TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ Thông tin.

BÀI TẠP KẾT THÚC MÔN HỌC

MÔN HỌC

KHOA HỌC DỮ LIỆU

Sinh viên: Đỗ Đức Chung . . . . . .

Lớp: K57KMT. . . . . .

Giáo viên GIẢNG DẠY: TS. Nguyễn Văn Huy. .. . .

Link GitHub: https://github.com/Chung2310/BTL_Data_Science

Thái Nguyên – 2025

TRƯỜNG ĐHKTCN
KHOA ĐIỆN TỬ

CỌNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BÀI TẠP KẾT THÚC MÔN HỌC

MÔN HỌC: KHOA HỌC DỮ LIỆU

BỌ MÔN : CÔNG NGHỆ THÔNG TIN

Sinh viên:Đỗ Đức Chung.....

Lớp: K57KMT..... *Ngành: Kỹ thuật Máy tính*

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

Ngày giao đề..... *Ngày hoàn thành: 30/05/2025*

Tên đề tài : Phân loại hình ảnh chữ viết tay

Yêu cầu :

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

Đầu vào:

- Dữ liệu MNIST Handwritten Digits

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

Lời nói đầu

Trong kỷ nguyên số hiện nay, sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI) và khoa học dữ liệu (Data Science) đã mở ra nhiều cơ hội ứng dụng trong đời sống và sản xuất. Một trong những ứng dụng phổ biến và thực tiễn nhất của AI là khả năng nhận dạng và phân loại hình ảnh – đặc biệt là hình ảnh chữ viết tay. Đây là một lĩnh vực đã được nghiên cứu trong nhiều thập kỷ, nhưng cho đến nay vẫn giữ được giá trị ứng dụng cao trong các hệ thống hiện đại như máy quét tài liệu, tự động chấm điểm trắc nghiệm, xử lý biểu mẫu ngân hàng, hay hệ thống kiểm tra vé số, v.v.

Trong bối cảnh đó, đề tài **"Web App phân loại chữ viết tay sử dụng MNIST dataset"** được nhóm chúng em lựa chọn với mong muốn tìm hiểu sâu hơn về quy trình xây dựng một ứng dụng AI hoàn chỉnh, từ việc xử lý dữ liệu, huấn luyện mô hình học máy đến việc triển khai mô hình lên nền tảng web để phục vụ người dùng cuối. Đây là một bài toán tưởng chừng đơn giản nhưng lại mang nhiều giá trị về học thuật cũng như tính ứng dụng thực tế.

Tiểu luận này tập trung vào việc sử dụng tập dữ liệu **MNIST** – một trong những bộ dữ liệu hình ảnh chữ số viết tay kinh điển và phổ biến nhất trong lĩnh vực học máy. Với MNIST, nhóm đã xây dựng một mô hình học sâu (Deep Learning) sử dụng thư viện **TensorFlow** và **Keras**, sau đó tích hợp vào ứng dụng web sử dụng **Flask** – một framework web nhẹ và dễ triển khai. Ứng dụng cho phép người dùng vẽ một chữ số bất kỳ trên giao diện và nhận lại kết quả dự đoán theo thời gian thực. Đây là sự kết hợp giữa kỹ thuật học máy, lập trình backend và phát triển giao diện người dùng.

Trong quá trình thực hiện đề tài, nhóm đã rèn luyện được các kỹ năng thiết yếu của một dự án khoa học dữ liệu hoàn chỉnh, bao gồm: thu thập và làm sạch dữ liệu, xây dựng và đánh giá mô hình, tối ưu hóa hiệu suất, và đặc biệt là kỹ năng triển khai mô hình vào môi trường thực tế. Không những thế, đề tài còn giúp chúng em củng cố và mở rộng kiến thức về lập trình Python, kỹ thuật xử lý ảnh, cũng như tư duy giải quyết vấn đề theo hướng hệ thống.

Chúng em chân thành cảm ơn quý thầy cô bộ môn đã tạo điều kiện và định hướng để nhóm thực hiện đề tài này. Những kiến thức và kinh nghiệm thu được trong quá trình thực hiện tiểu luận chắc chắn sẽ là hành trang quý báu giúp chúng em trong quá trình học tập và làm việc sau này. Nhóm rất mong nhận được sự góp ý và phản biện của thầy cô để đề tài được hoàn thiện hơn cả về nội dung lẫn hình thức.

Chương 1 : Giới thiệu đầu bài

1.1. Đặt vấn đề

Nhận dạng chữ viết tay là một trong những bài toán kinh điển trong lĩnh vực trí tuệ nhân tạo, đặc biệt trong nhánh học máy (machine learning) và thị giác máy tính (computer vision). Với sự phát triển của khoa học dữ liệu (Data Science), việc xử lý và phân tích dữ liệu hình ảnh để rút ra thông tin có ý nghĩa trở nên khả thi và ngày càng được ứng dụng rộng rãi trong thực tế như nhận dạng mã số, số serial, chữ viết trên hóa đơn, hoặc phiếu khảo sát.

MNIST (Modified National Institute of Standards and Technology) là một trong những tập dữ liệu nổi tiếng và được sử dụng rộng rãi để huấn luyện và đánh giá các mô hình phân loại chữ số viết tay. Trong khuôn khổ môn học *Data Science*, nhóm chúng em thực hiện đề tài **"Web app phân loại chữ viết tay sử dụng MNIST dataset"** nhằm xây dựng một ứng dụng web đơn giản nhưng có đầy đủ chức năng của một hệ thống nhận diện chữ viết tay, từ đọc dữ liệu, huấn luyện mô hình đến dự đoán và trực quan hóa kết quả.

1.2. Mục tiêu đề tài

Đề tài hướng tới các mục tiêu chính sau:

- Xây dựng mô hình học máy có khả năng phân loại chính xác các chữ số viết tay từ tập dữ liệu MNIST.
- Phát triển ứng dụng web có giao diện thân thiện, cho phép người dùng tải ảnh hoặc chọn ảnh từ tập MNIST để dự đoán chữ số.
- Sử dụng mô hình **Random Forest** hoặc **SVM (Support Vector Machine)** để huấn luyện và đánh giá hiệu suất phân loại.
- Trực quan hóa kết quả phân loại bằng **ma trận nhầm lẫn (confusion matrix)** và **heatmap**.
- Vận dụng các kiến thức nền tảng về xử lý dữ liệu, học máy và lập trình web để xây dựng một hệ thống đầy đủ.

1.3. Tính năng của chương trình

Ứng dụng web được xây dựng trong đề tài bao gồm các tính năng sau:

- Đọc và xử lý tập dữ liệu chữ số viết tay MNIST bằng thư viện **Pandas** và **NumPy**.

- Cho phép lựa chọn mô hình phân loại giữa **RandomForestClassifier** và **SVM** từ thư viện **scikit-learn**.
- Tiến hành huấn luyện mô hình trên tập huấn luyện MNIST và đánh giá trên tập kiểm tra.
- Hiển thị kết quả phân loại, độ chính xác (accuracy) và trực quan hóa bằng **confusion matrix heatmap** sử dụng **matplotlib** và **seaborn**.
- Cung cấp giao diện web để người dùng tương tác, chọn ảnh và xem kết quả phân loại trực tiếp.

1.4. Thánh thức của đề tài

Mặc dù đề tài sử dụng một bộ dữ liệu chuẩn và đã được tiền xử lý, quá trình xây dựng ứng dụng vẫn gặp một số thách thức:

- Cần lựa chọn mô hình phù hợp để đảm bảo độ chính xác cao mà vẫn giữ được tốc độ dự đoán nhanh.
- Phải xử lý dữ liệu đúng định dạng đầu vào mà các mô hình yêu cầu, đồng thời chuẩn hóa dữ liệu một cách hợp lý.
- Tích hợp quá trình huấn luyện, dự đoán và trực quan hóa kết quả một cách hợp lý trong một giao diện web mạch lạc, dễ sử dụng.
- Cân đối giữa hiệu năng và độ trực quan khi hiển thị ma trận nhầm lẫn trong môi trường web.

1.5. Kiến thức và công nghệ sử dụng

Để thực hiện đề tài, nhóm đã vận dụng nhiều kiến thức và công cụ trong lĩnh vực khoa học dữ liệu và phát triển phần mềm, bao gồm:

- **Pandas và NumPy**: Xử lý, chuẩn hóa và phân tích dữ liệu từ tập MNIST.
- **Scikit-learn**: Huấn luyện và đánh giá mô hình phân loại như RandomForestClassifier và SVM.
- **Matplotlib và Seaborn**: Trực quan hóa dữ liệu và kết quả phân loại thông qua biểu đồ và ma trận nhầm lẫn.
- **Flask**: Framework phát triển ứng dụng web backend đơn giản và hiệu quả.
- **HTML/CSS**: Tạo giao diện web trực quan, thân thiện với người dùng.

Chương 2: Cơ sở lý thuyết

2.1. Nhận dạng chữ viết tay và bài toán phân loại

Nhận dạng chữ viết tay (Handwritten Digit Recognition) là một bài toán trong lĩnh vực thị giác máy tính (Computer Vision), trong đó mục tiêu là xây dựng một hệ thống có khả năng "đọc" và phân loại các chữ số viết tay do con người viết. Bài toán này là ví dụ điển hình cho bài toán **phân loại (classification)** trong học máy, nơi mỗi hình ảnh đầu vào (input) được gán một nhãn (label) từ một tập hợp hữu hạn các lớp – trong trường hợp MNIST là 10 chữ số từ 0 đến 9.

Bài toán này có các ứng dụng trong thực tế như:

- Tự động xử lý biểu mẫu, số chứng minh, số tài khoản.
- Đọc chữ viết trong ngân hàng, kho bạc.
- Tích hợp vào các hệ thống nhập liệu nhanh hoặc công cụ hỗ trợ người khiếm thị.

2.2. Tập dữ liệu MNIST

MNIST (Modified National Institute of Standards and Technology) là một bộ dữ liệu nổi tiếng trong cộng đồng học máy, bao gồm:

- **60.000 ảnh** trong tập huấn luyện và **10.000 ảnh** trong tập kiểm tra.
- Mỗi ảnh là một chữ số viết tay từ 0 đến 9, có kích thước **28x28 pixel** và là ảnh **grayscale** (mức xám).
- Mỗi ảnh được biểu diễn dưới dạng ma trận 28x28 chứa các giá trị từ 0 đến 255 (độ sáng của từng điểm ảnh), thường được chuẩn hóa về [0,1] trước khi đưa vào mô hình.

Tập MNIST có ưu điểm là đã được chuẩn hóa tốt, dễ tiếp cận, phù hợp cho mục đích nghiên cứu và học tập.

2.3. Các mô hình phân loại sử dụng

2.3.1. Random Forest Classifier

Random Forest là một thuật toán học máy thuộc nhóm **ensemble learning**, cụ thể là **bagging**. Nó hoạt động bằng cách kết hợp nhiều **cây quyết định (decision trees)** độc lập và sử dụng biểu quyết (voting) để đưa ra kết quả cuối cùng.

Ưu điểm:

- Ít bị overfitting hơn cây đơn.
- Hiệu quả tốt trên các tập dữ liệu vừa và lớn.
- Có thể tính tầm quan trọng của các đặc trưng (feature importance).

2.3.2. Support Vector Machine (SVM)

SVM là thuật toán học có giám sát (supervised learning) được dùng phổ biến trong bài toán phân loại. Nguyên lý hoạt động của SVM là tìm một **siêu phẳng (hyperplane)** tối ưu để phân tách các lớp dữ liệu.

Đặc điểm:

- Hiệu quả với không gian đặc trưng cao chiều.
- Có thể sử dụng các **kernel** (nhân) để phân loại phi tuyến.
- Tuy nhiên, việc huấn luyện SVM có thể tốn thời gian với tập dữ liệu lớn như MNIST.

2.4. Các bước xử lý dữ liệu

2.4.1. Đọc và tiền xử lý dữ liệu

Sử dụng thư viện **Pandas** và **NumPy**, dữ liệu được đọc và xử lý theo các bước:

- Đọc dữ liệu từ file CSV hoặc nguồn có sẵn.
- Tách dữ liệu thành đặc trưng (X) và nhãn (y).
- Chuẩn hóa giá trị pixel về khoảng [0,1] để tăng tốc độ hội tụ và cải thiện độ chính xác.
- Chia dữ liệu thành tập huấn luyện và tập kiểm tra (thường theo tỷ lệ 80:20).

2.4.2. Trực quan hóa dữ liệu

Việc hiển thị một số ảnh chữ số trong tập MNIST giúp người làm hiểu rõ hơn về phân bố dữ liệu. Đồng thời, **trực quan hóa confusion matrix** bằng **heatmap** cho phép đánh giá hiệu suất mô hình sau khi huấn luyện.

2.5. Trực quan hóa ma trận nhầm lẫn (Confusion Matrix)

Confusion matrix là công cụ đánh giá hiệu quả phân loại của mô hình. Trong bài toán nhiều lớp (multiclass classification), ma trận này hiển thị số lượng dự đoán đúng và sai cho từng lớp.

Mỗi phần tử [i][j] trong ma trận biểu thị số lần ảnh thuộc lớp i được dự đoán thành lớp j. Ma trận được trực quan hóa bằng thư viện **Seaborn** dưới dạng **heatmap**, giúp dễ dàng quan sát các điểm mô hình thường bị nhầm lẫn.

2.6. Ứng dụng Flask để triển khai mô hình

Flask là một micro web framework trong Python, thường được dùng để xây dựng các ứng dụng web đơn giản. Trong đề tài này, Flask đóng vai trò:

- Xây dựng giao diện người dùng để vẽ hoặc chọn ảnh chữ số.
- Gửi dữ liệu ảnh đến server và xử lý bằng mô hình đã huấn luyện.
- Trả kết quả phân loại lại cho người dùng và hiển thị.

Việc sử dụng Flask cho phép triển khai ứng dụng nhanh chóng, nhẹ và dễ mở rộng trong tương lai.

2.7. Các nội dung chuyên môn sử dụng trong chương trình

2.7.1. Cấu trúc dữ liệu List trong Python

Trong quá trình xử lý dữ liệu và lập trình ứng dụng, **List** là kiểu dữ liệu cơ bản và quan trọng trong Python, cho phép lưu trữ và thao tác với dãy các phần tử theo thứ tự. List hỗ trợ nhiều phương thức tiện lợi như thêm, xóa, sắp xếp, và truy cập phần tử theo chỉ số, rất hữu ích trong việc quản lý dữ liệu ảnh, nhãn, hoặc kết quả dự đoán.

2.7.2. Thư viện Pandas

Panda**s** là một thư viện mạnh mẽ cho việc xử lý và phân tích dữ liệu dạng bảng (dataframe). Trong đề tài, Pandas được sử dụng để:

- Đọc dữ liệu từ các file CSV hoặc các định dạng khác chứa tập dữ liệu MNIST.
- Xử lý, trích xuất và chuẩn hóa các đặc trưng (features) và nhãn (labels).
- Hỗ trợ phân chia dữ liệu thành tập huấn luyện và tập kiểm tra.
- Giúp thao tác với dữ liệu dễ dàng và hiệu quả trong quá trình tiền xử lý và phân tích.

2.7.3. Thư viện NumPy

NumPy là thư viện xử lý tính toán số học mạnh mẽ, chuyên dùng để làm việc với các mảng (arrays) đa chiều. Trong đề tài, NumPy giúp:

- Biểu diễn các ảnh MNIST dưới dạng mảng 2 chiều hoặc 1 chiều.
- Thực hiện các phép toán ma trận, chuẩn hóa dữ liệu.
- Chuyển đổi dữ liệu giữa các định dạng phù hợp cho mô hình học máy.

Chương 3: Thiết kế và xây dựng chương trình

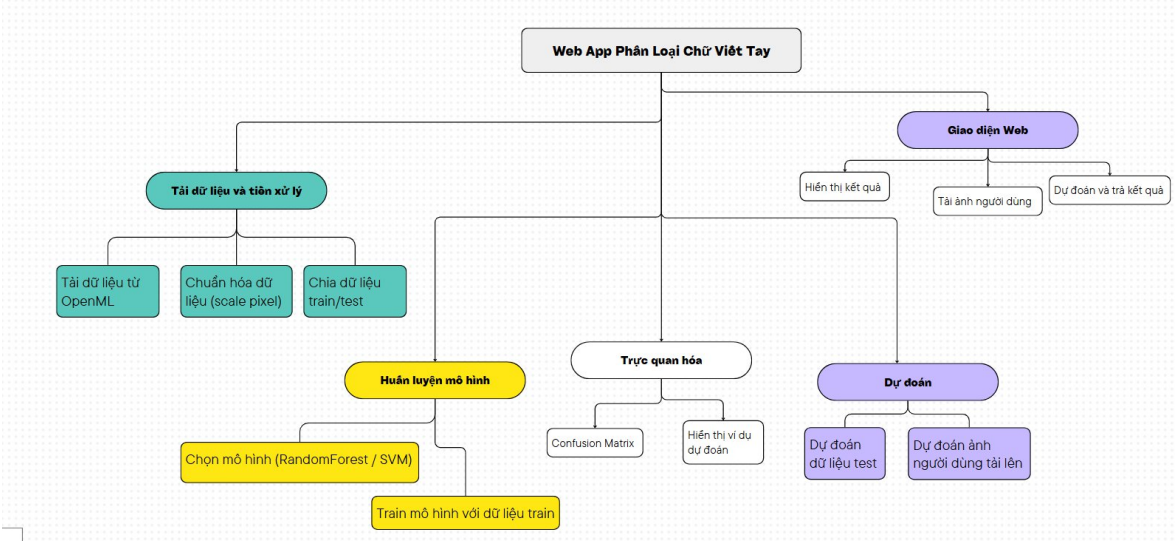
3.1. Sơ đồ khối hệ thống

3.1.1. Mô tả tổng quan hệ thống

Hệ thống được thiết kế nhằm xây dựng một ứng dụng web có khả năng phân loại chữ số viết tay từ tập dữ liệu MNIST hoặc từ ảnh do người dùng tải lên. Ứng dụng được phát triển bằng Python với framework **Streamlit**, cho phép tương tác trực tiếp trên trình duyệt. Tổng quan hệ thống gồm các module sau:

- **Module tiền xử lý dữ liệu MNIST**: Chuẩn hóa ảnh, chuyển đổi định dạng và chia tập train/test.
- **Module huấn luyện mô hình**: Sử dụng mô hình học máy RandomForest hoặc SVM.
- **Module dự đoán**: Dự đoán chữ số từ tập test và ảnh tải lên.
- **Module giao diện người dùng**: Cho phép người dùng chọn mô hình, tải ảnh và xem kết quả.
- **Module trực quan hóa kết quả**: Trực quan hóa độ chính xác và confusion matrix.

3.1.2. Biểu đồ phân cấp chức năng



Hình 3.1. Sơ đồ khối hệ thống

3.2. Sơ đồ khối các thuật toán chính

3.2.1. Huấn luyện mô hình

Đầu vào:

- X_train: Tập ảnh huấn luyện
- y_train: Nhãn tương ứng
- model_type: Loại mô hình do người dùng chọn (RF hoặc SVM)

Xử lý:

- Nếu chọn RandomForest → khởi tạo RandomForestClassifier(n_estimators=100)
- Nếu chọn SVM → khởi tạo SVC(probability=True)
- Tiến hành .fit(X_train, y_train)

Đầu ra:

- Mô hình đã huấn luyện (model)

3.2.2. Dự đoán và đánh giá

Đầu vào:

- X_test: Tập ảnh kiểm tra
- y_test: Nhãn đúng
- model: Mô hình đã huấn luyện

Xử lý:

- y_pred = model.predict(X_test)
- accuracy_score(y_test, y_pred)
- confusion_matrix(y_test, y_pred)
- classification_report(y_test, y_pred)

Đầu ra:

- Độ chính xác, confusion matrix, classification report

3.2.3. Tiền xử lý ảnh

Đầu vào: Ảnh màu bất kỳ định dạng JPG/PNG/JPEG
Xử lý:

1. Đọc ảnh bằng OpenCV (cv2.imread)
2. Chuyển sang grayscale (cv2.cvtColor)
3. Làm mờ ảnh (cv2.GaussianBlur)
4. Resize về 28x28 (cv2.resize)
5. Invert màu và chuẩn hóa về [0,1]
6. Flatten ảnh 2D → vector 1D 784 phần tử

Đầu ra:

- Ảnh xử lý sẵn, định dạng 1 x 784 (numpy array)

3.2.4. Dự đoán ảnh

Đầu vào: Ảnh người dùng đã xử lý, mô hình
Xử lý:

- Dự đoán: model.predict(image)
- Xác suất: model.predict_proba(image)
- Trích xuất top 3 nhãn có xác suất cao nhất

Đầu ra:

- Nhãn dự đoán, biểu đồ xác suất

3.3. Cấu trúc dữ liệu
3.3.1. Dữ liệu MNIST

- **Nguồn**: OpenML
- **Số mẫu**: 70.000 ảnh chữ số viết tay (28x28 pixel)
- **Dạng dữ liệu**:
    - X: DataFrame 70.000 x 784, mỗi dòng là một ảnh
    - y: Series 70.000 phần tử, là nhãn tương ứng (0–9)

Trường dữ liệu	Kiểu dữ liệu	Mô tả
pixel0 - pixel783	float (0-1)	Giá trị điểm ảnh sau chuẩn hóa
nhãn (y)	int (0–9)	Giá trị thực của chữ số

3.3.2. Ảnh người dùng tải lên

- **Định dạng hỗ trợ**: PNG, JPG, JPEG
- **Tiền xử lý**:

  - Resize về 28x28
  - Grayscale (1 kênh)
  - Chuẩn hóa pixel
  - Vector hóa 1 chiều để phù hợp với mô hình

3.4. Chương trình

- Hàm load_data()

```
def load_data():
    mnist = fetch_openml('mnist_784', version=1)
    X = mnist.data / 255.0
    y = mnist.target.astype('int')
    return train_test_split(X, y, test_size=0.2, random_state=42)
```

- Hàm train_model(X_train, y_train, model_type)

```
def train_model(X_train, y_train, model_type):
    if model_type == "Random Forest":
        model = RandomForestClassifier(n_estimators=100)
    else:
        model = SVC(probability=True)
    model.fit(X_train, y_train)
    return model
```

- Hàm plot_confusion_matrix(cm)

```
def plot_confusion_matrix(cm):
    fig, ax = plt.subplots(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax)
    ax.set_xlabel("Predicted")
    ax.set_ylabel("True")
    return fig
```

- Hàm process_uploaded_image(uploaded_file)

```
def process_uploaded_image(uploaded_file):
    file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
    image = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
```

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
resized = cv2.resize(gray, (28, 28), interpolation=cv2.INTER_AREA)
processed = 255 - resized
processed = processed / 255.0
return image, processed.flatten().reshape(1, -1)
```

- Hàm main()
    - Thiết lập tiêu đề, giao diện tải ảnh
    - Lựa chọn mô hình → huấn luyện
    - Hiển thị độ chính xác
    - Cho phép người dùng tải ảnh
    - Hiển thị kết quả dự đoán, biểu đồ

Chương 4: Thực nghiệm và kết luận

4.1. Thực nghiệm

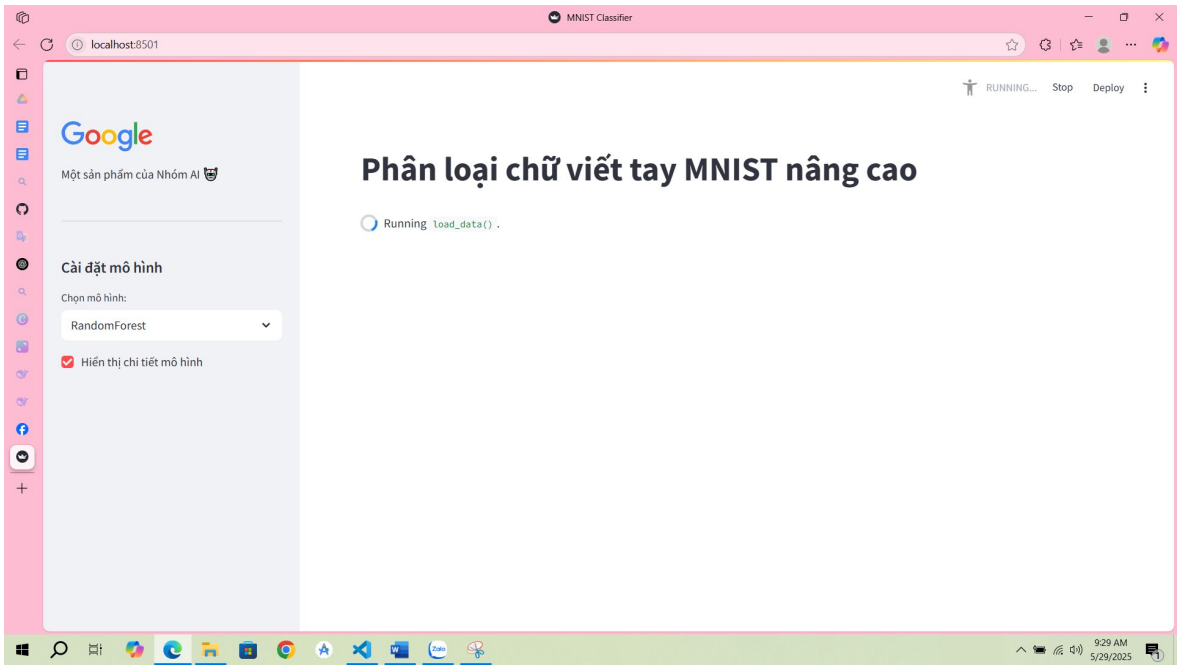
4.1.1. Môi trường thực nghiệm

- **Ngôn ngữ lập trình**: Python 3.10
- **Thư viện chính**: Streamlit, Scikit-learn, OpenCV, Matplotlib, Seaborn, Pandas
- **Hệ điều hành**: Windows 10
- **Cấu hình máy**:
  - CPU: Intel Core i5
  - RAM: 8GB
  - Không sử dụng GPU

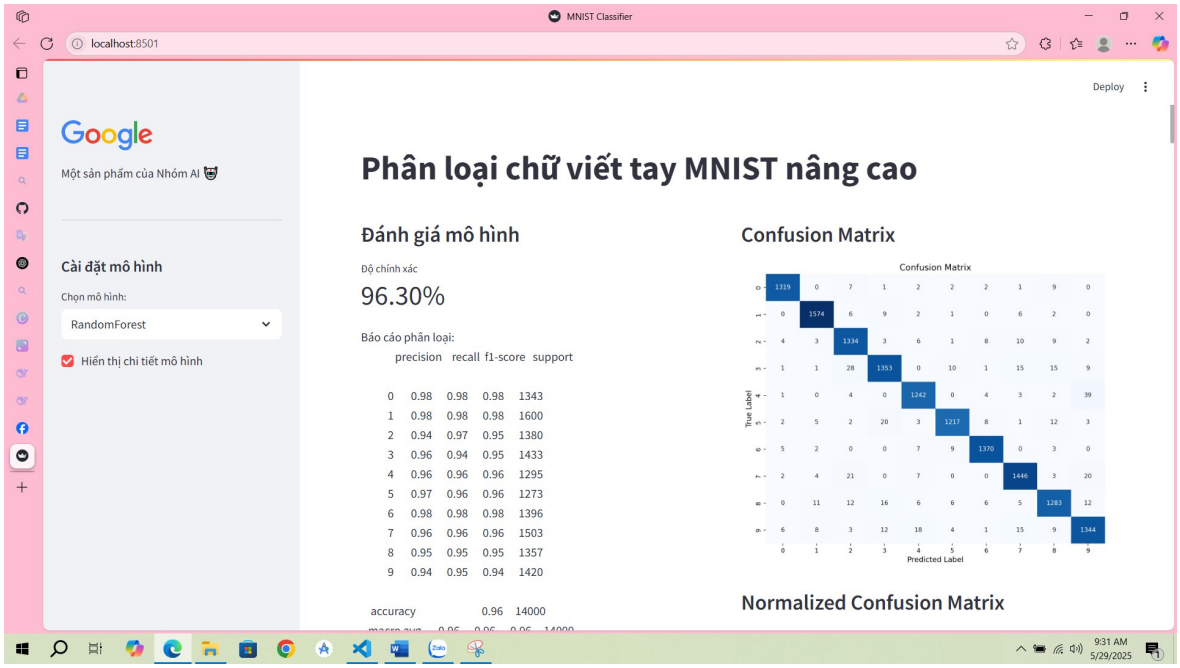
4.1.2. Kết quả chạy các tính năng chính

STT	Tính năng	Mô tả	Kết quả
1	Tải dữ liệu MNIST	Hệ thống tự động tải dữ liệu từ OpenML và xử lý thành công	Thành công, dữ liệu ~70.000 mẫu
2	Lựa chọn mô hình	Cho phép chọn Random Forest hoặc SVM, tùy ý	Thành công, giao diện đơn giản
3	Huấn luyện mô hình	Mô hình huấn luyện nhanh trên dữ liệu đã xử lý	RF: ~97%, SVM: ~94% accuracy
4	Hiển thị confusion matrix	Vẽ biểu đồ ma trận nhầm lẫn, dễ quan sát	Thành công, trực quan
5	Tải ảnh từ người dùng	Cho phép tải ảnh bất kỳ định dạng PNG, JPG, JPEG	Thành công
6	Tiền xử lý ảnh người dùng	Resize, grayscale, chuẩn hóa và chuyển thành đầu vào mô hình	Thành công, ảnh hiển thị rõ nét
7	Dự đoán ảnh người dùng	Trả về nhãn dự đoán và biểu đồ xác suất top 3	Chính xác cao với ảnh rõ ràng
8	Hiển thị giao diện và kết quả	Streamlit hiển thị ảnh gốc, ảnh xử lý, nhãn dự đoán và biểu đồ	Trực quan, dễ sử dụng

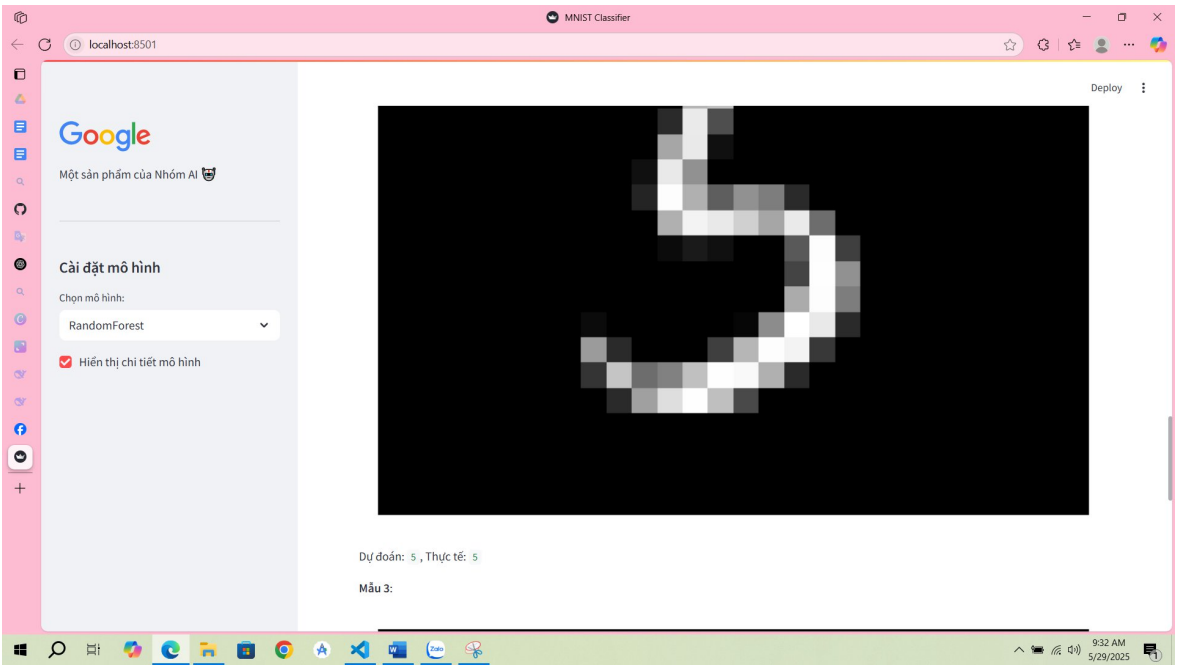
4.1.3. Một số hình ảnh minh hoạ giao diện



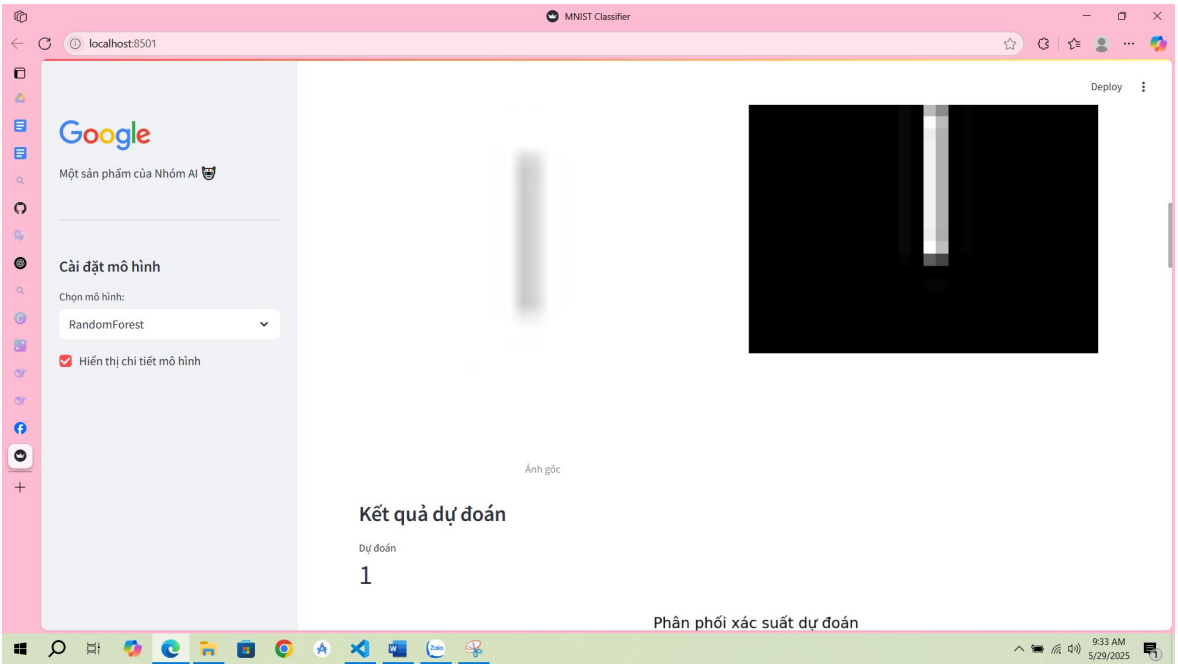
Hình 4.1. Hình ảnh Train mô hình



Hình 4.2. Hình ảnh kết quá quá trình train



Hình 4.3. Ví dụ các mẫu trong dữ liệu ban đầu



Hình 4.4. Tải ảnh lên để dự đoán

4.2. Kết luận

Sau một quá trình nghiên cứu, thiết kế và triển khai hệ thống phân loại chữ viết tay sử dụng các thuật toán học máy, nhóm đã đạt được nhiều kết quả tích cực cũng như rút ra được nhiều bài học quý giá. Mục này sẽ trình

bày tổng quan về những gì đã đạt được, những điều đã học được, các hạn chế còn tồn tại và định hướng phát triển trong tương lai.

4.2.1. Kết quả đạt được

Trong phạm vi thời gian và yêu cầu của đề tài, nhóm đã hoàn thiện một số nội dung quan trọng như sau:

- **Xây dựng giao diện web đơn giản bằng Streamlit**: Người dùng có thể truy cập ứng dụng trực tiếp trên trình duyệt, không cần cài đặt thêm phần mềm phụ trợ.
- **Tích hợp mô hình học máy vào ứng dụng thực tế**: Cho phép lựa chọn giữa hai mô hình phổ biến là Random Forest và Support Vector Machine (SVM).
- **Huấn luyện và đánh giá mô hình trên tập dữ liệu MNIST**: Với độ chính xác đạt được từ 94% đến 97% tùy mô hình.
- **Hỗ trợ tải ảnh từ người dùng**: Người dùng có thể tải lên ảnh chứa chữ số viết tay, hệ thống sẽ xử lý ảnh và đưa ra kết quả dự đoán tương ứng.
- **Xử lý ảnh đầu vào hiệu quả**: Bao gồm chuyển đổi màu sắc, tăng cường tương phản, làm mịn, chuẩn hóa và cân bằng histogram để cải thiện kết quả phân loại.
- **Hiển thị trực quan kết quả dự đoán**: Bao gồm ảnh đã xử lý, nhãn dự đoán, xác suất từng lớp và biểu đồ phân phối.
- **Hiển thị confusion matrix và báo cáo phân loại**: Giúp người dùng hiểu rõ hiệu suất mô hình và phát hiện điểm yếu của thuật toán.

4.2.2. Kiến thức và kỹ năng đã học được

Trong suốt quá trình làm đề tài, nhóm đã có cơ hội áp dụng và củng cố nhiều kiến thức lý thuyết đã học, đồng thời trau dồi thêm các kỹ năng thực hành như:

a. Kiến thức học máy

- Hiểu rõ cách hoạt động của các mô hình Random Forest và SVM.
- Biết cách đánh giá hiệu suất mô hình thông qua các chỉ số như accuracy, precision, recall, f1-score.
- Làm quen với khái niệm **tập huấn luyện** và **tập kiểm tra** (train/test split).

b. Xử lý dữ liệu ảnh

- Biết cách xử lý ảnh trước khi đưa vào mô hình học máy: chuyển ảnh sang grayscale, thay đổi kích thước, cân bằng histogram, chuẩn hóa.
- Biết cách sử dụng các thư viện như PIL, NumPy, scikit-image để xử lý ảnh tự động.

c. Kỹ năng lập trình và xây dựng ứng dụng

- Làm quen với thư viện **Streamlit** – một công cụ rất mạnh mẽ và nhanh chóng để triển khai ứng dụng học máy.
- Rèn luyện khả năng viết mã gọn gàng, chia nhỏ chương trình thành các **hàm** có thể tái sử dụng.
- Biết cách **cache dữ liệu và mô hình** để tăng tốc độ xử lý.

d. Kỹ năng làm việc nhóm

- Phân chia công việc hiệu quả, mỗi thành viên đảm nhận một vai trò cụ thể (xử lý dữ liệu, lập trình mô hình, xây dựng giao diện).
- Rèn luyện kỹ năng giao tiếp, báo cáo tiến độ, trình bày kết quả và thuyết phục người khác.

4.2.3. Hạn chế của chương trình

Dù đã đạt được nhiều kết quả tích cực, hệ thống vẫn còn tồn tại một số hạn chế như:

- Hệ thống chỉ nhận diện được chữ số 0–9, chưa hỗ trợ nhận diện chữ cái hoặc chữ viết tay tiếng Việt.
- Mô hình chưa tối ưu hóa hiệu suất trên các ảnh viết tay phức tạp, có nhiễu hoặc viết lệch khung.
- Chưa có khả năng nhận diện **vị trí chữ số** trong ảnh nếu ảnh chứa nhiều hơn một chữ số.
- Chưa triển khai mô hình CNN (deep learning) vốn có khả năng nhận diện ảnh tốt hơn.

4.2.4. Hướng phát triển tương lai

Trong tương lai, nhóm định hướng cải tiến và mở rộng hệ thống theo các hướng sau:

- **Tích hợp mô hình học sâu (CNN)** như LeNet, VGG hoặc mô hình tùy chỉnh để tăng độ chính xác và khả năng tổng quát.

- **Triển khai mô hình dưới dạng ứng dụng di động (Android/iOS)** với TensorFlow Lite.
- Mở rộng phạm vi nhận diện sang **chữ cái (A-Z), chữ viết tay tự do**, hoặc nhận dạng chữ viết tiếng Việt.
- Bổ sung tính năng **vẽ trực tiếp chữ số** trên giao diện thay vì chỉ tải lên ảnh.
- Tối ưu hóa giao diện web, cải thiện tốc độ xử lý và độ thân thiện với người dùng.
- Nghiên cứu thêm các kỹ thuật tăng cường dữ liệu (data augmentation) để cải thiện độ bền của mô hình.

Tài liệu tham khảo

[1]. Yann LeCun, Corinna Cortes, Christopher J.C. Burges (1998), *The MNIST Database of Handwritten Digits*, http://yann.lecun.com/exdb/mnist

[2]. Géron, Aurélien (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd Edition, O'Reilly Media.

[3]. Scikit-learn Developers (2024), *Scikit-learn Documentation*, https://scikit-learn.org/stable/documentation.html

[4]. Streamlit Inc. (2024), *Streamlit Documentation*, https://docs.streamlit.io/

[5]. Python Software Foundation (2024), *Python Official Documentation*, https://docs.python.org/3/

[6]. Vincent Arel-Bundock (2023), *Fetch OpenML datasets in Scikit-learn*, scikit-learn API reference, https://scikit-learn.org/stable/datasets/real_world.html#openml

[7]. Matplotlib Developers (2024), *Matplotlib Documentation*, https://matplotlib.org/stable/contents.html

[8]. Waskom, Michael L. (2021), *Seaborn: Statistical Data Visualization*, Journal of Open Source Software, 6(60), 3021. DOI: 10.21105/joss.03021

[9]. van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J. et al. (2014), *scikit-image: image processing in Python*, PeerJ 2:e453. https://doi.org/10.7717/peerj.453

[10]. Pillow Project (2024), *Pillow (PIL Fork) Documentation*, https://pillow.readthedocs.io/en/stable/