|  |  |
| --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI



BÁO CÁO BÀI TẬP LỚN HỌC PHẦN

Học Máy

Ứng dụng Học máy trong dự đoán

điểm thi kết thúc môn học của sinh viên

Nhóm thực hiện: Nhóm 3 - 64KTPM2

Thành viên: Nguyễn Quốc Huy

Nguyễn Thị Thanh Thủy

Đỗ Phương Quỳnh

Ngô Chúc Ngọc

Hà nội, năm 2024

LỜI CÁM ƠN

Chúng em xin trân trọng cảm ơn Tiến sĩ Tạ Quang Chiểu, người đã tận tình hướng dẫn và đóng góp ý kiến quý báu trong suốt quá trình thực hiện dự án này. Nhờ sự chỉ dẫn và hỗ trợ của thầy, nhóm đã có thể vượt qua những khó khăn, hoàn thiện dự án và nâng cao hiểu biết về lĩnh vực học máy.

Chúng em cũng xin gửi lời cảm ơn đến các thầy cô và bạn bè đã luôn đồng hành, hỗ trợ về kiến thức và tinh thần trong quá trình thực hiện dự án này.

LỜI CAM ĐOAN CỦA TÁC GIẢ

Chúng em xin cam đoan rằng báo cáo này là kết quả nghiên cứu và thực hiện của nhóm chúng em, dưới sự hướng dẫn của Tiến sĩ Tạ Quang Chiểu. Các số liệu, kết quả và phân tích trong báo cáo là trung thực và chưa từng được sử dụng trong bất kỳ công trình nào khác. Trong quá trình thực hiện, chúng em đã tuân thủ nghiêm ngặt các quy định về đạo đức nghiên cứu và sử dụng các nguồn tham khảo phù hợp, được liệt kê đầy đủ trong danh mục tài liệu tham khảo.

Chúng em xin hoàn toàn chịu trách nhiệm về tính chính xác và trung thực của nội dung báo cáo này.

MỤC LỤC

[DANH MỤC CÁC HÌNH ẢNH vi](#_Toc181120811)

[DANH MỤC BẢNG BIỂU vii](#_Toc181120812)

[DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ viii](#_Toc181120813)

[CHƯƠNG 1 GIỚI THIỆU 1](#_Toc181120814)

[1.1 Lý do chọn đề tài 1](#_Toc181120815)

[1.2 Vấn đề thực tiễn cần giải quyết 1](#_Toc181120816)

[1.3 Mục tiêu của dự án 1](#_Toc181120817)

[1.4 Mô tả bài toán 2](#_Toc181120818)

[1.5 Phương pháp tiếp cận 2](#_Toc181120819)

[1.5.1 Ridge Regression: 2](#_Toc181120820)

[1.5.2 CART Decision Tree: 2](#_Toc181120821)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 4](#_Toc181120822)

[2.1 Các thuật toán được sử dụng 4](#_Toc181120823)

[2.1.1 Ridge Regression 4](#_Toc181120824)

[2.1.2 Decision Tree (CART) 6](#_Toc181120825)

[2.2 Cơ chế chuẩn hóa và giảm thiểu quá khớp 7](#_Toc181120826)

[2.2.1 Ridge Regression 7](#_Toc181120827)

[2.2.2 Decision Tree CART 9](#_Toc181120828)

[2.3 Đánh giá mô hình 9](#_Toc181120829)

[2.3.1 Mean Squared Error (MSE) 9](#_Toc181120830)

[2.3.2 Độ chính xác (Accuracy) 10](#_Toc181120831)

[CHƯƠNG 3 DỮ LIỆU VÀ QUY TRÌNH THỰC HIỆN 11](#_Toc181120832)

[3.1 Dữ liệu 11](#_Toc181120833)

[3.1.1 Mô tả tập dữ liệu 11](#_Toc181120834)

[3.1.2 Tiền xử lý dữ liệu 12](#_Toc181120835)

[3.2 Cài đặt mô hình 16](#_Toc181120836)

[3.2.1 Cách thức triển khai các mô hình 16](#_Toc181120837)

[3.2.2 Gọi hàm để huấn luyện, dự đoán và đánh giá. 22](#_Toc181120838)

[CHƯƠNG 4 KẾT QUẢ VÀ ĐÁNH GIÁ 27](#_Toc181120839)

[4.1 Ridge regression: 27](#_Toc181120840)

[4.1.1 Kết quả 27](#_Toc181120841)

[4.1.2 Đánh giá hiệu quả mô hình: 30](#_Toc181120842)

[4.2 CART Decision Tree 31](#_Toc181120843)

[4.2.1 Kết quả 31](#_Toc181120844)

[4.2.2 Đánh giá hiệu quả mô hình: 34](#_Toc181120845)

[4.3 So sánh hiệu năng của hai mô hình Ridge Regression và CART Decision Tree 35](#_Toc181120846)

[TÀI LIỆU THAM KHẢO 36](#_Toc181120847)

[PHỤ LỤC 1: CẤU TRÚC THƯ MỤC VÀ MÔ TẢ 37](#_Toc181120848)

[PHỤ LỤC 2: HƯỚNG DẪN SỬ DỤNG THƯ MỤC ĐÍNH KÈM 39](#_Toc181120849)

DANH MỤC CÁC HÌNH ẢNH

[Hình 2.1 Công thức mục tiêu cho Ridge Regression 4](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119586)

[Hình 2.2 Công thức tính giá trị dự đoán 4](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119587)

[Hình 2.3 Hàm mất mát trong Ridge regression 7](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119588)

[Hình2.4.3 Hàm mất mát trong Ridge regression 7](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119589)

[Hình 2.5 Công thức tính độ chính xác 10](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119590)

[Hình 3.1 Dữ liệu ban đầu 11](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119591)

[Hình 3.2 Dữ liệu sau khi loại bỏ đặc trưng thừa 13](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119592)

[Hình 3.3 Dữ liệu sau khi quy đổi 13](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119593)

[Hình 3.4 Thông tin tổng quan dữ liệu sau khi chuyển đổi 14](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119594)

[Hình 4.1 Biểu đồ lỗi theo lamda bằng triển khai thủ công 27](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119595)

[Hình 4.2 Biểu đồ lỗi theo lamda bằng triển khai dùng thư viện sklearn 28](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119596)

[Hình 4.3 Biểu đồ so sánh trọng số w của 2 cách triển khai 28](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119597)

[Hình 4.4 So sánh giá trị lỗi trung bình khi triển khai theo 2 cách 29](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119598)

[Hình 4.5 Biểu đồ so sánh giá trị dự đoán so với thực tế trên tập test theo cách thủ công 29](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119599)

[Hình 4.6 Biểu đồ so sánh giá trị dự đoán của nhãn so với thực tế trên tập test sử dụng thư viện sklearn 30](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119600)

[Hình 4.7 Biểu đồ lỗi theo giá trị độ sâu của cây bằng cách thủ công 31](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119601)

[Hình 4.8 Biểu đồ lỗi theo giá trị độ sâu của cây bằng thư viện Sklearn 32](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119602)

[Hình 4.9 Biểu đồ so sánh giá trị lỗi trung bình theo 2 cách triển khai 32](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119603)

[Hình 4.10 Biểu đồ so sánh giá trị dự đoán so với giá trị điểm thực tế theo cách 33](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119604)

[Hình 4.11 Biểu đồ so sánh giá trị điểm dự đoán so với thực tế khi sử dụng thư viện 33](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119605)

[Hình 4.12 So sánh độ chính xác của 2 mô hình với 2 cách triển khai 35](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119606)

DANH MỤC BẢNG BIỂU

[Bảng 4.1 Bảng so sánh độ chính xác của Ridge Regression model theo 2 cách triển khai 30](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119607)

[Bảng 4.2 Bảng so sánh độ chính xác của Decision Tree - CART model theo 2 cách triển khai 34](file:///F:\School_slides\Third%20year%20-%20Junior\Semester%2005\Machine%20Learning\Báo%20cáo.docx#_Toc181119608)

DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ

# GIỚI THIỆU

## Lý do chọn đề tài

- Trong bối cảnh giáo dục hiện đại, khả năng dự đoán kết quả học tập của sinh viên là một chủ đề được quan tâm. Dự đoán điểm thi kết thúc môn học có thể giúp giảng viên và sinh viên xác định sớm các khó khăn trong học tập và đưa ra các biện pháp hỗ trợ kịp thời.

- Với sự phát triển của các phương pháp học máy, việc áp dụng các mô hình học máy để dự đoán điểm số của sinh viên trở nên khả thi và có thể đạt được độ chính xác cao hơn so với các phương pháp truyền thống.

- Đề tài này được chọn nhằm mục tiêu tìm hiểu, áp dụng các mô hình học máy vào bài toán dự đoán điểm thi kết thúc môn học của sinh viên, từ đó đánh giá độ hiệu quả và khả năng áp dụng trong thực tiễn.

## Vấn đề thực tiễn cần giải quyết

- Đề tài này nhắm đến giải quyết bài toán dự đoán dự đoán điểm thi kết thúc môn học của sinh viên, sử dụng dữ liệu từ các bài kiểm tra và bài tập trong suốt khóa học để đánh giá sự tiến bộ và khả năng đạt được mục tiêu học tập của mỗi sinh viên.

- Bài toán này sẽ góp phần hỗ trợ giảng viên trong việc đưa ra nhận định về năng lực của sinh viên và điều chỉnh phương pháp giảng dạy phù hợp hơn.

## Mục tiêu của dự án

**- Xây dựng mô hình học máy:** Tìm hiểu và áp dụng các thuật toán học máy, bao gồm Ridge Regression và CART Decision Tree, để giải quyết bài toán dự đoán điểm cuối kỳ của sinh viên.

**- Đánh giá hiệu quả của mô hình:** Thực hiện các bước tiền xử lý, huấn luyện, và đánh giá các mô hình trên dữ liệu thực tế để tìm ra mô hình có độ chính xác cao nhất.

## Mô tả bài toán

**Bài toán dự đoán điểm thi kết thúc môn học:** Với dữ liệu học tập của sinh viên trong suốt môn học, bao gồm điểm các bài kiểm tra và bài tập, bài toán đặt ra là dự đoán điểm thi kết thúc môn học (final score) dựa trên các đặc trưng đã được lựa chọn. Các đặc trưng đầu vào là điểm các bài kiểm tra giữa kỳ, bài tập lớn, và các bài kiểm tra nhỏ.

**- Các đặc trưng và mục tiêu dự đoán:**

**+ Đặc trưng (input features):** Điểm bài tập (cw1), điểm giữa kỳ (mid-term), điểm bài tập lớn thứ hai (cw2).

**+ Mục tiêu dự đoán (target):** Điểm thi kết thúc môn học (final).

Mô hình dự đoán sẽ được huấn luyện để dự đoán điểm thi cuối kỳ của sinh viên dựa trên các đặc trưng này.

## ****Phương pháp tiếp cận****

### ****Ridge Regression****:

**- Giới thiệu**: Ridge Regression là một phương pháp hồi quy tuyến tính với thêm một điều kiện ràng buộc để giảm hiện tượng quá khớp (overfitting). Phương pháp này thêm một tham số điều chỉnh lambda vào hàm mất mát để kiểm soát độ phức tạp của mô hình.

**- Lý do chọn**: Ridge Regression phù hợp với dữ liệu tuyến tính và là một mô hình đơn giản, dễ cài đặt, dễ phân tích hiệu quả dự đoán, phù hợp với nhóm khi mới lần đầu tiếp xúc với kiến thức học máy.

### ****CART Decision Tree****:

**- Giới thiệu**: CART (Classification and Regression Trees) là một thuật toán cây quyết định, dùng để phân loại hoặc dự đoán các giá trị liên tục. Mô hình này phân chia dữ liệu dựa trên các ngưỡng đặc trưng để tối ưu hóa dự đoán.

**- Lý do chọn**: CART trực quan và dễ giải thích, mô hình không đòi hỏi quá nhiều kiến thức về dữ liệu. CART cũng xử lý tốt các mối quan hệ phi tuyến giữa đặc trưng và mục tiêu, là một lựa chọn hiệu quả nếu dữ liệu không tuyến tính.

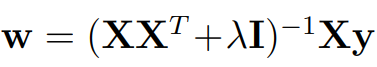
# CƠ SỞ LÝ THUYẾT

## Các thuật toán được sử dụng

### Ridge Regression

#### Lý thuyết

- Ridge Regression là một phiên bản mở rộng của hồi quy tuyến tính, sử dụng kỹ thuật chuẩn hóa để giảm thiểu hiện tượng quá khớp (overfitting) trong các mô hình có nhiều đặc trưng. Đặc điểm chính của Ridge Regression là thêm một tham số điều chỉnh (regularization parameter) vào hàm mất mát của hồi quy tuyến tính nhằm ràng buộc giá trị của các trọng số.

[2]

Hình 2.1 Công thức mục tiêu cho Ridge Regression

Trong đó:

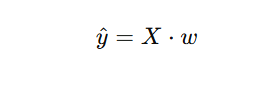
- X là ma trận đặc trưng của dữ liệu huấn luyện.

- y là vector các nhãn.

- w là trọng số cần tìm của mô hình.

- λ là tham số regularization, đóng vai trò quan trọng trong việc kiểm soát độ phức tạp của mô hình.

- Công thức tính giá trị dự đoán cho Ridge Regression:



Hình 2.2 Công thức tính giá trị dự đoán

Hình 2.2 Công thức tính giá trị dự đoán

Trong đó :

- X là ma trận đầu vào (các biến đầu vào đã thêm cột bias)

- w là vector trọng số tối ưu

**- Ảnh hưởng của λ:**

+ λ**=**0: Mô hình trở về dạng hồi quy tuyến tính thông thường, không có chuẩn hóa.

+ λ **lớn**: Mô hình càng ít phụ thuộc vào giá trị của từng đặc trưng riêng lẻ, giảm hiện tượng quá khớp nhưng có thể gây ra underfitting.

#### Xây dựng mô hình Ridge Regression

**- Chuẩn bị dữ liệu**:

+ Tách biến đầu vào và đầu ra từ tập dữ liệu, với các biến đầu vào (cw1, mid-term, cw2) và biến đầu ra (final).

+ Thêm cột bias (cột chứa giá trị 1) vào ma trận thuộc tính X để xử lý cho trường hợp intercept (giao điểm trục) của mô hình, đảm bảo rằng mô hình có thể tính toán được cả khi không có intercept trong dữ liệu.

**- Xây dựng hàm Ridge Regression**:

+ Xác định công thức Ridge Regression với thành phần regularization λ, tính trọng số w.

+ Sử dụng phép nhân ma trận để tính toán trọng số w và lưu lại các giá trị này để sử dụng cho dự đoán.

**- Huấn luyện và đánh giá trên từng fold (K-Fold Cross-Validation)**:

+ Chia tập dữ liệu thành các fold cho quá trình huấn luyện và kiểm tra.

+ Thực hiện Ridge Regression trên mỗi fold với các giá trị λ khác nhau.

+ Tính lỗi MSE trung bình trên các fold để chọn λ tối ưu có lỗi thấp nhất, giúp đạt được độ tổng quát tốt nhất.

**- Dự đoán và đánh giá trên tập kiểm thử**:

+ Sử dụng trọng số tối ưu www để dự đoán giá trị đầu ra trên tập kiểm thử.

+ Tính MSE trên tập kiểm thử để đánh giá hiệu quả của mô hình với dữ liệu chưa thấy trước đó.

### ***Decision Tree (CART)***

#### Lý thuyết

CART (Classification and Regression Trees) là một thuật toán cây quyết định có thể được sử dụng để giải quyết bài toán phân loại và hồi quy. Mô hình sử dụng phương pháp đệ quy để chia tách dữ liệu thành các nhóm nhỏ hơn dựa trên tiêu chí giảm thiểu lỗi. Đối với bài toán hồi quy, tiêu chí phổ biến để đo lường mức độ phân chia là Mean Squared Error (MSE), giúp xác định mức độ phù hợp của mô hình tại mỗi nhánh.

Quá trình xây dựng cây được điều khiển bởi các tham số như độ sâu tối đa của cây, số lượng điểm chia tối thiểu tại mỗi nút, nhằm ngăn chặn cây phát triển quá mức và tránh quá khớp.

#### Xây dựng mô hình CART

**- Chuẩn bị dữ liệu:**

+ Tách biến đầu vào và biến đầu ra từ tập dữ liệu, tương tự như Ridge Regression, với các biến đầu vào là (cw1, mid-term, cw2) và biến đầu ra là (final).

**- Xây dựng cây quyết định bằng thuật toán CART:**

+ Xác định các điểm chia dữ liệu bằng cách tính toán MSE tại mỗi nút và chọn điểm chia với MSE thấp nhất, giúp cây quyết định tại mỗi bước chia giảm thiểu được sai số dự đoán.

+ Tiến hành phân chia dữ liệu và xây dựng các nhánh đệ quy cho đến khi đạt độ sâu tối đa hoặc khi dữ liệu tại một nút không thể chia nhỏ hơn.

**- Tối ưu hóa độ sâu của cây và đánh giá qua K-Fold Cross-Validation:**

+ Tương tự như Ridge Regression, dữ liệu được chia thành các fold để thực hiện K-Fold Cross-Validation, thử nghiệm với nhiều giá trị độ sâu khác nhau.

+ Độ sâu của cây được chọn dựa trên kết quả MSE trung bình, giúp xác định cấu hình cây tốt nhất, giảm thiểu sai số mà vẫn tránh được quá khớp.

**- Dự đoán và đánh giá trên tập kiểm thử:**

+ Sử dụng cây với độ sâu tối ưu để dự đoán trên tập kiểm thử.

+ Tính MSE trên tập kiểm thử, giúp đánh giá khả năng tổng quát của cây với dữ liệu mới, chưa từng thấy trong quá trình huấn luyện.

## Cơ chế chuẩn hóa và giảm thiểu quá khớp

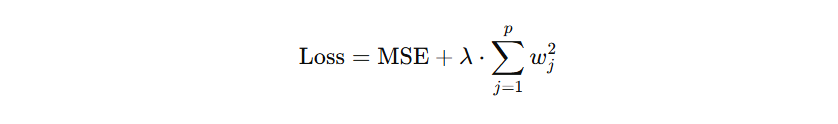
### Ridge Regression

#### Cơ chế chuẩn hóa trong Ridge Regression

- Hàm mất mát trong Ridge Regression

- Ridge Regression là một biến thể của hồi quy tuyến tính, trong đó một hệ số chuẩn hóa λ được thêm vào hàm mất mát (loss function) để kiểm soát độ phức tạp của mô hình.

- Hàm mất mát trong Ridge Regression có dạng:



Hình 2.3 Hàm mất mát trong Ridge regression

Hình2.4.3 Hàm mất mát trong Ridge regression

Trong đó:

- MSE (Mean Squared Error) là sai số bình quân giữa giá trị dự đoán và giá trị thực tế.

- λ là tham số điều chỉnh (regularization parameter), kiểm soát độ mạnh của thuật ngữ phạt.

- wj là các trọng số trong mô hình (coefficients).

#### Cơ chế hoạt động

- Giảm thiểu quá khớp: Bằng cách thêm thuật ngữ phạt vào hàm mất mát, Ridge Regression khuyến khích các trọng số w có giá trị nhỏ hơn, giúp làm giảm độ phức tạp của mô hình. Điều này làm cho mô hình ít nhạy cảm hơn với biến động trong dữ liệu huấn luyện, từ đó cải thiện khả năng tổng quát của mô hình trên dữ liệu mới.

- Chọn tham số tối ưu λ: Giá trị của λ quyết định mức độ phạt được áp dụng lên các trọng số. Nếu λ=0, mô hình trở thành hồi quy tuyến tính thông thường, trong khi nếu λ lớn, nó sẽ dẫn đến việc trọng số tiệm cận về 0. Quá trình tìm kiếm giá trị λ tối ưu thường được thực hiện thông qua phương pháp K-fold Cross-Validation, nơi mô hình được đào tạo và kiểm tra trên nhiều tập dữ liệu khác nhau để xác định giá trị λ nào cho hiệu suất tốt nhất.

#### *Tìm tham số tối ưu lambda*

- K-fold Cross-Validation: Để tìm giá trị lambda tối ưu, dự án đã được triển khai quy trình K-fold cross-validation, trong đó tập dữ liệu được chia thành K tập con và thực hiện huấn luyện, đánh giá trên mỗi tập. Giá trị lambda nào cho kết quả MSE (Mean Squared Error) trung bình thấp nhất sẽ được chọn là tham số tối ưu.

- Quy trình tối ưu hóa:

+ Chọn một loạt giá trị lambda để kiểm tra (0.01 đến 10).

+ Với mỗi giá trị lambda, huấn luyện mô hình bằng Ridge Regression và tính toán MSE trên tập kiểm thử qua từng fold.

+ Tính MSE trung bình của tất cả các fold cho mỗi lambda, và chọn giá trị lambda có MSE trung bình thấp nhất.

- Lựa chọn tối ưu: Khi chọn được lambda tối ưu, mô hình Ridge Regression sẽ có khả năng cân bằng giữa việc giảm thiểu lỗi huấn luyện và hạn chế quá khớp, đảm bảo mô hình không quá phức tạp nhưng vẫn chính xác trên tập dữ liệu kiểm thử.

### Decision Tree CART

- **Giới hạn độ sâu của cây:** CART thường có xu hướng chia nhỏ dữ liệu quá mức, làm cho cây trở nên quá phức tạp và dẫn đến quá khớp. Để tránh điều này, độ sâu tối đa của cây (maximum depth) có thể được giới hạn. Độ sâu thấp hơn giúp giảm khả năng mô hình hóa các nhiễu trong dữ liệu, từ đó tránh quá khớp.

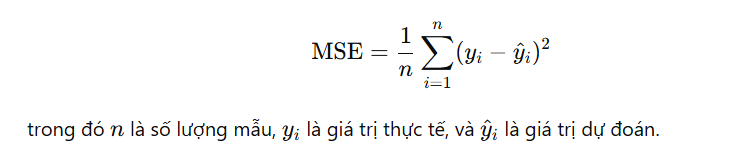
**- Sử dụng K-fold cross-validation để chọn tham số tối ưu:** CART được tối ưu hóa bằng cách chọn tham số độ sâu dựa trên kết quả của K-fold cross-validation. Điều này đảm bảo rằng mô hình sẽ ít bị phụ thuộc vào dữ liệu huấn luyện và có khả năng tổng quát hóa tốt hơn khi dự đoán trên dữ liệu mới.

## Đánh giá mô hình

### Mean Squared Error (MSE)

- Định nghĩa: MSE là trung bình bình phương sai lệch giữa giá trị yi^ dự đoán và giá trị yi thực tế.

- Công thức:

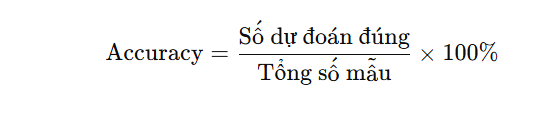


Hình 2.4 Công thức tính MSE

- Ý nghĩa: MSE đo lường mức độ sai lệch trung bình của dự đoán so với giá trị thực tế. MSE càng thấp, mô hình càng chính xác. Với bài toán dự đoán điểm thi, MSE thấp đồng nghĩa với việc mô hình có khả năng dự đoán điểm thi gần đúng với điểm thực tế của sinh viên.

### Độ chính xác (Accuracy)

- Đối với dự án này, độ chính xác được tính toán dựa trên tỷ lệ các dự đoán nằm trong một khoảng sai số chấp nhận được so với giá trị thực tế. Công thức tính độ chính xác như hình dưới:



Hình 2.5 Công thức tính độ chính xác

Trong đó:

- "Dự đoán đúng" được định nghĩa là các dự đoán nằm trong một ngưỡng sai số chấp nhận được, ví dụ như ±0.5 điểm so với giá trị thực tế.

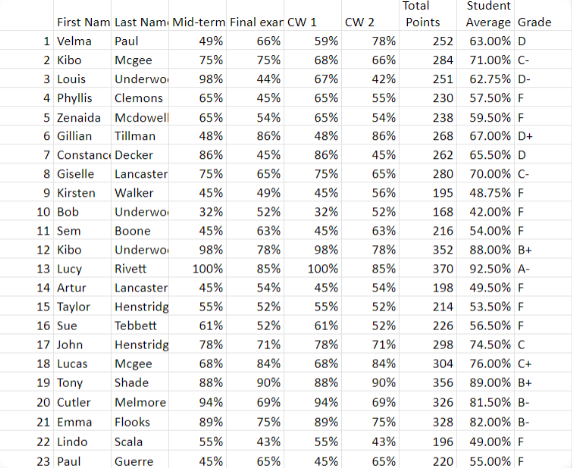
- Ý nghĩa: Độ chính xác cung cấp một cái nhìn tổng thể về khả năng của mô hình trong việc dự đoán chính xác các giá trị thực tế. Trong dự án này, việc tính toán độ chính xác với ngưỡng sai số ±0.5 điểm giúp đảm bảo rằng mô hình không chỉ cung cấp dự đoán gần đúng mà còn phản ánh đúng mức độ phân bố điểm thi thực tế.

# DỮ LIỆU VÀ QUY TRÌNH THỰC HIỆN

## Dữ liệu

### Mô tả tập dữ liệu

- Nguồn dữ liệu: Tập dữ liệu được sử dụng trong dự án này được thu thập từ Kaggle **(**[Nhấn vào đây xem dữ liệu](https://www.kaggle.com/datasets/alejandropaige/student-grades-record)**)** [1], với tổng cộng 1.029 bản ghi. Dữ liệu ban đầu chưa được chuẩn hóa và chứa thông tin về tên sinh viên, điểm số các môn học và đánh giá tổng thể về kết quả học tập.



Hình 3.1 Dữ liệu ban đầu

- Dữ liệu thô bao gồm các cột:

+ First Name: Họ của sinh viên

+ Last Name: Tên của sinh viên

+ Mid-term: Điểm thi giữa kỳ (thể hiện dưới dạng tỷ lệ phần trăm)

+ CW1: Điểm bài tập 1 (thể hiện dưới dạng tỷ lệ phần trăm)

+ CW2: Điểm bài tập 2 (thể hiện dưới dạng tỷ lệ phần trăm)

+ Final Exam: Điểm thi cuối kỳ (thể hiện dưới dạng tỷ lệ phần trăm)

+ Total Point: Tổng điểm của sinh viên (tính từ CW1, CW2, Mid-term, và Final Exam)

+ Student Average: Điểm trung bình toàn khóa của sinh viên

+ Grade: Xếp loại của sinh viên theo các mức (A, B, C,…)

**Nhận xét:**

- Mặc dù tập dữ liệu chứa nhiều thông tin chi tiết về kết quả học tập của sinh viên, nhưng không phải tất cả các thuộc tính đều liên quan đến bài toán dự đoán điểm thi cuối kỳ.

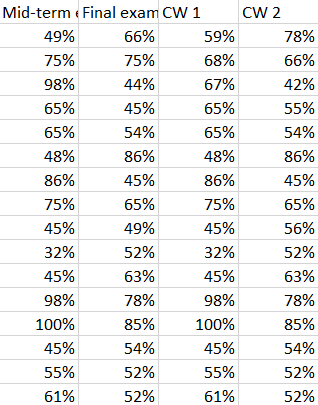
- Các đặc trưng cần thiết cho bài toán này chỉ bao gồm: CW1, CW2, Mid-term, Final Exam

- Các đặc trưng khác như tên sinh viên, điểm trung bình, và tổng điểm không ảnh hưởng trực tiếp đến khả năng dự đoán, vì vậy sẽ được loại bỏ trong quá trình tiền xử lý.

### ***Tiền xử lý dữ liệu***

#### Chuẩn hóa dữ liệu

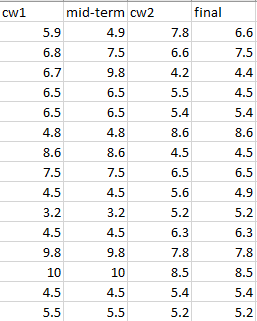
- Dữ liệu ban đầu bao gồm các cột như: first name, last name, mid-term, final exam, cw1, cw2, total point, student average, và grade. Tuy nhiên, chỉ các cột cw1, cw2, mid-term, và final exam là liên quan trực tiếp đến bài toán dự đoán điểm cuối kỳ. Do đó, các đặc trưng không liên quan đã bị loại bỏ.



Hình 3.2 Dữ liệu sau khi loại bỏ đặc trưng thừa

- Chuyển đổi điểm từ tỷ lệ phần trăm về thang điểm 10

Công thức chuyển đổi: Điểm (thang 10) = Điểm (%) / 10



Hình 3.3 Dữ liệu sau khi quy đổi

- Kiểm tra dữ liệu thiếu (Missing Values) bằng lệnh python:

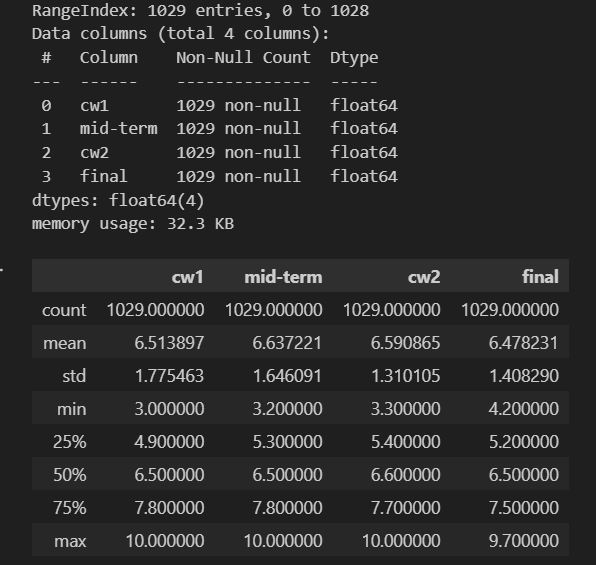
missing\_data = data.isnull().sum()

missing\_data[missing\_data > 0]

- Kiểm tra tổng quan dữ liệu

data.info()

data.describe()



Hình 3.4 Thông tin tổng quan dữ liệu sau khi chuyển đổi

- Nhận xét:

+ Tất cả các cột đều chứa dữ liệu số thực sau khi chuyển đổi.

+ Các giá trị đã được chuẩn hóa về thang điểm 10.

+ Tập dữ liệu cuối cùng bao gồm 1.029 bản ghi, mỗi bản ghi chứa thông tin về CW1, CW2, Mid-term, và Final Exam.

#### Chuẩn bị và Chia Dữ Liệu trong Dự Án

Bước 1: Đọc dữ liệu và chuẩn bị biến đầu vào (X) và đầu ra (y)

data = pd.read\_csv('data/processed/grade-records\_processed.csv')

X = data[['cw1', 'mid-term', 'cw2']]

y = data['final']

Bước 2: Chia dữ liệu thành tập huấn luyện và kiểm tra (train/test split)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Bước 3: Lưu thành tệp CSV

Bước 4: Thiết lập K-Fold Cross-Validation trên tập huấn luyện

train\_data = X\_train.copy()

train\_data['final'] = y\_train

train\_data.to\_csv('data/split/train\_data.csv', index=False)

test\_data = X\_test.copy()

test\_data['final'] = y\_test

test\_data.to\_csv('data/split/test\_data.csv', index=False)

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

Bước 5: Lặp qua từng fold và lưu vào file CSV

for fold, (train\_index, val\_index) in enumerate(kf.split(train\_data)):

    fold\_train\_data = train\_data.iloc[train\_index].reset\_index(drop=True)

    fold\_val\_data = train\_data.iloc[val\_index].reset\_index(drop=True

    fold\_train\_data.to\_csv(f'data/split/K-folds/fold\_{fold}\_train.csv', index=False)

    fold\_val\_data.to\_csv(f'data/split/K-folds/fold\_{fold}\_val.csv', index=False)

    print(f'Saved fold {fold} train and validation data to CSV files.')

## **Cài đặt mô hình**

### Cách thức triển khai các mô hình

#### Xây dựng hàm xử lí dữ liệu

**load\_fold\_data(fold\_number)** : Dùng để đọc và tách dữ liệu từ các file CSV cho từng fold trong quá trình K-Fold Cross-Validation.

def load\_fold\_data(fold\_number):

    train\_data = pd.read\_csv(f'data/split/K-folds/fold\_{fold\_number}\_train.csv')

    val\_data = pd.read\_csv(f'data/split/K-folds/fold\_{fold\_number}\_val.csv')

    X\_train = train\_data[['cw1', 'mid-term', 'cw2']].values

    y\_train = train\_data['final'].values

    X\_val = val\_data[['cw1', 'mid-term', 'cw2']].values

    y\_val = val\_data['final'].values

    return X\_train, y\_train, X\_val, y\_val

**load\_data\_from (path):** Dùng để đọc dữ liệu từ bất kỳ tệp CSV được cung cấp và tách các biến đầu vào (X) và biến đầu ra (y) cho quá trình huấn luyện và kiểm tra

def load\_data\_from(path):

    input = pd.read\_csv(f'{path}')

    data = input[['cw1', 'mid-term', 'cw2']].values

    label = input[['final']].values

    return data, label

#### Ridge Regression

- Cài đặt Thư viện: Các thư viện cần thiết như NumPy, Pandas, và Scikit-learn được cài đặt để hỗ trợ cho quá trình phân tích.

import numpy as np

import pandas as pd

import ridge\_utils as utils

- Tạo hàm **ridge\_regression(X, y, lamda)**: Viết hàm để tính toán trọng số cho Ridge Regression với giá trị lamda chỉ định.

def ridge\_regression(X, y, lamda):

    ones = np.ones((X.shape[0], 1))

    Xbar = np.concatenate((ones, X), axis=1)

    A = np.dot(Xbar.T, Xbar) + lamda \* np.identity(Xbar.shape[1])

    b = np.dot(Xbar.T, y)

    w = np.dot(np.linalg.pinv(A), b)

    return w

- Hàm **Huấn luyện**: Hàm huấn luyện được viết để thực hiện K-Fold Cross-Validation, giúp đánh giá mô hình trên từng fold và tính toán lỗi trung bình cho các giá trị λ khác nhau.

def train\_on\_folds(fold\_count, lamda\_values):

    best\_lamda = None

    lowest\_error = float('inf')

    avg\_errors = []  # Danh sách lưu trữ lỗi trung bình cho mỗi λ

    for lamda in lamda\_values:

        fold\_errors = []

        for i in range(fold\_count):

            # Đọc dữ liệu cho fold thứ i

            X\_train, y\_train, X\_val, y\_val = utils.load\_fold\_data(i)

            w = ridge\_regression(X\_train, y\_train, lamda)

            y\_pred = np.dot(np.concatenate((np.ones((X\_val.shape[0], 1)), X\_val), axis=1), w)

            # Tính lỗi (Mean Squared Error)

            error = np.mean((y\_val - y\_pred) \*\* 2)

            fold\_errors.append(error)

        # Tính lỗi trung bình qua các fold

        avg\_error = np.mean(fold\_errors)

        avg\_errors.append(avg\_error)  # Lưu lỗi trung bình vào danh sách

        if avg\_error < lowest\_error:

            lowest\_error = avg\_error

            best\_lamda = lamda

        print(f"Lambda = {lamda}, Average Fold Error = {avg\_error}")

    return best\_lamda, avg\_errors

#### Cart

**- Cài đặt Thư viện**:

import numpy as np

+ Hàm **Tính Độ Lệch (Error)** : đánh giá mức độ sai lệch của các dự đoán so với giá trị thực tế.

- **Xây dựng lớp DecisionTree**

def \_calculate\_split\_error(self, y):

        return np.mean((y - np.mean(y)) \*\* 2)

+ Hàm **Tìm Split Tốt Nhất**: Xem xét tất cả các thuộc tính và ngưỡng phân chia có thể, tính toán độ lệch cho mỗi split và chọn split với độ lệch thấp nhất.

def \_best\_split(self, X, y):

        best\_split = {}

        best\_error = float('inf')

        for feature in range(X.shape[1]):

            for threshold in np.unique(X[:, feature]):

                left\_indices = X[:, feature] <= threshold

                right\_indices = X[:, feature] > threshold

                if np.sum(left\_indices) == 0 or np.sum(right\_indices) == 0:

                    continue

+ Hàm **Xây Dựng Cây**: Xác định nút phân chia tại mỗi bước và tiếp tục xây dựng cây cho các nhánh trái và phải. Khi cây đạt đến độ sâu tối đa hoặc tất cả các nhãn đều giống nhau, nó trả về giá trị dự đoán (trung bình của các nhãn).

                left\_error = self.\_calculate\_split\_error(y[left\_indices])

                right\_error = self.\_calculate\_split\_error(y[right\_indices])

                split\_error = (np.sum(left\_indices) \* left\_error + np.sum(right\_indices) \* right\_error) / len(y)

            if split\_error < best\_error:

                    best\_error = split\_error

                    best\_split = {

                        'feature': feature,

                        'threshold': threshold,

                        'left\_indices': left\_indices,

                        'right\_indices': right\_indices

                    }

        return best\_split

def \_build\_tree(self, X, y, depth=0):

        if len(np.unique(y)) == 1 or depth == self.max\_depth:

            return np.mean(y)

        split = self.\_best\_split(X, y)

        if split == {}:

            return np.mean(y)

        left\_tree = self.\_build\_tree(X[split['left\_indices']], y[split['left\_indices']], depth + 1)

        right\_tree = self.\_build\_tree(X[split['right\_indices']], y[split['right\_indices']], depth + 1)

        return {'feature': split['feature'], 'threshold': split['threshold'], 'left': left\_tree, 'right': right\_tree}

+ Hàm **Huấn Luyện Mô Hình**: Khởi tạo quá trình xây dựng cây bằng cách gọi hàm build\_tree và lưu trữ cây đã được xây dựng trong mô hình.

def fit(self, X, y):

        self.tree = self.\_build\_tree(X, y)

### ***Gọi hàm để huấn luyện, dự đoán và đánh giá.***

#### Ridge Regression

Bước 1: Thiết lập giá trị λ cần kiểm tra

lamda\_values = np.logspace(-2, 1, num=100)

Bước 2: Huấn luyện mô hình và tìm λ tối ưu bằng K-Fold Cross-Validation

Mô hình được huấn luyện trên mỗi fold, và giá trị λ tốt nhất được chọn dựa trên lỗi trung bình (MSE) của 5 lần kiểm định.

best\_lamda, avg\_errors = model.train\_on\_folds(fold\_count=5, lamda\_values=lamda\_values)

print(f'Giá trị λ tối ưu (thủ công): {best\_lamda}')

Bước 3: Lưu trọng số của mô hình với λ tốt nhất

X\_train, y\_train = utils.load\_data\_from('data/split/train\_data.csv')

w = model.ridge\_regression(X\_train, y\_train, best\_lamda)

print('w = ', w)

np.save('ridge\_model\_weights.npy', w)

Bước 4: Dự đoán trên tập kiểm tra (test set)

Dữ liệu test chưa được nhìn thấy trong quá trình huấn luyện, giúp kiểm tra khả năng tổng quát hóa của mô hình.

X\_test, y\_test = utils.load\_data\_from('data/split/test\_data.csv')

weights = np.load('ridge\_model\_weights.npy')

y\_pred = model.predict(X\_test, weights)

Bước 5: Tính lỗi MSE và độ chính xác (accuracy)

Dự án sử dụng Mean Squared Error (MSE) làm thước đo lỗi và thiết lập ngưỡng sai số để tính độ chính xác.

mse\_test = np.mean((y\_test - y\_pred) \*\* 2)

print(f'Lỗi trung bình mse trên tập test (thủ công): {mse\_test}')

error\_threshold = 0.5

correct\_predictions = np.sum(np.abs(y\_test - y\_pred) < error\_threshold)

accuracy = (correct\_predictions / len(y\_test) ) \* 100

print(f'Accuracy (ngưỡng {error\_threshold}): {accuracy:.2f}%')

#### Cart

Bước 1: Thiết lập độ sâu và ngưỡng của cây cần kiểm tra Sử dụng độ sâu từ 1 đến 10 để tìm giá trị tốt nhất. Ngưỡng chênh lệch lỗi được sử dụng để dừng quá trình huấn luyện sớm nếu không có cải thiện.

depth\_values = range(1, 11)

threshold = 0.01

Bước 2: Huấn luyện mô hình và tìm độ sâu tối ưu :

- Mỗi lần huấn luyện, dữ liệu được chia làm 5 folds (K-Fold). Mỗi fold lần lượt đóng vai trò làm tập kiểm định, các folds còn lại dùng để huấn luyện.

- Dừng sớm được kích hoạt nếu lỗi giữa hai lần huấn luyện liên tiếp có sự chênh lệch nhỏ hơn threshold.

for depth in depth\_values:

    fold\_errors = []

    for fold in range(5):

        X\_train, y\_train, X\_val, y\_val = utils.load\_fold\_data(fold)

        tree = model.DecisionTree(max\_depth=depth)

        tree.fit(X\_train, y\_train)

        y\_pred = tree.predict(X\_val)

        mse = np.mean((y\_val - y\_pred) \*\* 2)

        fold\_errors.append(mse)

    avg\_error = np.mean(fold\_errors)

    errors.append(avg\_error)

    # So sánh lỗi trung bình với ngưỡng dừng sớm

    if abs(previous\_error - avg\_error) < threshold:

        print(f"Sự khác biệt lỗi giữa độ sâu {depth-1} và {depth} nhỏ hơn ngưỡng {threshold}. Dừng sớm.")

        break

    # Cập nhật giá trị lỗi và độ sâu tốt nhất

    if avg\_error < lowest\_error:

        lowest\_error = avg\_error

        best\_depth = depth

    previous\_error = avg\_error  # Cập nhật lỗi của độ sâu trước đó

    print(f"Depth = {depth}, Average Fold Error = {avg\_error}")

Bước 3 : Lưu trữ mô hình với độ sâu tốt nhất

print(f'Độ sâu tối ưu (thủ công): {best\_depth}')

Bước 4: Dự đoán trên tập test

Dữ liệu test được sử dụng để kiểm tra khả năng tổng quát hóa của mô hình.

X\_train, y\_train = utils.load\_data\_from('data/split/train\_data.csv')

X\_test, y\_test = utils.load\_data\_from('data/split/test\_data.csv')

tree = model.DecisionTree(max\_depth=best\_depth)

tree.fit(X\_train, y\_train)

y\_pred = tree.predict(X\_test)

Bước 5 : Tính lỗi MSE và độ chính xác (accuracy)

Accuracy được tính dựa trên số lượng dự đoán có sai số nhỏ hơn ngưỡng cho trước.

mse\_test = np.mean((y\_test - y\_pred) \*\* 2)

print(f'Lỗi trung bình trên tập test (thủ công): {mse\_test}')

error\_threshold = 0.5  # Ngưỡng sai số chấp nhận được

correct\_predictions = np.sum(np.abs(y\_test - y\_pred) < error\_threshold)

accuracy = (correct\_predictions / len(y\_test) )

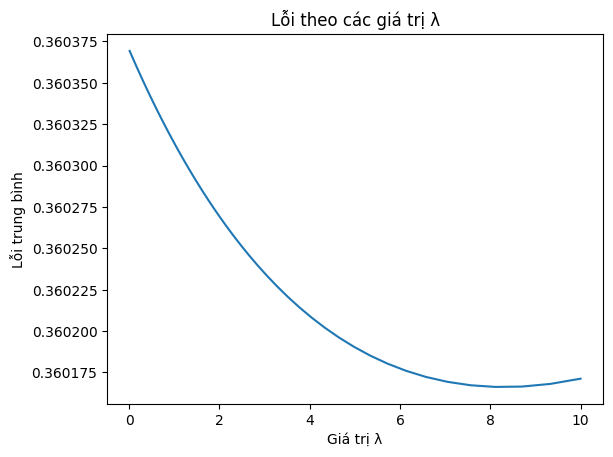
print(f'Accuracy (ngưỡng {error\_threshold}): {accuracy:.2f}%')

# KẾT QUẢ VÀ ĐÁNH GIÁ

## Ridge regression:

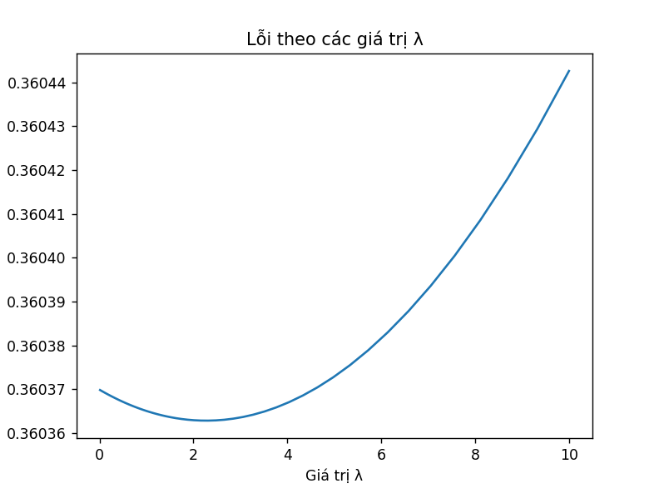
### Kết quả

- Biểu đồ lỗi theo các giá trị lamda trên tập test giúp xác định giá trị lambda tối ưu với lỗi trung bình thấp nhất:



Hình 4.1 Biểu đồ lỗi theo lamda bằng triển khai thủ công

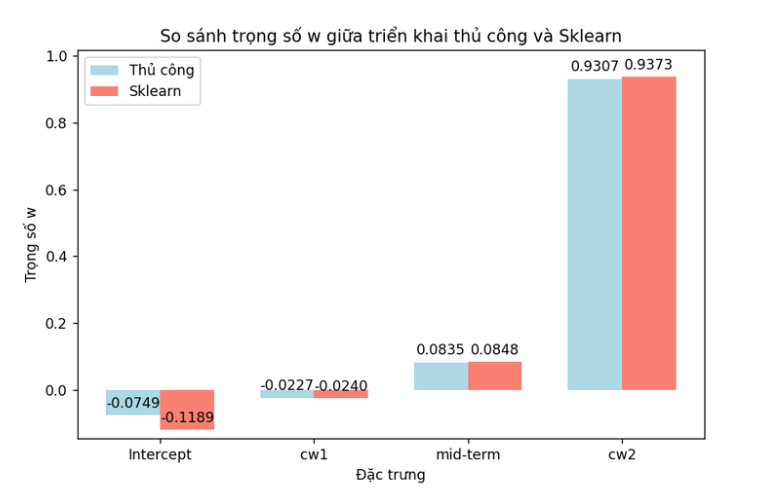
→ Giá trị λ tối ưu được chọn: 8.111308307896872



Hình 4.2 Biểu đồ lỗi theo lamda bằng triển khai dùng thư viện sklearn

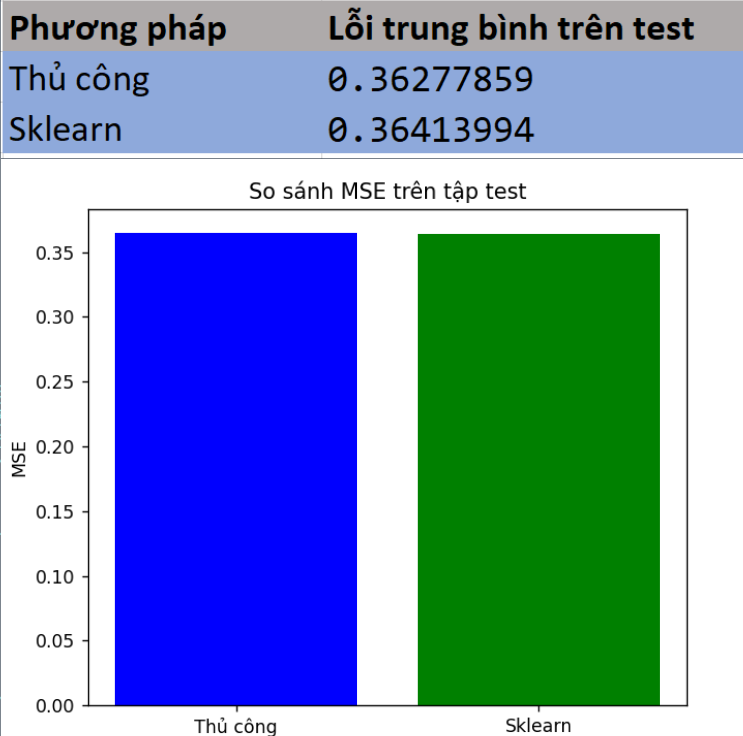
→ Giá trị λ tối ưu được chọn: 2.31012970008316

*- Biểu đồ so sánh trọng số w được lựa chọn của 2 cách triển khai:*



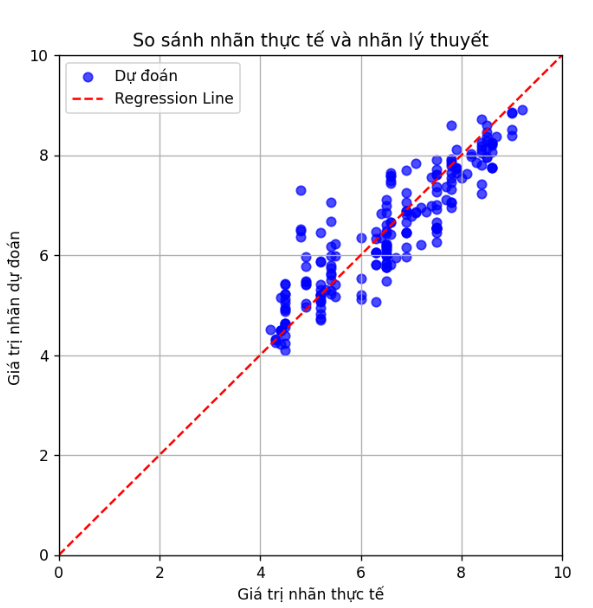
Hình 4.3 Biểu đồ so sánh trọng số w của 2 cách triển khai

*- Giá trị lỗi MSE của thuật toán thủ công và thư viện:*

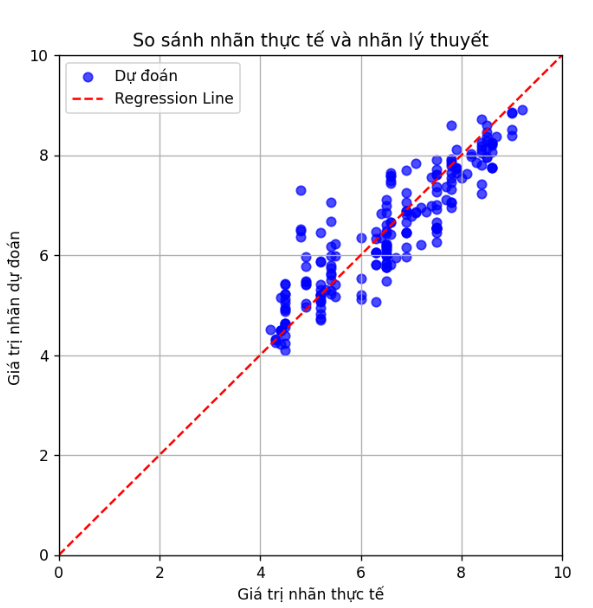


Hình 4.4 So sánh giá trị lỗi trung bình khi triển khai theo 2 cách

- Biểu đồ so sánh giá trị thực và giá trị dự đoán trên tập test**:**



Hình 4.5 Biểu đồ so sánh giá trị dự đoán so với thực tế trên tập test theo cách thủ công



Hình 4.6 Biểu đồ so sánh giá trị dự đoán của nhãn so với thực tế trên tập test sử dụng thư viện sklearn

- *Bảng so sánh Accuracy (độ chính xác):*

Bảng 4.1 Bảng so sánh độ chính xác của Ridge Regression model theo 2 cách triển khai

|  |  |  |
| --- | --- | --- |
| Accuracy (ngưỡng 0.5) | Thủ công | Thư viện |
| Ridge Regression | 64.08% | 64.56% |

### *Đánh giá hiệu quả mô hình*:

Cả 2 cách triển khai mô hình (thủ công và sử dụng thư viện sklearn) đều có hiệu suất tương đồng nhau.

- **Độ chính xác tổng thể:** Nhìn chung, các điểm dự đoán (chấm xanh) nằm khá sát đường gạch đứt đỏ (điểm thực tế), điều này cho thấy mô hình dự đoán tương đối chính xác cho phần lớn các trường hợp, thể hiện rằng mô hình đã học được mối quan hệ hợp lý giữa các biến đầu vào (cw1, mid-term, cw2) và nhãn (final) từ dữ liệu huấn luyện.

- **Sai số ở một số điểm ngoại lệ:** Ngoại lệ là điều không thể tránh khỏi khi dự đoán, do dữ liệu huấn luyện có thể chứa một số yếu tố nhiễu nhưng chúng không làm ảnh hưởng nhiều đến hiệu suất tổng thể của mô hình vì các điểm ngoại lệ này chỉ chiếm một phần rất nhỏ của toàn bộ tập dữ liệu.

- Kết quả giữa 2 phương pháp triển khai chỉ chênh lệch rất nhỏ (0.48%). Điều này cho thấy cả 2 cách triển khai đều hoạt động khá tốt và có độ nhất quán cao.

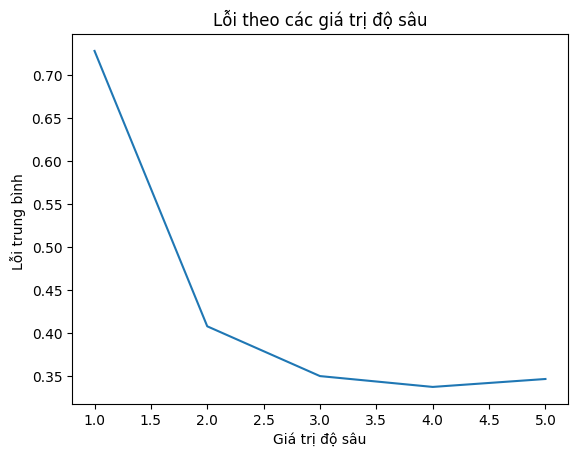
- **Hiệu năng:** Với độ chính xác hơn 64%, Ridge Regression hoạt động tương đối tốt cho bài toán này. Ridge Regression thường hiệu quả với các dữ liệu có tính tuyến tính và có nhiễu, và kết quả ở đây phù hợp với tính chất của phương pháp.

- **Độ tin cậy:** Độ tin cậy của mô hình khá tốt vì khi so sánh với thư viện uy tín như sklearn, chênh lệch rất nhỏ, chứng tỏ việc triển khai thủ công đã đúng với lý thuyết và không có lỗi lớn.

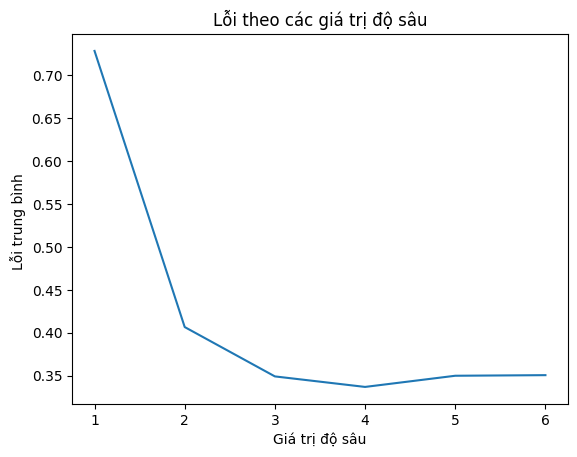
- **Tính ổn định:** Ridge Regression là một mô hình ổn định khi xử lý các tập dữ liệu có đa cộng tuyến (tính tương quan cao giữa các biến độc lập), và ở đây kết quả cũng ổn định với sai số thấp giữa hai phương pháp.

## CART Decision Tree

### Kết quả

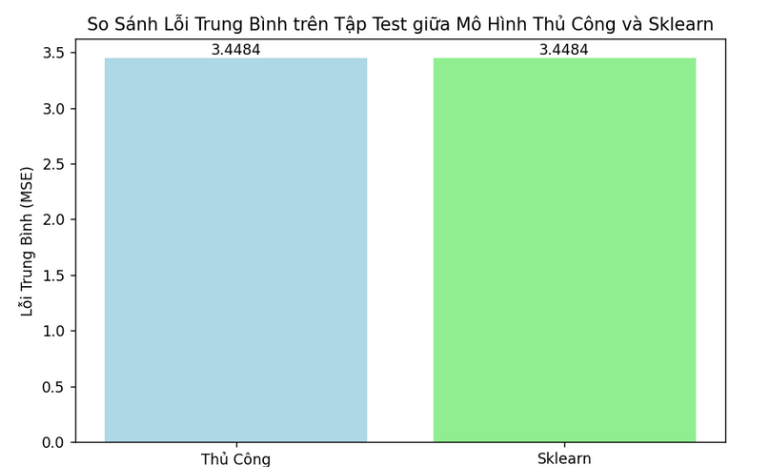


Hình 4.7 Biểu đồ lỗi theo giá trị độ sâu của cây bằng cách thủ công



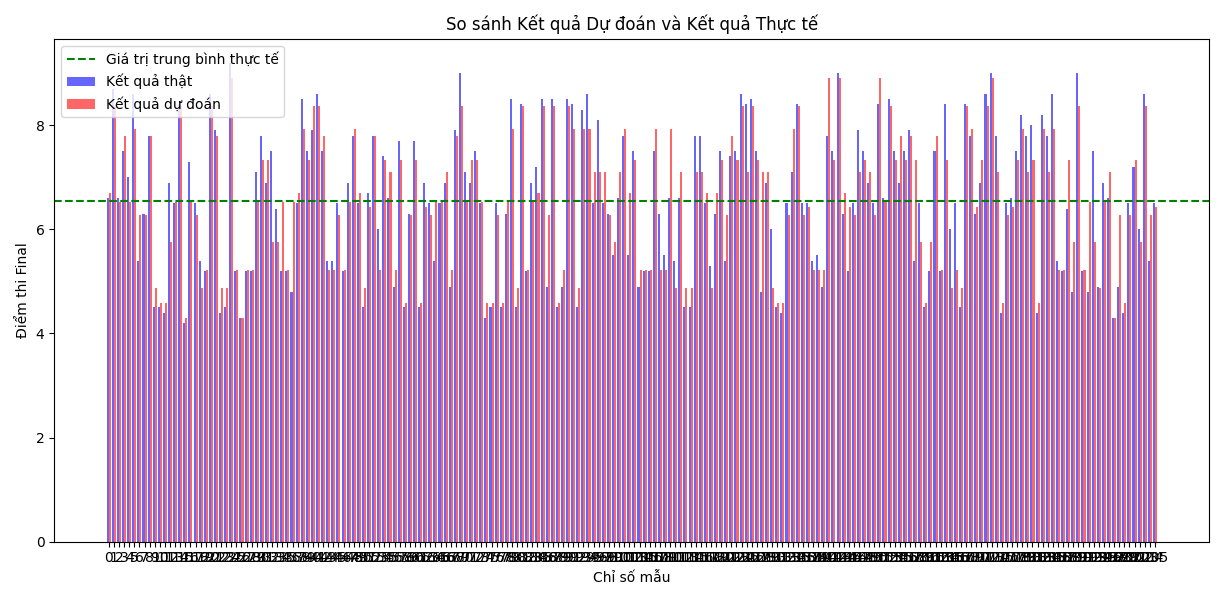
Hình 4.8 Biểu đồ lỗi theo giá trị độ sâu của cây bằng thư viện Sklearn

*- So sánh giá trị lỗi trung bình của 2 cách:*



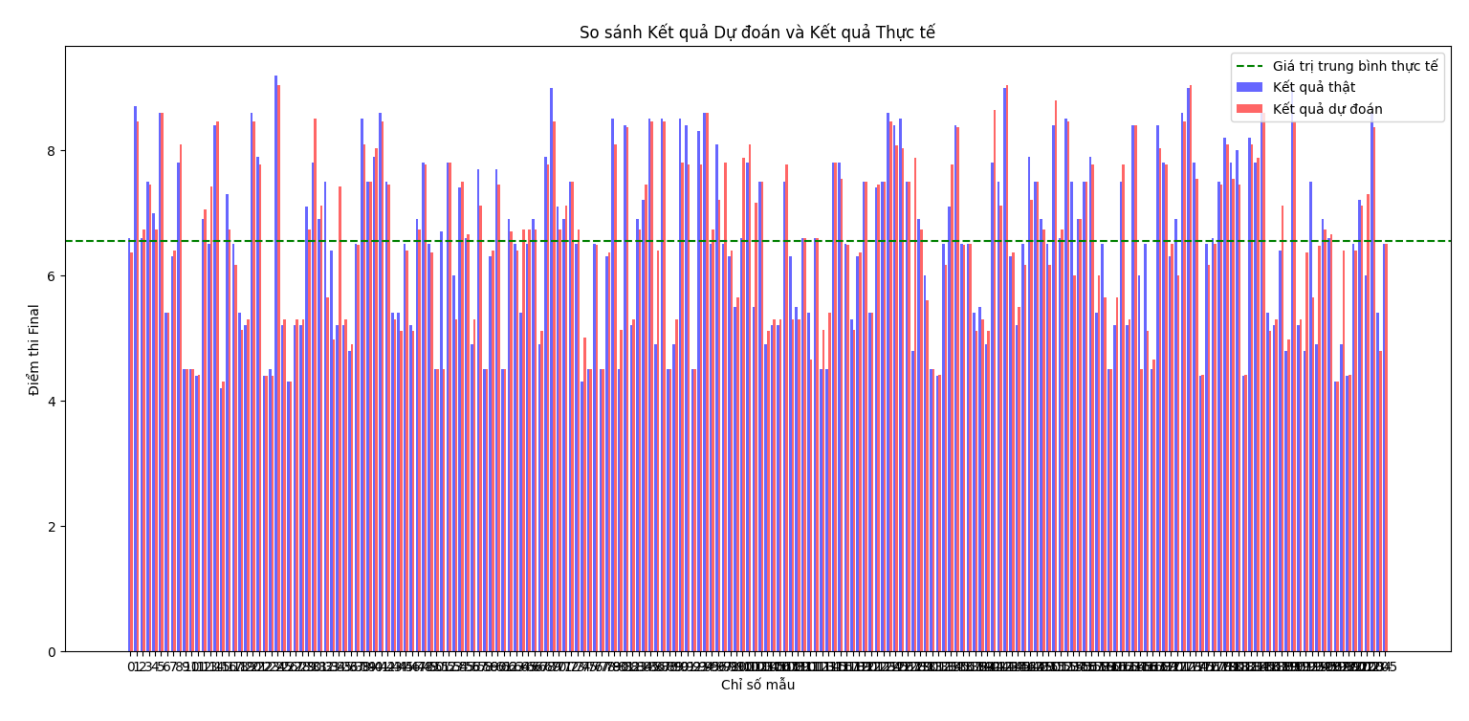
Hình 4.9 Biểu đồ so sánh giá trị lỗi trung bình theo 2 cách triển khai

*- So sánh kết quả dự đoán của mô hình:*



Hình 4.10 Biểu đồ so sánh giá trị dự đoán so với giá trị điểm thực tế theo cách

thủ công



Hình 4.11 Biểu đồ so sánh giá trị điểm dự đoán so với thực tế khi sử dụng thư viện

Bảng 4.2 Bảng so sánh độ chính xác của Decision Tree - CART model theo 2 cách triển khai

|  |  |  |
| --- | --- | --- |
| **Accuracy (ngưỡng 0.5)** | **Thủ công** | **Thư viện** |
| **Decision Tree - CART** | 42.60% | 42.05% |

### *Đánh giá hiệu quả mô hình*:

- Cả hai mô hình đều chọn độ sâu tối ưu là 4, cho thấy hiệu suất tương đương trong việc học từ dữ liệu.

- Lỗi trung bình trên tập test của cả hai mô hình là tương đương nhau, ở mức 3.7257. Điều này cho thấy cả hai mô hình đều có khả năng dự đoán tương đồng khi áp dụng trên tập test.

- Các lỗi trung bình theo từng độ sâu cũng tương tự nhau, cho thấy rằng cả 2 mô hình đều đang tối ưu hóa tốt theo độ sâu cây. Mô hình Scikit-learn cung cấp một số lỗi thấp hơn ở độ sâu 2 và 3, nhưng sự khác biệt không đáng kể.

- Mô hình tự xây dựng dừng lại ở độ sâu 4, trong khi mô hình Scikit-learn có thể kiểm tra độ sâu 5 và dừng ở độ sâu 6.

- Mức độ chính xác của hai phương pháp cũng khá tương đương (chênh lệch 0.55%), cho thấy việc triển khai thủ công rất gần với kết quả chuẩn từ sklearn.

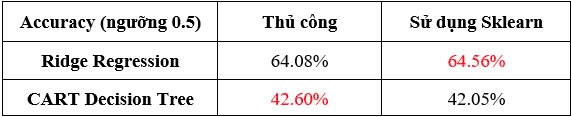
**- Hiệu năng:** Độ chính xác của mô hình CART thấp hơn rất nhiều so với Ridge Regression, chỉ ở mức khoảng 42%. Cho thấy Decision Tree không thực sự phù hợp (không tối ưu) cho bài toán và tập dữ liệu này.

**- Độ tin cậy:** Dù hiệu suất không cao, nhưng việc triển khai thủ công vẫn rất tin cậy, vì kết quả chỉ chênh lệch nhỏ so với sklearn.

**- Tính ổn định:** Decision Tree thường nhạy cảm với sự thay đổi nhỏ trong dữ liệu, có thể dẫn đến sự thiếu ổn định trong một số trường hợp. Ở đây, kết quả ổn định với sự sai lệch nhỏ giữa các phương pháp, nhưng hiệu suất tổng thể vẫn thấp.

→ **Kết luận:** Kết quả cho thấy rằng mô hình tự xây dựng và mô hình Scikit-learn đạt được hiệu suất tương đương, mặc dù mô hình Scikit-learn có lợi thế về việc xử lý các điều kiện dừng và kiểm tra thêm độ sâu cây.

## So sánh hiệu năng của hai mô hình Ridge Regression và CART Decision Tree



Hình 4.12 So sánh độ chính xác của 2 mô hình với 2 cách triển khai

Dựa vào Accuracy (độ chính xác) với ngưỡng được chọn là 0.5 cho cả 2 mô hình:

Ridge-Regression và Decision Tree - CART theo cách triển khai thủ công và sử dụng thư viện, ta rút ra được những nhận xét sau:

**- Hiệu năng:** Ridge Regression có hiệu năng tốt hơn đáng kể so với Decision Tree cho bài toán này, cho thấy dữ liệu tuân theo 1 mối quan hệ gần tuyến tính. Ridge Regression có khả năng chống overfitting tốt hơn đã có thể học được các quan hệ cần thiết giữa các biến đầu vào.

**- Tính phức tạp:** Decision Tree trở nên phức tạp khi các cây quyết định phát triển quá mức (overfitting), nguyên nhân của hiệu suất kém. Trong khi đó, Ridge Regression đơn giản hơn và khả năng điều chỉnh lambda giúp tránh overfitting hiệu quả hơn.

**KẾT LUẬN:** Ridge Regression là mô hình tốt hơn cho bài toán này, với hiệu suất cao hơn, độ tin cậy cao hơn và ổn định hơn trong dự đoán.

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | P. Alejandro, "Kaggle," 2018. [Online]. Available: https://www.kaggle.com/datasets/alejandropaige/student-grades-record. |
| [2] | V. H. Tiệp, Machine Learning cơ bản, 2018. |

PHỤ LỤC 1: CẤU TRÚC THƯ MỤC VÀ MÔ TẢ

**A. Thư mục dữ liệu (data)**

**1. raw**

**- grade-records\_raw.csv**: Tệp dữ liệu thô ban đầu chứa thông tin điểm số của sinh viên, được sử dụng làm nguồn đầu vào trước khi tiền xử lý.

**2. processed**

**- grade-records\_processed.csv**: Tệp dữ liệu đã qua xử lý, chứa các cột và thông tin đã chuẩn hóa, sẵn sàng cho việc huấn luyện và kiểm thử mô hình.

**3. split**

**- K-folds**: Chứa các tệp dữ liệu đã được chia thành 5 folds để thực hiện K-fold Cross Validation.

**+ fold\_0\_train.csv, fold\_0\_val.csv** đến **fold\_4\_train.csv, fold\_4\_val.csv**: Mỗi cặp tệp chứa tập huấn luyện và tập kiểm thử tương ứng cho từng fold.

**- train\_data.csv**: Tập huấn luyện dùng cho huấn luyện mô hình.

**- test\_data.csv**: Tập kiểm thử cuối cùng dùng để đánh giá mô hình sau khi đã chọn được các tham số tối ưu.

#### **B. Thư mục mã nguồn (src)**

**- decision-tree-CART**

**+ cart\_model.py**: Chứa thuật toán CART cho mô hình cây quyết định.

**+ cart\_utils.py**: Chứa các hàm hỗ trợ như đọc dữ liệu và xử lý đầu vào cho cây quyết định.

**+ cart\_plots.py**: Tạo biểu đồ để biểu diễn kết quả và hiệu suất của mô hình cây quyết định.

**+ cart\_sklearn.py**: Sử dụng thư viện sklearn để xây dựng và kiểm thử mô hình CART để so sánh với mô hình tự xây dựng.

**+ run-cart-model.py**: Chương trình chính để chạy mô hình CART, tìm độ sâu cây tối ưu, lưu mô hình và tính toán lỗi trên tập kiểm thử.

**- ridge-regression**

**+ ridge\_model.py**: Chứa thuật toán hồi quy Ridge được xây dựng thủ công.

**+ ridge\_utils.py**: Các hàm tiện ích để đọc và xử lý dữ liệu cho hồi quy Ridge.

**+ ridge\_plots.py**: Tạo các biểu đồ để biểu diễn lỗi theo các giá trị lambda.

**+ ridge\_sklearn.py**: Sử dụng thư viện sklearn để xây dựng và kiểm thử mô hình Ridge để so sánh với mô hình tự xây dựng.

**+ run-ridge-model.py**: Chương trình chính để chạy mô hình Ridge Regression, tìm giá trị lambda tối ưu, lưu trọng số mô hình, dự đoán và tính toán lỗi trên tập kiểm thử.

**- pre-processing**

**+ data-split.py**: Chứa mã để chia dữ liệu thành các fold phục vụ cho K-fold Cross Validation.

**+ data-visualize.py**: Chứa mã để tạo các biểu đồ cho quá trình khám phá và phân tích dữ liệu.

#### **C. Thư mục Kết quả và Biểu đồ (results/plots)**

**- decision-tree-CART, ridge-regression, eda**: Chứa các biểu đồ và hình ảnh được sinh ra trong quá trình huấn luyện, kiểm thử và khám phá dữ liệu.

**+ decision-tree-CART**: Biểu đồ hiệu suất của mô hình CART theo các độ sâu khác nhau của cây.

**+ ridge-regression**: Biểu đồ lỗi theo giá trị lambda trong mô hình Ridge Regression.

**+ eda (Exploratory Data Analysis)**: Các biểu đồ cho việc phân tích dữ liệu ban đầu.

#### **D. Thư mục Notebook (notebooks)**

**- 02\_Ridge\_Training.ipynb**: Notebook chứa các bước chi tiết để huấn luyện và kiểm thử mô hình Ridge Regression, bao gồm cả phân tích dữ liệu và tối ưu hóa tham số.

**- 03\_Decision\_Tree\_Cart.ipynb**: Notebook cho việc huấn luyện và kiểm thử mô hình CART, bao gồm phân tích dữ liệu và tối ưu hóa độ sâu cây.

**- exploratory\_analysis.ipynb**: Notebook chứa phân tích dữ liệu ban đầu, khám phá đặc trưng của tập dữ liệu và các biểu đồ để hiểu rõ về dữ liệu đầu vào.

**E. Thư mục Mô hình (models)**

Thư mục này chứa các mô hình đã được huấn luyện và lưu trữ dưới định dạng pickle, bao gồm:

- ridge\_model.pkl: Tệp chứa trọng số của mô hình hồi quy Ridge đã được huấn luyện.

- cart\_model\_best\_depth.pkl: Tệp chứa mô hình cây quyết định CART với độ sâu tối ưu đã được xác định.

**F. Tệp tin Dự đoán (predictApp.py)**

Đây là tệp tin giao diện người dùng được xây dựng bằng Tkinter, cho phép người dùng nhập điểm CW1, điểm giữa kỳ, và điểm CW2, sau đó dự đoán điểm cuối bằng cách sử dụng các mô hình đã được huấn luyện sẵn. Tệp này bao gồm:

- Nhập dữ liệu từ người dùng và xử lý dữ liệu đầu vào.

- Tải các mô hình đã được huấn luyện từ thư mục models dưới dạng tệp pickle.

- Cung cấp khả năng dự đoán dựa trên mô hình hồi quy Ridge hoặc cây quyết định CART đã được chọn.

PHỤ LỤC 2: HƯỚNG DẪN SỬ DỤNG THƯ MỤC ĐÍNH KÈM

**1. Chạy các tập mã nguồn:**

- Thực thi run-ridge-model.py để huấn luyện và đánh giá mô hình Ridge Regression.

- Thực thi run-cart-model.py để huấn luyện và đánh giá mô hình CART.

**2. Sử dụng notebook:**

- Mở các notebook trong thư mục notebooks để xem quá trình phân tích dữ liệu và kiểm thử mô hình chi tiết.

**3. Chạy ứng dụng dự đoán:**

- Mở terminal hoặc command prompt và điều hướng đến thư mục chứa tệp predictApp.py.

- Chạy lệnh python predictApp.py để khởi động giao diện người dùng.

- Nhập điểm CW1, điểm giữa kỳ, và điểm CW2 vào các ô nhập liệu.

- Chọn mô hình muốn sử dụng (Ridge hoặc CART) từ danh sách.

- Nhấn nút "Predict Final Score" để nhận dự đoán điểm cuối.

**4. Quản lý dữ liệu:**

- Trong thư mục data, có thể tìm thấy các tệp dữ liệu gốc và đã qua xử lý.

- Kiểm tra các tệp trong thư mục split để xem dữ liệu đã được chia thành các folds cho K-fold Cross Validation.

- Có thể thay thế hoặc cập nhật dữ liệu trong thư mục raw và thực hiện lại quá trình xử lý dữ liệu nếu cần.

**5. Phân tích kết quả:**

- Trong thư mục results/plots, tìm thấy các biểu đồ và hình ảnh được tạo ra trong quá trình huấn luyện và kiểm thử các mô hình.

- Mở các tệp hình ảnh để phân tích hiệu suất của mô hình và kết quả huấn luyện.

**6. Thay đổi cấu hình:**

- Có thể điều chỉnh các tham số trong tệp mã nguồn, chẳng hạn như giá trị lambda cho mô hình hồi quy Ridge hoặc độ sâu của cây cho mô hình CART, để xem ảnh hưởng của chúng đến hiệu suất mô hình.

- Mọi thay đổi cần được lưu lại và thực hiện lại các bước huấn luyện để áp dụng cấu hình mới.