

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM
KHOA KHOA HỌC MÁY TÍNH



BÁO CÁO ĐỒ ÁN MÔN HỌC

CS114 - MÁY HỌC

Đề tài: A. NHẬN DẠNG CHỮ SỐ VIẾT TAY
B. DỰ ĐOÁN ĐIỂM MÔN HỌC

Giảng viên hướng dẫn: ThS. Phạm Nguyễn Trường An

PGS. TS Lê Đình Duy

Nhóm sinh viên thực hiện:

- | | | |
|------------------------|----------|-------------|
| 1. Lê Ngọc Phương Thảo | 23521467 | Nhóm trưởng |
| 2. Nguyễn Phú Thành | 23521452 | Thành viên |
| 3. Trần Trọng Đức Tài | 23521379 | Thành viên |

Link Github repository: [lnghgthao/cs114-hand-written-digit-classification](https://github.com/lnghgthao/cs114-hand-written-digit-classification)

MỤC LỤC

A. NHẬN DẠNG CHỮ SỐ VIẾT TAY.....	3
Chương 0: Những thay đổi sau khi vắn đáp.....	3
Chương 1: Giới thiệu bài toán và dữ liệu.....	3
1.1. Mô tả bài toán.....	3
1.2 Input và Output của hệ thống.....	3
1.3 Dữ liệu và các nhóm tham gia.....	3
Chương 2: Data processing.....	4
2.1. Nạp dữ liệu.....	4
2.2. Tăng cường dữ liệu.....	4
2.2.1. Tăng cường cơ bản (Basic augmentation) – Hiệu quả nhất.....	5
2.2.2. Tăng cường nâng cao (Advanced augmentation).....	5
2.2.3. Không tăng cường (Only resize) – Kém hiệu quả nhất.....	5
2.3. Xử lý mẫu lỗi trong quá trình nạp dữ liệu.....	6
2.3.1. Trả về None khi gặp ảnh lỗi trong Dataset.....	6
2.3.2. Định nghĩa hàm collate_fn để loại bỏ các mẫu lỗi trong batch.....	6
2.4. Chia dữ liệu train/val.....	7
Chương 3: Training.....	7
3.1. Mô hình thử nghiệm.....	7
3.2. Cấu hình huấn luyện.....	8
3.3. Quy trình huấn luyện.....	9
Chương 4: Evaluation.....	10
4.1. Kết quả huấn luyện.....	10
4.2. Nhận xét.....	11
Chương 5: Kết luận.....	11
B. DỰ ĐOÁN ĐIỂM MÔN HỌC.....	13
Chương 0: Những thay đổi sau khi vắn đáp.....	13
Chương 1: Giới thiệu bài toán và dữ liệu.....	13
1.1 Mô tả bài toán.....	13
1.2 Input và output của hệ thống.....	13

1.3 Nguồn dữ liệu.....	13
1.4 Đặc điểm dữ liệu.....	14
Chương 2: Data processing.....	15
2.1 Import các thư viện và cài đặt.....	15
2.2 Tải dữ liệu và thử nghiệm.....	15
2.3 Làm sạch dữ liệu.....	15
2.3.1 Loại bỏ các kí tự đặc biệt.....	15
2.3.2 Trích xuất thông tin từ cột judgement.....	16
2.4 Trích xuất đặc trưng (Feature engineering).....	16
Chương 3: Training.....	18
3.1 Lựa chọn model, cách thử nghiệm và đánh giá.....	18
3.1.1 Lựa chọn model.....	18
3.1.2 Cách thử nghiệm và đánh giá.....	18
3.2 Huấn luyện model.....	18
3.2.1. Dự đoán điểm thực hành:.....	18
3.2.2. Dự đoán điểm quá trình:.....	19
3.2.3. Dự đoán điểm cuối kỳ:.....	20
3.2.4. Dự đoán điểm trung bình tích lũy:.....	21
3.3. Tổng quát:.....	21
Chương 4: Evaluation.....	22
Chương 5: Kết luận.....	22

NỘI DUNG

A. NHẬN DẠNG CHỮ SỐ VIẾT TAY

Chương 0: Những thay đổi sau khi vấn đáp

1. Bổ sung thêm phần huấn luyện bằng dữ liệu gốc (không tăng cường) để so sánh, đánh giá các mô hình.
2. Bổ sung thêm phần visualize các ảnh để thấy rõ hiệu quả của việc tăng cường.

Chương 1: Giới thiệu bài toán và dữ liệu

1.1. Mô tả bài toán

- Bài toán nhận diện chữ số viết tay là một trong những bài toán cơ bản và quan trọng trong lĩnh vực thị giác máy tính (computer vision) và học máy (machine learning). Đây là bài toán phân loại đa lớp (multi-class classification) với mục tiêu xác định chính xác chữ số từ 0 đến 9 từ các hình ảnh chữ số được viết tay.

1.2 Input và Output của hệ thống

- **Input (Đầu vào):** Hình ảnh chữ số viết tay ở các định dạng khác nhau (JPEG, PNG, v.v.).
- **Output (Đầu ra):** Một số nguyên từ 0 đến 9 đại diện cho chữ số được dự đoán.

1.3 Dữ liệu và các nhóm tham gia

- Dữ liệu được thu thập từ nhiều nhóm sinh viên trong lớp CS114.P21, với cấu trúc thư mục được tổ chức có thư mục có tên **hand_written_digit**, trong thư mục này chứa các folder có pattern chung là ??52???? đại diện cho MSSV của mỗi thành viên trong nhóm.
- Tổng cộng có 9987 ảnh trong tập test và 7059 ảnh được hệ thống xử lý từ các nhóm sinh viên, tạo thành một dataset đa dạng với nhiều phong cách (màu sắc, kiểu chữ) viết tay khác nhau.

Chương 2: Data processing

2.1. Nạp dữ liệu

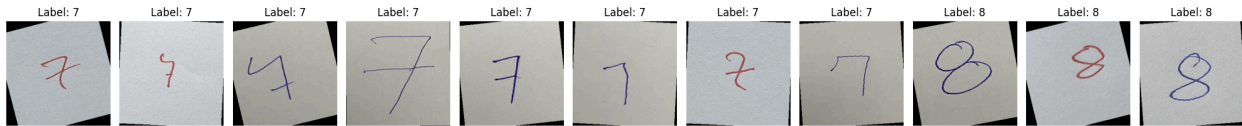
- Dựa trên phần code baseline do giảng viên cung cấp, nhóm đã tiến hành điều chỉnh và mở rộng hàm nạp dữ liệu để phù hợp hơn với yêu cầu bài toán và cấu trúc bộ dữ liệu thực tế. Cụ thể, dữ liệu được tổ chức trong các thư mục con thuộc thư mục lớn ML-Do-An. Trong mỗi thư mục con, ảnh chữ số viết tay được đặt trong thư mục `hand_written_digit`, và tên ảnh được đặt theo định dạng `<số>_*.jpg` hoặc các đuôi mở rộng khác như `.png`, `.jpeg`. Để đảm bảo tính chính xác trong việc lấy đúng các ảnh cần thiết, nhóm sử dụng thư viện `glob` để lọc chính xác các đường dẫn chứa chuỗi ký tự `??52????` tương ứng với các mssv.
- Sau khi thu thập đầy đủ đường dẫn ảnh, nhóm xây dựng một lớp `DigitDataset` kế thừa từ `torch.utils.data.Dataset` để hỗ trợ việc load ảnh theo batch thông qua `DataLoader` của PyTorch. Trong lớp này, ảnh được đọc bằng thư viện `PIL`, chuyển về định dạng `RGB` và tự động gán nhãn dựa trên chữ số đầu tiên của tên file. Ngoài ra, nhóm cũng tích hợp sẵn chức năng truyền `augmentation` thông qua tham số `aug`, giúp dễ dàng thử nghiệm các phương pháp xử lý ảnh khác nhau ở bước sau.

2.2. Tăng cường dữ liệu

- Để cải thiện hiệu suất mô hình và khả năng tổng quát hóa, nhóm đã thử nghiệm ba phương pháp tăng cường dữ liệu (`data augmentation`) với độ phức tạp khác nhau, áp dụng cho tập huấn luyện ảnh chữ số viết tay từ 0 đến 9.
- Quá trình thử nghiệm cho thấy phương pháp **tăng cường cơ bản (`basic augmentation`)** cho kết quả tốt nhất — giúp mô hình đạt độ chính xác cao hơn, huấn luyện ổn định, và tránh được hiện tượng `overfitting`. Các phương pháp được sắp xếp dưới đây theo thứ tự từ hiệu quả nhất đến kém hơn.

2.2.1. Tăng cường cơ bản (Basic augmentation) – *Hiệu quả nhất*

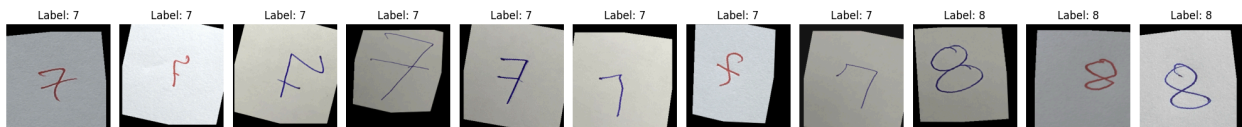
- Phương pháp này áp dụng xoay ảnh ngẫu nhiên ($\pm 15^\circ$), kết hợp với chuẩn hóa, giúp mô hình học được tính biến thiên trong cách viết mà không làm mất đặc trưng của chữ số.



Các ảnh được xoay vừa đủ, không bị biến đổi quá mức so với ảnh gốc

2.2.2. Tăng cường nâng cao (Advanced augmentation)

- Mặc dù sử dụng nhiều kỹ thuật biến đổi như lật ảnh, biến dạng phối cảnh, thay đổi độ sáng, dịch chuyển,... nhưng phương pháp này lại gây ra hiện tượng **quá nhiễu (noise)**, khiến mô hình khó học hoặc học sai đặc trưng chữ số. Kết quả không tốt bằng augmentation cơ bản.



Các ảnh số 7 bị lật ngược

2.2.3. Không tăng cường (Only resize) – *Kém hiệu quả nhất*

- Đây là phương pháp baseline, trong đó ảnh chỉ được **resize** về kích thước cố định và chuyển sang định dạng tensor. **Không có bất kỳ phép biến đổi nào khác**, bao gồm cả chuẩn hóa (normalize). Tuy đơn giản và dễ triển khai, nhưng cách tiền xử lý này lại cho kết quả **tệ nhất** trong các phương pháp đã thử nghiệm.








Các ảnh có màu sắc không rõ nét

- Cụ thể, do thiếu bước chuẩn hóa, ảnh đầu vào khi được visualize cho thấy có hiện tượng **màu sắc mờ nhạt, thiếu tương phản**, khiến một số chữ số khó nhận diện hơn cả bằng mắt thường. Điều này ảnh hưởng tiêu cực đến quá trình học của mô hình, dẫn đến hiệu suất huấn luyện thấp, độ chính xác không cải thiện nhiều qua các epoch và kết quả khi thực hiện dự đoán trên test dataset rất thấp.
- Điều đó cho thấy việc **bỏ qua bước chuẩn hóa ảnh** là không phù hợp trong bài toán này, kể cả khi không áp dụng các phép tăng cường khác.

2.3. Xử lý mẫu lỗi trong quá trình nạp dữ liệu

- Trong quá trình kiểm tra dữ liệu thủ công, nhóm nhận thấy có một số ảnh trong tập dữ liệu bị lỗi.

 0_5	5/21/2025 11:27 AM	JPEG File	1 KB
 0_6	5/21/2025 11:27 AM	JPEG File	1 KB
 3_6	5/21/2025 11:27 AM	JPEG File	1 KB
 4_9	5/21/2025 11:27 AM	JPEG File	1 KB
 6_1	5/21/2025 11:27 AM	JPEG File	1 KB

- Để giải quyết vấn đề này, nhóm đã xử lý theo hai bước:

2.3.1. Trả về *None* khi gặp ảnh lỗi trong *Dataset*

- Trong lớp **DigitDataset**, nếu ảnh không thể đọc hoặc chuyển đổi được, hàm **__load_image** sẽ không trả về tensor ảnh và nhãn như bình thường, mà sẽ trả về **None**. Điều này giúp đánh dấu các mẫu lỗi một cách rõ ràng để loại bỏ sau đó.

2.3.2. Định nghĩa hàm `collate_fn` để loại bỏ các mẫu lỗi trong batch

- Do `torch.utils.data.DataLoader` gom dữ liệu theo batch, nếu một phần tử trong batch là `None`, toàn bộ batch đó có thể bị lỗi. Nhóm đã định nghĩa một hàm `remove_none_samples` để lọc bỏ các mẫu **None** trước khi ghép batch. Hàm này được truyền vào **DataLoader** thông qua tham số `collate_fn`. Nhờ đó, quá trình huấn luyện không bị dừng lại ngay cả khi tồn tại ảnh lỗi trong tập dữ liệu. Đồng thời, nhóm vẫn có thể theo dõi được số lượng ảnh lỗi qua các thông báo được in ra từ `__load_image`.

2.4. Chia dữ liệu train/val

- Sau khi thu thập đầy đủ các đường dẫn ảnh và xử lý lỗi, nhóm tiến hành chia dữ liệu thành hai phần: tập huấn luyện (train set) và tập kiểm tra (validation set) theo tỷ lệ 90% – 10%. Cách chia này được lựa chọn nhằm tối ưu hóa quá trình học của mô hình, trong khi vẫn giữ lại một phần nhỏ dữ liệu để đánh giá khả năng tổng quát hoá trong suốt quá trình huấn luyện.
- Tập huấn luyện bao gồm phần lớn ảnh, được sử dụng để điều chỉnh trọng số của mô hình. Tập kiểm tra (validation) giúp theo dõi hiệu suất mô hình sau mỗi epoch, đồng thời hỗ trợ cơ chế dừng sớm (early stopping) nếu mô hình không tiếp tục cải thiện.
- Mỗi phần dữ liệu được kết hợp với một pipeline xử lý ảnh riêng:
 - + Tập huấn luyện: áp dụng một trong các phương pháp tăng cường (augmentation) được mô tả tại mục 2.2.
 - + Tập kiểm tra (validation): luôn sử dụng phương pháp xử lý cố định gồm resize và chuẩn hóa (normalize), nhằm đảm bảo tính nhất quán và công bằng trong đánh giá.

Chương 3: Training

3.1. Mô hình thử nghiệm

- Trong quá trình triển khai, nhóm đã thử nghiệm ba kiến trúc mô hình học sâu thuộc dòng CNN hiện đại, được huấn luyện sẵn trên tập ImageNet và hỗ trợ tốt cho việc fine-tune:

Mô hình thử nghiệm	Trọng số pretrained được sử dụng
EfficientNet-B1	EfficientNet_B1_Weights.IMAGENET1K_V2
EfficientNet-B2	EfficientNet_B2_Weights.IMAGENET1K_V1
RegNet-Y 1.6GF	RegNet_Y_1_6GF_Weights.IMAGENET1K_V2

- Nhóm lựa chọn các mô hình này dựa trên các yếu tố tham khảo từ bảng thống kê “**Table of all available classification weights**” trên tài liệu chính thức của PyTorch:

Mô hình và trọng số tương ứng	Acc@1	Acc@5	Params	GFLOPS
EfficientNet_B1_Weights.IMAGENET1K_V2	79.838	94.934	7.8M	0.69
EfficientNet_B2_Weights.IMAGENET1K_V1	80.608	95.31	9.1M	1.09
RegNet_Y_1_6GF_Weights.IMAGENET1K_V2	80.876	95.444	11.2M	1.61

- + **Số lượng tham số (params)** ở mức vừa phải, phù hợp với dữ liệu và nền tảng huấn luyện.
- + **Độ chính xác top-1 (Acc@1)** và **top-5 (Acc@5)** cao, đảm bảo khả năng nhận diện hiệu quả.
- + **Độ phức tạp tính toán (GFLOPS)** giúp tiết kiệm thời gian huấn luyện.
- Với mỗi mô hình, nhóm đã tinh chỉnh lớp cuối (classifier head) để phù hợp với bài toán phân loại 10 lớp (ứng với các chữ số từ 0 đến 9). Ngoài ra, nhóm cũng thêm lớp **Dropout** nhằm hạn chế hiện tượng overfitting.

3.2. Cấu hình huấn luyện

- Các tham số được thiết lập thống nhất cho cả ba mô hình như sau:

Thành phần	Giá trị thiết lập	Lý do chọn
Loss function	CrossEntropyLoss	Là hàm mất mát phổ biến trong các bài toán phân loại nhiều lớp
Optimizer	Adam	Optimizer phổ biến, hội tụ nhanh và ổn định , đặc biệt hiệu quả với tập dữ liệu vừa và không cần tinh chỉnh nhiều như SGD
Learning rate	0.0001	Là giá trị đủ nhỏ để mô hình học từ từ, không nhảy quá nhanh
Weight decay	0.0001	Giảm overfitting nhẹ , giúp mô hình không học quá khớp với tập train.
Batch size	64	Đưa được nhiều ảnh vào mỗi batch để tính toán hiệu quả hơn mà không bị tràn bộ nhớ GPU. Nhóm đã thử nghiệm các giá trị khác và thấy 64 là tối ưu nhất .
Số epoch tối đa	15	Đủ để mô hình học hiệu quả, vì qua quan sát thì độ chính xác đã ổn định sau khoảng dưới 10 epoch , nên không cần huấn luyện quá lâu.
Early stopping patience	2	Dừng sớm nếu mô hình không cải thiện thêm, nhằm tiết kiệm thời gian huấn luyện và tránh overfitting.
Thiết bị huấn luyện	GPU	CPU quá chậm so với GPU

3.3. Quy trình huấn luyện

- Trong mỗi vòng lặp huấn luyện, các mô hình đều được huấn luyện với quy trình gồm **hai pha chính trong mỗi epoch**:
 - + **Train phase**: Mô hình được đặt ở chế độ huấn luyện, thực hiện lan truyền xuôi và lan truyền ngược để cập nhật trọng số dựa trên tập huấn luyện.

- + **Validation phase:** Mô hình được chuyển sang chế độ đánh giá, tính toán loss và độ chính xác (accuracy) trên tập validation mà không cập nhật trọng số.
- Sau mỗi epoch:
 - + Nếu **độ chính xác trên tập validation cao hơn** so với các lần trước đó, trọng số mô hình sẽ được lưu lại (checkpoint).
 - + Ngược lại, nếu **không có cải thiện nào sau 2 epoch liên tiếp**, quá trình huấn luyện sẽ **dừng sớm (early stopping)** để tiết kiệm tài nguyên tính toán và tránh overfitting. Khi đó, mô hình được chọn là mô hình có kết quả validation tốt nhất trước đó.
- Ngoài ra, nhóm cũng sử dụng cơ chế lọc mẫu lỗi trong batch thông qua hàm **collate_fn** nhằm đảm bảo rằng dữ liệu bị lỗi không ảnh hưởng đến quá trình huấn luyện (phần này đã trình bày chi tiết ở Chương 2).

Chương 4: Evaluation

4.1. Kết quả huấn luyện

- Sau khi huấn luyện toàn bộ mô hình, nhóm đã dùng các mô hình để dự đoán trên tập dữ liệu test, sau đó gửi lên wecode để xem hiệu quả của từng mô hình. Nhóm ghi nhận kết quả như sau:

Large dataset	EffNet_B1._V2	EffNet_B2._V1	RegNet_Y_1_6GF._V2
Raw Data	5233/9987	4933/9987	5850/9987
Basic Preprocess	9716/9987	9738/9987	9766/9987
Advanced Preprocess	9665/9987	9682/9987	9690/9987

Small dataset	EffNet_B1._V2	EffNet_B2._V1	RegNet_Y_1_6GF._V2
Raw Data	1604/2939	1242/2939	1609/2939
Basic Preprocess	2825/2939	2850/2939	2856/2939
Advanced Preprocess	2787/2939	2824/2939	2808/2939

4.2. Nhận xét

- **Hiệu quả của tăng cường dữ liệu:** Tăng cường cơ bản (basic augmentation) giúp tăng đáng kể độ chính xác so với khi không tăng cường hoặc tăng cường quá mức. Cách xoay ảnh nhẹ, kết hợp chuẩn hóa, giúp mô hình học được tốt hơn mà không bị nhiễu.
- **Tăng cường quá mức:** Dù áp dụng thêm nhiều kỹ thuật nâng cao như lật ảnh, biến dạng phối cảnh,... nhưng kết quả không vượt trội. Một số biến đổi gây khó khăn cho mô hình nhận diện, đặc biệt với những chữ số dễ nhầm như 6 và 9.
- **Hiệu quả của mô hình:** RegNet_Y_1.6GF cho kết quả tốt hơn so với 2 model còn lại.
- **EffNet_B1 với B2:** Sự khác biệt không đáng kể, cho thấy việc mở rộng mô hình (từ B1 sang B2) không cải thiện quá nhiều.

Chương 5: Kết luận

- Sau khi thực hiện đồ án, nhóm chúng em nhận thấy đây là một trải nghiệm thú vị và bổ ích. Việc trực tiếp triển khai một hệ thống nhận diện chữ số viết tay đã giúp nhóm hiểu rõ hơn về các bước chính trong một pipeline học sâu — từ tải và xử lý dữ liệu, xây dựng mô hình, huấn luyện, cho đến đánh giá kết quả.

- Thông qua quá trình thử nghiệm, nhóm đặc biệt ấn tượng với **tầm quan trọng của bước tăng cường dữ liệu (data augmentation)**. Chỉ với cùng một tập dữ liệu, việc áp dụng các kỹ thuật tiền xử lý hợp lý như xoay nhẹ ảnh, chuẩn hóa,... đã giúp cải thiện đáng kể hiệu suất mô hình, giảm hiện tượng overfitting, và tăng độ chính xác trên tập kiểm thử. Tuy nhiên, nhóm cũng nhận ra rằng **không phải lúc nào các kỹ thuật tăng cường nâng cao cũng mang lại kết quả tốt hơn**. Trong một số trường hợp, việc lật ảnh, biến dạng phối cảnh hoặc làm thay đổi quá nhiều đặc trưng chữ số có thể khiến mô hình học sai và giảm hiệu quả dự đoán.
- Bên cạnh đó, đồ án còn giúp nhóm **hiểu thêm về các kiến trúc CNN hiện đại như EfficientNet và RegNet**, cách fine-tune mô hình có sẵn từ ImageNet để áp dụng vào bài toán thực tế, và cách sử dụng nền tảng huấn luyện GPU như Kaggle hiệu quả. Nhóm cũng được củng cố kỹ năng làm việc nhóm, phân chia nhiệm vụ rõ ràng và xử lý các lỗi phát sinh trong quá trình triển khai.
- Qua đồ án này, nhóm chúng em cảm thấy tự tin hơn khi tiếp cận các bài toán thị giác máy tính phức tạp hơn trong tương lai, đồng thời rút ra được nhiều bài học thực tiễn từ quá trình làm việc với dữ liệu thực và mô hình học sâu.

B. DỰ ĐOÁN ĐIỂM MÔN HỌC

Chương 0: Những thay đổi sau khi vấn đáp

1. Phân tích độ tương quan giữa các đặc trưng gốc, đặc trưng trích xuất và output.
2. Thay đổi các đặc trưng dùng để dự đoán, đảm bảo ý nghĩa tương quan với điểm số.
3. Bổ sung postprocessing đảm bảo dự đoán nằm trong khoảng điểm từ 0 đến 10.

Chương 1: Giới thiệu bài toán và dữ liệu

1.1 Mô tả bài toán

- Bài toán đưa ra yêu cầu dự đoán điểm số học tập của sinh viên dựa trên hành vi lập trình của họ trên hệ thống Wecode. Đây là bài toán regression với output là một số thực biểu diễn điểm của sinh viên trong khoảng 0 đến 10 điểm.

1.2 Input và output của hệ thống

- Input (Đầu vào): Các thông tin về các lần nộp và bài làm của sinh viên được trích xuất từ data ban đầu.
- Output (Đầu ra): Một số thực nằm trong khoảng từ 0 đến 10 thể hiện điểm của học sinh.
- Cụ thể, hệ thống cần dự đoán bốn loại điểm:
 - + TH: Điểm thực hành
 - + QT: Điểm quá trình học tập
 - + CK: Điểm thi cuối kỳ
 - + TBTL: Điểm trung bình tích lũy

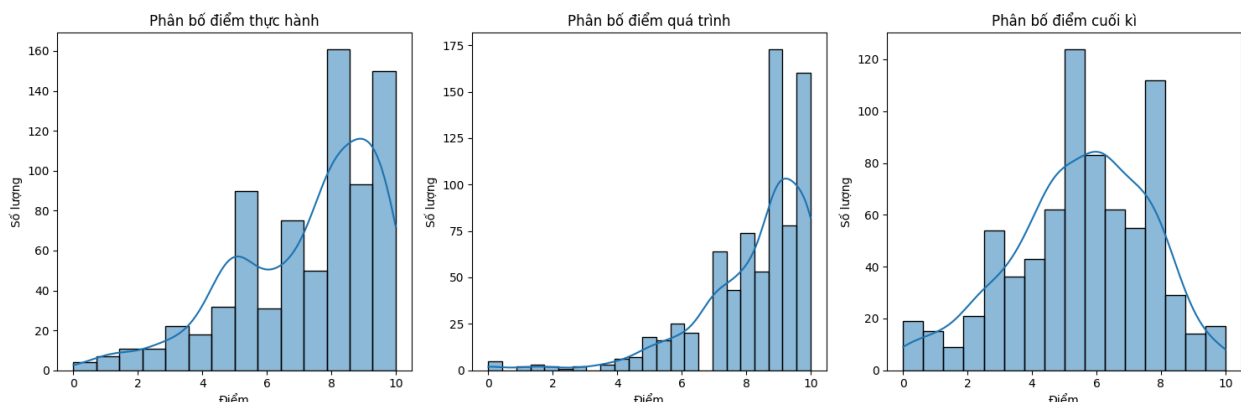
1.3 Nguồn dữ liệu

- Dataset chính được thu thập từ hệ thống Wecode với 11 cột thông tin và tổng cộng 295198 bản ghi submission. Dataset bao gồm:

- + Dataset chính (anonimized.csv):
 - Thông tin định danh: assignment_id, problem_id, hash (student ID).
 - Thông tin trạng thái: is_final, status, pre_score, coefficient.
 - Thông tin thời gian: created_at, updated_at.
 - Thông tin kỹ thuật: language_id, judgement.
- + Dataset điểm thật của sinh viên:
 - th-public.csv: Điểm thực hành thật của sinh viên.
 - qt-public.csv: Điểm quá trình thật của sinh viên.
 - ck-public.csv: Điểm cuối kỳ thật của sinh viên.

1.4 Đặc điểm dữ liệu

- Dataset chính:
 - + Gồm 11 cột và 295198 dòng.
 - + Dữ liệu đã được mã hóa các cột về thông tin cá nhân để bảo vệ quyền riêng tư của sinh viên. Ví dụ:
 - assignment_id: 90ce27571176d87961b565d5ef4b3de33ede04ac
 - problem_id: 789454427dd4097a14749e3dde63346b7a8d3811
 - hash: ed9eae6a707f50154024b24d7efcb874a9795dd
 - + Trong dữ liệu có sử dụng format JSON để lưu một số thông tin khác.
- Các dataset điểm thật:
 - + Gồm 2 cột và 761 dòng. Với cột thứ nhất là MSSV đã được mã hoá, cột thứ hai là điểm TH/QT/CK/TBTL của sinh viên.
 - + Phân bố điểm của sinh viên ở từng loại



Chương 2: Data processing

2.1 Import các thư viện và cài đặt

- Dự án sử dụng các thư viện:
 - + pandas, numpy: Thư viện dùng để xử lý và làm việc với dữ liệu.
 - + sklearn: Thư viện các công cụ và mô hình máy học, ở đây ta sử dụng mô hình RandomForestRegressor, công cụ phân chia dữ liệu train_test_split, công cụ tính kết quả r2_score.
 - + xgboost: Thư viện chứa mô hình XGBRegressor, áp dụng gradient boosting lên mô hình regression.
 - + json: Thư viện sử dụng để chuyển format JSON sang kiểu dữ liệu trong Python, trong bài toán này ta dùng cho cột thông tin judgement.

2.2 Tải dữ liệu và thử nghiệm

- Quá trình tải dữ liệu được thực hiện đồng thời cho cả 4 datasets.
- Sau khi load, dataset chính có cấu trúc với 11 cột và 295198 dòng. Các cột có thể chia ra thành các thể loại:
 - + Thông tin định danh: assignment_id, problem_id, hash (student ID).
 - + Thông tin trạng thái: is_final, status, pre_score, coefficient.
 - + Thông tin thời gian: created_at, updated_at.
 - + Thông tin kỹ thuật: language_id, judgement.

2.3 Làm sạch dữ liệu

2.3.1 Loại bỏ các kí tự đặc biệt

- Hàm remove_special_char được khai báo để loại bỏ kí tự đặc biệt xuất hiện trong các dataset nhằm tránh xảy ra các lỗi trong quá trình train và infer. Ngoài ra hàm cũng xử lý kiểu dữ liệu và dữ liệu rỗng.
 - + Loại bỏ ký tự non-breaking space ('\xa0').
 - + Loại bỏ khoảng trắng đầu cuối.
 - + Chuyển đổi kiểu dữ liệu điểm từ string sang float.

- + Thay đổi các record có giá trị cột điểm là rỗng thành điểm 0.
- Do dataset điểm thật Quá trình có sự không đồng nhất trong tên gọi của cột với các dataset khác nên ta thực hiện thêm việc đổi tên gọi cho cột 'diemqt' thành 'QT'.

2.3.2 Trích xuất thông tin từ cột judgement

- Hàm parse_judgement được khai báo để trích xuất thông tin từ cột judgement có format là JSON string. Ví dụ một mẫu của cột judgement:

```
{"times":[0,0,0,0,0,0,0,0,0,0,0],"mems":[0,0,0,0,0,0,0,0,0,0,0],"verdicts":{"WRONG":10}}
```

- Sau khi chuyển đổi, ta có được một dictionary với 3 cặp key-value:
 - + Key 'times' với value là một list các số nguyên biểu diễn thời gian chạy của từng test trong problem.
 - + Key 'mems' với value là một list các số nguyên biểu diễn bộ nhớ sử dụng của từng test trong problem.
- Từ cột judgement ta có thể trích xuất được thêm 4 đặc trưng về bài nộp của sinh viên:
 - + total_tests: Tổng số test cases = len(data.get('times', [])) = Độ dài của mảng times hoặc mems.
 - + total_wrong: Số test cases sai = data.get('verdicts', {}).get('WRONG', 0) = Giá trị của verdicts với key là WRONG.
 - + total_time: Tổng thời gian thực thi = sum(data.get('times', [])) = Tổng giá trị các phần tử trong mảng times.
 - + total_mem: Tổng memory sử dụng = sum(data.get('mems', [])) = Tổng giá trị các phần tử trong mảng mems.

2.4 Trích xuất đặc trưng (Feature engineering)

- Dataset gốc bao gồm 11 cột với các thông tin:

Tên cột	Kiểu dữ liệu	Ý nghĩa
assignment_id	string	Mã ID cho lần giao bài. Trong một assignment_id có thể có nhiều problem_id.
problem_id	string	Mã ID của từng bài con trong assignment.
hash	string	Mã số sinh viên đã được hash để đảm bảo an toàn thông tin cho sinh viên.
is_final	int/bool	Đánh dấu submission có phải là submission cuối cùng được chấm chính thức. Với 1 là submission cuối cùng và ngược lại.
status	string	Trạng thái của bài nộp. Gồm 4 giá trị là 'SCORE': đã chấm điểm, 'Compilation Error': Lỗi thực thi, 'Syntax Error': Lỗi cú pháp, 'pending': đang chấm điểm.
pre_score	int	Điểm cuối cùng sau khi hệ thống chấm và tổng hợp.
coefficient	int	Hệ số nhân của bài tập cho điểm tổng cuối cùng của lần giao bài.
created_at	string	Thời gian nộp của submission.
updated_at	string	Thời gian chấm xong của submission.
language_id	string	Ngôn ngữ được dùng để lập trình trong submission.
judgement	JSON string	Thông tin về từng test trong problem này.

- Vì đây là bài toán dự đoán điểm cho từng sinh viên, do đó, ta tổng hợp các thông tin thông qua từng mã số sinh viên (hash):
 - + 'num_assignments': Số lượng assignment đã làm.
 - + 'num_problems': Tổng số lượng problem đã làm.
 - + 'num_submissions': Tổng số lượng submission đã thực hiện.
 - + 'num_tests': Tổng số lượng test trong tất cả bài tập.
 - + 'wrong_sum': Tổng số lượng test sai trong tất cả bài tập.
 - + 'final_pre_score_mean': Điểm trung bình các bài nộp chính thức của sinh viên.
 - + 'accuracy': Tỷ lệ test chính xác trong tổng số lượng test.
 - + 'num_days': Số ngày khác nhau mà sinh viên đã thực hiện submit trên hệ thống.

Chương 3: Training

3.1 Lựa chọn model, cách thử nghiệm và đánh giá

3.1.1 Lựa chọn model

- Dự án sử dụng 2 thuật toán chính với các ưu điểm:
 - + **Random Forest**: Đơn giản, dễ hiểu, ít siêu tham số, chịu được nhiễu và ít overfitting.
 - + **XGBoost**: State-of-the-art gradient boosting, performance cao.

3.1.2 Cách thử nghiệm và đánh giá

- Tỷ lệ split Train và Test set là 70/30 với random_state = 42.
- Metric dùng để đánh giá: R^2 score. 20 ha oke

3.2 Huấn luyện model

3.2.1. Dự đoán điểm thực hành:

- Đối với điểm thực hành, nhóm chọn các đặc trưng sau để thực hiện huấn luyện model:

- + 'num_assignments'
- + 'num_submissions'
- + 'num_problems'
- + 'tests_sum'
- + 'final_pre_score_mean'
- + 'accuracy'

Model	Hyperparameters	Performance (R^2 score)
Random Forest	n_estimators=400, max_depth=7, min_samples_split=5, max_features=2	0.3631
XGBoost	n_estimators=300, learning_rate=0.01, max_depth=3, verbosity=1	0.3489

3.2.2. Dự đoán điểm quá trình:

- Đối với điểm quá trình, nhóm chọn các đặc trưng sau để thực hiện huấn luyện model:
 - + 'num_assignments'
 - + 'num_submissions'
 - + 'num_problems'
 - + 'tests_sum'
 - + 'num_days'
 - + 'final_pre_score_mean'

Model	Hyperparameters	Performance (R^2 score)
Random Forest	n_estimators=50, max_depth=5,	0.1206

	min_samples_split=5, max_features=2	
XGBoost	n_estimators=300, learning_rate=0.01, max_depth=4, verbosity=1	0.1292

3.2.3. Dự đoán điểm cuối kỳ:

- Đối với điểm cuối kỳ, nhóm chọn các đặc trưng sau để thực hiện huấn luyện model:

- + 'num_assignments'
- + 'num_submissions'
- + 'num_problems'
- + 'tests_sum'
- + 'wrong_sum'
- + 'num_days'
- + 'final_pre_score_mean'
- + 'accuracy'

Model	Hyperparameters	Performance (R^2 score)
Random Forest	n_estimators=400, max_depth=10, min_samples_split=5, max_features=2,	0.3513
XGBoost	n_estimators=300, learning_rate=0.05, max_depth=4, verbosity=1	0.3312

3.2.4. Dự đoán điểm trung bình tích lũy:

- Đối với điểm trung bình tích lũy, nhóm chọn các đặc trưng sau để thực hiện huấn luyện model:

- + 'num_assignments'
- + 'num_submissions'
- + 'num_problems'
- + 'tests_sum'
- + 'wrong_sum'
- + 'num_days'
- + 'final_pre_score_mean'
- + 'accuracy'

Model	Hyperparameters	Performance (R^2 score)
Random Forest	n_estimators=400, max_depth=20, min_samples_split=5, max_features=4, min_impurity_decrease=0.0001	0.3188
XGBoost	n_estimators=500, learning_rate=0.01, max_depth=5, verbosity=1	0.3067

3.3. Tổng quát:

Model	TH	QT	CK	TBTL
Random Forest	0.3631	0.1206	0.3513	0.3188
XGBoost	0.3489	0.1292	0.3312	0.3067

- **Nhận xét:** Random Forest có điểm số tốt hơn ở 3/4 dự đoán điểm (TH, CK, TBTL), còn XGBoost chỉ vượt trội hơn ở 1 dự đoán (QT). Ngoài ra, Random Forest cho kết quả ổn định hơn, nhưng sự chênh lệch không quá lớn so với XGBoost.

Chương 4: Evaluation

- Hệ thống tạo ra 8 file submission để nộp và chấm điểm trên wecode:
 - + submission_{TH/QT/CK/TBTL}_rfr.csv: kết quả dự đoán bằng RandomForest
 - + submission_{TH/QT/CK/TBTL}_xgboost.csv: kết quả dự đoán bằng XGBoost
- Thống kê điểm được chấm trên wecode:

Model	TH	QT	CK	TBTL
RandomForest	36	6	28	-137
XGBoost	28	12	20	-203

- Nhận xét: RandomForest vẫn đạt điểm cao hơn XGBoost ở các điểm TH, CK, TBTL, trong khi XGBoost chỉ nhỉnh hơn ở QT. Điểm âm ở TBTL cho thấy mô hình chưa dự đoán tốt với dạng điểm này, có thể do các đặc trưng nhóm tìm ra chưa phản ánh hết các yếu tố ảnh hưởng đến điểm trung bình tích lũy.

Chương 5: Kết luận

- Việc xây dựng hệ thống dự đoán điểm số cho sinh viên từ dữ liệu Wecode đã mang lại cho nhóm chúng em những trải nghiệm quý giá về machine learning trong lĩnh vực giáo dục. Khác với các bài toán machine learning truyền thống, việc làm việc với dữ liệu hành vi học tập đòi hỏi sự hiểu biết sâu sắc về cách sinh viên tương tác với hệ thống lập trình.

- Trong quá trình xử lý dữ liệu, nhóm chúng em đã nhận ra rằng việc xử lý JSON trong trường Judgement là bước then chốt để có thể trích xuất ra các metric quan trọng như: total_tests, total_wrong. Ngoài ra, việc làm sạch dữ liệu với các ký tự đặc biệt như “\xa0” đã dạy cho chúng em tầm quan trọng của data preprocessing trong môi trường thực tế. Những vấn đề nhỏ như vậy có thể ảnh hưởng lớn đến chất lượng model.
- Kết quả chấm điểm trên wecode không được như chúng em kỳ vọng, chỉ được từ 6 đến 36 điểm trên toàn bộ lần nộp, thậm chí đối với bài dự đoán điểm TBTL còn bị âm, R^2 score cũng chỉ đạt được từ 0.12 đến 0.36. Từ đó cho thấy vẫn còn nhiều “hidden pattern” mà nhóm chúng em vẫn chưa phát hiện ra. Điều này nhắc nhở rằng educational data mining là một lĩnh vực phức tạp, đòi hỏi sự kết hợp giữa kỹ thuật machine learning và hiểu biết sâu về tâm lý học tập.
- Qua quá trình thực hiện đồ án, nhóm chúng em đã tích lũy thêm nhiều kinh nghiệm thực tế trong việc xây dựng pipeline xử lý dữ liệu và huấn luyện mô hình học máy trên dữ liệu giáo dục. Đồ án không chỉ giúp nhóm rèn luyện kỹ năng làm việc nhóm, phân tích dữ liệu, mà còn giúp nhóm hiểu rõ hơn những khó khăn khi dự đoán điểm số của sinh viên do sự phức tạp trong hành vi học tập và các yếu tố ngoài dữ liệu. Dù kết quả mô hình vẫn còn nhiều điểm cần cải thiện, nhóm đánh giá đây là bước khởi đầu quan trọng, giúp nhóm tự tin hơn để có thể tiếp cận và triển khai các bài toán machine learning phức tạp hơn trong các dự án và nghiên cứu tương lai.