ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

NHẬP MÔN HỌC MÁY VÀ KHAI PHÁ DỮ LIỆU

ĐỀ TÀI: XÂY DỰNG MỘT MÁY HỌC CÓ KHẢ NĂNG DỰ BÁO THỜI TIẾT

Giáo viên hướng dẫn: PGS. TS. Thân Quang Khoát

Mã lớp: 156421 - Nhóm: 21

Sinh viên thực hiện: Phạm Việt Hoàng - 20224854
Nguyễn Thị Trà My - 20225049
Đoàn Mạnh Hùng – 20224995
Phạm Thanh An - 20224911

Hà Nội – Tháng 12/2024

MỤC LỤC

LỜI NÓI ĐẦU

Ngày nay, sự phát triển bùng nổ của Trí tuệ nhân tạo (AI) mở ra một kỉ nguyên mới cho con người, nơi mà AI có thể hỗ trợ con người rất nhiều công việc hàng ngày như tự động hóa một số hoạt động trong nhà, xe tự lái, hệ thống tự nhận diện khuôn mặt, hệ thống chatbot … Ở mỗi lĩnh vực AI có vai trò khác nhau nhưng nhìn chung chúng đều mang lại lợi ích to lớn cho con người.

Một lĩnh vực cụ thể trong AI đấy là Học máy, ở đó con người huấn luyện những máy học, giúp chúng có khả năng đưa ra phán đoán, quyết định, cao cấp hơn là một số lĩnh vực như Thị giác máy tính và Xử lý ngôn ngữ tự nhiên. Trong học máy, bài toán Học có giám sát là phổ biến hơn cả, đặc biệt là bài toán phân loại có rất nhiều ý nghĩa thực tiễn, khi máy học có thể phân loại và hoàn thành 1 số nhiệm vụ mà con người đặt ra, với một độ chính xác thậm chí là rất cao.

Qua quá trình học tập, nghiên cứu và tiếp thu kiến thức từ môn học, chúng em thấy rằng bài toán phân loại trong Học máy là vô cùng rộng và có rất nhiều phương pháp để giải quyết, cũng như rất nhiều đề tài thú vị để lựa chọn. Để có thể hiểu rõ bản chất các mô hình Học máy thông qua bài toán hiện thực trực quan nhất, chúng em quyết định chọn đề tài "XÂY DỰNG MỘT MÁY HỌC CÓ KHẢ NĂNG DỰ BÁO THỜI TIẾT" làm nội dung báo cáo bài tập lớn môn học. Chúng em hy vọng rằng qua quá trình thực hiện dự án, nhóm sẽ tích lũy được thêm nhiều kỹ năng làm việc nhóm cũng như củng cố nền tảng kiến thức đã được học.

Trong môn học Nhập môn Học máy và khai phá dữ liệu, việc lựa chọn và triển khai một dự án cụ thể là phương pháp tiếp cận hiệu quả nhất để hiểu rõ các phương pháp và quy trình xây dựng một hệ thống học máy với đầy đủ các bước, từ Tiền xử lý dữ liệu, cho đến Huấn luyện model, và cuối cùng là Đánh giá kết quả. Vậy nên trong bài tập lớn này chúng em cũng đã làm đầy đủ các bước đã đề cập trên, với hi vọng tạo ra được một máy học phán đoán thật tốt. Đặc biệt, trong dự án này các phương pháp học máy được sử dụng, chúng em đều chia ra làm 2 phần là sử dụng Thư viện có sẵn để kiểm tra độ chính xác, và Tự code lại toàn bộ thuật toán, để hiểu sâu hơn.

Một vấn đề vui được đặt ra: "Giả sử hôm nay bạn ra ngoài đường nhận thấy trời âm u, bạn thu thập được một số dữ liệu thời tiết. Nhưng bạn thắc mắc liệu rằng ngày mai trời có mưa hay không? Bạn có cần phải mang ô theo bên người hay không?" Máy học của nhóm chúng em sẽ giải đáp thắc mắc của người dùng phía trên bằng cách đưa ra phán đoán lên đến 90% chính xác.

Dù đã nỗ lực hết mình để hoàn thiện bài tập lớn, nhưng do hạn chế về kinh nghiệm, chương trình của chúng em không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý quý báu từ thầy để nhóm có thể cải thiện và hoàn thiện sản phẩm của mình hơn nữa. Những đóng góp ấy không chỉ giúp chúng em nâng cao chất lượng dự án hiện tại mà còn là bài học kinh nghiệm quý giá cho các dự án trong tương lai.

Trân trọng!
Nhóm 21

CHƯƠNG 1: MỞ ĐẦU

1.1. Mục đích nghiên cứu

Nhóm chúng em lựa chọn nghiên cứu dự án này bởi vì bài toán phân loại nói riêng hay học có giám sát nói chung thuộc nhóm bài toán quan trọng và chiếm đa số trong Học máy. Chính vì vậy khi chọn chủ đề này cả nhóm có thể học được nhiều kiến thức hay và bổ ích mới trong lĩnh vực Học máy mà nằm ngoài nội dung môn học, cũng như ôn tập những kiến thức, những phương pháp đã học từ trước đến nay. Đây là cơ sở để chúng em có thể nghiên cứu những bài toán khó và phức tạp hơn trong tương lai.

Dự đoán thời tiết không phải là đề tài phổ biến, thậm chí với tài năng của nhân loại, chúng ta hiện nay đã có thể dự báo được thời tiết, phán đoán nắng mưa, giông bão trong phạm vi có thể lên đến từ 2 tuần đến 1 tháng đổ lại. Tuy nhiên để AI có thể dự đoán được thời tiết thì không phải là điều dễ dàng, cần phải huấn luyện cho máy học nhiều dữ liệu, sử dụng nhiều kĩ thuật khác nhau để cải tiến, giúp máy có thể đưa ra phán đoán tốt nhất, mục tiêu là độ chính xác khi đưa ra phán đoán từ 85-90% trở lên.

1.2. Giới thiệu tập dữ liệu

Trong dự án của nhóm, chúng em sử dụng tập dữ liệu Rain in Australia để xây dựng mô hình học máy có khả năng dự đoán khả năng có mưa vào ngày tiếp theo. Đây là một tập dữ liệu phong phú và đa dạng cung cấp thông tin chi tiết về thời tiết tại nhiều địa điểm khác nhau trên khắp nước Úc, bao gồm các thông tin khí tượng học được thu thập trong khoảng thời gian 10 năm từ 2007-2017. Đầu vào là các đặc trưng của thời tiết tại một địa điểm trong ngày, đầu ra là một nhãn nhị phân dự đoán rằng ngày sau đó liệu trời có mưa hay không. Tập dữ liệu được thiết kế để sử dụng trong các bài toán phân tích, trực quan hóa dữ liệu, và học máy.

Thông tin tổng quan về bộ dữ liệu:

Số lượng mẫu: 145.460.

Số lượng đặc trưng: 22

Nhãn cần phán đoán: 1/0 (Mưa/Không mưa)

Tập dữ liệu được có sẵn trên Kaggle: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package

Thông tin chi tiết về tập dữ liệu cùng các đặc trưng, thuộc tính sẽ được đề cập ở phần “KHAI PHÁ DỮ LIỆU VÀ TIỀN XỬ LÝ”

CHƯƠNG 2: KHAI PHÁ DỮ LIỆU VÀ TIỀN XỬ LÝ

2.1. Phân tích bài toán và tập dữ liệu

Bài toán dự báo thời tiết trong dự án mà nhóm lựa chọn thuộc lớp bài toán học có giám sát, cụ thể là lớp bài toán phân loại nhị phân. Đối với tập dữ liệu Rain in Australia, có tổng 145.460 mẫu dữ liệu, với đầu vào là 22 thuộc tính, máy học sẽ đưa ra đầu ra là 1 nhãn phán đoán là Yes (Có mưa) hoặc No (Không mưa). Dưới đây là chi tiết từng thuộc tính và nhãn phán đoán:

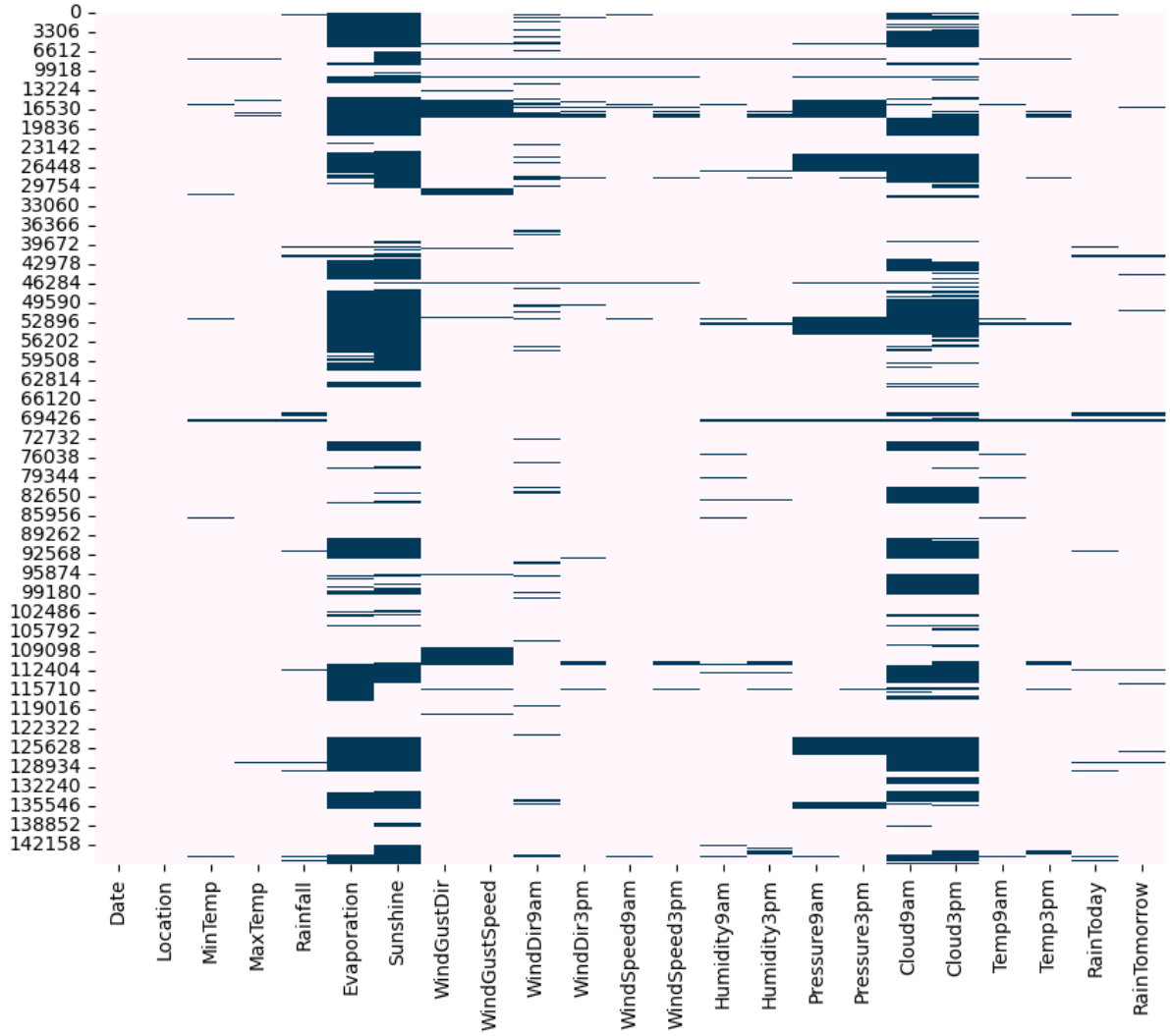
- Date: thời gian quan sát (từ 2007-07-01 đến 2017-06-25)
- Location: tên thành phố được quan sát
- MinTemp: Nhiệt độ thấp nhất đo được (độ C)
- MaxTemp: Nhiệt độ cao nhất đo được (độ C)
- Rainfall: Lượng mưa đo được trong ngày (mm)
- Evaporation: Lượng nước bốc hơi (mm)
- Sunshine: Số giờ nắng trong ngày
- WindGustDir: Hướng của cơn gió mạnh nhất
- WindGustSpeed: Tốc độ của cơn gió mạnh nhất (km/h)
- WindDir9am: Hướng gió lúc 9 giờ sáng
- WindDir3pm: Hướng gió lúc 3 giờ chiều
- WindSpeed9am: Tốc độ gió lúc 9 giờ sáng (km/h)
- WindSpeed3pm: Tốc độ gió lúc 3 giờ chiều (km/h)
- Humidity9am: Độ ẩm lúc 9 giờ sáng (%)
- Humidity3pm: Độ ẩm lúc 3 giờ chiều (%)
- Pressure9am: Áp suất khí quyển lúc 9 giờ sáng (hpa)
- Pressure3pm: Áp suất khí quyển lúc 3 giờ chiều (hpa)
- Cloud9am: Mức độ che phủ bầu trời bởi mây lúc 9 giờ sáng (oktas)
- Cloud3pm: Mức độ che phủ bầu trời bởi mây lúc 3 giờ chiều (oktas)
- Temp9am: Nhiệt độ lúc 9 giờ sáng (độ C)
- Temp3pm: Nhiệt độ lúc 3 giờ chiều (độ C)
- RainToday: Biến nhị phân (Yes/No) biểu thị rằng ngày hôm nay có mưa hay không (lượng mưa >1mm tức là có mưa)
- **RainTomorrow: Nhãn nhị phân (Yes/No) đầu ra của bài toán, thể hiện phán đoán ngày mai có mưa hay không**

2.1.1. Kiểm tra lượng dữ liệu trống

Bởi vì dữ liệu được đo trực tiếp trong thời gian dài trên 10 năm, nên lượng dữ liệu còn thiếu là tương đối, đặc biệt ở 1 số thuộc tính, lượng dữ liệu thiếu lên tới gần 1 nửa:

	missing_values	percent_missing %	data type
Date	0	0.000000	object
Location	0	0.000000	object
MinTemp	1485	1.020899	float64
MaxTemp	1261	0.866905	float64
Rainfall	3261	2.241853	float64
Evaporation	62790	43.166506	float64
Sunshine	69835	48.009762	float64
WindGustDir	10326	7.098859	object
WindGustSpeed	10263	7.055548	float64
WindDir9am	10566	7.263853	object
WindDir3pm	4228	2.906641	object
WindSpeed9am	1767	1.214767	float64
WindSpeed3pm	3062	2.105046	float64
Humidity9am	2654	1.824557	float64
Humidity3pm	4507	3.098446	float64
Pressure9am	15065	10.356799	float64
Pressure3pm	15028	10.331363	float64
Cloud9am	55888	38.421559	float64
Cloud3pm	59358	40.807095	float64
Temp9am	1767	1.214767	float64
Temp3pm	3609	2.481094	float64
RainToday	3261	2.241853	object
RainTomorrow	3267	2.245978	object

Bảng 1: Số lượng dữ liệu thiếu tại mỗi cột



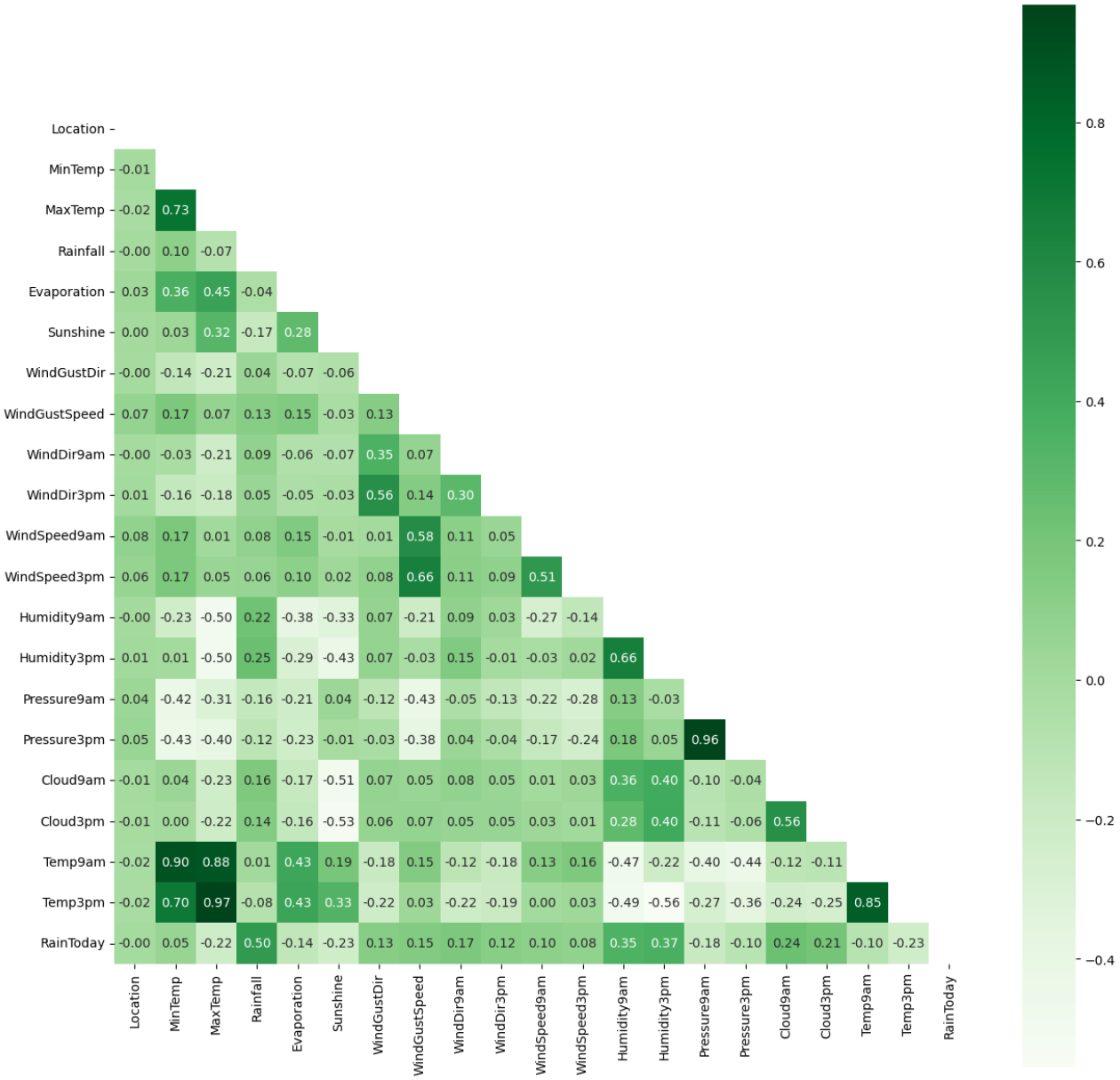
Hình 1: Biểu đồ thể độ mất mát dữ liệu của các cột (Phần màu xanh tức là dữ liệu thiếu)

Thông qua Bảng 1 và Hình 1, có thể thấy các cột có lượng dữ liệu thiếu nhiều nhất là cột Evaporation (43%), Sunshine(48%), Cloud9am(38%), Cloud3pm(90%), các cột này nói riêng và toàn bộ dữ liệu bị thiếu nói chung sẽ được xử lý ở phần Tiền xử lý.

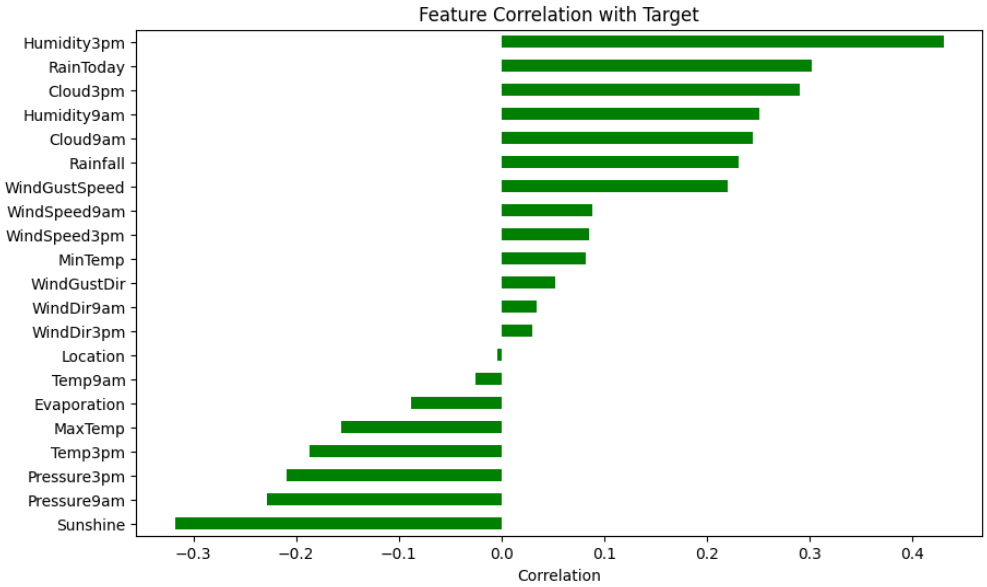
2.1.2. Độ tương quan giữa các đặc trưng

Độ tương quan thể hiện mối liên hệ giữa các cặp đặc trưng với nhau, hoặc giữa các đặc trưng với nhãn đầu ra. Nếu độ tương quan dương, 2 đặc trưng tỉ lệ thuận với nhau, tức là đặc trưng này tăng lên đồng nghĩa với đặc trưng kia tăng lên. Ngược lại, nếu độ tương quan âm, 2 đặc trưng tỉ lệ nghịch, nếu đặc trưng này tăng lên thì đặc trưng kia giảm đi. Nếu độ tương quan gần giá trị 0, 2 đặc trưng gần như không có mối liên hệ.

Độ tương quan có vai trò tương đối quan trọng, khi so sánh giữa các đặc trưng với nhau, các đặc trưng có giá trị độ tương quan cao sẽ thường được lược bớt đi vì không có ý nghĩa nhiều. Và đặc biệt khi so sánh độ tương quan với nhãn đầu ra, các đặc trưng có giá trị độ tương quan cao sẽ thể hiện được rằng thuộc tính đó có độ quan trọng lớn, có ý nghĩa lớn đối với mô hình, và cần phải được chú ý. Dưới đây lần lượt là ma trận hệ số tương quan và biểu đồ độ tương quan:

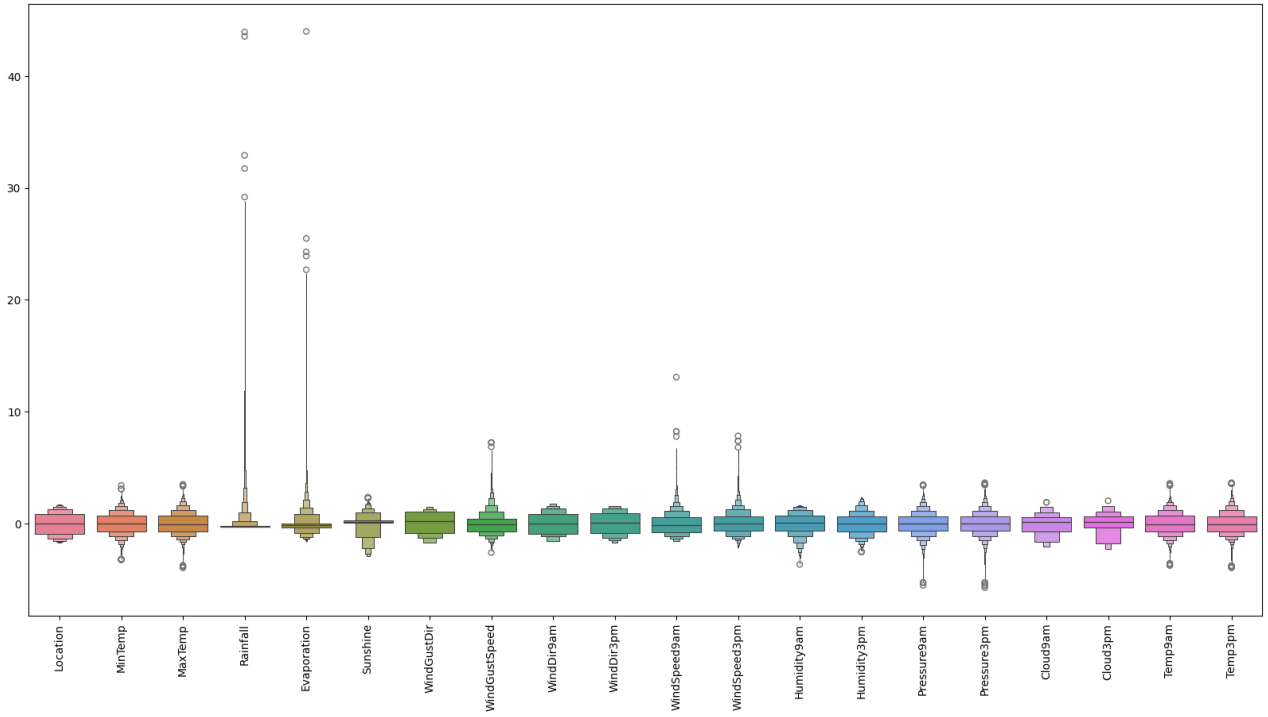


Hình 2: Ma trận hệ số tương quan



Hình 3: Biểu đồ độ tương quan giữa các cột đối với cột Target (Bỏ cột Date)

2.1.3. Dữ liệu nhiễu

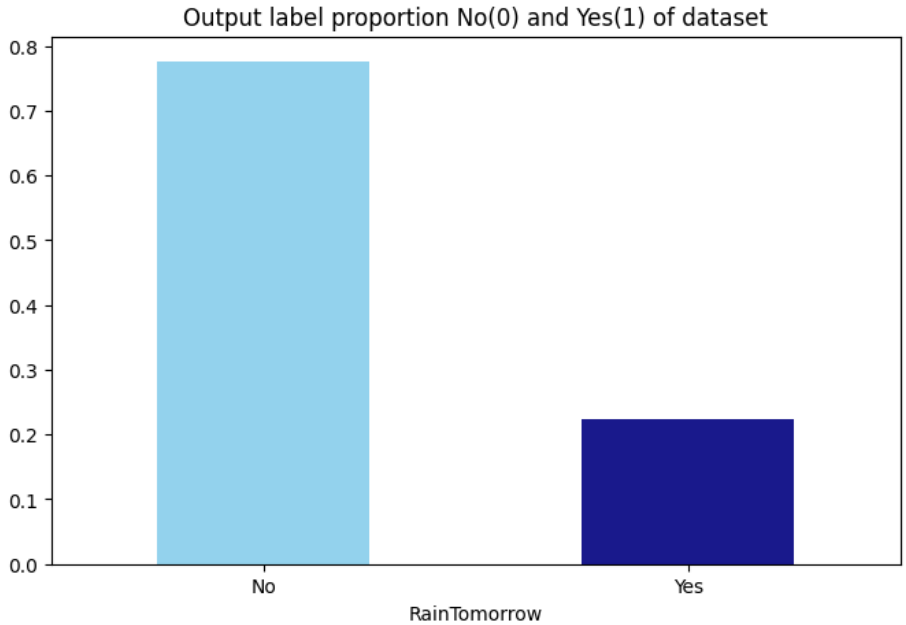


Hình 4: Biểu đồ thể hiện phân phối của dữ liệu (Sử dụng z-score để xác định)

Dữ liệu nhiễu là những mẫu có giá trị lệch đi so với phần lớn tập dữ liệu, có thể do nhiều lí do từ người thu thập, từ máy, hay từ tự nhiên, điều này ảnh hưởng lớn đến độ chính xác của mô hình. Tuy nhiên đây cũng có thể là những mẫu đem lại 1 xu hướng quan trọng của dữ liệu, vì vậy không thể bỏ chúng đi mà cần có cách giải quyết hợp lý nhất

Trong Hình 4, có thể thấy thông qua điểm *z-score* các cột Rainfall, Evaporation, WindSpeed9am, WindSpeed3pm có khá nhiều mẫu nhiễu. Xử lí các mẫu nhiễu này sẽ được thực hiện ở phần sau.

2.1.4. Phân phối nhãn đầu ra



Hình 5: Phân phối nhãn đầu ra

Từ Hình 5 có thể thấy, tập dữ liệu mắc phải 1 vấn đề là phân phối nhãn đầu ra không đồng đều, trong đó nhãn No (tức trời không mưa) chiếm tới gần 80%, trong khi nhãn Yes (tức trời mưa) chỉ chiếm trên 20%. Đây là 1 vấn đề nghiêm trọng sẽ ảnh hưởng đến tỉ lệ dự đoán đúng của từng nhãn, có thể thấy rằng tỉ lệ dự đoán đúng nhãn 0 sẽ chính xác hơn nhiều so với tỉ lệ dự đoán đúng nhãn 1. Vấn đề này sẽ được xử lý ở phần sau.

2.2. Các kỹ thuật tiền xử lý

2.2.1. Xử lý dữ liệu thiếu và mã hóa

Như đã đề cập ở phần trên, dữ liệu bị thiếu sẽ ảnh hưởng đến khả năng dự đoán của máy học, đồng thời trong phạm vi của môn học cùng với các kĩ thuật sử dụng trong dự án này, thì máy học chỉ có thể nhận đầu vào là dữ liệu dạng số. Vì vậy công việc cần làm đầu tiên đấy là xử lý những dữ liệu còn thiếu và dữ liệu dạng số.

Trước tiên, các thuộc tính của tập dữ liệu có thể chia ra thành:

+ Dạng số: `['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'RainTomorrow']`
+ Không phải dạng số: `['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm']`

- Đối với nhãn của tập dữ liệu và cột RainToday: chuẩn hóa từ Yes/No trở thành 1/0
- Đối với các cột dữ liệu không phải dạng số:

+ Cột Date: loại bỏ cột Date, thay vào đó dữ lại các tháng trong cột Date, vì các tháng có tính chu kì và nó ảnh hưởng đến mùa và lượng mưa (tháng hè thì mưa nhiều hơn), các giá trị tháng sẽ được mã hóa dưới tỉ lệ sin-cos (vì có tính chu kì), và sẽ tạo thành 2 cột mới (month-sin và month-cos)
+ Các cột còn lại: Điền các ô còn thiếu bằng giá trị phổ biến nhất sau đó mã hóa chúng bằng LabelEncoder

- Đối với dữ liệu dạng số: Điền các ô còn thiếu sử dụng kĩ thuật SimpleImputer hoặc IterativeImputer tùy chọn:

+ SimpleImputer: thay thế các giá trị bị thiếu bằng một thống kê đã tính toán từ dữ liệu (mean, median…), đơn giản và nhanh.
+ IterativeImputer: Sử dụng mô hình dự đoán (ví dụ Hồi quy tuyến tính) để dự đoán các giá trị bị thiếu, phức tạp và linh hoạt hơn, tuy nhiên rất chậm.

Kết thúc bước 1, các dữ liệu đã đều ở dạng số và không trống:

	missing_values	percent_missing %	data type
Location	0	0.0	int64
MinTemp	0	0.0	float64
MaxTemp	0	0.0	float64
Rainfall	0	0.0	float64
Evaporation	0	0.0	float64
Sunshine	0	0.0	float64
WindGustDir	0	0.0	int64
WindGustSpeed	0	0.0	float64
WindDir9am	0	0.0	int64
WindDir3pm	0	0.0	int64
WindSpeed9am	0	0.0	float64
WindSpeed3pm	0	0.0	float64
Humidity9am	0	0.0	float64
Humidity3pm	0	0.0	float64
Pressure9am	0	0.0	float64
Pressure3pm	0	0.0	float64
Cloud9am	0	0.0	float64
Cloud3pm	0	0.0	float64
Temp9am	0	0.0	float64
Temp3pm	0	0.0	float64
RainToday	0	0.0	float64
RainTomorrow	0	0.0	float64
month_sin	0	0.0	float64
month_cos	0	0.0	float64

Hình 6: Dữ liệu bị thiếu ở từng cột

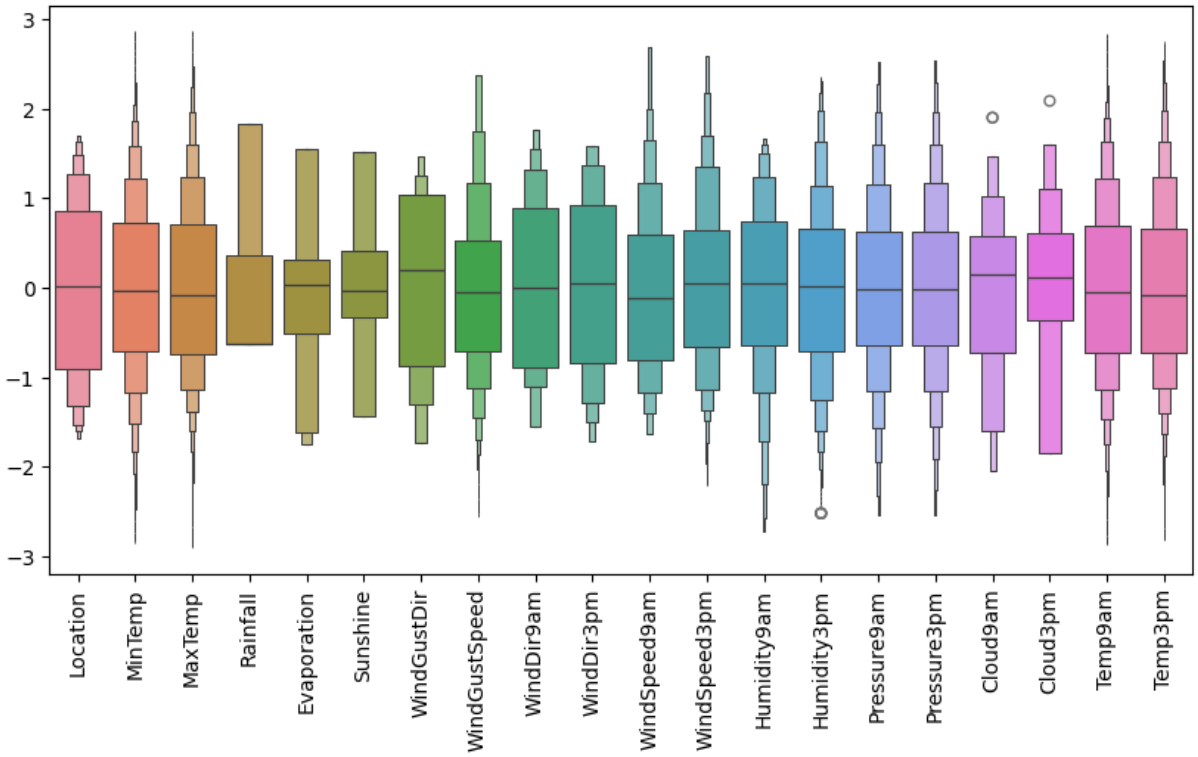
2.2.2. Xử lý nhiễu

Như đã đề cập ở trên, dữ liệu bị nhiễu có thể do nhiều lí do từ người thu thập, từ máy, hay từ tự nhiên, điều này ảnh hưởng lớn đến độ chính xác của mô hình. Tuy nhiên đây cũng có thể là những mẫu đem lại 1 xu hướng quan trọng của dữ liệu, vì vậy không thể bỏ chúng đi mà cần có cách giải quyết hợp lý nhất.

Ở đây nhóm đã sử dụng phương pháp Interquantile Range để giải quyết vấn đề này.

- Trước tiên, phương pháp này đi tính các giá trị gọi là khoảng không nhiễu, những giá trị nằm trong khoảng này sẽ không được coi là nhiễu, và không cần phải xử lý.
- Đối với các điểm nằm ngoài khoảng trên, ta thay thế chúng bằng các ngưỡng của khoảng không nhiễu, hoặc thay bằng các đại lượng trung bình, trung vị

Sau cùng, ta thu được dữ liệu với phân bố tính theo điểm z-score như hình dưới đây:



Hình 7: Dữ liệu sau khi xử lí nhiễu

Có thể thấy, kết quả ở hình 7 đã "đẹp" hơn nhiều so với trước khi xử lý (Hình 4)

2.2.3. Chia thành các tập Train/Test

Chia tập dữ liệu thành các tập train và test là một bước quan trọng và hầu như bắt buộc trong quá trình xây dựng mô hình học máy. Trong đó:

- **Train set**: Dữ liệu được sử dụng để **huấn luyện** mô hình. Mô hình học từ dữ liệu này để tối ưu hóa các tham số
- **Test set**: Dữ liệu được sử dụng để **đánh giá** hiệu quả của mô hình. Tập này giúp kiểm tra xem mô hình hoạt động tốt đến đâu trên dữ liệu mà nó chưa từng thấy trước đó.

Việc chia chúng thành các tập là cần thiết vì nó đảm bảo rằng mô hình có thể áp dụng được với dữ liệu mới ngoài tập huấn luyện, đồng thời giúp phát hiện tốt các mô hình bị overfitting (quá khớp, tức là hoạt động rất tốt trên tập train nhưng kém trên tập test)

Thông thường một bài toán lớn sẽ ưu tiên chia tập dữ liệu thành 3 tập bao gồm **Train set, Dev set, Test set** thay vì 2 tập, trong đó **Dev set** có vai trò giúp mô hình có thể tìm ra bộ siêu tham số tốt nhất, khách quan nhất trước khi mang mô hình cuối cùng đi dự đoán với **Test set**.

Tuy nhiên trong phạm vi của dự án lần này, nhóm nhận thấy rằng việc chia ra thêm tập **Dev set** hay không chia không ảnh hưởng nhiều đến kết quả của mô hình, bởi vì quy mô của bài toán này chưa phải lớn. Chính vì vậy ở phần Tiền xử lý, nhóm chỉ thực hiện chia tập dữ liệu ra làm 2 tập là **Train set** và **Test set** như mục đích ban đầu thôi. Đối với một số phương pháp cụ thể, nhóm sẽ chia ra thêm **Dev set** nếu cần thiết.

Cụ thể, nhóm đã chia ra 80% dữ liệu cho **Train set** và 20% dữ liệu cho **Test set**, đồng thời, sử dụng phương pháp lấy mẫu phân tầng, để đảm bảo rằng phân phối của nhãn đều được đảm bảo ở cả 2 tập, tránh trường hợp nhãn 1 ít dữ liệu dồn hết vào một tập

```
Original Class Distribution: RainTomorrow
0.0    90240
1.0    26128
Name: count, dtype: int64
Train set size: (116368, 23)
Test set size: (29092, 23)
```

Hình 8: Số lượng dữ liệu tại các tập cũng như nhãn của tập Train sau khi chia

2.2.4. Cân bằng lại phân phối nhãn

Dữ liệu từ bộ dữ liệu có sự mất cân bằng về nhãn đầu ra (cụ thể nhãn "No" chiếm gần 80%, nhãn "Yes" chỉ hơn 20%). Điều này có thể ảnh hưởng nhiều đến quá trình huấn luyện vì có thể mô hình huấn luyện ra có thể bị "bias". Do đó cần phải cân bằng lại dữ liệu.

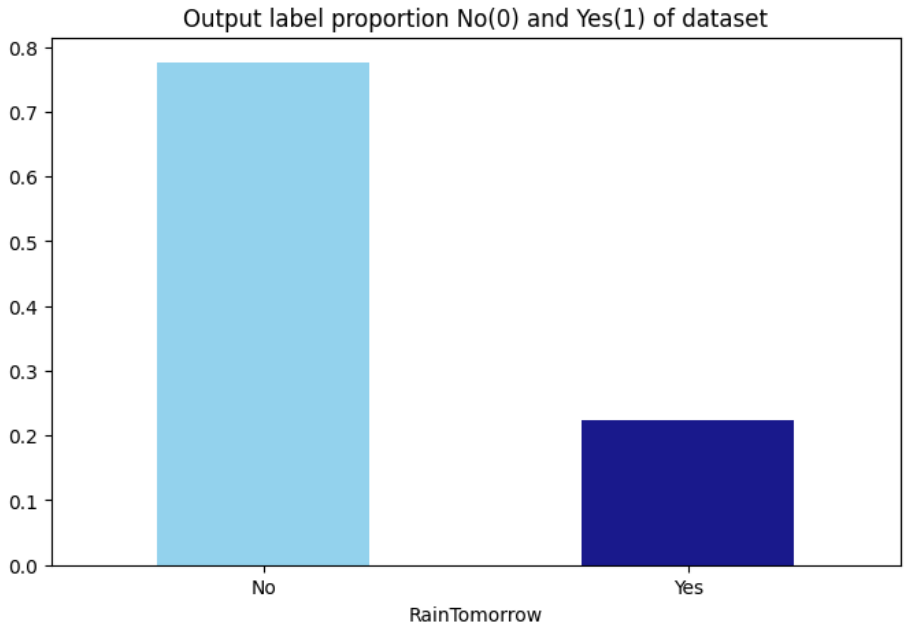
Ở đây, nhóm sử dụng 2 phương pháp SMOTE và Undersampling để giải quyết vấn đề mất cân bằng dữ liệu:

- Phương pháp SMOTE (Synthetic Minority Over-sampling Technique) là một phương pháp phổ biến xử lý vấn đề mất cân bằng dữ liệu. Ý tưởng chính của phương pháp SMOTE là gia tăng số lượng các dữ liệu còn thiếu bằng cách tạo ra các mẫu tổng hợp mới. Mẫu dữ liệu mới được tổng hợp bằng cách :
  + Chọn một mẫu dữ liệu từ lớp thiểu số (gọi là x_i).
  + Xác định k-láng giềng gần nhất của x_i trong lớp thiểu số.
  + Ngẫu nhiên chọn một láng giềng x_j từ k-láng giềng.
  + Tạo mẫu tổng hợp mới bằng cách nội suy giữa x_i và x_j . Mẫu mới là 1 tổ hợp lồi tuyến tính bất kì của x_i và x_j.

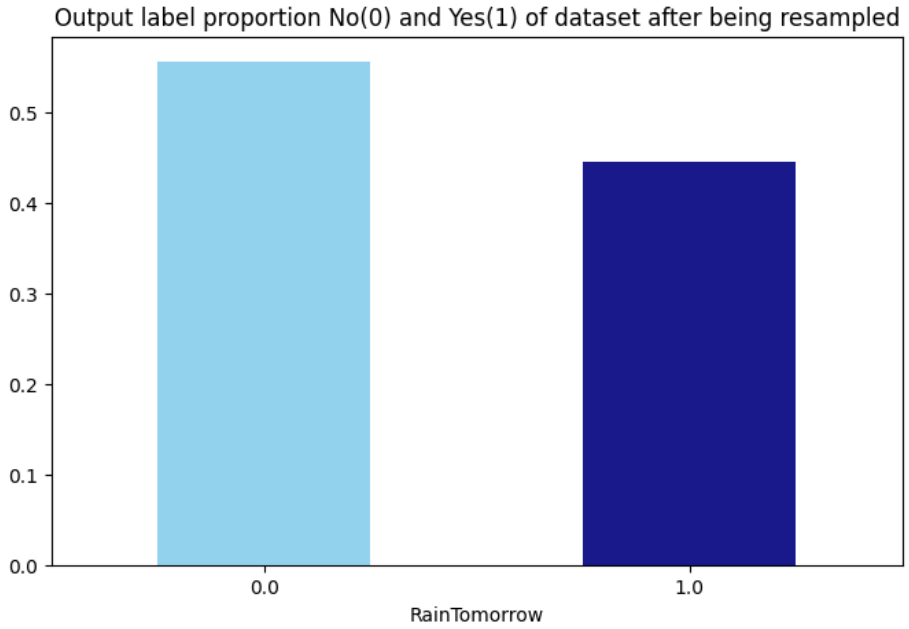
- Phương pháp Undersampling là một phương pháp xử lý mất cân bằng dữ liệu bằng cách giảm số lượng mẫu của lớp đa số để cân bằng tỷ lệ giữa các lớp. Ý tưởng chính của phương pháp Undersampling là giảm số lượng mẫu dữ liệu lớp đa số để 2 lớp có tỉ lệ tương đồng. Có một số cách bỏ đi các mẫu dữ liệu như Random Undersampling(lựa chọn ngẫu nhiên một tập con từ lớp đa số và loại bỏ các mẫu còn lại), Tomek Links,...

Nhóm kết hợp cả 2 phương pháp trên Đối với phương pháp SMOTE, nhóm tăng lớp thiểu số lên 50% của lớp đa số. Đối với phương pháp Undersampling, nhóm giảm lớp đa số xuống 80%, chiến lược giảm là Random Undersampling.

Sau khi áp dụng, các nhãn "Yes" và "No" trở nên cân bằng hơn(cụ thể nhãn "No" chỉ còn khoảng 50%, nhãn "Yes" gần 45%)



Hình 9: Phân phối nhãn trước khi thực hiện resampled



Hình 10: Phân phối nhãn sau khi thực hiện resampled

MỘT CHÚ Ý QUAN TRỌNG: Vì để so sánh hiệu quả 1 cách khách quan, tận dụng tối đa tài nguyên sẵn có, nhóm sẽ sử dụng **cả tập dữ liệu gốc ban đầu và cả tập dữ liệu resampled** đem đi đưa vào mô hình, để có thể so sánh một cách dễ dàng, thuận tiện và khách quan hơn.

2.2.5. Chuẩn hóa dữ liệu

Các cột dữ liệu với các thang đo khác nhau, và các giá trị của các thuộc tính cũng không có sự đồng đều (cụ thể, trước khi chuẩn hóa dữ liệu, cột "Pressure9am" và cột "Pressure3pm" các giá trị lớn hơn 1000, trong khi cột "Cloud9am", "Cloud3pm" chỉ dưới 10). Điều này ảnh hưởng không tốt trong quá trình huấn luyện mô hình vì việc khoảng dữ liệu quá lớn có thể gây ra hiện tượng mô hình nhạy cảm với các các điểm ngoại lai, điểm nhiễu(chỉ cần 1 thay đổi nhỏ trong tập dữ liệu cũng làm cho đầu ra có sự thay đổi lớn).

Do đó nhóm quyết định chọn phương pháp StandardScaler để giải quyết vấn đề này. Standard Scaler là một phương pháp chuẩn hóa dữ liệu (data normalization) được sử dụng để đảm

bảo các đặc trưng trong tập dữ liệu có cùng thang đo. Ý tưởng của Standard Scaler là chuyển đổi dữ liệu trong một cột sao cho giá trị trung bình (mean) của mỗi đặc trưng bằng 0, độ lệch chuẩn (standard deviation) của mỗi đặc trưng bằng 1.

Sau khi áp dụng Standard Scaler, nhóm đã thu được tập dữ liệu với các giá trị trong các cột thuộc tính sau khi chuẩn hóa trong khoảng từ -1 đến 1.

Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity3pm
39	9.3	23.7	0.0	4.8	9.45	1.0	39.0	8	1	...	44.0
46	7.7	21.4	0.0	4.8	8.40	7.0	48.0	7	7	...	59.0
30	18.5	39.4	0.0	4.8	8.40	7.0	59.0	6	14	...	16.0
27	15.5	18.4	0.0	4.8	7.45	15.0	50.0	13	15	...	75.0
1	8.3	20.4	0.0	3.6	9.40	13.0	39.0	15	13	...	67.0

Hình 11: Trực quan dữ liệu của mô hình trước khi chuẩn hóa

Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity3pm
1.067965	-0.455112	0.067269	-0.626561	0.033317	1.517114	-1.511997	-0.053827	0.221841	-1.498443	...	-0.369041
1.559973	-0.706545	-0.258476	-0.626561	0.033317	-0.035101	-0.233283	0.684994	0.000485	-0.176076	...	0.364688
0.435384	0.990630	2.290833	-0.626561	0.033317	-0.035101	-0.233283	1.587997	-0.220871	1.366685	...	-1.738668
0.224524	0.519193	-0.683361	-0.626561	0.033317	-1.439487	1.471669	0.849176	1.328623	1.587080	...	1.147332
-1.602934	-0.612257	-0.400104	-0.626561	-0.792558	1.443199	1.045431	-0.053827	1.771336	1.146291	...	0.756010

Hình 12: Trực quan dữ liệu của mô hình sau khi chuẩn hóa

CHƯƠNG 3: CÁC THUẬT TOÁN ĐỀ XUẤT

Lưu ý:

- Trong cả 8 thuật toán mà nhóm đề xuất dưới đây, nhóm đều thực hiện 2 công đoạn là sử dụng Thư viện có sẵn và Tự code toàn bộ mô hình, ngoài ra ở mỗi công đoạn, cả nhóm đều thực hiện 2 lần với 2 tập dữ liệu là tập dữ liệu gốc (No resampled) và tập dữ liệu được cân bằng (Resampled)
- Để giải quyết hiện tượng quá khớp (*overfitting*), việc thực hiện tinh chỉnh siêu tham số là một trong các giải pháp có thể tiến hành. Do đó, nhóm đã tiến hành tinh chỉnh các siêu tham số đã đề cập ở trên bằng cách sử dụng Grid Search Cross-Validation. Ý tưởng đằng sau Grid Search CV là định ra một lưới các giá trị siêu tham số, sau đó thử nghiệm tất cả các tổ hợp có thể trong lưới đó. Grid Search thuận tiện trong với thử nghiệm các bộ tham số nhằm đánh giá hiệu quả của mô hình. Tuy nhiên, Grid Search có thể tốn nhiều thời gian và tài nguyên nếu không gian siêu tham số quá lớn.

3.1. Logistic Regression:

3.1.1. Cơ sở lý thuyết

Hồi quy Logistic là một trong những phương pháp huấn luyện mô hình học máy giải quyết bài toán học có giám sát. Hồi quy Logistic thường được sử dụng trong bài toán phân loại, đặc biệt đối với phân loại nhị phân như trong dự án này, LR thể hiện được khả năng phán đoán vô cùng tốt với độ chính xác cao trong khi mô hình đơn giản

Những ưu điểm có thể kể đến của LR đó là:

- Dễ hiểu và dễ triển khai: đây là một thuật toán đơn giản và dễ hiểu, dựa trên mối quan hệ giữa đầu vào và xác suất xảy ra của một nhãn (0 hoặc 1).
- Hiệu quả với dữ liệu tuyến tính
- Tốc độ nhanh
- Dễ áp dụng các kĩ thuật hiệu chỉnh

Đối với LR, hàm loss ở đây được sử dụng là entropy, với hàm kích hoạt là sigmoid

$$f(x) = \frac{1}{1+e^{-w^T x}}$$

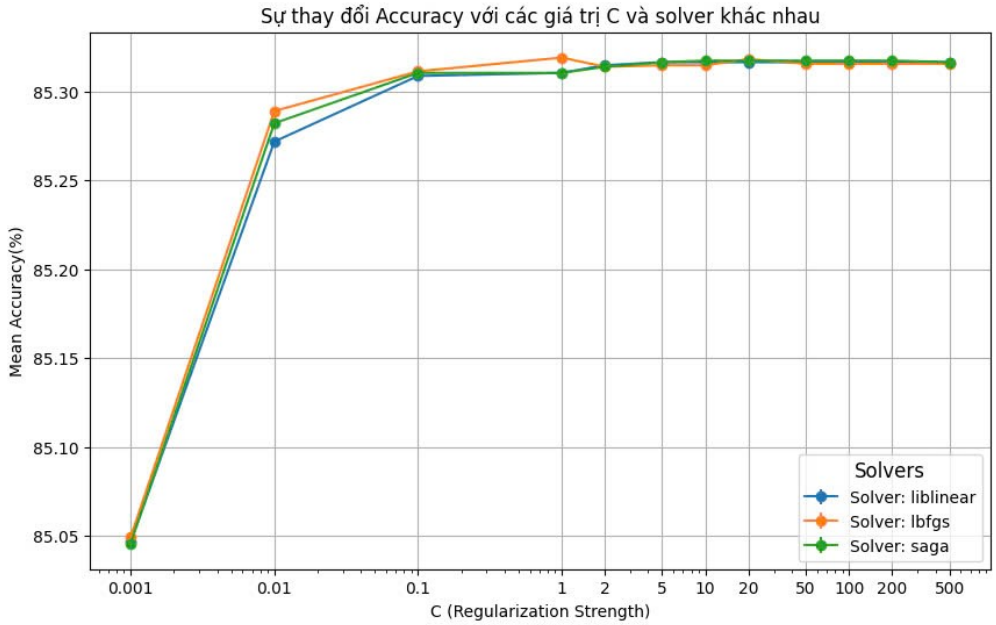
Khi phán đoán: Với 1 điểm dữ liệu mới, mô hình sẽ tính toán xác suất *P* của điểm dữ liệu thuộc vào lớp có nhãn 1 bằng hàm đã học được. Chọn ngưỡng xác định: thường là *0.5:*

- Nếu *P < 0.5* thì điểm đầu vào thuộc lớp có nhãn 0
- Ngược lại *P >= 0.5,* điểm đầu vào thuộc lớp có nhãn 1

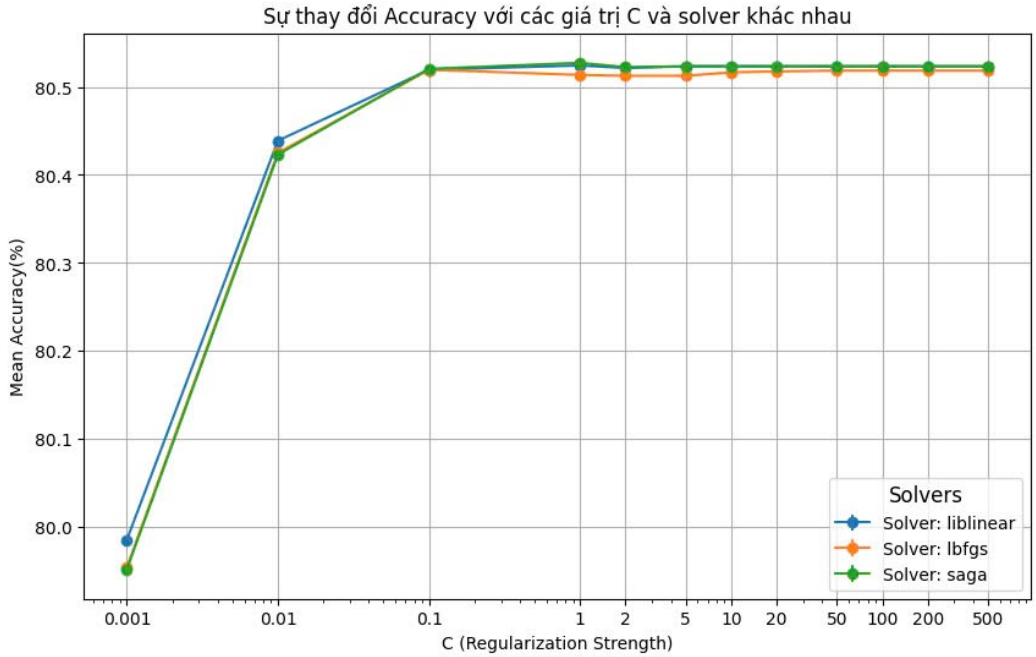
3.1.2. Thực nghiệm

Để chạy được LR, nhóm đã tiến hành GridSearch để chọn ra các siêu tham số phù hợp nhất cho model, các siêu tham số bao gồm:

```
'C': [0.001, 0.01, 0.1, 1, 2, 5, 10, 20, 50, 100, 200, 500]

'solver' = ['liblinear', 'lbfgs', 'saga']
```

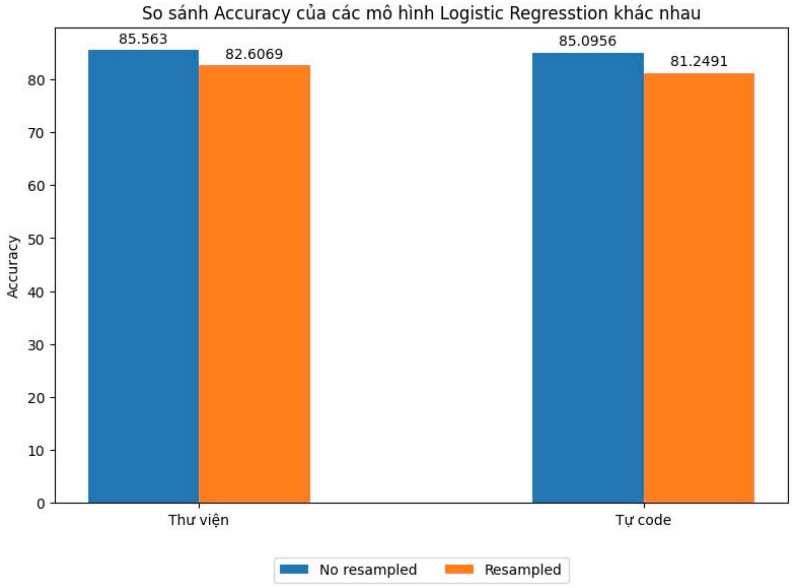


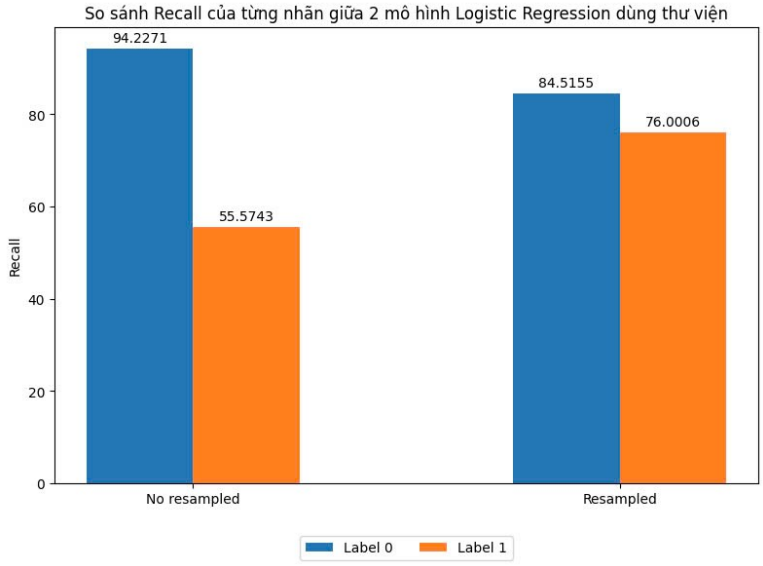
Hình 13: GridSeach với tập dữ liệu gốc

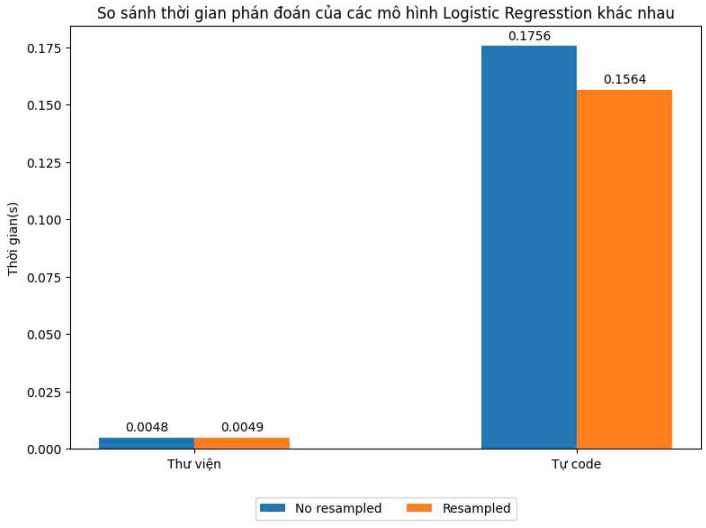


Hình 14: GridSeach với tập dữ liệu đã được cân bằng

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:







Hình 15, 16, 17: Một số kết quả đánh giá được khi sử dụng LR

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp LR là vô cùng tốt, accuracy rơi vào khoảng 85%, thuộc mức tốt đối với 1 bài toán phân loại và trung bình trong nhóm thuật toán mà nhóm đề xuất.
- Khi sử dụng resampled độ chính xác giảm một chút (do dữ liệu giảm đi), nhưng recall lại thay đổi rất tích cực đối với nhãn 1 khi so với trường hợp ban đầu.
- Tốc độ đưa ra dự đoán (predict) rất nhanh kể cả khi dùng thư viện hay khi tự code

3.2. K-Nearest Neighbors

3.2.1. Cơ sở lý thuyết:

K-Nearest Neighbours(KNN) là một trong các thuật toán học máy giải quyết bài toán học có giám sát. KNN có thể sử dụng để giải quyết bài toán hồi quy và phân loại,

KNN có 1 đặc điểm nổi bật, đó là trong quá trình huấn luyện, mô hình không thực sự "học" được một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới dựa vào tập dữ liệu đã lưu, do đó KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

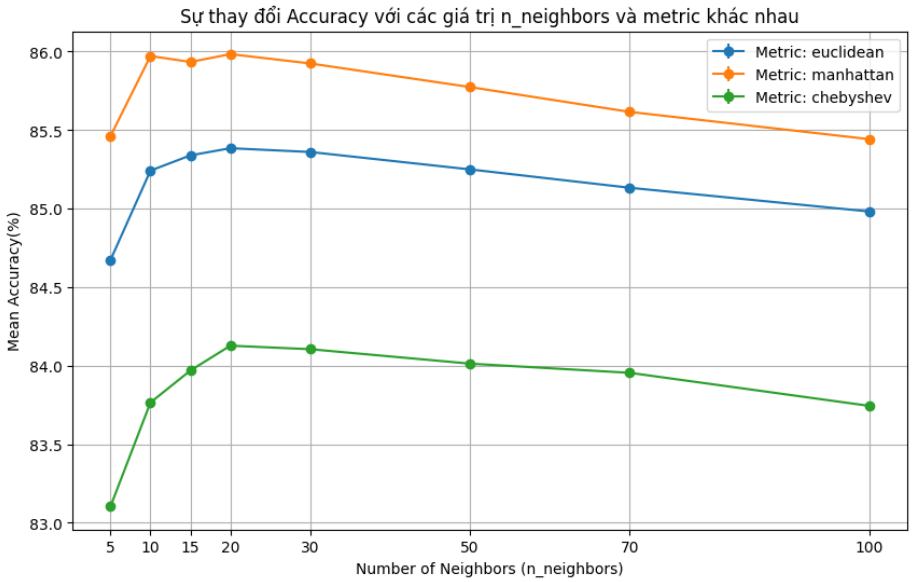
Quá trình phán đoán 1 mẫu dữ liệu mới sẽ dựa trên k hàng xóm gần nhất của nó trong tập dữ liệu. Để phán đoán 1 mẫu dữ liệu trong tương lai, cần phải chọn: độ đo tương đồng giữa 2 điểm dữ liệu và số lượng hàng xóm *k*.

3.2.2. Thực nghiệm

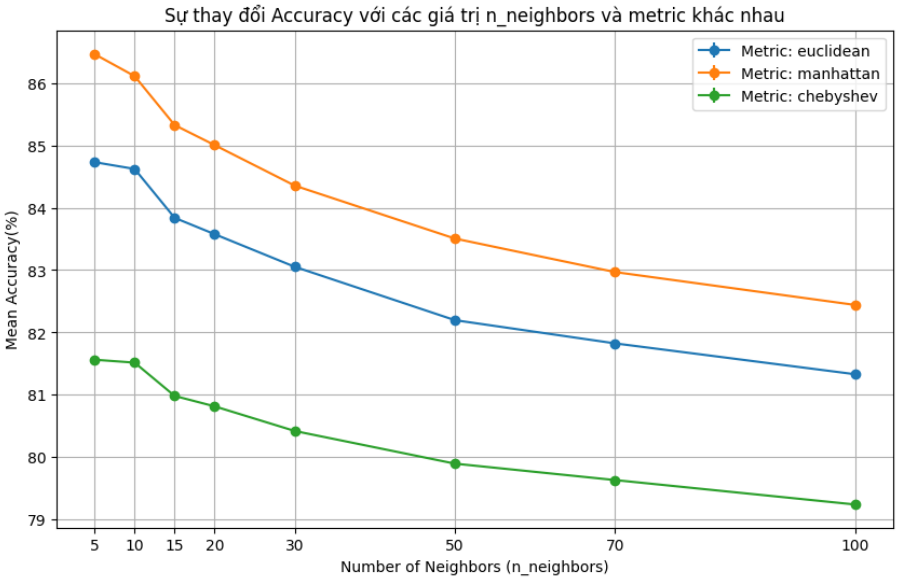
Các siêu tham số cần phải chọn ra bao gồm:

```
'n_neighbors': [5, 10, 15, 20, 30, 50, 70, 100]
```

```
‘metric’ = ['euclidean', 'manhattan', 'chebyshev']
```

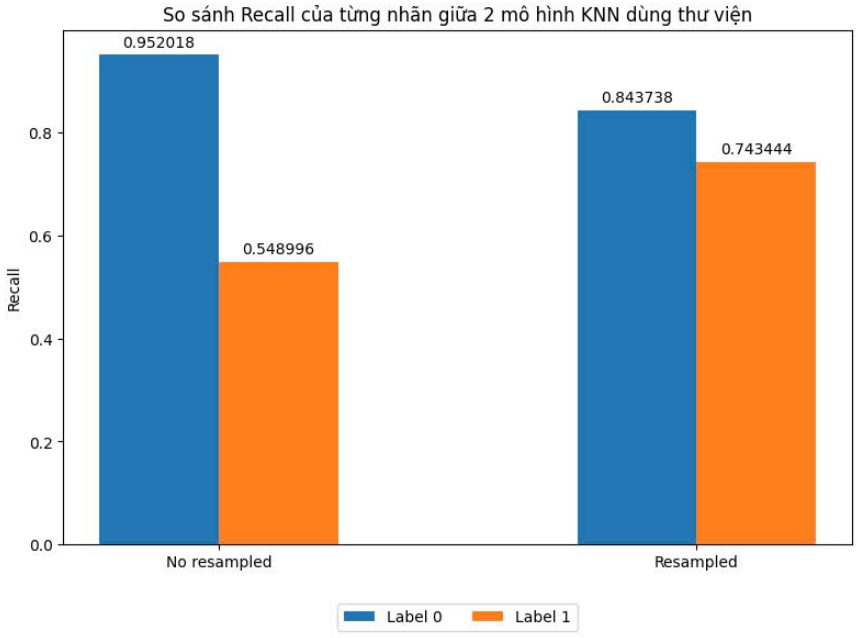


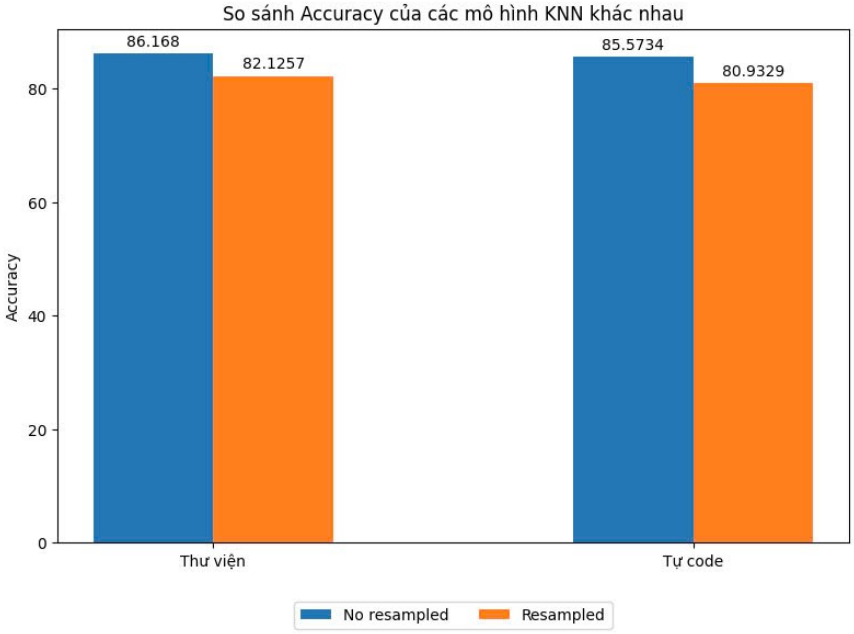
Hình 18: GridSeach với tập dữ liệu gốc

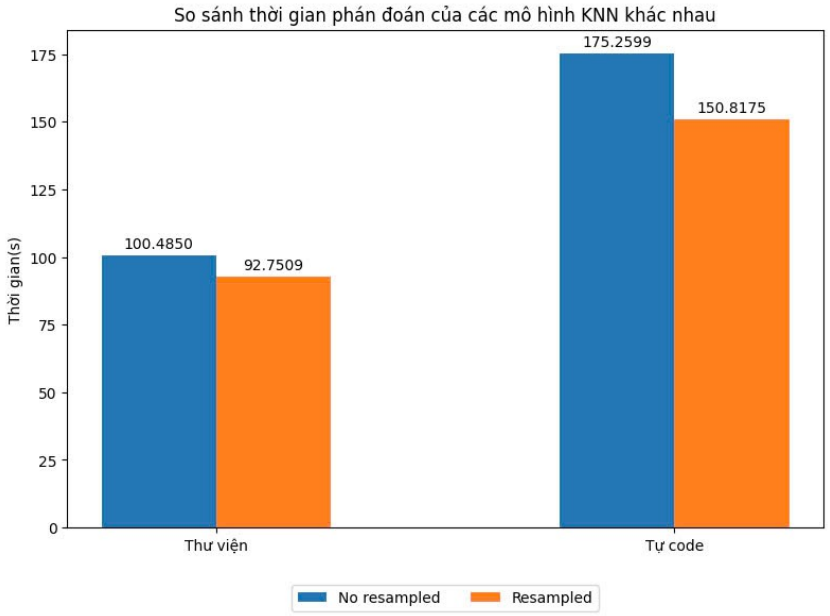


Hình 19: GridSeach với tập dữ liệu đã cân bằng (Resampled)

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:







Hình 20, 21, 22: Một số kết quả đánh giá được khi sử dụng KNN

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp KNN là vô cùng tốt, accuracy rơi vào khoảng 86%, thuộc mức tốt đối với 1 bài toán phân loại và trung bình trong nhóm thuật toán mà nhóm đề xuất.
- Đây là một thuật toán đơn giản, dễ hiểu
- Tốc độ đưa ra dự đoán (predict) tương đối chậm kể cả khi dùng thư viện hay khi tự code, quá trình gridsearch rất lâu

3.3. Naive Bayes

3.3.1. Cơ sở lý thuyết

Naive Bayes là một trong các phương pháp học máy giải quyết bài toán học có giám sát.

Naive Bayes huấn luyện ra một mô hình xác suất (mô hình thống kê) dùng để dự đoán xác suất của các sự kiện dựa trên trên dữ liệu đầu vào. Cơ sở của phương pháp Naive Bayes là các lý thuyết xác suất thống kê và định lý Bayes.

Naive Bayes chủ yếu được sử dụng trong bài toán phân loại.

Quá trình huấn luyện sẽ ước lượng 2 đại lượng: $P(c_i)$ và $P(x|c_i)$

Giả sử các thuộc tính là độc lập với nhau, giả sử mỗi thuộc tính t của các mẫu dữ liệu trong lớp i là 1 biến ngẫu nhiên tuân theo 1 phân phối xác F_{it}. Khi đó ta có thể tính được $P(x_k|c_i)$

Khi đó có xác suất $P(x|c_i) = \prod_{k=1}^{d} P(x_k|c_i)$

Đối với đại lượng $P(c_i)$ tính bằng tỉ số dữ liệu nhãn c_i trên tổng số dữ liệu tập dữ liệu.

Quá trình phán đoán:

Với một mẫu dữ liệu mới x, ta sẽ phán đoán nhãn cho x : $c = argmax_{c \in \{c_1,c_2,\dots,c_C\}} P(c|x)$

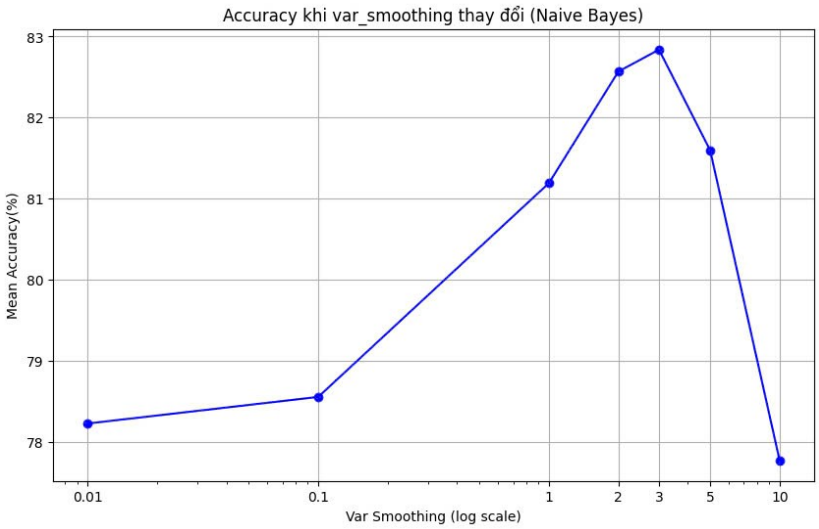
Tính $P(c|x) = \frac{P(c)P(x|c)}{P(x)} = \frac{P(c)\prod_{k=1}^{d} P(x_k|c_i)}{P(x)}$

$$c = argmax_{c \in \{c_1,c_2,\dots,c_C\}} P(c|x) = argmax_{c \in \{c_1,c_2,\dots,c_C\}} P(c) \prod_{k=1}^{d} P(x_k|c_i)$$

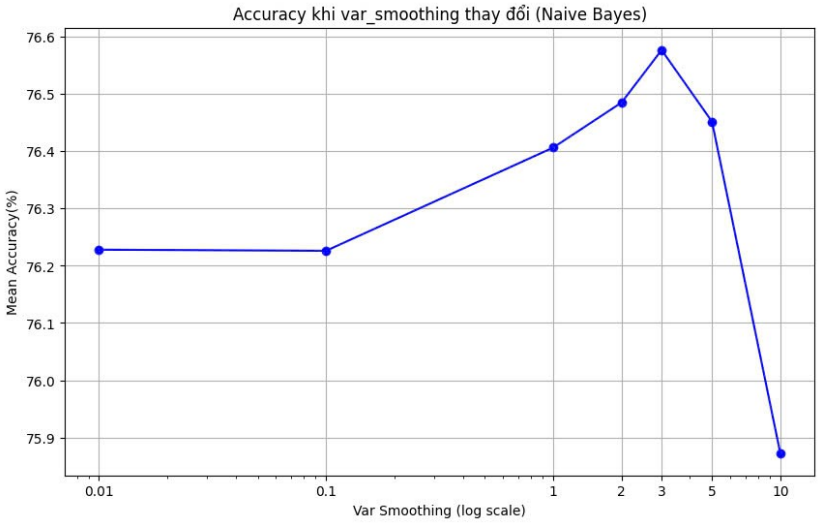
3.3.2. Thực nghiệm

Các siêu tham số cần tìm:

```
GridSearch: 'var_smoothing': [0.01, 0.1, 1, 2, 3, 5, 10]
```

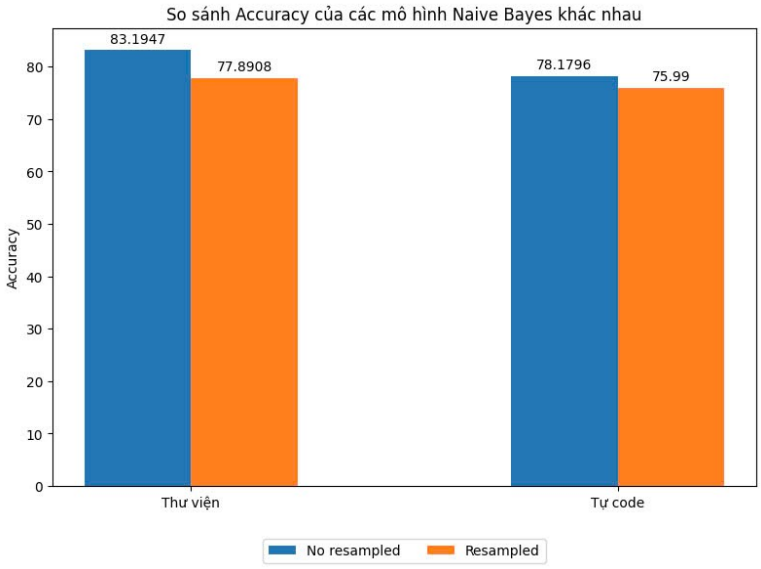


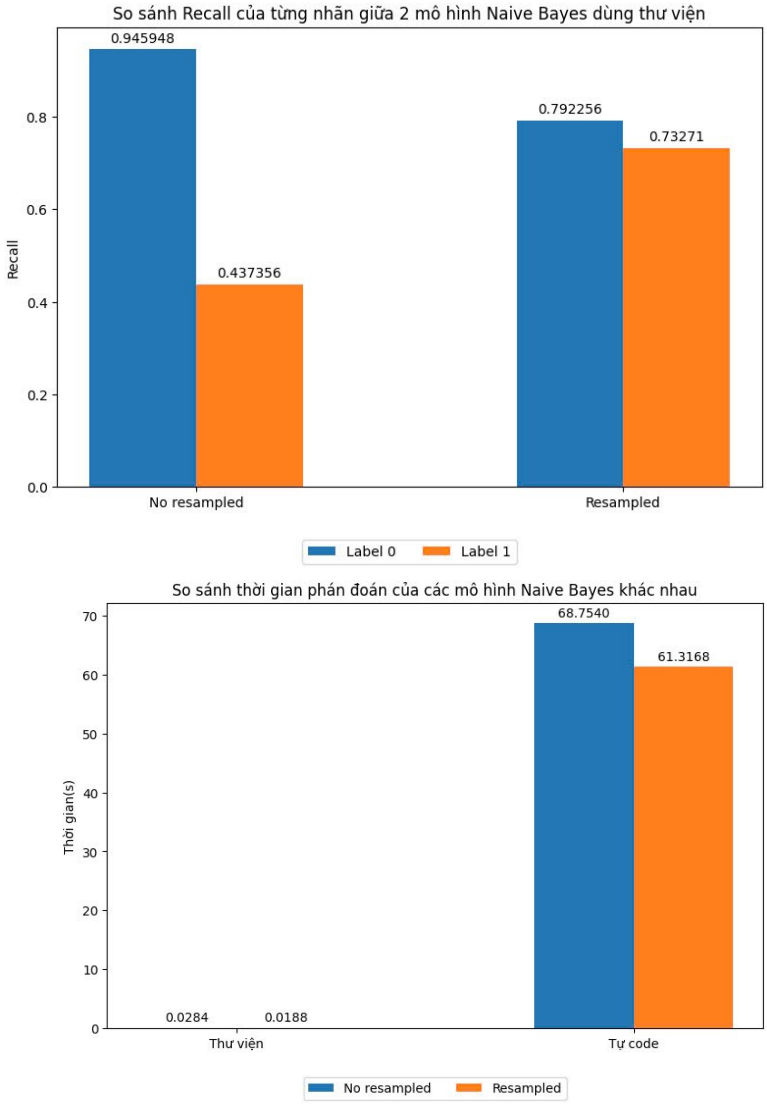
Hình 23: GridSeach với tập dữ liệu gốc



Hình 24: GridSeach với tập dữ liệu đã cân bằng

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:





Hình 25, 26, 27: Một số kết quả đánh giá được khi sử dụng NaiveBayes

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp NaiveBayes chỉ ở mức trung bình, accuracy rơi vào khoảng 80%, tốt so với 1 bài toán phân loại và nhưng so với nhóm thuật toán mà nhóm đề xuất, đây thuộc kết quả tệ nhất
- Tuy nhiên đây là một thuật toán đơn giản, dễ hiểu, dễ xây dựng

3.4. Support Vector Machine

3.4.1. Cơ sở lý thuyết

SVM (Support Vector Machine) có thể được sử dụng cho cả phân loại tuyến tính và không tuyến tính

- Phân loại tuyến tính: SVM tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân tách hai lớp trong không gian đặc trưng. Đây là trường hợp mà các dữ liệu có thể được phân tách bằng một đường thẳng (hoặc mặt phẳng trong không gian nhiều chiều)
- Phân loại không tuyến tính: Khi các dữ liệu không thể phân tách bằng một siêu phẳng tuyến tính, SVM có thể sử dụng hàm nhân(kernel function) để ánh xạ dữ liệu vào một không gian đặc trưng cao hơn, nơi mà chúng có thể được phân tách tuyến tính

a, Trường hợp dữ liệu có thể phân chia tuyến tính

Cho tập dữ liệu $D = \{(x_1, y_1), (x_2, y_2), \dots (x_r, y_r)\}$. với r quan sát

Mỗi x_i là một vector n chiều, mỗi chiều biểu diễn một thuộc tính.

Với bài toán phân loại, đầu ra y_i là một nhãn cho nằm trong $\{-1;\ 1\}$ với -1 là lớp âm, 1 là lớp dương .

Mục tiêu của SVM là tìm một siêu phẳng có dạng: $w^T x + b = 0$ với w là trọng số, b là bias

Siêu phẳng này phải phân tách các dữ liệu của hai lớp sao cho khoảng cách từ siêu phẳng đến các điểm gần nhất là lớn nhất. Khoảng cách này gọi là *margin*.

Điều kiện phân tách dữ liệu: $y_i(w^T x_i + b) \geq 1$ với mọi i

b, Trường hợp dữ liệu không thể phân chia tuyến tính

Đưa dữ liệu vào một không gian khác, có nhiều chiều hơn, ở đó dữ liệu có thể phân chia tuyến tính. Sau đó sử dụng SVM trong không gian mới này

Điều đó có nghĩa là ta phải tìm được một hàm ánh xạ ϕ biến tập dữ liệu ban đầu $X = \{(x_1, y_1), (x_2, y_2), \dots (x_r, y_r)\}$ thành $F = \{(\phi(x_1), y_1), (\phi(x_2), y_2), \dots (\phi(x_r), y_r)\}$

ứng với chiều không gian mới

Quá trình huấn luyện

Quá trình huấn luyện SVM nhằm tìm ra siêu phẳng tối ưu để phân tách các điểm dữ liệu của hai lớp.

a, Lựa chọn siêu phẳng (Hyperplane)

Tìm siêu phẳng dưới dạng $w^T x + b = 0$, sao cho khoảng cách từ các điểm dữ liệu thuộc hai lớp đến siêu phẳng là lớn nhất.

Nếu dữ liệu không thể tách biệt tuyến tính, sử dụng **soft margin SVM** (cho phép vi phạm điều kiện ràng buộc).

b, Xây dựng bài toán tối ưu hóa

Tìm **w** và **b** sao cho:

Cực tiểu hóa:

$L(w, b) = \frac{1}{2} \| w \|^2 + C \sum \xi_i$

(Cân bằng giữa việc tối ưu hóa khoảng cách và cho phép một số điểm dữ liệu vi phạm điều kiện ràng buộc.)

Với điều kiện :

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Với các bài toán phi tuyến tính, sử dụng **kernel trick** để ánh xạ dữ liệu sang không gian đặc trưng cao hơn.

c, Tìm nghiệm bài toán tối ưu

Dùng các kỹ thuật như *Gradient Descent, Quadratic Programming (QP)*, hoặc các thư viện tối ưu hóa như *libSVM* để giải bài toán.

Kết quả là vector trọng số **w**, bias **b**, và các $\boldsymbol{\alpha_i}$ (trọng số cho các vector hỗ trợ - support vectors).

d,Lựa chọn kernel (nếu cần)

Chọn hàm kernel phù hợp (Polynomial, RBF, Sigmoid, ...) tùy thuộc vào bài toán và dạng phân bố dữ liệu.

Quá trình phán đoán

Quá trình phán đoán sử dụng siêu phẳng hoặc hàm kernel đã được huấn luyện để phân loại một điểm dữ liệu mới **x**. Các bước như sau:

a, Siêu phẳng phân loại

+ Trong SVM tuyến tính:

$$f(x) = w^T x + b$$

+ Trong SVM phi tuyến tính (sử dụng kernel):

$$f(x) = \sum_{\blacksquare}^{\blacksquare} \alpha_i y_i K(x_i, x) + b$$

$K(x_i, x)$: Giá trị kernel giữa điểm dữ liệu mới x và các vector hỗ trợ x_i.

b, Dự đoán nhãn:

+ Nếu $f(x) \geq 0$, gán nhãn $y = 1$.
+ Nếu $f(x) < 0$, gán nhãn $y = -1$.

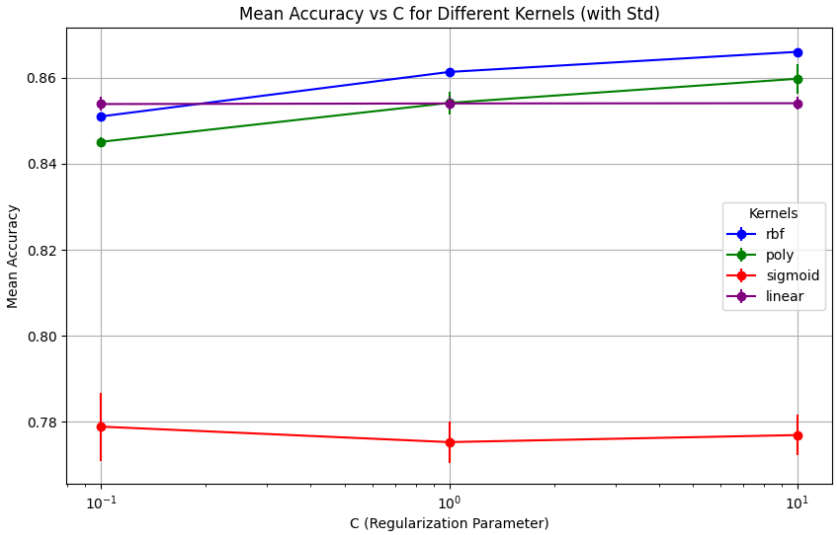
Ưu và nhược điểm của SVM

+ Ưu điểm:
  - Hiệu quả trên dữ liệu nhỏ**:** Hoạt động tốt với dữ liệu ít mẫu nhưng nhiều chiều.
  - Độ chính xác cao**:** Phân loại tốt cả dữ liệu tuyến tính và không tuyến tính.
  - Ngăn overfitting**:** Sử dụng regularization và support vectors để giảm overfitting.
  - Kernel trick**:** Linh hoạt khi xử lý dữ liệu phức tạp bằng cách ánh xạ sang không gian cao hơn.
+ Nhược điểm:
  - Thời gian tính toán cao**:** Thời gian huấn luyện dài với dữ liệu lớn.
  - Không phù hợp với dữ liệu nhiều nhiễu: Outliers có thể ảnh hưởng đến hiệu quả.

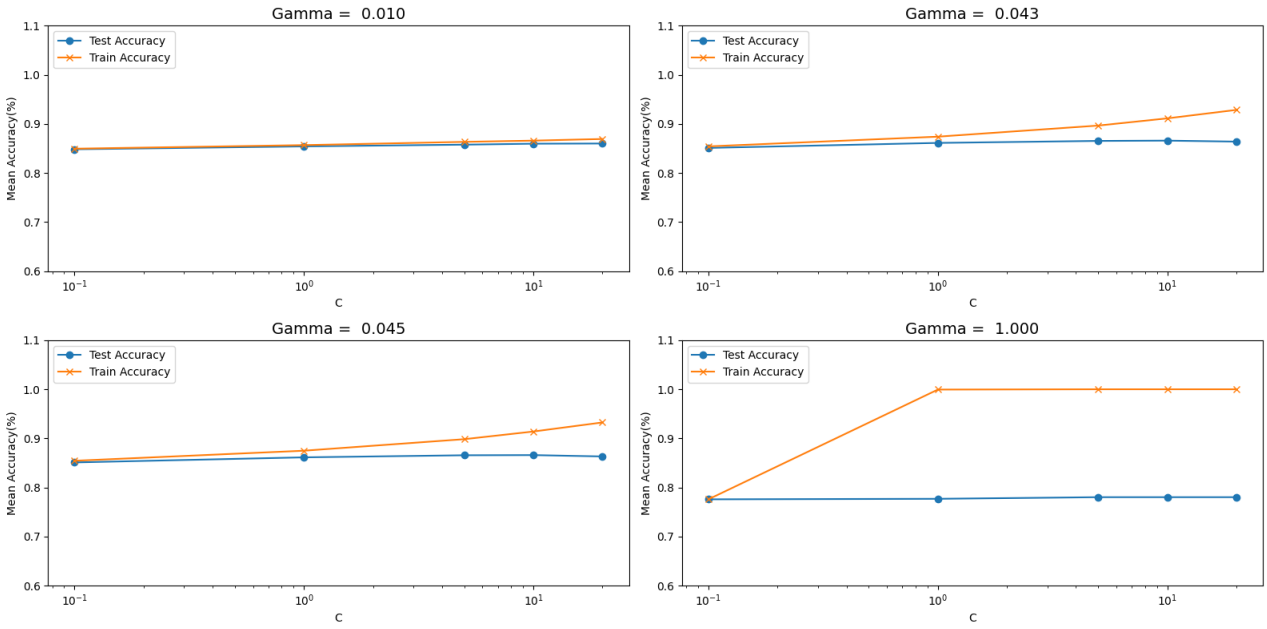
3.4.2. Thực nghiệm

Các siêu tham số mà mô hình cần tìm kiếm:

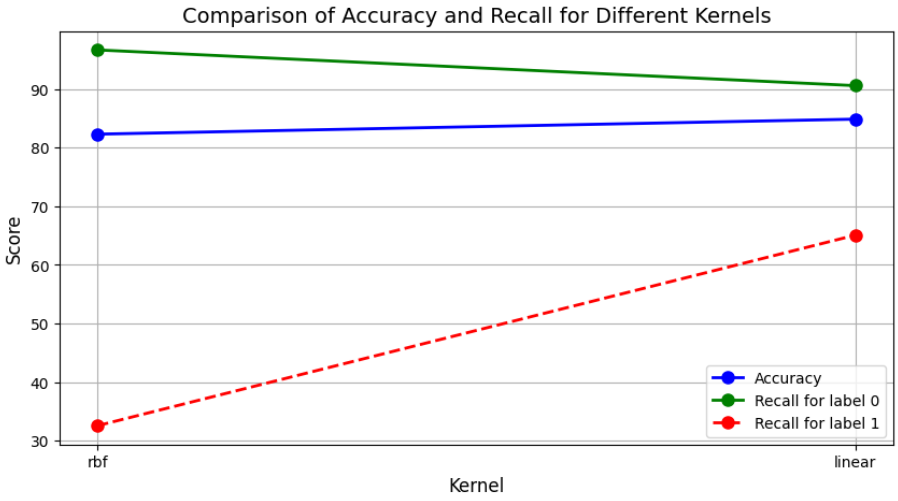
```
'C': [0.1, 1, 10]

'kernel': ['linear', 'poly', 'rbf', 'sigmoid']
```

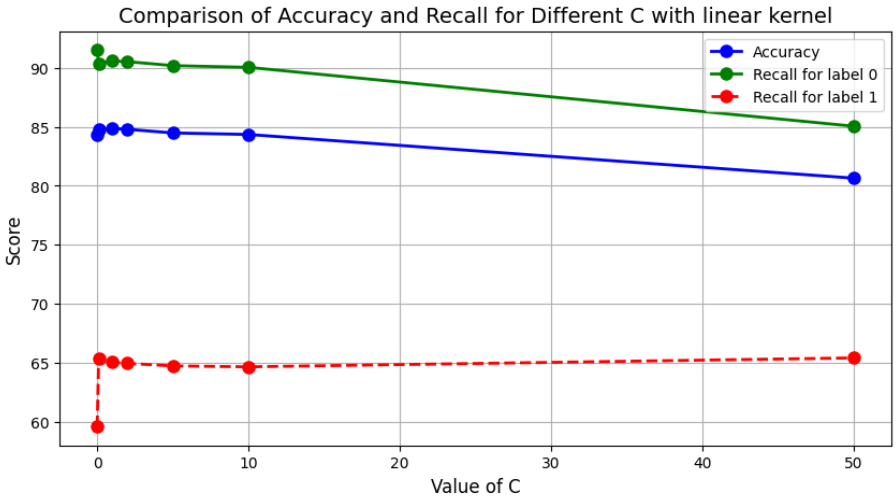


Hình 28: Gridsearch với tập dữ liệu gốc

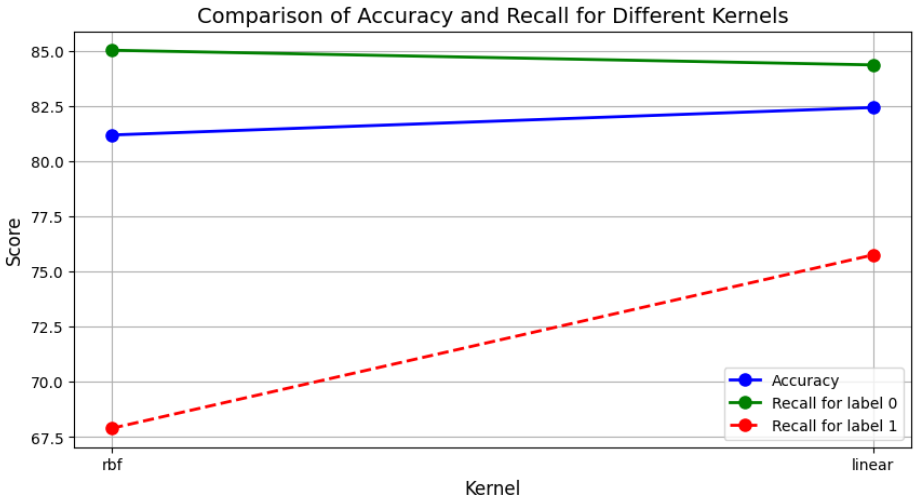


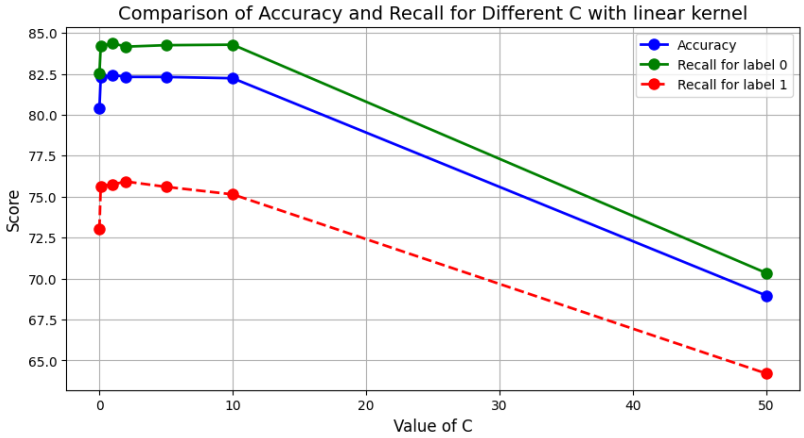
Hình 29: Sự thay đổi của Accuracy khi thay đổi Gamma





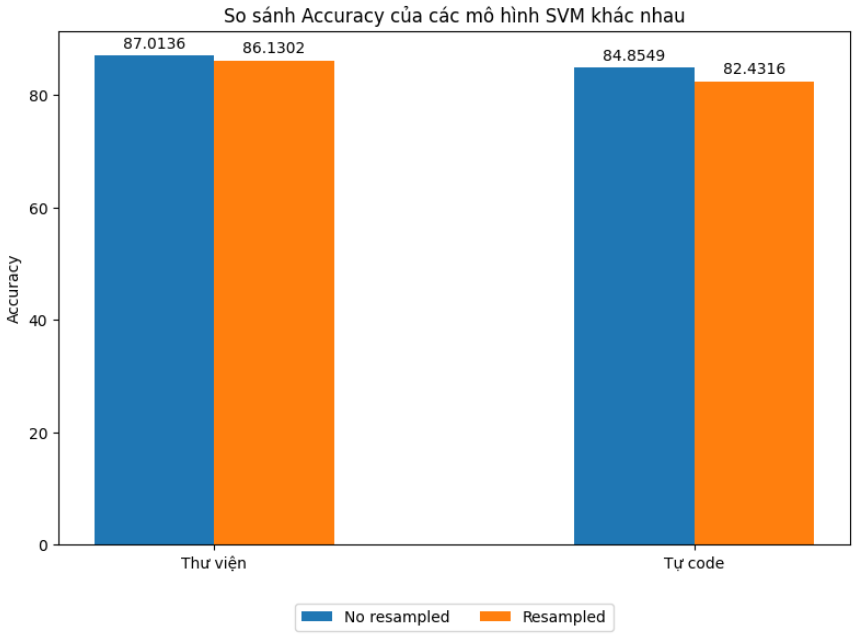
Hình 30, 31: Sự thay đổi của Accuracy và Recall trong tập dữ liệu gốc

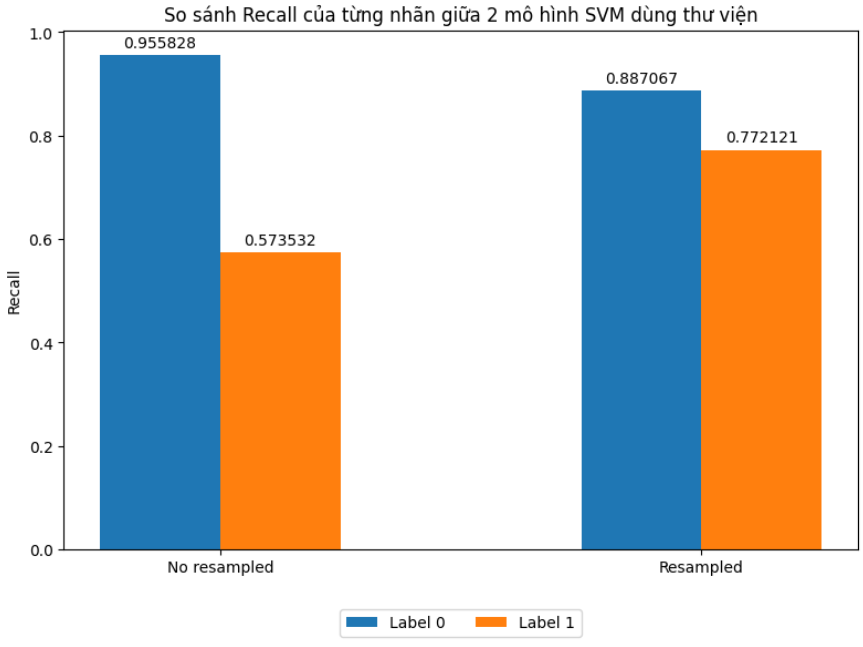




Hình 32, 33: Sự thay đổi của Accuracy và Recall trong tập dữ liệu đã cân bằng (resampled)

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:





Hình 34, 35. Một số kết quả thu được khi sử dụng SVM

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp SVM khá tốt, accuracy rơi vào khoảng 86%, thuộc nhóm các thuật toán có kết quả tốt mà nhóm đã đề xuất
- Tốc độ chạy của SVM tương đối chậm.

3.5. Decision Tree

3.5.1. Cơ sở lý thuyết

Decision Tree là một thuật toán học máy được sử dụng phổ biến cho cả bài toán phân loại (classification) và hồi quy (regression). Cây quyết định tổ chức các điều kiện quyết định theo dạng cây, nơi mà mỗi nút đại diện cho một thuộc tính, các nhánh là kết quả của các điều kiện, và lá là các nhãn hoặc giá trị dự đoán.

Các ưu điểm của Decision Tree:

+ Dễ hiểu và giải thích.
+ Có thể xử lý cả dữ liệu số và dữ liệu phân loại.
+ Không yêu cầu chuẩn hóa dữ liệu.

Quá trình huấn luyện

Quá trình huấn luyện Decision Tree bao gồm ba bước chính: lựa chọn thuộc tính tối ưu, phân chia dữ liệu, và dừng phân chia.

a. Lựa chọn thuộc tính tối ưu (Splitting Criterion)
Để chọn thuộc tính tốt nhất tại mỗi nút, cần đo lường **mức độ giảm hỗn loạn** của dữ liệu sau mỗi lần phân chia. Một số tiêu chí phổ biến là Entropy và Information Gain (Gain):

+ Entropy**:** Đo lường độ hỗn loạn hoặc độ không chắc chắn của dữ liệu:

$$Entropy(S) = -\sum_{i=1}^{c} p_i log_2(p_i)$$, Trong đó:

p_i: Tỷ lệ mẫu thuộc lớp i trong tập S.

+ Information Gain**:** Mức giảm entropy sau khi phân chia:

$$Gain = Entropy(S) - \sum_{k=1}^{m} w_k Entropy(S_k)$$, Trong đó:

$w_k = \frac{|S_k|}{|S|}$: Tỷ lệ của tập con S_k.

S_k: Tập dữ liệu sau khi phân chia theo giá trị k của thuộc tính.

b. Phân chia dữ liệu
Sau khi chọn thuộc tính tốt nhất:

+ Tạo một nhánh cho từng giá trị của thuộc tính.
+ Phân chia dữ liệu thành các tập con dựa trên giá trị của thuộc tính.

c. Dừng phân chia
Quá trình xây dựng cây dừng lại khi xảy ra một trong các điều kiện sau:

+ Tất cả các mẫu trong tập con thuộc về cùng một lớp.

+ Không còn thuộc tính nào khả dụng để phân chia.
+ Số mẫu trong tập con nhỏ hơn một ngưỡng tối thiểu.
+ Độ sâu tối đa của cây đã đạt tới (để tránh overfitting).

Quá trình phán đoán

a. Duyệt cây quyết định

+ Đi từ nút gốc, kiểm tra giá trị của thuộc tính quyết định.
+ Dựa trên giá trị của thuộc tính, đi qua các nhánh tương ứng.
+ Tiếp tục duyệt cây cho đến khi đến một nút lá.

b. Dự đoán nhãn hoặc giá trị

+ Phân loại**:** Nhãn tại nút lá là nhãn của lớp chiếm đa số.
+ Hồi quy: Giá trị tại nút lá là giá trị trung bình của các mẫu trong nút.

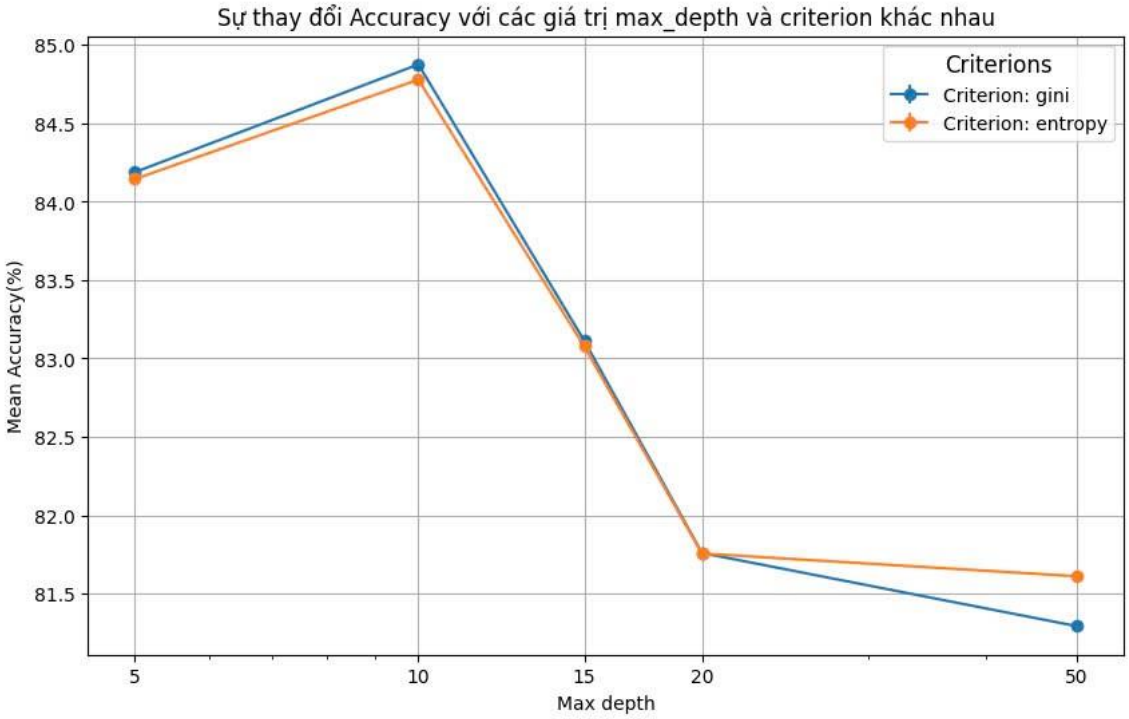
Ưu và nhược điểm của Decision Tree

+ Ưu điểm:
    - Dễ hiểu và giải thích.
    - Không yêu cầu chuẩn hóa dữ liệu.
    - Phù hợp với dữ liệu phân loại và số học.
    - Xử lý được dữ liệu thiếu.
    - Dự đoán nhanh sau khi huấn luyện.
+ Nhược điểm:
    - Dễ bị overfitting khi cây quá sâu.
    - Nhạy cảm với nhiễu dữ liệu.
    - Không tốt với mối quan hệ phức tạp.
    - Cần điều chỉnh tham số để tránh cây quá lớn.

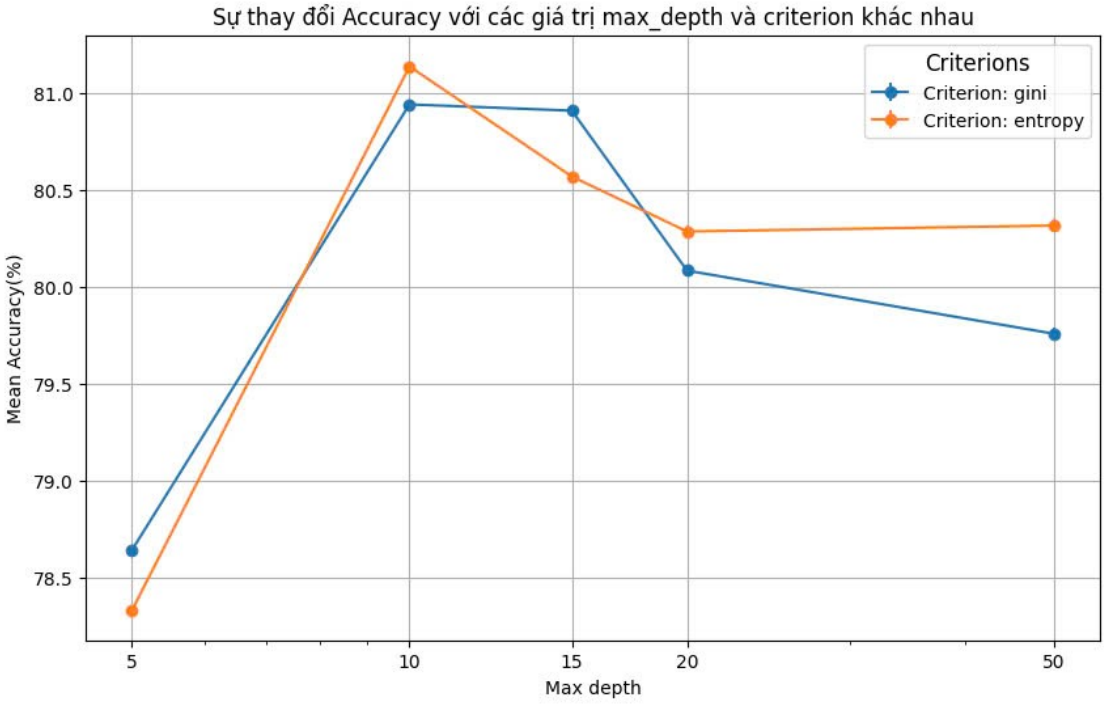
3.5.2. Thực nghiệm

Các siêu tham số cần phải đi tìm:

```
'max_depth': [5, 10, 15, 20, 50]
‘criterion’ = ['gini', 'entropy']
```

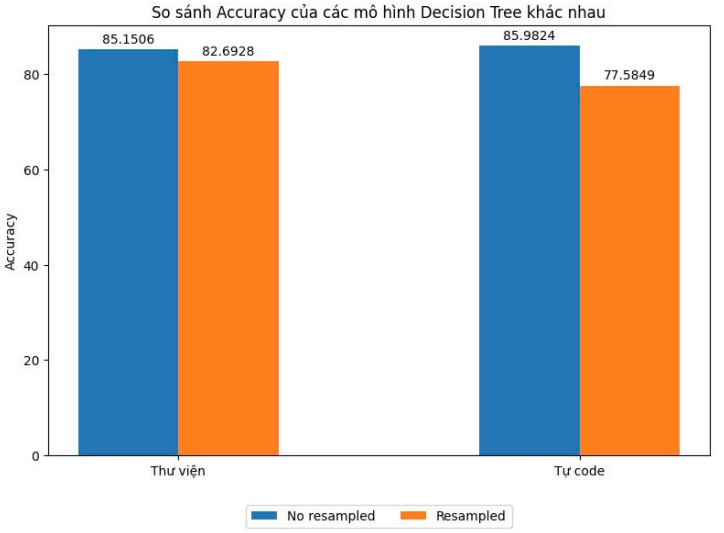


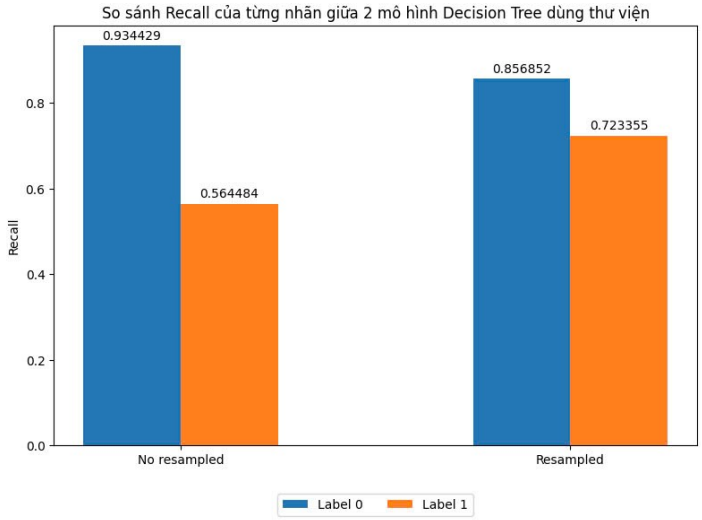
Hình 36. GridSearch với tập dữ liệu gốc

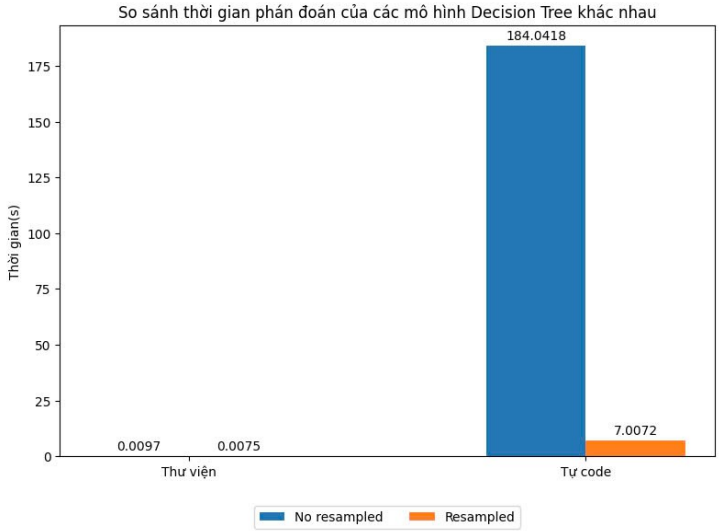


Hình 37. GridSearch với tập dữ liệu đã được cân bằng

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:







Hình 38, 39, 40: Một số kết quả thu được khi sử dụng Decision Tree

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp Decision Tree tương đối tốt, accuracy rơi vào khoảng 85%, thuộc nhóm các thuật toán có kết quả tốt mà nhóm đã đề xuất

- Tốc độ chạy của Decision Tree tương đối chậm, nguy cơ mắc phải Overfitting cao

3.6. Random Forest

3.6.1. Cơ sở lý thuyết

Random Forest là một thuật toán học máy có giám sát, đây là một thuật toán mạnh mẽ và linh hoạt, có thể được sử dụng cho cả bài toán phân loại (classification) và hồi quy (regression). Random Forest dựa trên khái niệm học kết hợp (ensemble learning), nơi mà nhiều mô hình (cụ thể là các cây quyết định) được kết hợp để giải quyết các vấn đề phức tạp, đồng thời cải thiện hiệu suất và độ chính xác của mô hình.

Thuật toán Random Forest hoạt động dựa trên việc xây dựng nhiều cây quyết định (decision trees) trên các tập con dữ liệu và thuộc tính được chọn ngẫu nhiên. Mỗi cây quyết định trong rừng ngẫu nhiên được huấn luyện với một tập con dữ liệu được tạo ra bằng kỹ thuật lấy mẫu bootstrapping (lấy mẫu ngẫu nhiên có lặp lại). Mỗi nút trong cây sẽ được phân nhánh dựa trên một tập con thuộc tính được chọn ngẫu nhiên. Điều này làm giảm sự tương quan giữa các cây, tăng sự đa dạng của mô hình và giảm thiểu tình trạng quá khớp (overfitting).

Thuật toán Random Forest:

1. Tạo tập con dữ liệu:
    - Lấy mẫu ngẫu nhiên có lặp lại (bootstrapping) để tạo ra K tập con Di từ tập dữ liệu gốc D.
2. Xây dựng cây quyết định:
   Xây dựng cây quyết định bằng cách:
    - Tại mỗi nút, chọn ngẫu nhiên một tập con các thuộc tính.
    - Phân nhánh cây dựa trên tập thuộc tính đó.

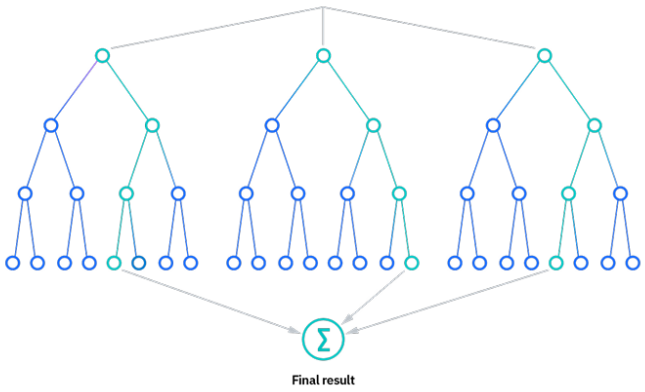
   Cây được xây dựng đến kích thước lớn nhất mà không cắt tỉa (pruning).

3. Dự đoán:

Khi có dữ liệu mới, kết quả dự đoán được tính bằng cách:

Trung bình kết quả của các cây (đối với hồi quy).

Lấy kết quả theo đa số từ các cây (đối với phân loại).



Hình 41: Minh họa Ensemble Learning

Khi dùng thuật toán Random Forests, chúng ta nên chú ý đến các siêu tham số như: số lượng cây quyết định sẽ xây dựng, số lượng thuộc tính dùng để xây dựng cây, độ sâu của các cây...

3.6.2. Thực nghiệm

3.6.2.1. Quá trình huấn luyện mô hình (Training phase)

Các siêu tham số (*hyperparameters*) là một trong những phần quan trọng quyết định hiệu quả của các mô hình học máy. Các siêu tham số quan trọng nhất của Random Forest có thể được điều chỉnh bao gồm:

- n_estimators: Số lượng cây quyết định trong một rừng ngẫu nhiên (*random forest*).
- criterion: Tiêu chí để phân chia (Gini/ Entropy/ Log Loss cho bài toán phân loại; MSE/ MAE cho bài toán hồi quy).
- max_depth: Độ sâu tối đa của mỗi cây quyết định.
- val_set: tập tối ưu được lấy ra một phần từ tập huấn luyện

Trong giai đoạn đầu của quá trình huấn luyện, mục tiêu là kiểm tra khả năng của thuật toán và tiềm năng của nó trong việc giải quyết bài toán phân loại nhị phân. Do đó, các siêu tham số được khởi tạo với các giá trị mặc định của thư viện sklearn:

- n_estimators = 100.
- criterion = Gini.
- max_depth = None, nghĩa là các cây sẽ tiếp tục phân chia đến khi các mẫu ở các nút lá có cùng nhãn.
- Chia train_set ban đầu thành train_set và val_set với tỉ lệ 80/20

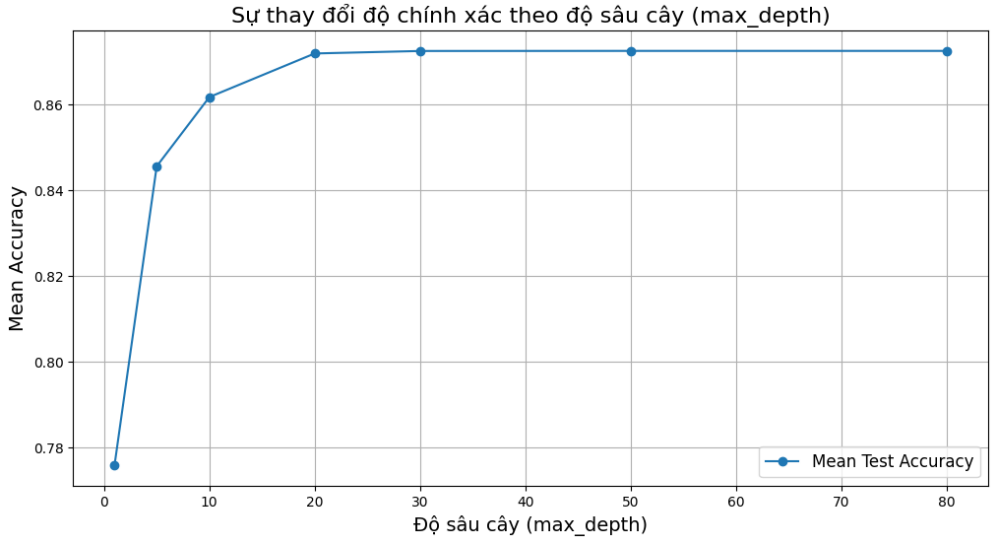
Với thiết lập ban, mô hình đạt độ chính xác 100% trên tập huấn luyện, và 87% trên tập tối ưu. Kết quả đạt được khá tốt tuy nhiên độ chính xác đạt được tối ưu vẫn thấp hơn khá nhiều so với tập huấn luyện, ta cần cân nhắc và tính tổng quát hoá của mô hình, xem liệu nó có đang gặp phải vấn đề quá khớp (overfitting hay không)

3.6.2.2. Lựa chọn các siêu tham số (Hyperparameter tuning)

Để tìm hiểu về hiệu suất tốt nhất mà mô hình có thể đạt được, nhóm đã thực hiện một số kiểm tra để chỉ ra mối tương quan giữa một số đặc trưng quan trọng và hiệu suất. Cụ thể, chúng em tập trung vào mối quan hệ giữa độ chính xác của mô hình trên tập huấn luyện, tập kiểm tra và số lượng cây trong rừng và độ sâu của các cây.

Ảnh hưởng của số lượng cây: Trong quá trình huấn luyện, nhóm nhận thấy rằng độ chính xác của mô hình tăng dần khi số lượng cây (n_estimators) trong Random Forest tăng lên. Tuy nhiên, khi số lượng cây đạt đến 100, độ chính xác hầu như không cải thiện đáng kể. Điều này cho thấy rằng sau một ngưỡng nhất định, việc tăng thêm số lượng cây chỉ mang lại hiệu quả không đáng kể đối với độ chính xác của mô hình.

Lý do chính cho hiện tượng này có thể được giải thích là Random Forest dựa vào cơ chế tổng hợp dự đoán của nhiều cây quyết định độc lập. Khi số lượng cây tăng lên, mô hình có khả năng học được nhiều mẫu hơn và giảm bớt ảnh hưởng của sự ngẫu nhiên trong dữ liệu. Tuy nhiên, khi đã đạt được một số lượng cây đủ lớn, việc thêm cây chỉ làm tăng chi phí tính toán mà không cải thiện hiệu suất rõ rệt. Dù vậy, việc sử dụng số lượng cây lớn vẫn đảm bảo mô hình đạt độ chính xác cao và ổn định hơn, nhờ khả năng kết hợp tốt giữa các dự đoán từ các cây quyết định.



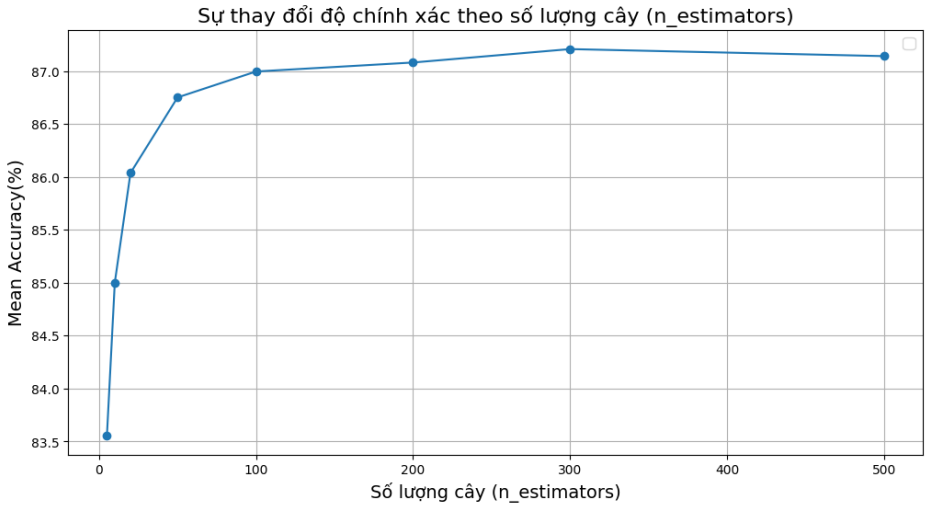
Hình 42: GridSearch tìm siêu tham số max_depth

Ảnh hưởng của độ sâu cây: Với độ sâu cây nhỏ, mô hình hoạt động với hiệu suất trung bình với độ chính xác đạt khoảng 75%. Hiệu suất cải thiện khi chúng ta tăng độ sâu cây và đạt đến một ngưỡng ổn định sau khi đạt độ sâu 20.

Hiện tượng này là hợp lý vì Random Forest (RF) được xây dựng từ nhiều cây quyết định (decision tree), và hiệu suất của từng cây quyết định phụ thuộc mạnh mẽ vào số lượng nút và độ sâu của nó. Cây càng sâu thì độ chính xác càng cao, nhưng cũng có nguy cơ bị quá khớp (overfitting) cao hơn, do mô hình có thể tập trung vào việc tìm kiếm các mẫu cụ thể trong một số trường hợp nhất định.

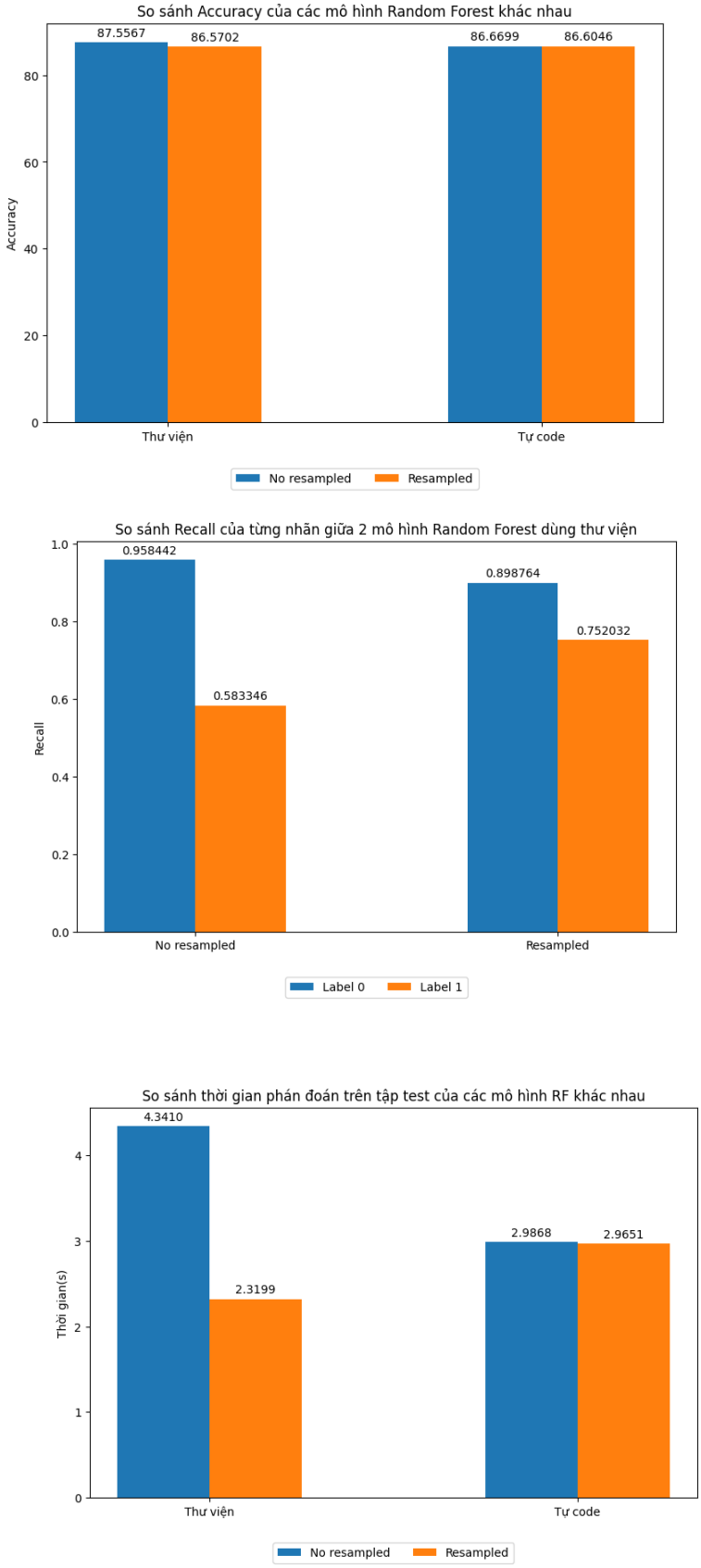
Quá trình được tiến hành tương tự với tập train đã được resampled cho thấy một vài điểm khác như criterion là entropy cho kết quả tốt hơn.

Kết quả cho thấy các tham số mặc định của thư viện sklearn cho kết quả chênh lệch rất ít sau quá trình tìm kiếm siêu tham số. Tuy nhiên việc tinh chỉnh tham số đã làm kết quả trên cả tập val và tập test tăng nhẹ



Hình 43: GridSearch tìm siêu tham số n-estimators

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:



Hình 44, 45, 46: Một số kết quả thu được khi sử dụng phương pháp RF

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp Random Forest khá tốt, accuracy rơi vào khoảng 87%, thuộc nhóm các thuật toán có kết quả tốt mà nhóm đã đề xuất. Nguyên nhân có kết quả tốt như vậy bởi thuật toán này đã sử dụng phương pháp Ensemble Learning.

- Tốc độ chạy của Random Forest tương đối chậm, nhưng khắc phục nhược điểm dễ Overfitting của Decision Tree

3.7. XGBoost (Extreme Gradient Boosting)

3.7.1. Cơ sở lý thuyết:

XGBoost là một thuật toán học máy sử dụng phương pháp boosting kết hợp với cây quyết định, nhằm cải thiện độ chính xác và tốc độ dự đoán. Thuật toán này rất mạnh mẽ trong các bài toán phân loại và hồi quy, đặc biệt khi dữ liệu có sự mất cân bằng.

+ Gradient Boosting**:** XGBoost xây dựng các cây quyết định theo cách tuần tự, mỗi cây được thêm vào để cải thiện dự đoán của mô hình hiện tại. Mỗi cây mới học từ lỗi (residual) của các cây trước.
+ Hàm mục tiêu: XGBoost tối ưu hóa hàm mất mát kết hợp với regularization để giảm overfitting và kiểm soát độ phức tạp của mô hình. Regularization giúp hạn chế việc cây quyết định quá khớp với dữ liệu huấn luyện.

Các kỹ thuật quan trọng trong XGBoost:

+ Shrinkage (Learning Rate): XGBoost sử dụng hệ số học tập (η) để điều chỉnh mức độ ảnh hưởng của mỗi cây vào dự đoán cuối cùng. Điều này giúp giảm nguy cơ overfitting.
+ Feature Subsampling: XGBoost chọn ngẫu nhiên một phần của các đặc trưng tại mỗi bước huấn luyện, giúp giảm overfitting và cải thiện khả năng tổng quát của mô hình.
+ Regularization: Sử dụng các thuật toán L1 và L2 để giảm thiểu độ phức tạp của mô hình và tránh overfitting.

Phán đoán:

+ Sau khi huấn luyện, mô hình dự đoán kết quả bằng cách tổng hợp dự đoán từ tất cả các cây quyết định đã học được. Nếu là bài toán phân loại, kết quả được quyết định dựa trên ngưỡng (thường là 0.5), còn nếu là bài toán hồi quy, kết quả là giá trị trung bình của các dự đoán từ các cây.

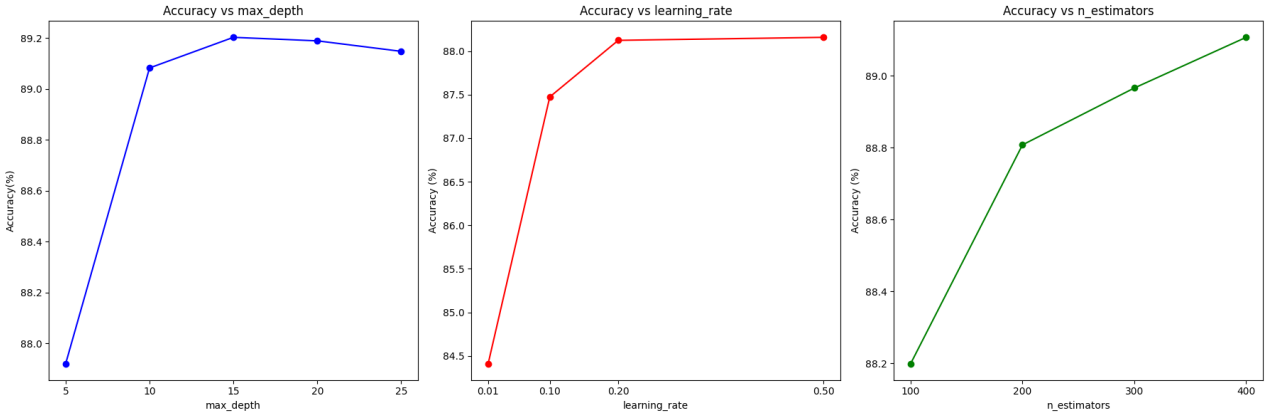
Ưu và Nhược điểm của XGBoost

+ Ưu điểm:
    - Hiệu suất cao: Tốc độ huấn luyện nhanh, xử lý dữ liệu lớn hiệu quả.
    - Regularization: Ngăn ngừa overfitting với L1 và L2.
    - Khả năng xử lý dữ liệu mất cân bằng: Xử lý tốt các bài toán với phân bố dữ liệu không đều.
+ Nhược điểm:
    - Mô hình phức tạp.
    - Yêu cầu tài nguyên lớn: Tốn bộ nhớ và thời gian tính toán khi dữ liệu lớn.

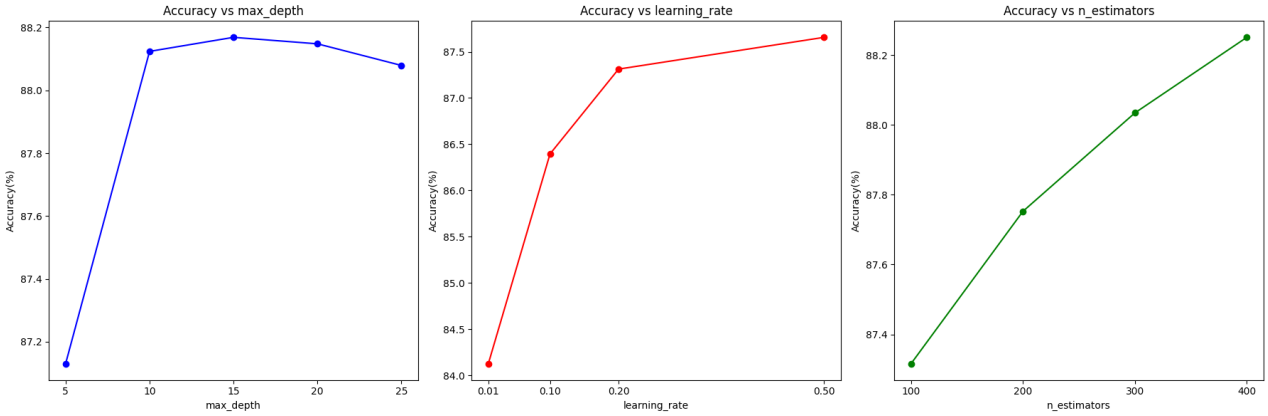
3.7.2. Thực nghiệm

Các siêu tham số cần phải đi tìm cho model:

```
    'max_depth': [5, 10, 15, 20, 25],

    'learning_rate': [0.01, 0.1, 0.2, 0.5],

    'n_estimators': [100, 200, 300, 400]
```

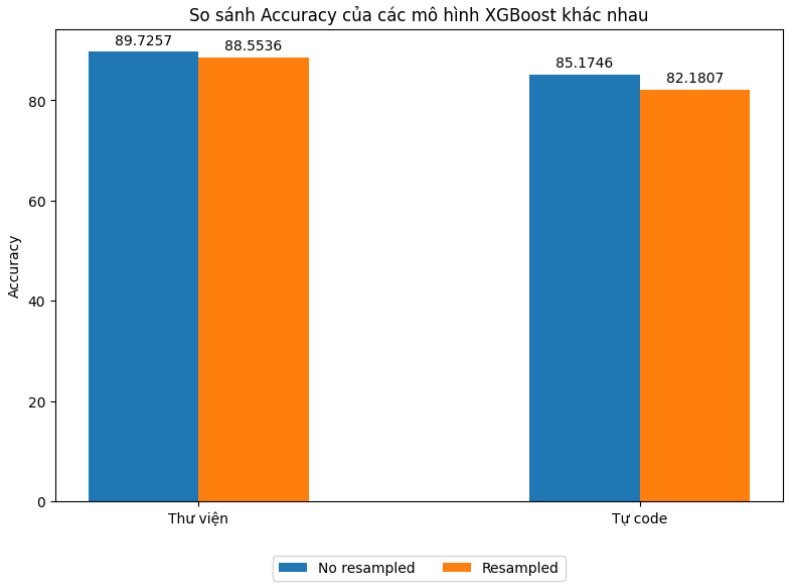


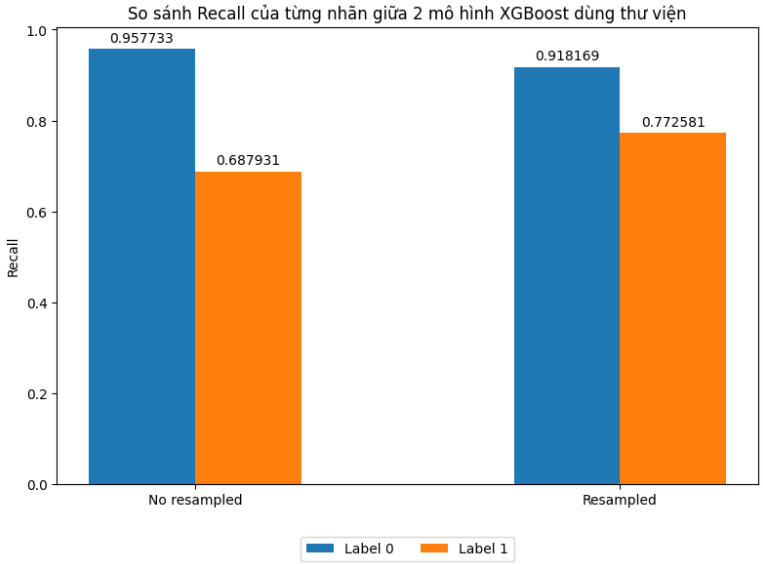
Hình 47: GridSearch với từng tham số lẻ trong tập dữ liệu gốc

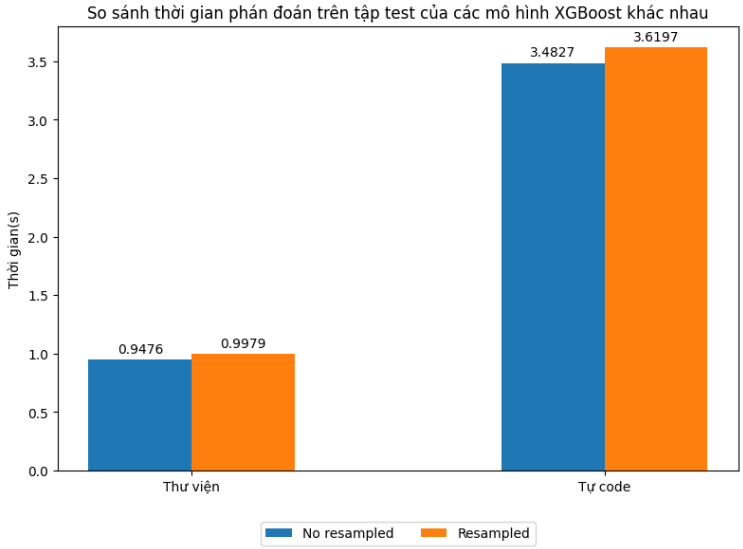


Hình 48: GridSearch với từng tham số lẻ trong tập dữ liệu đã cân bằng

Sử dụng các siêu tham số tốt nhất vừa tìm được đưa vào mô hình, thu được 1 số kết quả:







Hình 49, 50, 51: Một số kết quả thu được khi sử dụng phương pháp XGBoost

Đánh giá:

- Kết quả thu được khi sử dụng phương pháp XGBoost tốt, accuracy rơi vào khoảng 89%, thuộc nhóm các thuật toán có kết quả tốt nhất mà nhóm đã đề xuất. Sở dĩ là do cơ chế Boosting của Ensemble Learning, và huấn luyện trên 1 lượng lớn cây
- Giải quyết tốt vấn đề Overfitting, đây là lựa chọn nên dùng trong các bài toán

3.8. Neural Network

3.8.1. Cơ sở lý thuyết

Mạng nơ-ron (Neural Network) là một mô hình học máy dựa trên cấu trúc mô phỏng cách thức hoạt động của bộ não con người để xử lý thông tin. Mạng nơ-ron bao gồm các đơn vị tính toán được gọi là nơ-ron (hoặc nút), được kết nối với nhau thành các lớp để học và thực hiện các nhiệm vụ như phân loại, hồi quy, nhận diện hình ảnh, và xử lý ngôn ngữ tự nhiên.

Cấu trúc mạng nơ-ron

Mạng nơ-ron được tổ chức theo các lớp(layers), 1 lớp là tập hợp các nơ - ron. Các nơ-ron cùng 1 lớp không liên kết với nhau. Tuy nhiên, nơ-ron của lớp trước và lớp ngay sau thì có sự liên kết để truyền dẫn dữ liệu giữa các lớp.

Một mạng nơ-ron có thể có nhiều lớp, nhưng luôn đảm bảo cấu trúc sau:

Lớp đầu vào (Input layer): là lớp nhận các dữ liệu đầu vào từ bên ngoài và truyền nó đến các lớp tiếp theo.

Lớp đầu ra (Output layer): là lớp đưa kết quả ra ngoài.

Các lớp ẩn (Hidden layers) : là các lớp nằm giữa lớp đầu vào và lớp đầu ra, có nhiệm vụ tính toán để học được đặc trưng của đầu vào.

Quá trình huấn luyện

Quá trình huấn luyện mạng nơ-ron nhằm tìm ra bộ trọng số w cho mỗi nơ-ron để mạng có thể dự đoán chính xác.

Khởi tạo

Các trọng số w của các nơ-ron được khởi tạo giá trị ban đầu. Có nhiều cách để khởi tạo giá trị ban đầu cho bộ trọng số như khởi tạo ngẫu nhiên(khởi tạo từ 1 phân phối ngẫu nhiên như phân phối chuẩn), khởi tạo Xavier,...

Truyền tải dữ liệu (Forward Propagation)

Dữ liệu được đưa vào lớp đầu vào, sử dụng các trọng số hiện tạo và hàm kích hoạt các nơ-ron để tính toán đầu ra và truyền tải qua các lớp sau. Các lớp ẩn cũng thực hiện tương tự. Lớp đầu ra sẽ đưa ra dự đoán.

Tính toán lỗi (loss function)

Sau khi có đầu ra của mạng, ta so sánh giá trị dự đoán với giá trị thực bằng hàm mất mát(loss function)

Cập nhật trọng số

Để cập nhật tham số, nhiều phương pháp lặp dựa trên Gradient: Backpropagation, SGD, Adam, AdaGrad, etc.

Lặp lại quá trình

Lặp lại quá trình huấn luyện đến khi thỏa mãn điều kiện dừng

Quá trình phán đoán

Với một mẫu dữ liệu mới, mẫu sẽ được đưa vào mạng nơ-ron, mạng sẽ sử dụng các trọng số học được để tính toán qua từng lớp và đưa ra phán đoán ở lớp đầu ra.

3.8.2 Thực nghiệm

Trong dự án lần này, nhóm đã thực hiện sử dụng thư viện Keras phục vụ cho cách sử dụng Thư viện, và sử dụng thư viện Torch để tự xây lại quá trình huấn luyện và dự đoán.

Dưới đây là 1 số thông số quan trọng khi sử dụng thư viện Keras

- Kiến trúc mạng:

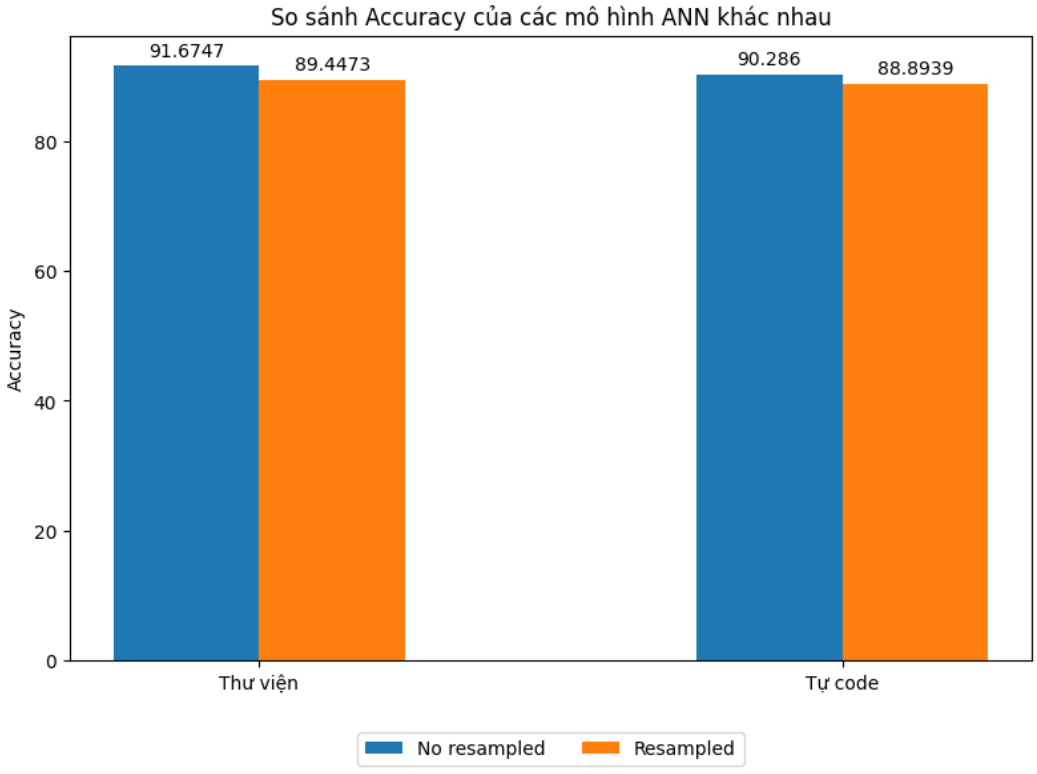
```
Input(shape=[X_train.shape[1]]),
BatchNormalization(),
Dense(128, activation='relu'),
Dropout(0.3),
Dense(96, activation='relu'),
Dropout(0.3),
Dense(64, activation='relu'),
Dense(32, activation='relu'),
Dense(1, activation='sigmoid'),
```

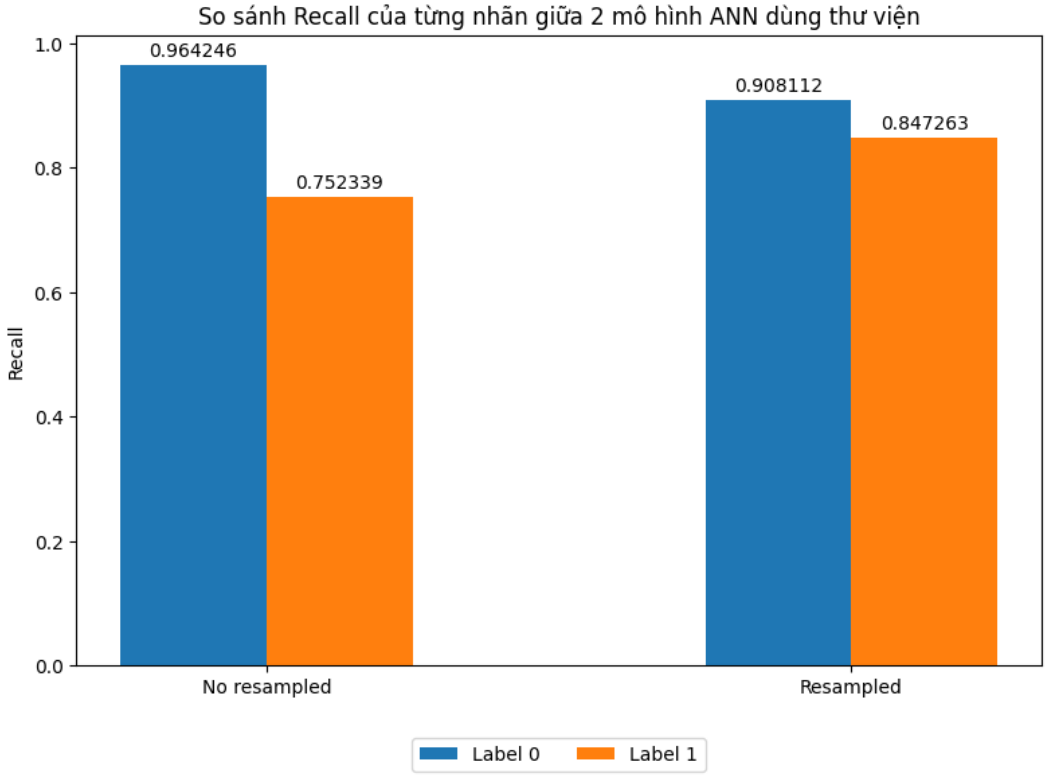
- Cập nhật trọng số: thuật toán Adam, learning_rate=0.0005
- batch_size=100, epochs=200

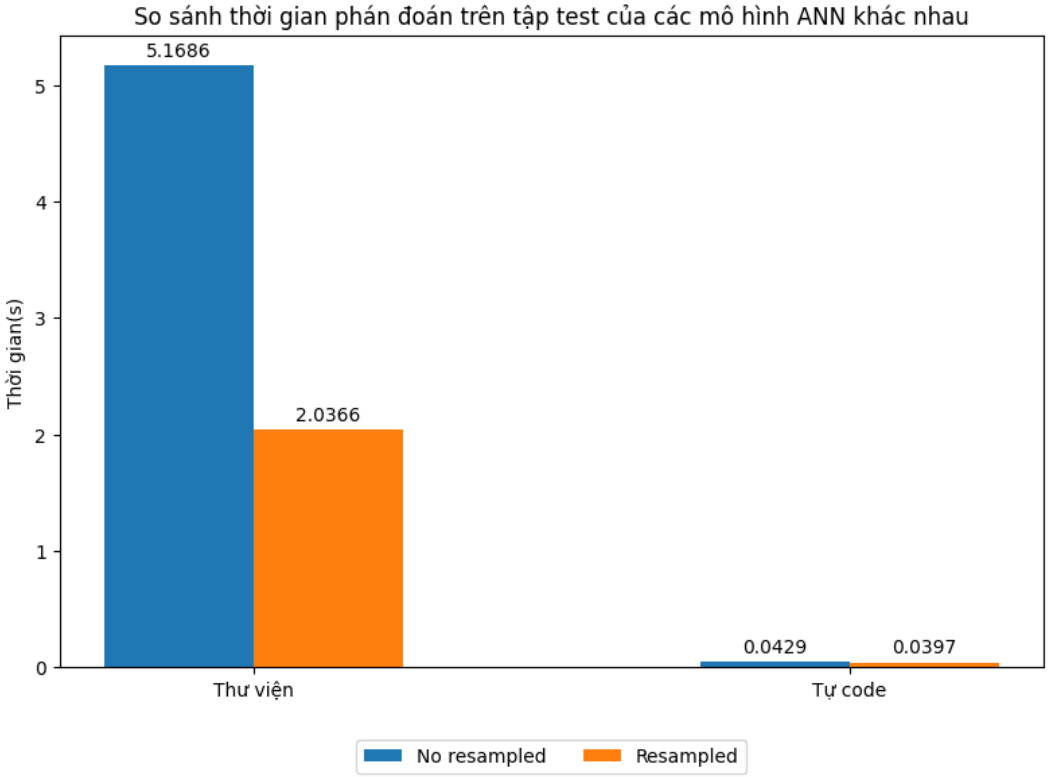
Một số thông số khi tự xây dựng quá trình huấn luyện bằng thư viện torch:

- Kiến trúc mạng: Tương tự phía trên
- Cập nhật trọng số: thuật toán Adam, learning_rate=0.005
- Learning rate decay: gamma=0.99
- Epochs=300

Kết quả thu được:







Hình 52, 53, 54: Một số kết quả thu được khi sử dụng Neural Network

Đánh giá:

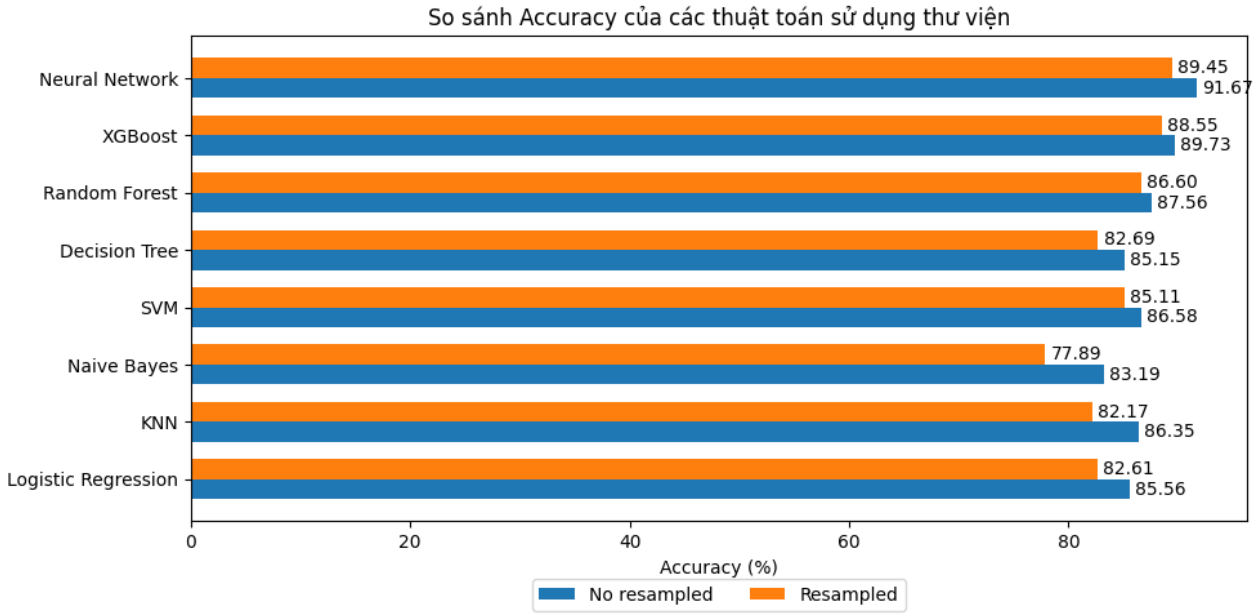
- Kết quả thu được khi sử dụng Neural Network là tốt nhất, accuracy rơi vào khoảng trên 90%, là thuật toán có kết quả tốt nhất mà nhóm đã đề xuất. Điều này là không bất ngờ với sức mạnh của mạng neuron đem lại. Thậm chí nếu tuning thêm các tham số, kết quả còn có thể tốt hơn nữa.
- Đổi ngược lại, kiến trúc mạng neuron là vô cùng phức tạp so với các thuật toán khác, thời gian huấn luyện cũng vô cùng lâu cả khi sử dụng Keras hay Torch. Tuy nhiên mạng neuron là vô cùng quan trọng và có ý nghĩa lớn cho các ứng dụng cao hơn như CV hay NLP. Vì thế đây vẫn là lựa chọn "đáng" khi giải quyết các bài toán.

CHƯƠNG 4: ĐÁNH GIÁ KẾT QUẢ VÀ TỔNG KẾT

4.1. Kết quả thu được

Tổng kết lại, đối với bài toán dự đoán thời tiết (ngày mai có mưa hay không), nhóm đã thử nghiệm các thuật toán học máy phổ biến bao gồm Logistic Regression, KNN, Naive Bayes, SVM, Decision Tree, Random Forest, XGBoost, và Neural Network.

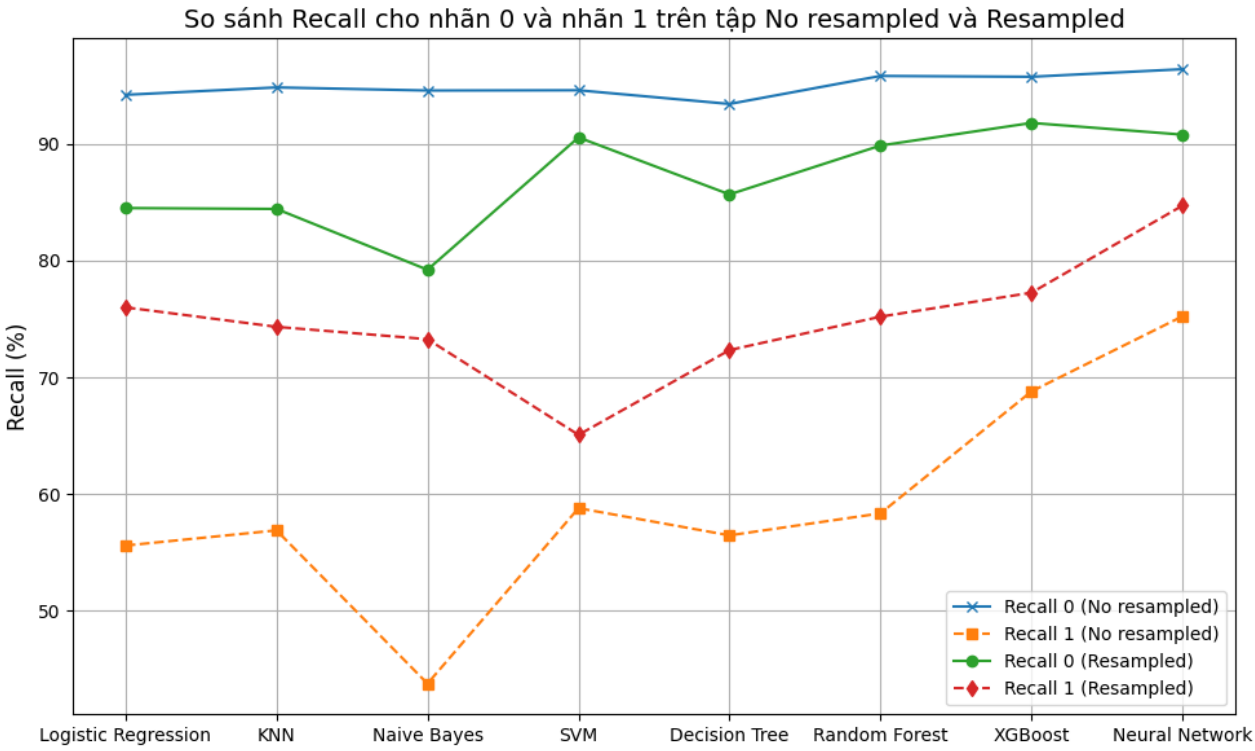
Chúng em lựa chọn các mô hình tốt nhất với từng thuật toán học máy và tiến hành đánh giá dựa trên độ chính xác (accuracy) và thời gian dự đoán trên tập kiểm tra (test set). Dưới đây là các kết quả quan trọng:



Hình 55: So sánh Accuracy của các thuật toán trước và sau resampled

Biểu đồ trên minh họa độ chính xác của các thuật toán. Từ biểu đồ ta có thể rút ra một số nhận xét:

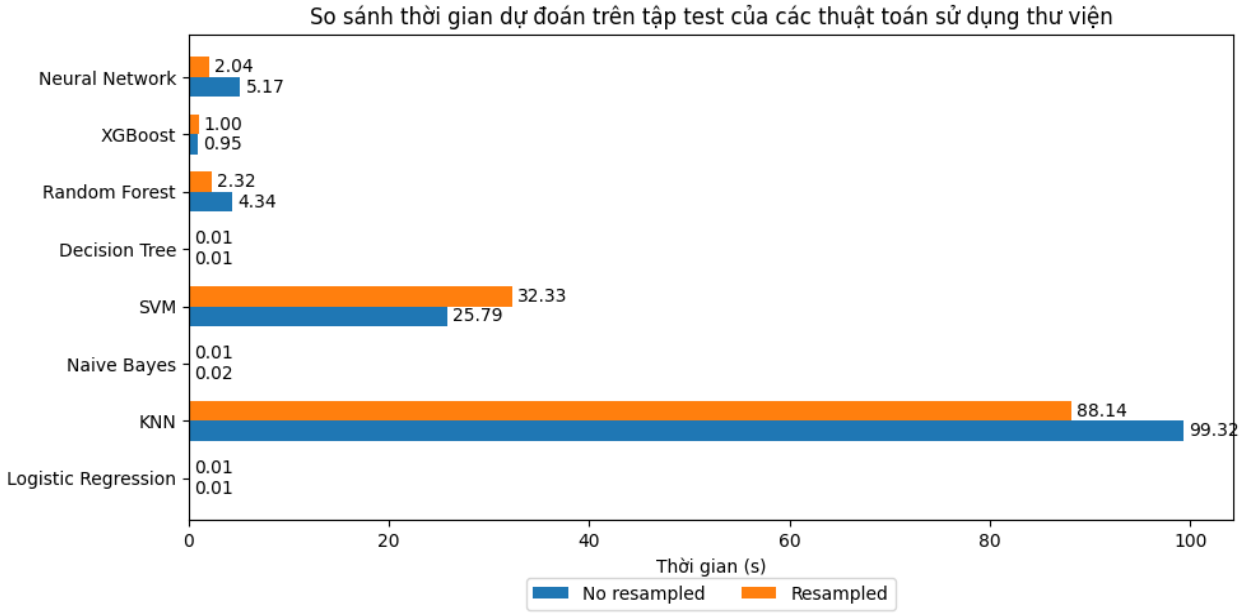
- Neural Network đạt độ chính xác cao nhất, lên đến 91.67%, cho thấy khả năng tổng quát hóa tốt trên tập dữ liệu lớn.
- XGBoost đứng thứ hai với độ chính xác đạt 89.72%, chỉ kém Neural Network một chút, cho thấy khả năng vượt trội của Ensemble Learning.
- Random Forest cũng đạt hiệu suất tốt với độ chính xác là 87.55%, cao hơn so với Logistic Regression (85.56%) và KNN (86.34%).
- Naive Bayes có độ chính xác thấp nhất, chỉ đạt 83.19%, nhưng vẫn chấp nhận được với tốc độ dự đoán nhanh
- Khi sử dụng resampled, độ chính xác bị giảm đi một chút, nguyên nhân là do phân phối của dữ liệu đã bị thay đổi trên tập Train, nhưng không thay đổi trên tập Test (do vẫn giữ nguyên), vì vậy tỉ lệ dự đoán đúng của nhãn đa số (0) đã bị giảm đi. Tuy nhiên bù lại, tổng thể tỉ lệ dự đoán đúng của 2 nhãn lại tăng lên khá tốt. Kết quả thu được nhìn ở Hình 56:



Hình 56: Recall cho 2 nhãn 0 và 1 trước và sau sử dụng Resampled

Có thể thấy Recall giữa 2 nhãn đã thay đổi từ khoảng 95% (Nhãn 0) và 45-55% (Nhãn 1), trở thành 85-90%(Nhãn 0) và 70-80%(Nhãn 1). Sự đánh đổi này được xem là chấp nhận.

Bên cạnh đó, thời gian dự đoán (predict) của các thuật toán cũng được so sánh:



Hình 57: So sánh thời gian phán đoán của các thuật toán

Qua sát thời gian dự đoán trên tập kiểm tra của từng thuật toán ta thấy:

- KNN có thời gian dự đoán lâu nhất, đặc biệt trước khi resampled, thời gian dự đoán đạt 99.31 giây, do thuật toán KNN cần tính toán khoảng cách giữa tất cả các điểm trong tập dữ liệu.

- SVM cũng có thời gian dự đoán khá cao so với các thuật toán khác, 32.32 giây sau khi resampled.
- Các mô hình như Logistic Regression, Naive Bayes, và Decision Tree có thời gian dự đoán nhanh nhất, chỉ mất chưa đến 0.02 giây.

Nhận xét tổng quan:

- Neural Network và XGBoost là các thuật toán có hiệu suất cao nhất trên dữ liệu gốc được tiền xử lý và dữ liệu sau khi resampled. Tuy nhiên, thời gian huấn luyện và dự đoán của Neural Network lớn hơn đáng kể so với XGBoost.
- KNN phù hợp trong bài toán với dữ liệu nhỏ, nhưng không hiệu quả khi số lượng dữ liệu lớn do thời gian dự đoán cao.
- Các thuật toán như Logistic Regression tuy có độ chính xác trung bình nhưng lại vượt trội về mặt thời gian, thích hợp cho các bài toán yêu cầu tốc độ xử lý nhanh.

4.2. Tổng kết

Có thể nói, thông qua dự án này, nhóm chúng em đã ít nhiều học được những kiến thức quan trọng để xây dựng được 1 mô hình thuật toán, nắm rõ hơn được ưu, nhược điểm của các thuật toán quan trọng trong bài toán phân loại. Đồng thời hiểu được cách xây dựng 1 model bằng cả thư viện lẫn tự code.

Trong quá trình làm vẫn còn một số khó khăn mắc phải, cả nhóm đã ngồi lại cùng bàn bạc và thảo luận giải pháp tốt nhất cho đề tài. Kết quả từ 85-90% trên toàn bộ thuật toán được xem là thành công đối với nhóm.

4.3. Hướng phát triển

Dựa trên kết quả trên, một số hướng phát triển có thể thực hiện để cải thiện mô hình trên:

1. Kết hợp các mô hình: Sử dụng ensemble methods để kết hợp Neural Network và XGBoost nhằm cải thiện độ chính xác.
2. Tăng cường dữ liệu: Áp dụng data augmentation để cải thiện khả năng tổng quát hóa của các mô hình.
3. Tối ưu hóa mô hình: Sử dụng các kỹ thuật như pruning cây hoặc giảm chiều dữ liệu để giảm thời gian dự đoán của các mô hình như Random Forest và SVM.

Trong tương lai, chúng em sẽ khắc phục được những nhược điểm cũ và tìm hiểu thêm về các bài toán mới, những bài toán khó và phức tạp hơn ví dụ như các bài toán liên quan đến Học sâu. Dự án lần này chính là nền tảng để cả nhóm có được để đi tiếp sau này.

4.4. Kết thúc

Trước khi kết thúc, chúng em muốn gửi lời cảm ơn sâu sắc đến thầy Thân Quang Khoát, vì đã dạy chúng em môn học khá quan trọng này. Những kiến thức thầy dạy chính là nền tảng cơ bản để chúng em gây dựng lên bài tập lớn lần này. Và chúng em cũng mong nhận được những góp ý và đánh giá của thầy về bài tập lớn lần này để chúng em có thể cải thiện hơn trong tương lai.