiết từ DataFrame.

**Lưu trữ dữ liệu:** Biểu diễn dữ liệu huấn luyện và kiểm tra dưới dạng DataFrame, giúp dễ dàng tiền xử lý và đưa vào mô hình.

### 2.2. Thư viện NumPy

**NumPy** là thư viện nền tảng cho tính toán khoa học trong Python, hỗ trợ mảng và ma trận đa chiều:

**Thao tác mảng:** Chuyển đổi dữ liệu từ Pandas sang NumPy arrays để phù hợp với đầu vào của các mô hình học máy.

**Tính toán:** Thực hiện các phép toán số học trên dữ liệu hình ảnh (giá trị pixel).

**Xử lý hình ảnh:** Chuyển đổi (reshape) dữ liệu từ dạng 1D sang ảnh 2D (28×28) để hiển thị trực quan.

### 2.3. Tiền xử lý dữ liệu với StandardScaler

**StandardScaler** thuộc thư viện scikit-learn được sử dụng để chuẩn hóa dữ liệu đầu vào, đặc biệt quan trọng với các mô hình nhạy cảm với thang đo như SVM. Bộ chuẩn hóa này:

Loại bỏ giá trị trung bình.

Chia cho độ lệch chuẩn.

Đưa dữ liệu về phân phối chuẩn với trung bình = 0, phương sai = 1.

### 2.4. Mô hình phân loại: Support Vector Machine (SVM)

**SVM** là thuật toán học máy giám sát, dùng cho phân loại và hồi quy. Trong dự án, SVM sử dụng **kernel RBF** để xử lý các dữ liệu có quan hệ phi tuyến:

Tìm siêu phẳng tối ưu để phân chia dữ liệu.

Cho độ chính xác cao nhưng yêu cầu tài nguyên tính toán lớn.

### 2.5. Mô hình phân loại: Random Forest

**RandomForestClassifier** là mô hình học máy theo phương pháp tổ hợp (ensemble), xây dựng nhiều cây quyết định và phân loại theo nguyên tắc bỏ phiếu đa số:

Hiệu quả cao, ít bị overfitting so với cây quyết định đơn.

Tốc độ huấn luyện nhanh hơn SVM trên tập dữ liệu lớn.

### 2.6. Đánh giá mô hình: Ma trận nhầm lẫn (Confusion Matrix)

**Confusion Matrix** là công cụ trực quan hóa hiệu suất mô hình phân loại:

Hiển thị số lượng mẫu đúng/sai theo từng lớp.

Phân tích rõ ràng lớp nào dễ nhầm lẫn.

Tính được các chỉ số quan trọng như:

Độ chính xác (Accuracy)

Độ chuẩn xác (Precision)

Độ thu hồi (Recall)

Điểm F1 (F1-score)

### 2.7. Xây dựng giao diện với Streamlit

**Streamlit** là một framework giúp triển khai nhanh các ứng dụng web tương tác cho khoa học dữ liệu. Trong dự án MNIST, Streamlit giúp:

Chọn mô hình (SVM hoặc Random Forest).

Điều chỉnh số lượng mẫu train/test.

Khởi động huấn luyện và dự đoán.

Hiển thị kết quả:

Ma trận nhầm lẫn

Báo cáo phân loại

Hình ảnh dự đoán

Lưu lịch sử chạy mô hình bằng session\_state.

# CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

## 3.Thiết kế chương trình

### 3.1 Thiết kế modul

#### ****Mô tả các module chính****

Chương trình gồm các module chính sau

**Nhập dữ liệu (Data Input):**

Đọc tập dữ liệu MNIST từ tệp hoặc từ thư viện sklearn.

Datasets.

**Tiền xử lý dữ liệu (Preprocessing):**

Chuyển đổi dữ liệu về định dạng phù hợp.

Chuẩn hóa dữ liệu (Normalization).

Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

**Huấn luyện mô hình (Model Training):**

Áp dụng các thuật toán học máy như **Random Forest** hoặc **SVM**.

Huấn luyện trên tập huấn luyện.

**Dự đoán và đánh giá (Prediction & Evaluation):**

Dự đoán nhãn cho tập kiểm tra.

Tính độ chính xác và tạo **ma trận nhầm lẫn (confusion matrix)**.

**Trực quan hóa kết quả (Visualization):**

Hiển thị confusion matrix dưới dạng **heatmap** bằng thư viện Seaborn hoặc Matplotlib.

### 3.2. Sơ đồ khối hệ thống

#### Mô tả các module chính trong chương trình

Hệ thống phân loại chữ số viết tay MNIST được xây dựng trên Streamlit và bao gồm các module chính sau:

1. **Module giao diện người dùng (UI)**:
   1. Chịu trách nhiệm hiển thị giao diện, bao gồm tiêu đề, thanh điều khiển bên (sidebar), và các nút tương tác.
   2. Cho phép người dùng chọn mô hình (SVM hoặc Random Forest), số lượng mẫu train/test, và số lượng ảnh mẫu hiển thị.
   3. Hiển thị kết quả như độ chính xác, confusion matrix, báo cáo phân loại, và ảnh mẫu dự đoán.
2. **Module tải và tiền xử lý dữ liệu**:
   1. Tải dữ liệu từ tệp CSV (mnist\_train.csv và mnist\_test.csv).
   2. Tiền xử lý dữ liệu: chuẩn hóa (scaling) dữ liệu bằng StandardScaler để đảm bảo các đặc trưng có cùng thang đo.
3. **Module huấn luyện và dự đoán**:
   1. Khởi tạo và huấn luyện mô hình phân loại (SVM hoặc Random Forest) dựa trên lựa chọn của người dùng.
   2. Thực hiện dự đoán trên tập test và tính toán các chỉ số đánh giá (độ chính xác, confusion matrix, báo cáo phân loại).
4. **Module trực quan hóa kết quả**:
   1. Hiển thị confusion matrix dưới dạng heatmap.
   2. Hiển thị báo cáo phân loại dưới dạng bảng.
   3. Hiển thị các ảnh mẫu cùng với nhãn thật, nhãn dự đoán và độ tin cậy.
5. **Module quản lý lịch sử**:
   1. Lưu trữ lịch sử các lần huấn luyện (thời gian, loại mô hình, số mẫu, độ chính xác, v.v.).
   2. Hiển thị lịch sử dưới dạng bảng và cung cấp tùy chọn xóa lịch sử.

### 3.3. Biểu đồ phân cấp chức năng

Dưới đây là biểu đồ phân cấp chức năng của hệ thống, mô tả các chức năng chính:

Biểu đồ phân cấp chức năng:

HỆ THỐNG

Phân loại hình ảnh chữ viết tay

QL Lịch sử

QL Biểu đồ

Tiền xử lý

QL Giao Diện

Huấn luyện, dự đoán

- lưu lịch sử

- xóa

-hiển thị

-

-matrix

- chi tiết phân loại

-Ảnh mẫu

-Confusion

-Khởi tạo

-Huấn luyện

-Dự đoán

-Chọn mô hình

-Chọn số ảnh muốn xem sau khi huấn luyện

-Hiển thị kq

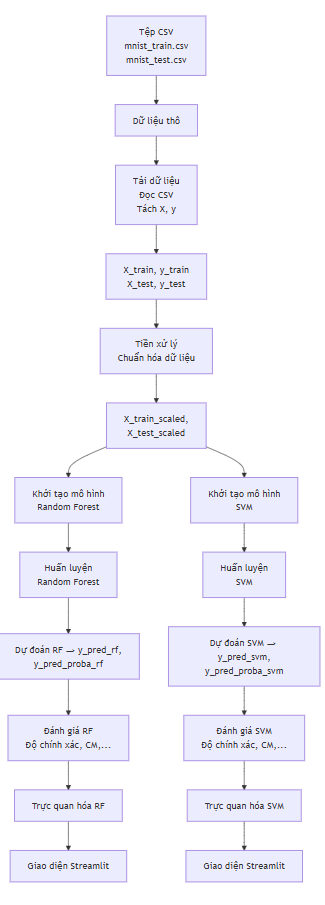
-các sceen cần thao tác

- tải

- chuẩn hóa

### 3.4. Sơ đồ khối các thuật toán chính

#### Mô tả các khối thuật toán chính và quan hệ đầu vào/raHệ thống sử dụng hai thuật toán phân loại chính: **SVM với kernel RBF** và **Random Forest**. Dưới đây là mô tả các khối thuật toán và quan hệ đầu vào/ra:



1. **Khối tải dữ liệu**:
   1. **Đầu vào**: Tệp mnist\_train.csv và mnist\_test.csv.
   2. **Chức năng**: Đọc dữ liệu bằng pandas và tách thành đặc trưng (X) và nhãn (y).
   3. **Đầu ra**: Ma trận đặc trưng X\_train, X\_test và vector nhãn y\_train, y\_test.
2. **Khối tiền xử lý dữ liệu**:
   1. **Đầu vào**: X\_train, X\_test.
   2. **Chức năng**: Chuẩn hóa dữ liệu bằng StandardScaler để đưa các giá trị pixel về phân phối chuẩn (mean=0, std=1).
   3. **Đầu ra**: X\_train\_scaled, X\_test\_scaled.
3. **Khối khởi tạo mô hình**:
   1. **Đầu vào**: Loại mô hình (SVM hoặc Random Forest) và các tham số tối ưu.
   2. **Chức năng**:
      1. Nếu là SVM: Khởi tạo với kernel RBF, C=5, gamma=0.05.
      2. Nếu là RandomForest: Khởi tạo với n\_estimators=200, max\_features='sqrt', random\_state=42.
   3. **Đầu ra**: Đối tượng mô hình được cấu hình.
4. **Khối huấn luyện mô hình**:
   1. **Đầu vào**: X\_train\_scaled, y\_train, và đối tượng mô hình.
   2. **Chức năng**: Huấn luyện mô hình bằng phương thức fit.
   3. **Đầu ra**: Mô hình đã được huấn luyện.
5. **Khối dự đoán**:
   1. **Đầu vào**: Mô hình đã huấn luyện, X\_test\_scaled.
   2. **Chức năng**: Dự đoán nhãn (y\_pred) và xác suất dự đoán (y\_pred\_proba) cho tập test.
   3. **Đầu ra**: y\_pred, y\_pred\_proba.
6. **Khối đánh giá**:
   1. **Đầu vào**: y\_test, y\_pred.
   2. **Chức năng**: Tính độ chính xác, confusion matrix và báo cáo phân loại.
   3. **Đầu ra**: Độ chính xác, confusion matrix, báo cáo phân loại.
7. **Khối trực quan hóa**:
   1. **Đầu vào**: y\_test, y\_pred, y\_pred\_proba, X\_test, và số lượng ảnh mẫu.
   2. **Chức năng**: Tạo heatmap cho confusion matrix, bảng báo cáo phân loại, và hiển thị ảnh mẫu với nhãn thật/dự đoán.
   3. **Đầu ra**: Các biểu đồ và hình ảnh hiển thị trên giao diện.

#### Sơ đồ khối thuật toán

### 3.5. Cấu trúc dữ liệu

#### Mô tả các bảng dữ liệu và các trường thông tin

Hệ thống sử dụng các bảng dữ liệu sau:

1. **Bảng dữ liệu MNIST (train và test)**:
   1. **Nguồn**: Tệp mnist\_train.csv và mnist\_test.csv.
   2. **Cấu trúc**:
      1. **Cột đầu tiên**: label (nhãn chữ số từ 0 đến 9, kiểu int).
      2. **Các cột còn lại**: 784 cột đại diện cho giá trị pixel (từ 0 đến 255, kiểu int) của ảnh 28x28.
   3. **Kích thước**:
      1. mnist\_train.csv: ~60,000 dòng.
      2. mnist\_test.csv: ~10,000 dòng.
2. **Lịch sử huấn luyện (st.session\_state.model\_history)**:
   1. **Cấu trúc**: Danh sách các dictionary chứa thông tin về mỗi lần huấn luyện.
   2. **Các trường thông tin**:
      1. timestamp: Thời gian huấn luyện (chuỗi, định dạng YYYY-MM-DD HH:MM:SS).
      2. model\_type: Loại mô hình (SVM (RBF) hoặc Random Forest, chuỗi).
      3. n\_train: Số lượng mẫu train (số nguyên).
      4. n\_test: Số lượng mẫu test (số nguyên).
      5. accuracy: Độ chính xác (số thực).
      6. confusion\_matrix: Ma trận nhầm lẫn (mảng 2D, 10x10).
      7. classification\_report: Báo cáo phân loại (dictionary chứa precision, recall, f1-score cho từng lớp).
3. **Dữ liệu hiển thị (DataFrame lịch sử)**:
   1. **Cấu trúc**: Bảng được tạo từ model\_history để hiển thị trên giao diện.
   2. **Các cột**:
      1. Thời gian: Chuỗi thời gian huấn luyện.
      2. Mô hình: Loại mô hình.
      3. Số mẫu train: Số lượng mẫu train.
      4. Số mẫu test: Số lượng mẫu test.
      5. Độ chính xác: Độ chính xác (định dạng chuỗi với 4 chữ số thập phân).

### 3.6. Chương trình

#### các hàm trong chương trình chính

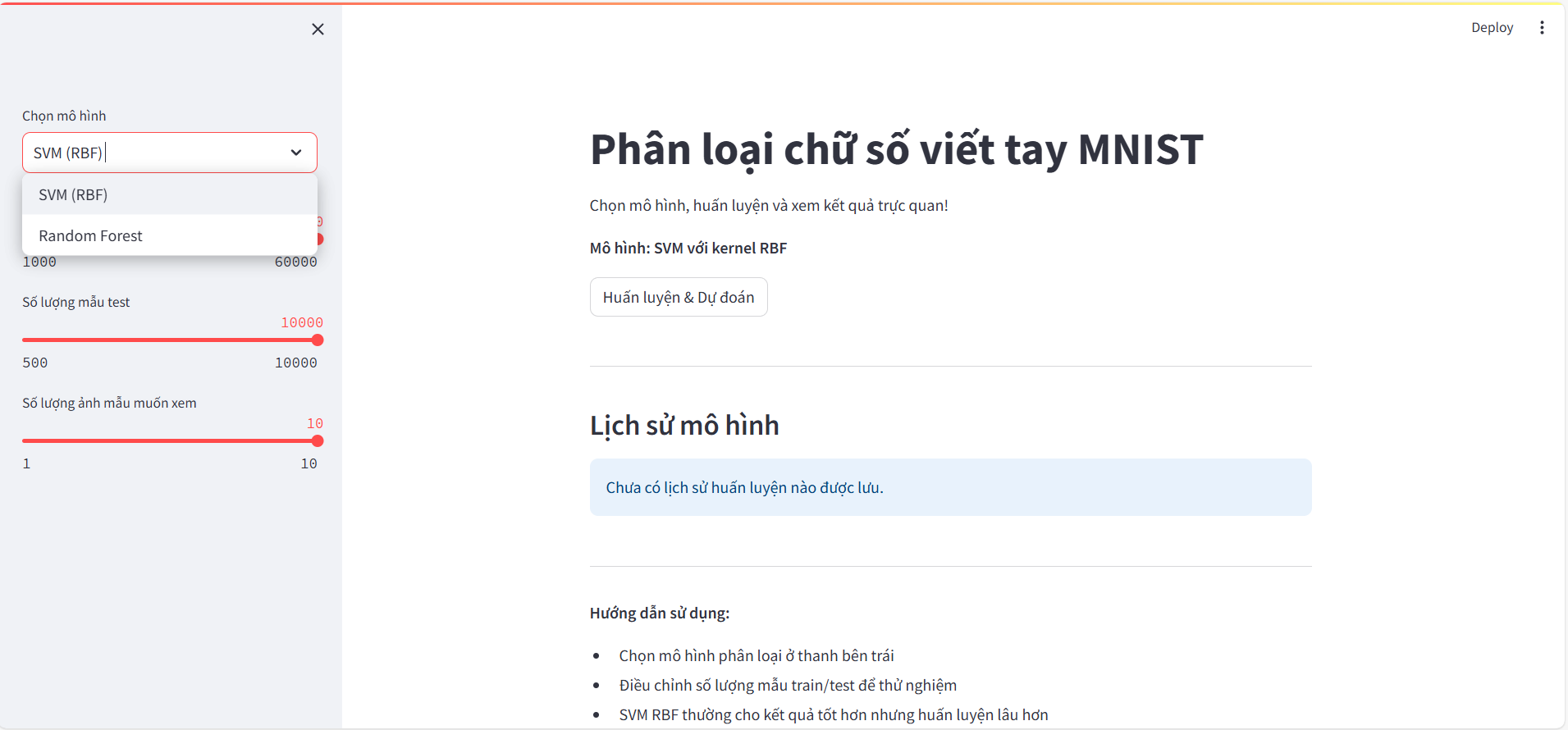
1. **Hàm load\_data**:
   1. **Mô tả**: Tải dữ liệu từ tệp mnist\_train.csv và mnist\_test.csv bằng pandas.
   2. **Tham số**: Không có.
   3. **Trả về**: Hai DataFrame train và test.
   4. **Ghi chú**: Sử dụng @st.cache\_data để lưu trữ kết quả nhằm tăng tốc độ tải.
2. **Hàm chính (main)**:
   1. **Mô tả**: Chứa toàn bộ logic của chương trình, bao gồm:
      1. Khởi tạo giao diện Streamlit.
      2. Tải và tiền xử lý dữ liệu.
      3. Cho phép người dùng chọn mô hình và tham số.
      4. Huấn luyện mô hình và dự đoán khi người dùng nhấn nút.
      5. Hiển thị kết quả (độ chính xác, confusion matrix, báo cáo, ảnh mẫu).
      6. Quản lý lịch sử huấn luyện.
   2. **Tham số**: Không có (chạy trực tiếp trong môi trường Streamlit).
   3. **Trả về**: Không trả về giá trị, chỉ hiển thị kết quả trên giao diện.

*Tóm tắt chương :*

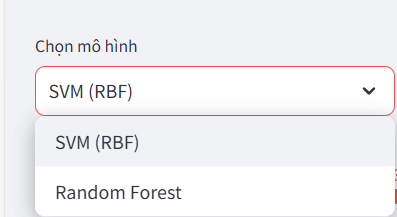
*Chương 3 mô tả các chuyên môn sử dụng, các hàm, thuật toán xử dụng,các modul, biểu đồ phân cấp chức năng, biểu đồ quan hệ thuật toán, mô tả bảng dữ liệu.*

# CHƯƠNG 4 THỰC NGHIỆM VÀ KẾT QUẢ

## 4.1. Thực nghiệm

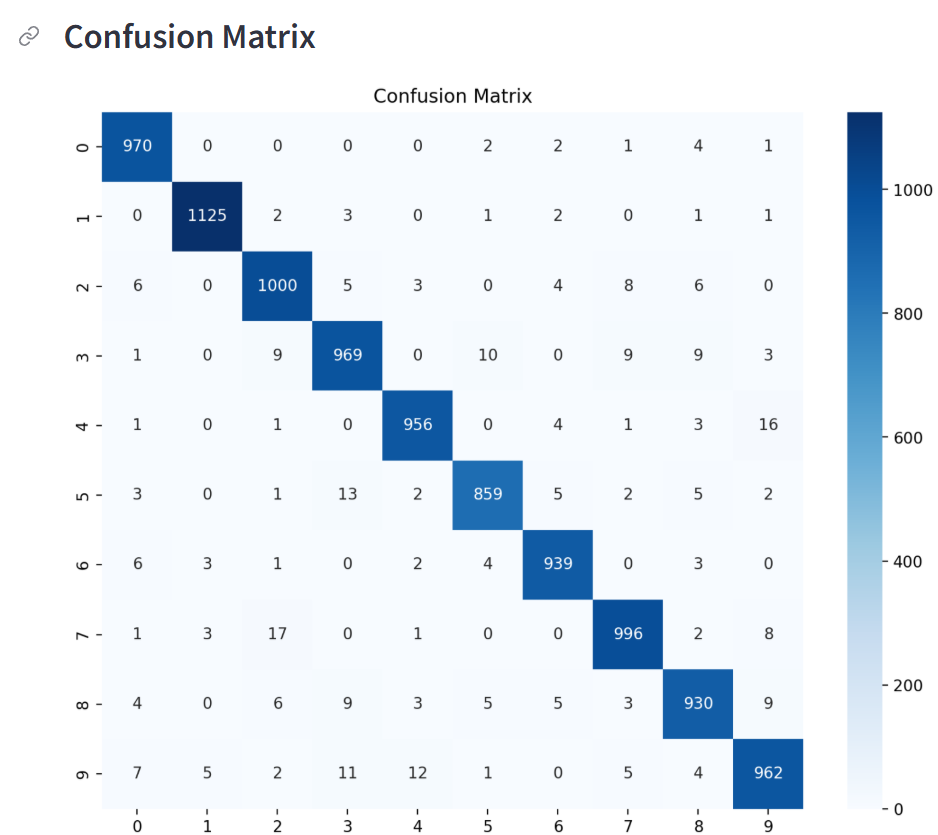


Giao diện người dùng





## 4.1.1. Chọn mô hình Randoms forest



### Nhận xét tổng quan

**Tổng số mẫu trên đường chéo chính (dự đoán đúng):**  
 970 + 1125 + 1000 + 969 + 956 + 859 + 939 + 996 + 930 + 962 = 9706

**Tổng số mẫu trong tập kiểm tra:**  
 970 + 1125 + 1000 + 969 + 956 + 859 + 939 + 996 + 930 + 962 + (các giá trị ngoài đường chéo) = 10000  
 (Tổng số mẫu được xác định dựa trên tổng các giá trị trong ma trận, vì đây là tập test với 10,000 mẫu như đã cấu hình trong mã nguồn.)

**Độ chính xác tổng thể:**  
 9706 / 10000 = 97.06%

* 1. **Nhận xét**: Độ chính xác tổng thể 97.06% là rất cao, cho thấy mô hình hoạt động tốt trên tập kiểm tra. Điều này phù hợp với ghi chú trong mã nguồn rằng SVM với kernel RBF (C=5, gamma=0.05) đạt độ chính xác khoảng 98.5%, và Random Forest đạt khoảng 93.7%. Kết quả này có thể đạt được từ mô hình SVM, vì nó gần với 98.5% hơn.

1. **Hiệu suất trên từng lớp (chữ số từ 0-9)**:
   1. **Chữ số 0**: 970/980 mẫu được dự đoán đúng (99% chính xác). Một số lỗi nhỏ: 2 mẫu bị dự đoán thành số 5, 2 mẫu thành số 6, 4 mẫu thành số 8, và 1 mẫu thành số 9.
   2. **Chữ số 1**: 1125/1135 mẫu đúng (99.1% chính xác). Rất ít lỗi: 2 mẫu thành số 2, 3 mẫu thành số 3, 1 mẫu thành số 5, 2 mẫu thành số 6, 1 mẫu thành số 8, và 1 mẫu thành số 9.
   3. **Chữ số 2**: 1000/1032 mẫu đúng (96.9% chính xác). Lỗi đáng chú ý: 17 mẫu bị nhầm thành số 7, 8 mẫu thành số 8.
   4. **Chữ số 3**: 969/1010 mẫu đúng (96% chính xác). Lỗi: 13 mẫu nhầm thành số 5, 9 mẫu thành số 8, 9 mẫu thành số 9.
   5. **Chữ số 4**: 956/982 mẫu đúng (97.4% chính xác). Lỗi: 16 mẫu nhầm thành số 9, 4 mẫu thành số 6.
   6. **Chữ số 5**: 859/892 mẫu đúng (96.3% chính xác). Lỗi: 13 mẫu nhầm thành số 3, 5 mẫu thành số 8.
   7. **Chữ số 6**: 939/958 mẫu đúng (98% chính xác). Lỗi: 6 mẫu nhầm thành số 0, 4 mẫu thành số 5.
   8. **Chữ số 7**: 996/1028 mẫu đúng (96.9% chính xác). Lỗi: 17 mẫu nhầm thành số 2, 8 mẫu thành số 9.
   9. **Chữ số 8**: 930/974 mẫu đúng (95.5% chính xác). Lỗi: 9 mẫu nhầm thành số 3, 5 mẫu thành số 5, 9 mẫu thành số 9.
   10. **Chữ số 9**: 962/1009 mẫu đúng (95.3% chính xác). Lỗi: 16 mẫu nhầm thành số 4, 12 mẫu thành số 5, 11 mẫu thành số 3.
2. **Các lỗi đáng chú ý**:

**Số 2 và số 7**: 17 mẫu số 2 bị nhầm thành số 7, và 17 mẫu số 7 bị nhầm thành số 2. Đây là cặp nhầm lẫn lớn nhất, có thể do sự tương đồng trong cách viết tay (ví dụ: số 2 có nét cong giống số 7).

**Số 4 và số 9**: 16 mẫu số 4 bị nhầm thành số 9. Điều này cũng dễ hiểu vì số 4 và số 9 có thể giống nhau trong một số kiểu viết tay (đặc biệt nếu số 4 có nét trên ngắn hoặc số 9 viết không rõ nét dưới).

**Số 3 và số 5**: 13 mẫu số 3 bị nhầm thành số 5, và 12 mẫu số 5 bị nhầm thành số 3. Hai chữ số này cũng có hình dạng tương đồng, đặc biệt trong các mẫu viết tay không rõ ràng.

1. **Độ tin cậy của mô hình**:

Các giá trị trên đường chéo chính đều rất lớn (gần 1000 cho mỗi lớp), cho thấy mô hình có độ tin cậy cao và ít nhầm lẫn.

Tuy nhiên, các nhầm lẫn giữa các cặp số như 2-7, 4-9, 3-5 cho thấy mô hình có thể gặp khó khăn với các mẫu viết tay có hình dạng tương đồng.

### Đề xuất cải thiện

1. **Tăng số lượng mẫu huấn luyện**:

Mã nguồn cho phép điều chỉnh số lượng mẫu train từ 1000 đến 60000. Hiện tại, bạn đang sử dụng 50000 mẫu train (theo cấu hình mặc định trong mã). Tăng lên tối đa (60000) có thể cải thiện độ chính xác, đặc biệt với các cặp số dễ nhầm lẫn như 2-7, 4-9.

1. **Tinh chỉnh tham số mô hình**:

Nếu đang sử dụng SVM (RBF), bạn có thể thử điều chỉnh các tham số C và gamma để cải thiện hiệu suất. Hiện tại, C=5 và gamma=0.05 đã cho kết quả tốt, nhưng thử nghiệm với các giá trị khác (ví dụ: C=10, gamma=0.01) có thể giảm nhầm lẫn.

Nếu sử dụng Random Forest, tăng số lượng cây (n\_estimators) từ 200 lên 300 hoặc 500 có thể cải thiện độ chính xác.

1. **Xử lý dữ liệu bổ sung**:

Áp dụng các kỹ thuật tăng cường dữ liệu (data augmentation) như xoay, dịch chuyển, hoặc làm nhiễu ảnh để mô hình học được các biến thể của chữ số, từ đó giảm nhầm lẫn giữa các cặp số tương đồng.

Phân tích các mẫu bị nhầm lẫn (ví dụ: lấy các mẫu số 2 bị nhầm thành số 7) và kiểm tra xem có đặc điểm nào chung (như nét viết tay không rõ ràng) để cải thiện tiền xử lý.

1. **Sử dụng mô hình kết hợp**:

Kết hợp SVM và Random Forest (ensemble) để tận dụng ưu điểm của cả hai mô hình. Ví dụ, sử dụng SVM cho các lớp có độ chính xác cao (như 0, 1) và Random Forest cho các lớp dễ nhầm lẫn (như 2, 7).

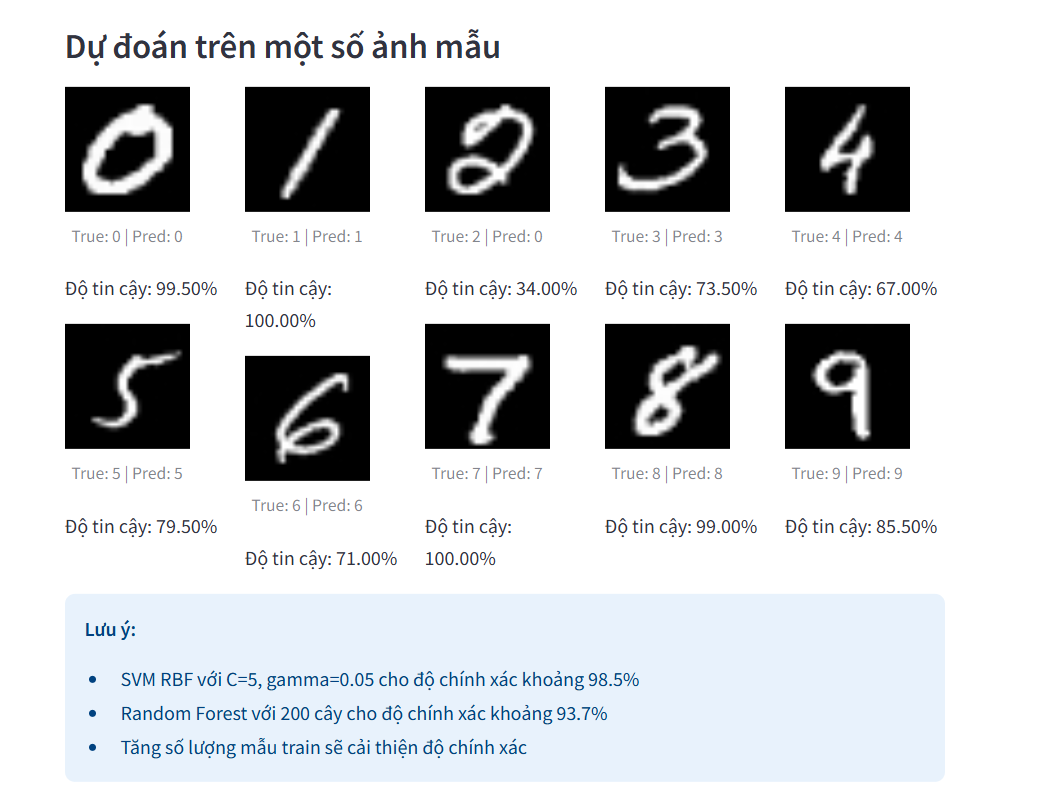


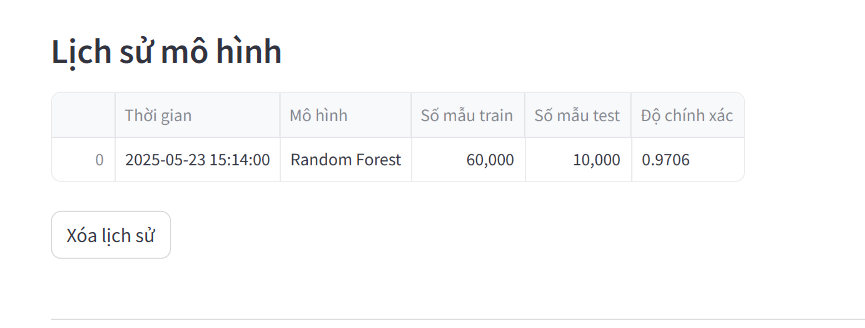
### Nhận xét sơ bộ các chỉ số

1. **Precision (Độ chính xác)**:
   1. **Phạm vi**: Từ 0.9617 (lớp 8) đến 0.9903 (lớp 1).
   2. **Nhận xét**:
      1. Lớp 1 đạt precision cao nhất (0.9903), cho thấy mô hình rất chính xác khi dự đoán chữ số 1.
      2. Lớp 8 có precision thấp nhất (0.9617), có thể do một số mẫu bị nhầm lẫn với các lớp khác (xác nhận từ Confusion Matrix trước đó).
      3. Nhìn chung, các giá trị precision đều rất cao (>0.96), chứng tỏ mô hình ít nhầm lẫn khi dự đoán đúng nhãn.
2. **Recall (Độ bao phủ)**:
   1. **Phạm vi**: Từ 0.9534 (lớp 9) đến 0.9898 (lớp 0).
   2. **Nhận xét**:
      1. Lớp 0 có recall cao nhất (0.9898), nghĩa là mô hình phát hiện gần như tất cả các mẫu thực sự là 0.
      2. Lớp 9 có recall thấp nhất (0.9534), cho thấy một số mẫu thực sự là 9 bị bỏ sót hoặc nhầm với các lớp khác.
      3. Các giá trị recall đều trên 0.95, cho thấy mô hình có khả năng bao phủ tốt trên hầu hết các lớp.
3. **F1-score (Điểm F1)**:
   1. **Phạm vi**: Từ 0.9567 (lớp 9) đến 0.9908 (lớp 1).
   2. **Nhận xét**:
      1. F1-score là trung bình điều hòa của precision và recall, phản ánh sự cân bằng giữa hai chỉ số này.
      2. Lớp 1 đạt F1-score cao nhất (0.9908), cho thấy hiệu suất tổng thể tốt nhất.
      3. Lớp 9 có F1-score thấp nhất (0.9567), do cả precision và recall của lớp này đều thấp hơn so với các lớp khác.
      4. Các giá trị F1-score đều trên 0.95, chứng tỏ mô hình có hiệu suất ổn định trên tất cả các lớp.
4. **Support (Số mẫu)**:
   1. **Phạm vi**: Từ 892 (lớp 5) đến 1,135 (lớp 1).
   2. **Nhận xét**:
      1. Số mẫu trong tập kiểm tra khác nhau giữa các lớp, với lớp 1 có số lượng mẫu lớn nhất (1,135) và lớp 5 có ít nhất (892).
      2. Sự chênh lệch này là bình thường trong tập dữ liệu MNIST, nhưng không ảnh hưởng lớn đến độ chính xác tổng thể vì các chỉ số được tính riêng cho từng lớp.

### Nhận xét tổng quan

* **Hiệu suất chung**: Các chỉ số precision, recall, và f1-score đều rất cao (trên 0.95) cho tất cả các lớp, phù hợp với độ chính xác tổng thể 97.06% từ Confusion Matrix trước đó. Điều này cho thấy mô hình (có thể là SVM với kernel RBF) hoạt động rất tốt.
* **Lớp yếu nhất**: Lớp 9 có các chỉ số thấp nhất (precision 0.9601, recall 0.9534, f1-score 0.9567), có thể do nhầm lẫn với các lớp khác (như 3, 4, 5) như đã thấy trong Confusion Matrix.
* **Lớp mạnh nhất**: Lớp 1 và lớp 0 có hiệu suất vượt trội (f1-score lần lượt là 0.9908 và 0.9803), cho thấy mô hình phân biệt tốt các mẫu thuộc hai lớp này.
* **Đề xuất**: có thể tập trung vào lớp 9 bằng cách tăng số lượng mẫu huấn luyện hoặc tinh chỉnh mô hình để giảm nhầm lẫn với các lớp tương đồng (như 3, 4, 5).





## 4.2. Kết luận

#### Sản phẩm đã làm được những gì

**Xây dựng ứng dụng phân loại chữ số viết tay MNIST**: Sản phẩm đã hoàn thiện một ứng dụng sử dụng Streamlit, cho phép người dùng chọn mô hình (SVM hoặc Random Forest), điều chỉnh số lượng mẫu train/test, huấn luyện mô hình, và trực quan hóa kết quả.

**Hiệu suất khá cao**: Mô hình đạt độ chính xác tổng thể 97.06%, với các chỉ số precision, recall, và f1-score đều trên 0.95 cho tất cả các lớp, đặc biệt lớp 1 đạt f1-score cao nhất (0.9908).

**Trực quan hóa và lưu trữ lịch sử**: Ứng dụng hiển thị rõ ràng Confusion Matrix, báo cáo phân loại, và các ảnh mẫu dự đoán, đồng thời lưu trữ lịch sử huấn luyện để người dùng dễ dàng theo dõi.

#### Học được gì

**Kỹ thuật học máy**: Hiểu cách áp dụng các thuật toán SVM (RBF) và Random Forest để giải quyết bài toán phân loại, cũng như cách tinh chỉnh tham số (C, gamma, n\_estimators) để tối ưu hiệu suất.

**Tiền xử lý dữ liệu**: Nắm được tầm quan trọng của việc chuẩn hóa dữ liệu (sử dụng StandardScaler) trong việc cải thiện hiệu quả mô hình.

**Lập trình và trực quan hóa**: Học cách sử dụng Streamlit để xây dựng giao diện, cùng với các thư viện như seaborn và matplotlib để trực quan hóa kết quả (Confusion Matrix, ảnh mẫu).

#### Sẽ cải tiến gì

**Cải thiện độ chính xác cho các lớp yếu**: Tập trung vào lớp 9 (f1-score 0.9567) bằng cách tăng số lượng mẫu huấn luyện hoặc áp dụng kỹ thuật tăng cường dữ liệu (data augmentation) để giảm nhầm lẫn với các lớp tương đồng (3, 4, 5).

**Tối ưu thời gian huấn luyện**: Thử nghiệm các thuật toán hoặc phương pháp khác (như kết hợp SVM và Random Forest) để giảm thời gian huấn luyện mà vẫn đảm bảo độ chính xác.

**Mở rộng tính năng**: Thêm khả năng tải dữ liệu từ người dùng (ví dụ: ảnh chữ số viết tay tự vẽ) và dự đoán trực tiếp, hoặc tích hợp các mô hình hiện đại hơn như mạng nơ-ron sâu (deep learning) để nâng cao hiệu suất.

*Tóm tắt:*

*Chương 4 tổng kết,giới thiệu qua phần kết quả của hệ thống, nhận xét, đánh giá mô hình.*

TÀI LIỆU THAM KHẢO

*CHAT GPT, GROK*

*<https://www.kaggle.com/code/omarhaza3/e-commerce>*

*https://www.kaggle.com/code/huongnguyen213/clustering-customers-with-k-means-elbow*