ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Dự đoán bệnh di truyền của thai nhi dựa trên dữ liệu bản ghi y tế

TRƯƠNG VĂN HIỂN
hien.tv194276@sis.hust.edu.vn

Ngành Kỹ thuật máy tính

Giảng viên hướng dẫn: TS. Nguyễn Hồng Quang _____

Chữ kí GVHD

Khoa: Kỹ thuật máy tính

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 01/2024

LỜI CẢM ƠN

Đường đời tuổi trẻ, như một cuộc hành trình ngược xuôi sóng lớn, đã đưa tôi đi qua muôn vàn thách thức, nơi mà mỗi bước chân đều là một bài học mới. Thanh xuân của tôi không chỉ là một trang sách ghi lại những kỷ niệm, mà còn là một bức tranh với những nét vẽ độc đáo trên con đường tìm đến sự trưởng thành. Trên hành trình đó, tôi đã lựa chọn Bách Khoa là điểm dừng chân của mình, là nguồn cảm hứng đầy sức sống, nơi tôi tìm thấy bản thân và khám phá đam mê. Nơi đó, tôi vượt qua những thử thách, trải qua bao thăng trầm cuộc sống, và chứng kiến sự trưởng thành của mình từ tuổi 19 đến tuổi 23. Bách Khoa là nơi tôi, một chàng trai trẻ, đắm chìm vào những cơn mưa cuồng nhiệt của thanh xuân.

Những năm tháng sinh viên, dù không dài nhưng cũng đủ để tạo nên những biến động lớn trong tâm hồn và tính cách của tôi. Mặc dù tương lai có trải qua những thách thức đầy sóng gió, mặc dù có nhiều giọt mồ hôi và nước mắt rơi, nhưng tôi tin rằng mình sẽ không bao giờ chấp nhận thất bại. Tôi sẽ luôn giữ nụ cười và bước đi vững vàng trên con đường đầy gian nan, bởi vì một phần của tôi chính là dòng máu Bách Khoa. Cảm ơn Bách Khoa vì đã cho tôi được ở trong một tập thể tuyệt vời, nơi quy tụ những con người tài năng, bản lĩnh nhất Việt Nam. Ngày hôm nay, bước chân ra khỏi cổng parabol, tôi biết sẽ khó có thể tìm được một môi trường nào tuyệt vời hơn thế nữa. Xin tạm biệt và hẹn gặp lại một ngày không xa.

Đồng thời, tôi xin gửi lời cảm ơn chân thành đến TS. Nguyễn Hồng Quang - giảng viên Khoa Kỹ thuật máy tính - Trường Công nghệ Thông tin và Truyền thông, người đã là nguồn động viên không ngừng, luôn tận tâm và hướng dẫn nhiệt tình trong suốt quá trình hiện đồ án tốt nghiệp cử nhân của tôi.

Đồng thời, lời cảm ơn này cũng xin gửi đến những người bạn, anh chị và các em, đặc biệt là gia đình VIT - nơi tôi đã luôn tự hào “Chúng ta là một gia đình”,. Họ là những người đã luôn là nguồn động viên mạnh mẽ, luôn chia sẻ niềm vui, nỗi buồn và luôn sát cánh cùng tôi. Hơn cả đó, tôi xin chân thành cảm ơn bố mẹ và gia đình mình. Họ không chỉ là nguồn động viên, mà còn là bờ vai vững chắc, luôn hỗ trợ và tin tưởng vào tôi. Sự ấm áp và yêu thương từ gia đình là động lực giúp tôi vượt qua mọi khó khăn và hoàn thành đồ án tốt nghiệp một cách trọn vẹn nhất.

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của một sinh viên, đồ án này không thể tránh được những thiếu sót. Tôi rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để tôi có điều kiện bổ sung, nâng cao vốn hiểu biết của mình, phục vụ tốt hơn cho công tác thực tế sau này.

TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong quá trình làm việc và triển khai đồ án tốt nghiệp với TS. Nguyễn Hồng Quang, tôi đã lựa chọn đề tài dự đoán bệnh di truyền của thai nhi dựa trên dữ liệu bản ghi y tế, cụ thể là phát hiện phân loại bệnh Trisomy 21 (Hội chứng down). Hiện nay trong cả lĩnh vực y tế và khoa học máy tính, này đặt ra những thách thức lớn, khiến cho việc đưa ra các phương pháp dự đoán hiệu quả là rất cần thiết do có nhiều hạn chế trong các hướng tiếp cận hiện có và không có mô hình dự đoán mạnh mẽ. Để giải quyết vấn đề này, tôi đã lựa chọn sử dụng các mô hình học máy phổ biến hiện nay như Logistic, XGBoost, LightGBM và Feature Tokenizer Transformer. Lựa chọn này của tôi được đưa ra dựa trên khả năng xử lý và huấn luyện dữ liệu phức tạp, cũng như khả năng tối ưu hóa hiệu suất dự đoán. Đóng góp chính trong Đồ án tốt nghiệp của tôi là việc kết hợp và áp dụng các công nghệ tiên tiến của trí tuệ nhân tạo để có sự so sánh giữa các mô hình học máy, học sâu phổ biến hiện nay cho bài toán dự đoán bệnh di truyền của thai nhi. Kết quả tôi đạt được có sự cải thiện và có thể đưa ra phán đoán ở ngưỡng trung bình về độ chính xác và khả năng dự đoán. Trong tương lai tới, tôi hi vọng có thể tiếp tục phát triển các mô hình mạnh mẽ hơn và chuyên biệt về xử lý các dữ liệu y tế dạng bảng sau khi hoàn thành xong đồ án tốt nghiệp.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

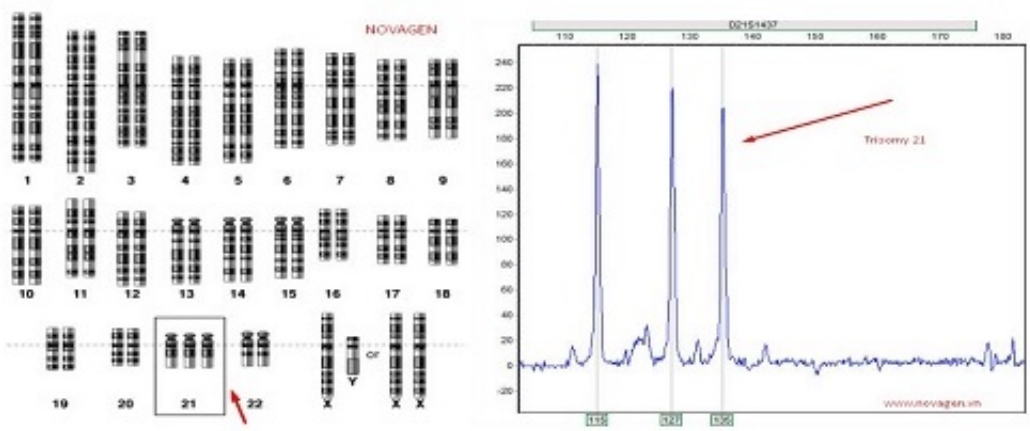
Thuật ngữ	**Ý nghĩa**
AI	Trí tuệ nhân tạo (Artificial Intelligence)
AUC	Area Under the Curve
CLS	Classification Token
DL	Học sâu (Deep Learning)
FN	False Negative
FP	False Positive
FPR	Flase Positive Rate
FTT	Feature Tokenizer Transformer
KNN	Học dựa trên K láng giềng gần nhất (K-Nearest Neighbors)
LightGBM	Light Gradient-Boosting Machine
ML	Học máy (Machine Learning)
NST	Nhiễm sắc thể (Chromosome)
ROC	Receiver Operating Characteristic
SMOTE	Lấy mẫu tổng hợp quá mức (Synthetic Minority Over-sampling)
TN	True Negative
TP	True Positive
TPR	True Positive Rate
XGBoost	Extreme Gradient Boosting

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề và tổng quan đề tài

Trisomy là hiện tượng bộ gen có ba nhiễm sắc thể, thay vì ở trạng thái người bình thường là một cặp nhiễm sắc thể. Bình thường thai nhi được thừa hưởng vật chất di truyền gồm có 46 nhiễm sắc thể (23 NST từ mẹ, 23 NST từ bố) tuy nhiên ở các trường hợp hội chứng trisomy, thai nhi có 47 nhiễm sắc thể do thừa một nhiễm sắc thể. Sự dư thừa vật chất di truyền này chính là nguyên nhân gây dị tật bẩm sinh ở trẻ. Phổ biến nhất là hội chứng Down (trisomy 21), tiếp sau đó là Edwards (trisomy 18), Patau (trisomy 13).

Hội chứng Down là bất thường NST thường gặp nhất ở người. Ở Trung Quốc, với phương pháp truyền thống và sự hỗ trợ của các bác sĩ, sàng lọc nguy cơ mang thai down được thực hiện dựa trên sinh hóa huyết thanh của người mẹ và siêu âm sự phát triển của thai nhi. Tuy nhiên, hiệu suất của các thử nghiệm này đạt khoảng 5% dương tính giả (false positive rate), và tỉ lệ dự đoán đúng chỉ $60 - 65\%$ [1]. Do vậy cần có các hệ thống tự động cho phép xác định khả năng trisomy với độ chính xác cao.



Hình 1.1: Trisomy21 - Hội chứng down [2]

Ở Việt Nam mỗi năm có khoảng hơn 40000 trường hợp trẻ sinh ra bị dị tật bẩm sinh, trong số đó có tới 2000 trường hợp mắc bệnh liên quan đến rối loạn di truyền trisomy. Hiện nay để phát hiện nguy cơ mắc các hội chứng trên, xét nghiệm sàng lọc và chẩn đoán trước sinh cho phép xác định thai nhi mắc lệch bội nhiễm sắc thể sớm ngay trong quý 1 và 2 của thai kì. Chẩn đoán sớm tạo điều kiện tiếp cận chăm sóc y tế bao gồm kế hoạch điều trị và can thiệp, chăm sóc sau sinh sớm, dịch vụ hỗ trợ và tư vấn cho các gia đình. Các phương pháp sàng lọc được áp dụng phổ biến như siêu âm, sinh hoá máu mẹ (Double test, Triple test), đều là các phương pháp

ít xâm lấn. Trong khi để chẩn đoán xác định sớm phải dùng các phương pháp xâm lấn như: chọc ối, sinh thiết gai rau,... Do đó, cần thiết phải cải tiến các phương pháp sàng lọc hiện tại để hạn chế những can thiệp không cần thiết.

Hiện nay, học máy và trí tuệ nhân tạo là những lĩnh vực phát triển cao trong lĩnh vực y tế, kết hợp khoa học máy tính và thống kê cho các vấn đề y học. Sự phát triển của trí tuệ nhân tạo đã mở ra nhiều cơ hội cho sự phát triển và cải tiến các công cụ sàng lọc trước sinh. Nhiều nhà khoa học trên thế giới đang tập trung, nghiên cứu ứng dụng trí tuệ nhân tạo trong sàng lọc trước sinh và bước đầu đã đạt được những kết quả và thành công nhất định. Tuy nhiên, ở Việt Nam, các mô hình học máy và học sâu chưa được ứng dụng trong sàng lọc trước sinh mà sử dụng phổ biến các phương pháp như siêu âm, Double test, Triple test... Vì vậy, việc sử dụng các mô hình trong tầm soát trước sinh trong nghiên cứu tại Việt Nam là cần thiết để nâng cao hiệu quả lâm sàng của việc tầm soát thai sản và các bất thường bẩm sinh.

Trong phạm vi đề tài đồ án này, tôi đề xuất các thuật toán học sâu để tầm soát chẩn đoán bệnh trisomy dựa trên các xét nghiệm trên các thai phụ đang mang thai ở Việt Nam bao gồm các tham số cơ bản của thai phụ như lứa tuổi, tuổi thai nhi, kết quả siêu âm, kết quả xét nghiệm Double test, Triple test.

1.2 Mục tiêu đề tài

Trong quá trình bắt đầu triển khai thực hiện đề tài "Dự đoán bệnh di truyền của thai nhi dựa trên dữ liệu bản ghi y tế", tôi hướng đồ án tới những mục tiêu sau:

1. Kiến thức thu thập được: Tìm hiểu làm quen với bài toán trong lĩnh vực Tin sinh học; Phương pháp tổng hợp, thu thập và thống kê dữ liệu; Các phương pháp xử lý dữ liệu dạng bảng; và Kiến trúc, hoạt động, phương pháp đánh giá của các mô hình học máy, học sâu trong lĩnh vực Medtech.
2. Công nghệ: Ngôn ngữ lập trình Python; Thư viện, công cụ hỗ trợ lập trình cho mô hình AI và DL như numpy, pandas, sklearn, torch...
3. Kỹ năng: Kỹ năng quản lý, phân bổ thời gian hợp lý; Kỹ năng giao tiếp, trao đổi vấn đề, đề nghị; Kỹ năng nghiên cứu, tìm kiếm, giải quyết vấn đề.
4. Sản phẩm kỳ vọng: Bộ dữ liệu chuẩn để dự đoán bệnh di truyền; Mô hình được triển khai dự đoán bệnh di truyền dựa vào dữ liệu bản ghi y tế; Báo cáo triển khai công việc và tổng hợp kết quả đồ án tốt nghiệp.
5. Vấn đề thực tiễn đồ án giải quyết: Ứng dụng công nghệ hiện của trí tuệ nhân tạo để thực hiện chăm sóc y tế, cung cấp giải pháp thiết thực cho cộng đồng về lĩnh vực di truyền học; Triển khai các mô hình để có thể dự đoán chính xác các bệnh di truyền trong khoảng thời gian thai kì sớm nhất.

1.3 Nội dung thực hiện và kế hoạch triển khai

Trong quá trình thực hiện đồ án trong học kì 2023.1, tôi đã triển khai đề tài theo các nội dung như sau:(i) Tìm hiểu tổng quan bài toán và lĩnh vực; (ii) Tìm hiểu công nghệ liên quan và công cụ hỗ trợ; (iii) Thu thập và xử lý dữ liệu; (iv) Xây dựng chương trình và đánh giá mô hình; (v) Thử nghiệm và đánh giá.

1.4 Đối tượng thực hiện và định hướng giải pháp

1.4.1 Đối tượng thực hiện

Đối tượng mà đề tài hướng đến là các hồ sơ bệnh án của thai phụ đến khám tại Bệnh viện Phụ sản Trung ương (Việt Nam). Bộ dữ liệu bao gồm các thông tin về thai phụ, các kết quả xét nghiệm siêu âm, kết quả sàng lọc sinh hoá (Double test, Triple test) và kết quả chẩn đoán thai nhi.

1.4.2 Định hướng giải pháp

Với bộ dữ liệu được cung cấp, tôi dự định sử dụng các mô hình học máy phổ biến mạnh mẽ hiện nay cho bài toán phân loại như: XGBoost, LightGBM, Logistic Regression để tính toán ra kết quả cho từng mô hình.

Tuy nhiên, bộ dữ liệu được cung cấp là dữ liệu dạng bảng (Tabular Data) và Deep Learning không phải lúc nào cũng hoạt động tốt với dữ liệu dạng này với một số khó khăn như dữ liệu bị nhiễu hoặc khuyết, khó áp dụng Transfer Learning... Gần đây trong lĩnh vực Deep Learning, các nhà nghiên cứu Yury Gorishniy và Ivan Rubachev đã thực hiện tổng quan về các kiến trúc cho dữ liệu dạng bảng. Đầu tiên là kiến trúc giống ResNet và mô hình thứ hai là sự điều chỉnh đơn giản về kiến trúc Transformer cho dữ liệu dạng bảng (FTT - Feature Tokenizer Transformer), vượt trội hơn các giải pháp khác trong hầu hết các tác vụ.

Do đó, trong quá trình thực hiện đồ án này, tôi tập trung triển khai kiến trúc mô hình FTT trên bộ dữ liệu được cung cấp. Sau đó thực hiện đánh giá mô hình; nhận ý kiến đánh giá, phản hồi từ giảng viên và bên nghiệp vụ y học; và cuối cùng là hoàn thiện báo cáo tổng kết quá trình thực hiện đồ án tốt nghiệp.

1.5 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau: Chương 2 giới thiệu về các nghiên cứu và thành tựu liên quan đến bài toán mà đồ án đang thực hiện. Chương 3 trình bày ngắn gọn về nền tảng lý thuyết và các kỹ thuật, phương pháp sử dụng để triển khai mô hình. Chương 4 mô tả tổng quan về luồng hoạt động. Chương 5 trình bày chi tiết về các phương pháp và mô hình tôi triển khai cùng với phân tích và đánh giá các kết quả thu được. Và cuối cùng, trong chương 6 tôi tổng kết lại thành các kết luận và đưa ra định hướng phát triển của đồ án trong tương lai.

CHƯƠNG 2. CÁC NGHIÊN CỨU VÀ THÀNH TỰU LIÊN QUAN

Trong những năm gần đây cũng đã có những nghiên cứu đề xuất các hệ thống tầm soát bệnh trisomy. Chẳng hạn như Amir Jamshidnezhad và cộng sự [3] sử dụng mô hình GA-ANN (Genetic Algorithm–Artificial neural network) để chẩn đoán trisomy 21 (nguyên nhân gây ra bệnh down ở trẻ). Tập dữ liệu gồm 308 thai phụ trong đó 286 trường hợp bình thường và 22 trường hợp mang bệnh. Các tác giả đã lựa chọn tham số đầu vào mô hình bao gồm tuổi thai phụ, cân nặng thai phụ, chiều dài đầu mông thai nhi, PAPP-A. Độ chính xác, độ nhạy và độ đặc hiệu lần lượt là $96.32\%, 90.91\%, 99.72\%$. Nhìn chung, giải thuật GA đã hỗ trợ trong việc thiết kế mạng ANN rất hiệu quả và đạt được độ chính xác cao. Tuy nhiên một nhược điểm chính của nghiên cứu đó là bộ dữ liệu quá mất cân bằng cũng như số mẫu bị bệnh khá nhỏ.

A. C. Neocleous và cộng sự [4] đề xuất phương pháp sử dụng mạng nơ ron nhân tạo (ANN: Artificial Neural Network) với dữ liệu các xét nghiệm tại tuần tuổi 11-13 của thai nhi, nhằm xác định các nguy cơ mắc chứng đa bội, thể tam nhiễm 21, OCA và các thể dị bội nhiễm sắc thể ở thai nhi. Các tác giả sử dụng bộ dữ liệu gồm 51208 trường hợp thai phụ mang thai đơn, đã được sàng lọc để tìm ra thể dị bội. Mô hình sử dụng 9 thuộc tính đầu vào bao gồm tuổi của thai phụ, trisomy 21 ở lần mang thai trước, độ mờ da gáy, β-hCG MoM, Pap-A MoM, chiều dài đầu mông, xương mũi, dòng chảy van ba lá và dòng chảy tĩnh mạch chủ. Phương pháp này bao gồm hai mô hình, trong đó mô hình 1 được sử dụng để phân biệt thể dị bội và không dị bội, mô hình 2 phân biệt trisomy 21 với OCA. Kết quả cho thấy mô hình phân tích đúng tất cả trường hợp trisomy 21 và $96,1\%$ của thể dị bội với 3.9% dương tính giả.

He, Falin và cộng sự [1] đã sử dụng mô hình rừng ngẫu nhiên (RF: Random Forest) để dự đoán khả năng thai nhi mắc bệnh down. Nghiên cứu này được thực hiện với tập dữ liệu bao gồm 58972 phụ nữ mang thai trong đó có 49 ca thai nhi mắc hội chứng down. Các thuộc tính đầu vào cho mô hình bao gồm các kết quả xét nghiệm Double test và Triple test (nồng độ của AFP, uE3 and β-hCG) cũng như tuổi, cân nặng của thai phụ và số tuần tuổi của thai. Đồng thời các tác giả cũng sử dụng 1 bộ dữ liệu bệnh viện khác gồm 27170 phụ nữ mang thai để kiểm tra mô hình. Kết quả thử nghiệm mô hình cho thấy 66.7% độ chính xác, với tỷ lệ dương tính giả 5% với bộ dữ liệu huấn luyện. Với bộ dữ liệu tại bệnh viện khác, mô hình đạt 85.2% độ chính xác và 5% dương tính giả.

Catic, A. và cộng sự [5] đã sử dụng mạng nơ ron nhân tạo trong chẩn đoán phát

hiện sớm hội chứng turner, klinefelter, patau, edwards và down. Dữ liệu được sử dụng gồm 2500 thai phụ, với các dữ liệu được thu thập từ xét nghiệm Double test như PAPP-A, β-hCG. Đồng thời các thuộc tính như độ mờ da gáy, sự xuất hiện của xương mũi lấy từ kết quả siêu âm cũng được sử dụng. Kết quả đạt được độ chính xác trung bình là 89.6% và độ nhạy là 92.00%.

Durmuşoğlu và cộng sự [6] đã đưa ra phương pháp phân loại bệnh down gồm các thông số xét nghiệm Tripple test bao gồm 3 tham số UE3, AFP và HCG trên 81 bệnh nhân. Kỹ thuật SMOTE được sử dụng để khắc phục vấn đề mất cân bằng dữ liệu. Nhiều mô hình được sử dụng như: ZeroR, k-NN, Bayesian network, C4.5, Logistic Regresstion, MLP (ANN), SMO, FLDA... trong đó mô hình MLP đạt độ chính xác cao nhất lên đến 90%. Tuy nhiên nhược điểm của nghiên cứu là tập dữ liệu quá nhỏ khiến mô hình có thể không mang tính xác thực cao.

Ramanathan và cộng sự [7] đề xuất phương pháp xác định thai nhi bị bệnh down gồm các thông số xét nghiệm tiền sản. Dữ liệu được tăng cường bằng phương pháp ADASYN và SMOTE, kết quả cho thấy ADASYN chiếm ưu thế trong việc cân bằng dữ liệu. Các tác giả sử dụng 3 mô hình học máy bao gồm Random Forest, Naïve bayse và ANN trong đó mô hình Naïve bayse cho kết quả cao nhất với độ chính xác là 92%.

Zhang, Hong-Guo và cộng sự [8] đã đề xuất phương pháp tầm soát bệnh down với bộ dữ liệu gồm các thông số xét nghiệm Double test, Tripple test và siêu âm. Vấn đề mất cân bằng dữ liệu được xử lý bằng các phương pháp SMOTE, ADASYN, EasyEnsemble. Nghiên cứu thử nghiệm các mô hình học máy SVM, CART, ADABOOST. Kết quả cho thấy kết hợp giữa mô hình SVM với phương pháp tăng cường dữ liệu EasyEnsemble đạt độ nhạy cao nhất lên đến 99%. Có thể thấy các thuật toán tăng thêm dữ liệu góp phần rất quan trọng đến độ chính xác của các mô hình phân loại nhất là đối với dữ liệu y tế.

Neuocleous và các cộng sự [9] đề xuất sử dụng 2 mô hình ANN. Cụ thể, ở mô hình 1 các tác giả sử dụng tuổi thai phụ, free β-hCG, PAPP-A, độ mờ da gáy NT để xác định nguy cơ mắc bệnh trisomy 21. Những trường hợp nguy cơ cao thì mô hình 2 sẽ được sử dụng để phân thành 3 loại gồm không có nguy cơ, nguy cơ trung bình, nguy cơ cao trong đó dữ liệu đưa vào mô hình 2 có 7 thuộc tính gồm 4 thuộc tính cũ và thêm 3 thuộc tính gồm xương mũi, dòng chảy tĩnh mạch chủ và dòng chảy van ba lá. Tỷ lệ phát hiện đúng đạt được lên tới 95%.

Marttala và cộng sự [10] đã sử dụng phương pháp thống kê trên 56076 thai phụ ở khu vực Bắc Phần Lan để phát hiện các căn bệnh trisomy nhiễm sắc thể 21, 18, 13. Các tham số thống kê bao gồm tuổi thai phụ, NT, free β-hCG, PAPP-A,... Tỷ

lệ cut-off ở trisomy 21 là 1/250 trong khi đó trisomy 18 và 13 là 1/200. Ngưỡng của free β-hCG, PAPP-A là 0.2ng/ml và 5mU/L. Kết quả cho thấy tỷ lệ dương tính giả của trisomy 21 là dưới 5% trong khi trisomy 18 và trisomy 13 đều dưới 3%.

Khattak Momina và cộng sự [11] đã dùng phương pháp thống kê để nhận dạng và chẩn đoán bệnh down (trisomy21) và dị tật ống thần kinh. Dữ liệu đầu vào được thu thập từ 1989-2004 với các chỉ số chính là AFP, UE3, β-hCG, NT, ... Mô hình Logistics Regresstion được sử dụng cho cả 2 vấn đề phân loại. Tuy vậy, độ nhạy trên tập validation chỉ đạt khoảng $75 - 80\%$.

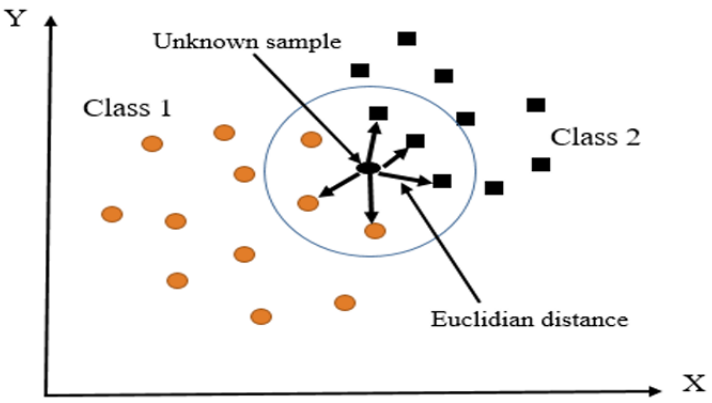
CHƯƠNG 3. NỀN TẢNG LÝ THUYẾT

3.1 Phương pháp xử lý dữ liệu bị khuyết (missing data) với KNNImputation

kNN là một thuật toán học có giám sát đơn giản nhất trong học máy. Khi huấn luyện, thuật toán này không học một điều gì từ dữ liệu huấn luyện (lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.

Ý tưởng của thuật toán là tìm những điểm lân cận của điểm thiếu dữ liệu, sau đó lấy trung bình giá trị của những điểm lân cận để điền vào dữ liệu bị thiếu. Trong quá trình thực hiện đồ án, để giải quyết vấn đề khuyết dữ liệu, tôi đã hướng tới sử dụng một mô hình để dự đoán giá trị khuyết đó, dựa vào những giá trị tồn tại trong tập dữ liệu.

Về lý thuyết, tôi có thể sử dụng bất kỳ thuật toán ML Classification/Regression nào để thực hiện việc Imputation cho dữ liệu khuyết, nhưng thực tế chứng minh rằng kNN mang lại hiệu quả tốt hơn, cả về khía cạnh độ chính xác và mức độ phức tạp khi thực hiện.



Hình 3.1: Minh hoạ thuật toán KNNImputation [12]

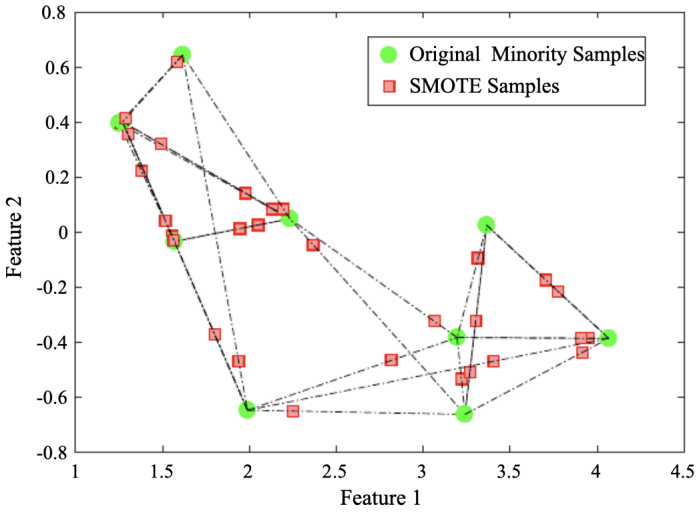
3.2 Phương pháp khắc phục mất cân bằng dữ liệu với thuật toán SMOTE

Phần lớn các trường hợp trong tập dữ liệu được cung cấp là người bình thường, điều này đã dẫn tới sự mất cân bằng nghiêm trọng giữa hai lớp: thai nhi bình thường và thai nhi mắc trisomy 21. Mất cân bằng dữ liệu là hiện tượng số mẫu của một lớp có sự chênh lệch rất lớn với các lớp còn lại. Trong lĩnh vực y tế, những dữ liệu bất thường thường xuất hiện ít hơn những dữ liệu bình thường, do đó dẫn tới vấn đề mất cân bằng dữ liệu, và trong bài toán này, tôi cần dự đoán khả năng liệu thai nhi có bị mắc trisomy 21 - lớp thiểu số trong dữ liệu.

Đề khắc phục sự mất cân bằng dữ liệu trisomy, tôi đã sử dụng kỹ thuật lấy lại

mẫu dữ liệu bằng phương pháp SMOTE. Ý tưởng chung của SMOTE là tạo ra dữ liệu tổng hợp giữa mỗi mẫu của lớp thiểu số từ k hàng xóm gần nhất của nó. Có nghĩa là, đối với mỗi một trong các mẫu của lớp thiểu số, k các láng giềng gần nhất của nó được chọn. Sau đó giữa các cặp điểm được tạo bởi mẫu và từng láng giềng của nó để nội suy ra dữ liệu cho các mẫu mới.

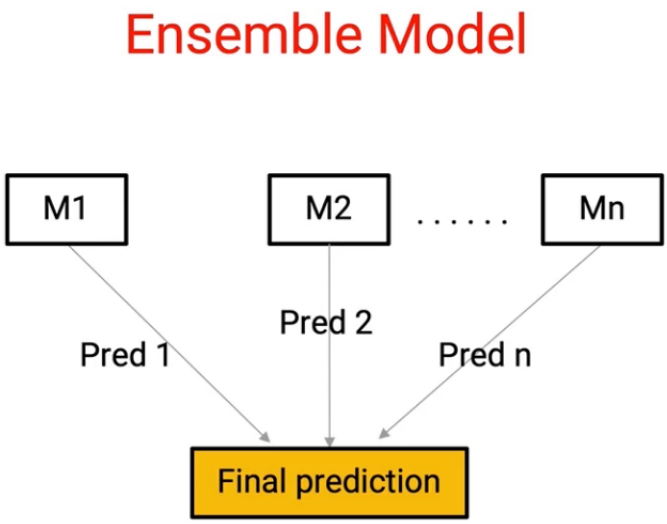
Thuật toán SMOTE:



Hình 3.2: Minh hoạ cơ chế nội suy của SMOTE [13]

3.3 Ensemble Learning

Ensemble Learning là phương pháp kết hợp kết quả của các mô hình yếu để tạo ra một mô hình dự đoán mạnh mẽ hơn.

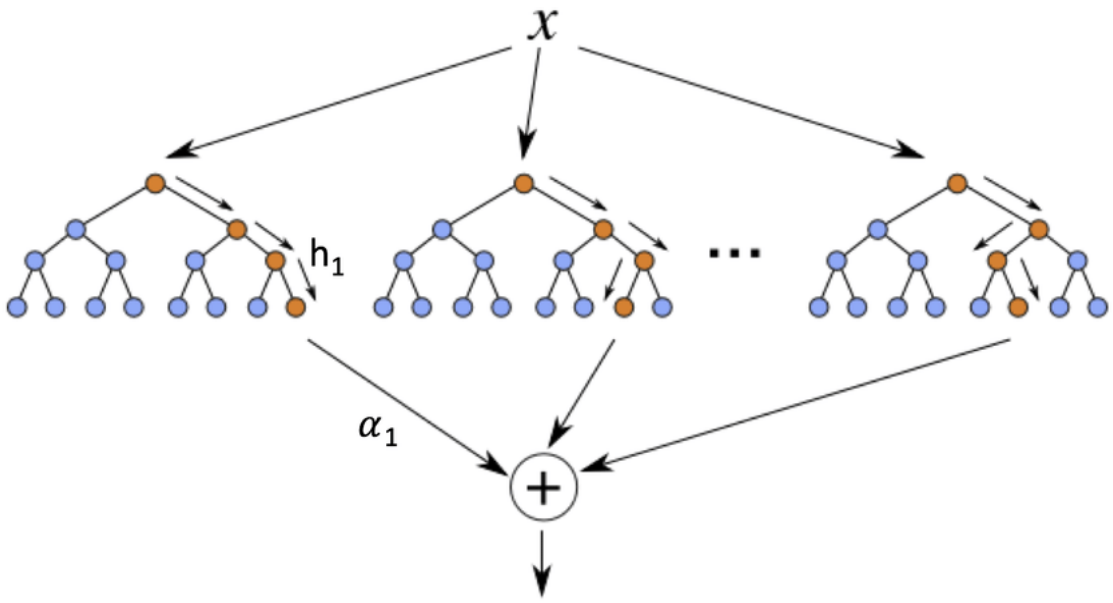


Hình 3.3: Minh hoạt thuật toán Ensemble Learning [14]

3.4 Thuật toán XGBoost

XGBoost là một thuật toán Ensemble Learning sử dụng kỹ thuật Boosting. Một trong những hiệu quả được chú ý của XGBoost là xử lý hiệu quả các giá trị bị khuyết mà không cần trải qua quá nhiều bước tiền xử lý dữ liệu. Ngoài ra, XGBoost có hỗ trợ tích hợp để xử lý song song, giúp huấn luyện các mô hình với dữ liệu lớn trong một khoảng thời gian hợp lý.

3.4.1 Tập hợp các cây quyết định (Decision Tree Ensembles)



Hình 3.4: Decision Tree Ensembles [15]

Ta có dự đoán đầu ra với mô hình Decision Tree Ensembles: $\hat{y} = \sum_{k=1}^{K} f_k(x_i)$

Trong đó:

K: số lượng cây quyết định

$f_k \in F$: một hàm trong không gian F

F: tập hợp tất cả các trường hợp có thể có

3.4.2 Hàm mất mát của XGBoost

Ta có hàm mất mát của XGBoost:

$$\Rightarrow \mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

Trong đó:

l: một hàm mất mát khả vi đo sự sai lệch giữa $\hat{y}_i$ và y_i

Cây $f_t(x) = w_{q(x)}$

$w \in \mathbb{R}^T$: vector trọng số trên lá

$q : \mathbb{R}^d \to \{1, 2, ..., T\}$ là hàm ánh xạ từng điểm dữ liệu tới lá tương ứng

T: số lượng lá trên cây

$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$: độ phức tạp của cây f_t

Nhận xét: Hàm mất mát của XGBoost là một tập các hàm và không thể tối ưu hóa bằng các phương pháp tối ưu hóa truyền thống trong không gian Euclide.

Sử dụng công thức lấy xấp xỉ Taylor bậc hai:

$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2$

$\Rightarrow \mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \Omega(f_t)$

Trong đó: $g_i = \partial_{\hat{y}_i^{t-1}} l\left(y_i, \hat{y}_i^{t-1}\right)$ và $h_i = \partial^2_{\hat{y}_i^{t-1}} l\left(y_i, \hat{y}_i^{t-1}\right)$

Bằng cách sử dụng công thức Taylor, ta đã chuyển đổi hàm mất mát ban đầu thành một hàm trong không gian Euclide, để có thể sử dụng các kỹ thuật tối ưu hóa truyền thống. Sau đó, ta có thể loại bỏ các hằng số để tối ưu hoá ở bước t:

$$\begin{aligned}
\tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \\
&= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T
\end{aligned}$$

Trong đó:

$I_j = \{i | q(x_i) = j\}$: tập index của các điểm dữ liệu được gán cho lá j trong cây

$G_j = \sum_{i \in I_j} g_i$ và $H_j = \sum_{i \in I_j} h_i$

Trong phương trình bên trên, ta thấy các w_j đều độc lập với nhau, thành phần $G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2$ là dạng bậc hai nên giá trị tối ưu dễ dàng nhận được là:

$$w_j^* = -\frac{G_j}{H_j+\lambda} \text{ và } \mathcal{L}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

Trong thực tế, để bắt đầu xây dựng quá trình huấn luyện, ta thường:

- Bước 1: Bắt đầy khởi tạo cây với một nút gốc (single root) chứa tất cả dữ liệu huấn luyện.
- Bước 2: Lặp lại từng giá trị của mỗi thuộc tính. Đánh giá từng mức giảm lỗi và tối ưu hóa từng cấp độ của cây một. Cụ thể, ta cố gắng chia một chiếc lá thành hai lá con (left leaf, right leaf).

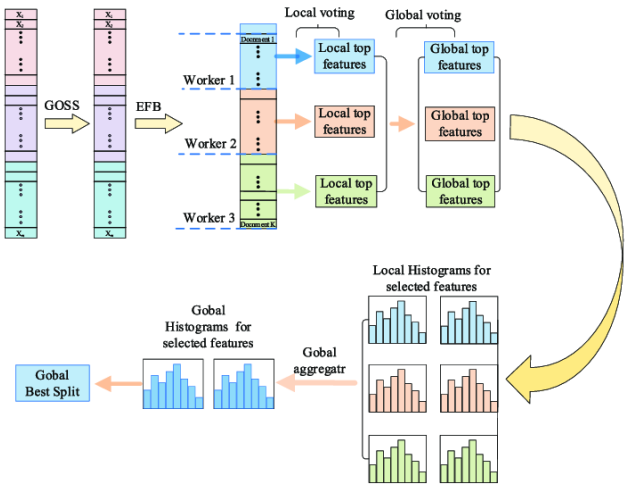
$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

- Bước 3: Nếu giá trị $Gain$ nhỏ hơn γ thì không nên thêm nhánh đó. Đây là kỹ thuật cắt tỉa trong các mô hình học dựa trên cây quyết định.

Thuật toán trên được gọi là "Thuật toán tham lam chính xác" (Exact Greedy Algorithm) và độ phức tạp của nó là $O(n * m)$ trong đó n là số lượng dữ liệu huấn luyện và m là số lượng thuộc tính.

3.5 Thuật toán LightGBM

LightGBM (Light Gradient Boosting Machine) là một framework tăng cường gradient (gradient boosting) sử dụng cây quyết định giúp tăng hiệu quả của mô hình và tối thiểu hoá mức sử dụng bộ nhớ. LightGBM sử dụng 2 kỹ thuật là: Lấy mẫu một phía bằng gradient (Gradient-based One Side Sampling - GOSS), và Gói thuộc tính độc quyền (Exclusive Feature Bundling - EFB).



Hình 3.5: Sơ đồ minh hoạ thuật toán LightGBM [16]

3.5.1 Gradient-based One Side Sampling – GOSS

Các dữ liệu khác nhau sẽ có vai trò khác nhau trong việc tính giá trị Information Gain. Các dữ liệu có gradient lớn hơn (tức là các dữ liệu chưa được huấn luyện) sẽ có ý nghĩa nhiều hơn cho giá trị Information Gain. GOSS giữ các dữ liệu đó với gradient lớn và thực hiện lấy mẫu ngẫu nhiên các dữ liệu có gradient nhỏ để giữ lại độ chính xác của giá trị Information Gain.

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models ← {}, fact ← $\frac{1-a}{b}$
topN ← a × len(I) , randN ← b × len(I)
for $i = 1$ ***to*** d **do**
  preds ← models.predict(I)
  g ← $loss$(I, preds), w ← {1,1,...}
  sorted ← GetSortedIndices(abs(g))
  topSet ← sorted[1:topN]
  randSet ← RandomPick(sorted[topN:len(I)], randN)
  usedSet ← topSet + randSet
  w[randSet] × = fact ▷ Assign weight $fact$ to the small gradient data.
  newModel ← L(I[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

Hình 3.6: Thuật toán Gradient-based One Side Sampling [17]

GOSS được sử dụng trong việc tăng cường gradient trên tập huấn luyện với n mẫu $\{x_1, \ldots, x_n\}$, trong đó mỗi mẫu x_i là một vectơ có kích thước s trong không gian X^s. Trong mỗi lần lặp lại khi tăng gradient, các gradient âm của hàm mất mát đối với đầu ra của mô hình được biểu diễn dưới dạng $\{g_1, \ldots, g_n\}$. Các mẫu trong tập huấn luyện được sắp xếp theo thứ tự giảm dần dựa trên các giá trị tuyệt đối của gradient và các mẫu topSet theo $a \times 100\%$ với gradient lớn nhất được chọn để tạo thành một tập con A.

Đối với tập A^c còn lại, bao gồm $(1-a) \times 100\%$ các mẫu với gradient nhỏ hơn, một tập con B với kích thước $b \times |A^c|$ được lấy mẫu ngẫu nhiên. Các mẫu sau đó được phân chia dựa trên giá trị phương sai của Gain tại vector $\tilde{V_j}(d)$ trên tập A, B:

$$\tilde{V_j}(d) = \frac{1}{n}\left[\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right]$$

Trong đó: $A_l = \{x_i \in A : x_{ij} \leq d\}, A_r = \{x_i \in A : x_{ij} > d\}$

$B_l = \{x_i \in B : x_{ij} \leq d\}, B_r = \{x_i \in A : x_{ij} > d\}$

$\frac{1-a}{b}$: hệ số chuẩn hoá tổng các gradient trên B theo kích thước của A^c

Bằng cách lấy mẫu một phía này, ta có thể tập trung hơn vào các mẫu b chưa được huấn luyện mà không làm thay đổi nhiều về phân phối dữ liệu ban đầu.

3.5.2 Exclusive Feature Bundling – EFB

Algorithm 3: Greedy Bundling

Input: F: features, K: max conflict count
Construct graph G
searchOrder ← G.sortByDegree()
bundles ← { }, bundlesConflict ← { }
for i ***in*** *searchOrder* **do**
  needNew ← True
  for $j = 1$ ***to*** *len(bundles)* **do**
    cnt ← ConflictCnt(bundles[j],F[i])
    if *cnt + bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add(F[i]), needNew ← False
      break
  if *needNew* **then**
    Add $F[i]$ as a new bundle to *bundles*
Output: *bundles*

Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges ← {0}, totalBin ← 0
for f ***in*** F **do**
  totalBin += f.numBin
  binRanges.append(totalBin)
newBin ← new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
  newBin[i] ← 0
  for $j = 1$ ***to*** *len*(F) **do**
    if F*[j].bin[i]* $\neq$ *0* **then**
      newBin[i] ← F[j].bin[i] + binRanges[j]
Output: $newBin$, $binRanges$

Hình 3.7: Thuật toán Exclusive Feature Bundling [17]

Dữ liệu thường rất thưa thớt, điều này mang lại khả năng thiết kế gần như không mất dữ liệu để giảm số lượng thuộc tính. Cụ thể, trong một không gian thưa thớt các thuộc tính, nhiều thuộc tính xung đột lẫn nhau, tức là chúng không bao giờ nhận các giá trị khác 0 cùng một lúc. Các thuộc tính này có thể được nhóm lại một cách an toàn thành một thuộc tính duy nhất (được gọi là Exclusive Feature Bundle). Do đó, độ phức tạp của thay đổi từ O(data × feature) thành O(data × bundle) trong khi bundle « feature. Do đó, tốc độ huấn luyện được cải thiện mà không ảnh hưởng đến độ chính xác.

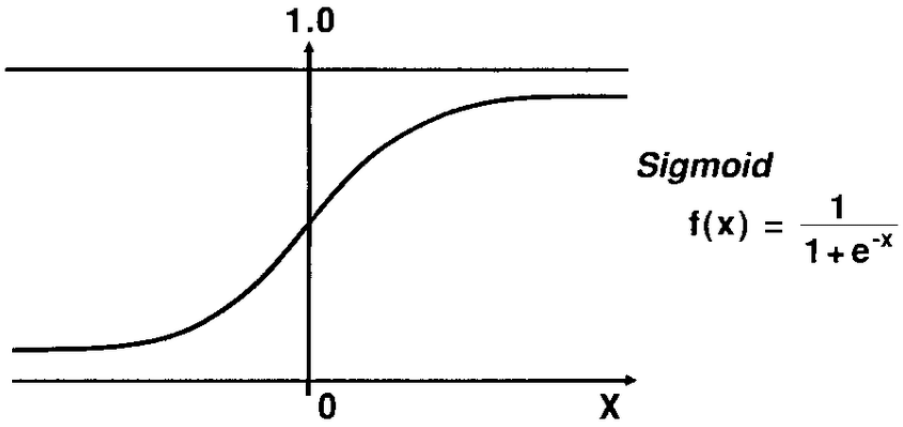
3.6 Hồi quy Logistic

Hồi quy Logistic (Logistic Regression) là một mô hình học máy áp dụng cho các bài toán phân loại nhị phân (dự đoán phân lớp một đối tượng vào một trong hai nhóm).

3.6.1 Hàm Sigmoid

Mô hình hồi quy Logistic hoạt động dựa trên nguyên tắc của hàm Sigmoid, hàm phi tuyến tự chuyển đổi dữ liệu đầu vào thành phân phối xác suất của nó.

Công thức hàm Sigmoid: $f(s) = \frac{1}{1+e^{-s}} \stackrel{\Delta}{=} \sigma(s)$



Hình 3.8: Đồ thị hàm Sigmoid [18]

3.6.2 Xác suất của Logistic

Đặt $\hat{y}_i = f(w^T x_i)$. Xác suất xảy ra sự kiện $y_i = 1$ hoặc $y_i = 0$ với điều kiện dữ liệu đầu vào x_i và trọng số w: $P(y_i|x_i; w) = \hat{y}_i^{y_i} \left(1 - \hat{y}_i\right)^{1-y_i}$

Xét tập huấn luyện $X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{d \times N}$ và $y = [y_1, y_2, \ldots, y_N]$, các mẫu là độc lập thống kê với nhau. Xác suất đồng thời của toàn bộ các mẫu trong tập huấn luyện:

$$P(y|X; w) = \prod_{i=1}^{N} P(y_i|x_i; w) = \prod_{i=1}^{N} \hat{y}_i^{y_i} \left(1 - \hat{y}_i\right)^{1-y_i}$$

3.6.3 Ước lượng hợp lý tối đa (Maximum Likelihood Estimation) và Hàm Cross Entropy

Bài toán đặt ra: Giải bài toán tối ưu hàm hợp lý (Maximum Likelihood Function) để tìm nghiệm w, còn gọi là ước lượng hợp lý tối đa (Maximum Likelihood Estimation).

Vì việc tối ưu trực tiếp là khó khăn nên ta sẽ lấy logarit để đưa về bài toán tối ưu hàm Log Likelihood:

$$\begin{aligned} \hat{w} &= \arg\max_{w} \log \left(\prod_{i=1}^{N} P(y_i|x_i; w) \right) \\ &= \arg\min_{w} \sum_{i=1}^{N} - \left[y_i \log \left(\hat{y}_i \right) + \left(1 - y_i \right) \log \left(1 - \hat{y}_i \right) \right] \end{aligned}$$

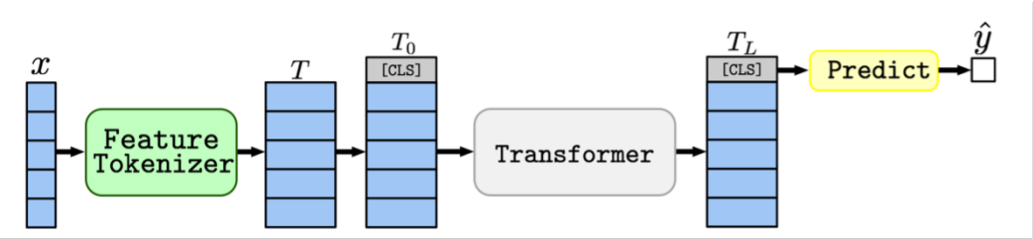
Hàm mất mát Cross Entropy: $\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{N} - \left[y_i \log \left(\hat{y}_i \right) + \left(1 - y_i \right) \log \left(1 - \hat{y}_i \right) \right]$

Bằng cách cập nhật nghiệm theo gradient descent trên từng điểm dữ liệu (x_i, y_i), ta sẽ tìm ra nghiệm tối ưu của hàm mất mát Cross Entropy là:

$w := w - \alpha x_i \left(y_i - \hat{y}_i \right)$ với α là tốc độ học tập (learning rate)

3.7 Mô hình FT – Transformer

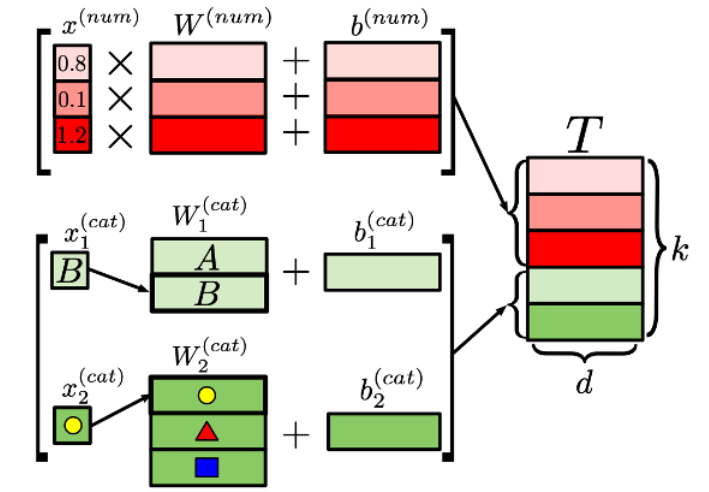
FT-Transformer (Feature Tokenizer – Transformer) là một mô hình được mở rộng từ kiến trúc của Transformer để xử lý các dữ liệu dạng bảng (Tabular Data).



Hình 3.9: Kiến trúc mô hình FT-Transformer [19]

Về tổng quan, đầu tiên Feature Tokenizer sẽ chuyển đổi toàn bộ thuộc tính bao gồm cả categorical và numerical thành các thành phần embeddings. Sau đó các thành phần embeddings này được xử lý tại các tầng Transformer và cuối cùng đưa ra kết quả dự đoán bằng cách sử dụng một token [CLS].

3.7.1 Module Feature Tokenizer



Hình 3.10: Sơ đồ module Feature Tokenizer [19]

Module Feature Tokenizer thực hiện đưa các thuộc tính x về các vector embeddings $T \in \mathbb{R}^{k \times d}$. Công thức chuyển đổi một thuộc tính x_j như sau:

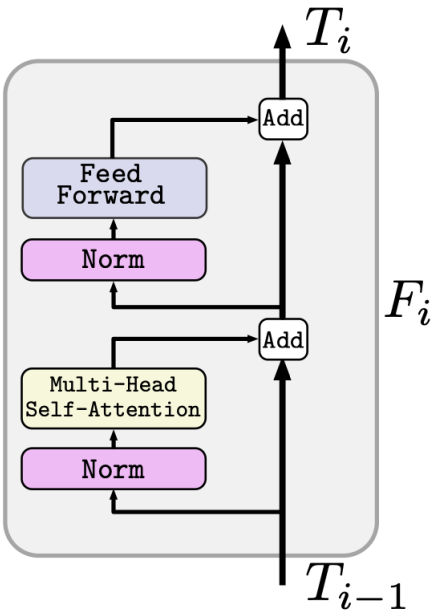
$$T_j = b_j + f_j\left(x_j\right) \in \mathbb{R}^d,\ f_j : \mathbb{X}_j \to \mathbb{R}^d$$

Trong đó: b_j: thuộc tính bias thứ j

$$T_j^{(num)} = b_j^{(num)} + x_j^{(num)}.W_j^{(num)} \in \mathbb{R}^d$$
$$T_j^{(cat)} = b_j^{(cat)} + e_j^T W_j^{(cat)} \in \mathbb{R}^d$$
$$T = stack\left[T_1^{(num)}, \ldots, T_{k^{(num)}}^{(num)}, T_1^{(cat)}, \ldots, T_{k^{(cat)}}^{(cat)}\right] \in \mathbb{R}^{k \times d}$$

e_j^T: one-hot vector sử dụng cho các thuộc tính phân loại

3.7.2 Tầng Transformer



Hình 3.11: Sơ đồ tầng Transformer [19]

Thêm vào vector T một token [CLS] (classification token) sau đó đưa tới các tầng Transformer $(F_1, \ldots, F_L)$:

$$T_0 = stack\left[[CLS], T\right]$$
$$T_i = F_i\left(T_{i-1}\right)\left(i = 1, \ldots, L\right)$$

3.7.3 Kết quả dự đoán đầu ra

Kết quả cuối cùng của token [CLS] được sử dụng để dự đoán giá trị đầu ra:

$$\hat{y} = Linear\left(ReLU\left(LayerNorm\left(T_L^{[CLS]}\right)\right)\right)$$

Ưu điểm chính của FT-Transformer là sự linh hoạt và có thể được sử dụng để phân tích nhiều loại dữ liệu dạng bảng (Tabular Data). Mô hình tương đối đơn giản để thực hiện, giúp tiếp cận người dùng dễ dàng.

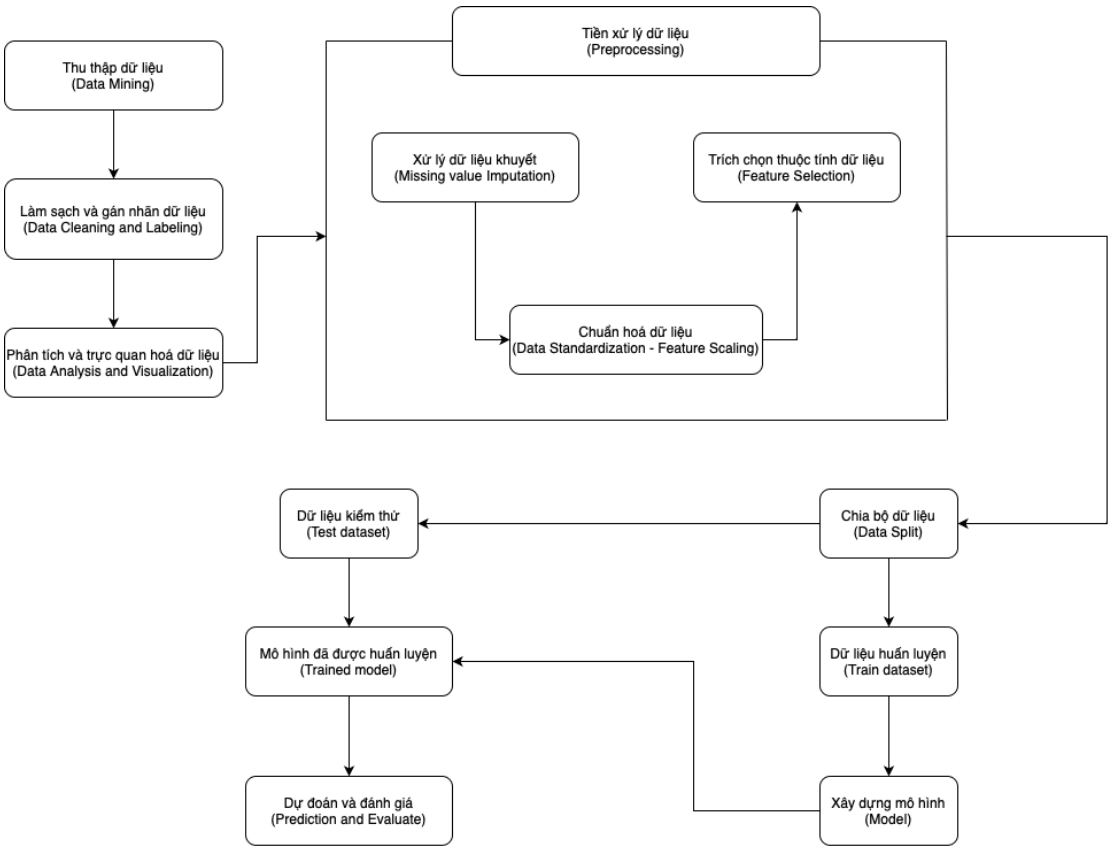
FT-Transformer linh hoạt có thể xử lý cả dữ liệu số (numeric data) và dữ liệu phân loại (categorical data).

Cuối cùng, FT-Transformer có thể sử dụng kiến trúc Transformer mạnh mẽ, có hiệu quả cao đối với nhiều tác vụ. Bằng cách điều chỉnh kiến trúc này để xử lý dữ liệu dạng bảng, FT-Transformer có thể tận dụng những ưu điểm này để phân tích dữ liệu có cấu trúc.

CHƯƠNG 4. PHƯƠNG PHÁP ĐỀ XUẤT

4.1 Tổng quan giải pháp

Để mô tả tổng quan giải pháp của bài toán khi thực hiện đồ án, tôi đã xây dựng luồng hoạt động được tôi mô tả như hình 4.1 dưới đây:



Hình 4.1: Luồng hoạt động

Cụ thể, tôi đã triển khai quá trình thực hiện theo các giai đoạn như sau:

- Thu thập và phân tích dữ liệu.
- Tiền xử lý dữ liệu.
- Xây dựng và huấn luyện mô hình.
- Đánh giá hiệu suất của mô hình.

4.2 Thu thập, phân tích và tiền xử lý dữ liệu

Bộ dữ liệu tôi được cung cấp thông tin về thai phụ và các kết quả xét nghiệm tại Bệnh viện Phụ sản Trung ương (Việt Nam). Đầu tiên, tôi xác định các thuộc tính liên quan đến đặc điểm của bệnh trisomy 21 dựa trên sự hướng dẫn của các chuyên gia y tế. Tôi đã giữ lại những thuộc tính cần chú ý và các thuộc tính khác có ít tác động hơn hoặc không liên quan đã bị xoá vì chúng không đóng góp nhiều cho kết

quả đầu ra. Sau đó, tôi tiếp tục phân tích để gán nhãn cho kết quả đầu ra cùng với phân tích các kết quả xét nghiệm (siêu âm, Double Ttest, Triple test) để bổ sung vào dữ liệu được cung cấp. Tiếp đó, dữ liệu được chia thành 2 phần, tập huấn luyện và tập kiểm tra, sau này được sử dụng để huấn luyện và kiểm tra tương ứng cho các mô hình học máy, học sâu đơn giản.

Sau khi thu thập dữ liệu thô, tôi đã thực hiện trực quan hoá để có một cái nhìn tổng quan về dữ liệu. Sau đó, tôi đã thực hiện một số kỹ thuật tiền xử lý dữ liệu như làm sạch dữ liệu và gán nhãn dữ liệu để bảo rằng dữ liệu được sạch và chính xác, xử lý dữ liệu khuyết để kiểm tra các giá trị bị thiếu và sử dụng các kỹ thuật điền thay thế, trích chọn thuộc tính giúp xác định các thuộc tính dữ liệu quan trọng nhất có thể được sử dụng cho bài toán này, và chuẩn hóa dữ liệu để đưa các thuộc tính dữ liệu có cùng phạm vi và trọng số.

4.3 Xây dựng và huấn luyện mô hình

Khi hoàn thành quá trình tiền xử lý dữ liệu, tôi tiếp tục xây dựng và huấn luyện các mô hình phân loại. Đối với bài toán phân loại thai nhi mang bệnh trisomy 21 và thai nhi bình thường, đầu tiên tôi đã chọn sử dụng ba mô hình để thử nghiệm và đưa ra so sánh: Hồi quy Logistic, XGBoost và LightGBM.

Đầu tiên là mô hình hồi quy Logistic. Tôi lựa chọn mô hình này để triển khai đầu tiên vì tính đơn giản và khả năng diễn giải tốt. Mô hình này phù hợp cho bài toán phân loại nhị phân và có thể cung cấp giả định về mức độ ảnh hưởng của từng thuộc tính độc lập.

Tiếp đó, để tận dụng sức mạnh của các mô hình ensemble, tôi triển khai XGBoost và LightGBM. Cả hai đều là mô hình dựa trên cây quyết định có khả năng xử lý tốt với dữ liệu phức tạp và đồng thời giảm thiểu rủi ro overfitting.

Cuối cùng, khi có được sự so sánh, tôi tập trung chủ yếu những tuần cuối để triển khai mô hình Feature Tokenizer Transformer với hi vọng sử dụng cho dự đoán thực tế, và mô hình này còn đang khá mới chưa có nhiều nghiên cứu phổ biến.

Tất cả các mô hình sau khi được cấu hình, tôi đều tiến hành quá trình huấn luyện trên tập huấn luyện đã được chia từ trước. Mục tiêu là tối ưu hóa tham số và hàm mất mát, tạo ra một mô hình có khả năng dự đoán chính xác trên dữ liệu mới.

4.4 Đánh giá hiệu suất của mô hình

Kết thúc quá trình huấn luyện, tôi thực hiện đánh giá hiệu suất của từng mô hình trên tập kiểm tra. Với mô hình phân loại cho dữ liệu mất cân bằng, việc chọn một metric như là thước đo để đánh giá mô hình cũng rất quan trọng, mà việc đánh giá hiệu suất của mô hình dựa trên độ chính xác (Accuracy) trở nên thiếu ý nghĩa và

thậm chí có thể dẫn đến những đánh giá sai lầm. Chẳng hạn, nếu có một lớp thiểu số và mô hình dự đoán hầu hết các mẫu thuộc lớp đa số, độ chính xác có thể cao nhưng không phản ánh đúng khả năng dự đoán trên lớp thiểu số.

Vậy nên trong phạm vi đồ án này, tôi sẽ quan tâm nhiều hơn đến những thước đo khác như độ nhạy (Sensitivity) và độ đặc hiệu (Specificity), đồng thời lựa chọn G-mean làm thước đo chính để đánh giá hiệu suất của mô hình. G-mean là một thước đo kết hợp độ nhạy và độ đặc hiệu, đồng thời cân nhắc cả hai khía cạnh này, đặc biệt là trong trường hợp dữ liệu bị mất cân bằng.

Tôi đưa ra định nghĩa các giá trị sau:

- P: lớp thai nhi mang bệnh Trisomy 21 (lớp thiểu số).
- N: lớp thai nhi bình thường (lớp đa số).
- TP: tổng số mẫu thai nhi mang bệnh Trisomy 21 được phân loại đúng vào lớp thai nhi mang bệnh Trisomy 21.
- FP: tổng số mẫu thai nhi bình thường nhưng bị phân loại vào lớp thai nhi mang bệnh Trisomy 21.
- TN: tổng số mẫu thai nhi bình thường được phân loại đúng vào lớp thai nhi mang bình thường.
- FN: tổng số mẫu thai nhi mang bệnh Trisomy 21 nhưng bị phân loại vào lớp thai nhi bình thường.

Từ đó tôi có một số giá trị thông số đánh giá mô hình như sau:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN},$$

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN},$$

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}},$$

$$Sensitivity = \frac{TP}{TP + FN},$$

$$Specificity = \frac{TN}{FP + TN},$$

$$Balanced\ Accuracy\ Score = \frac{Sensitivity + Specificity}{2},$$

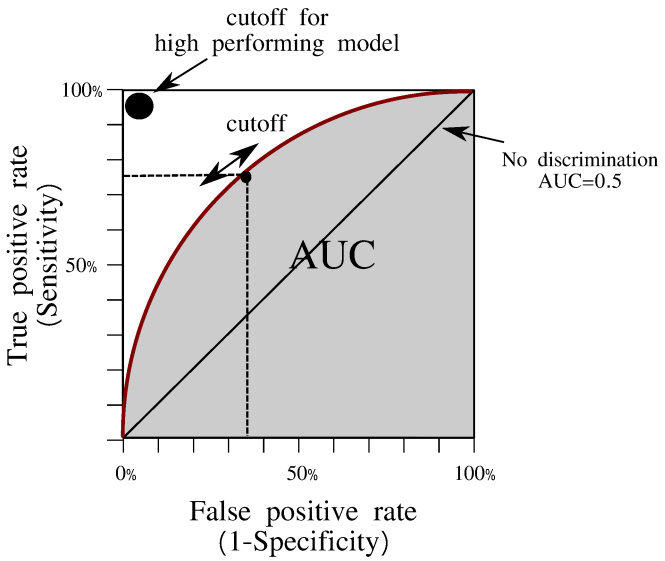
$$G - mean = \sqrt{Sensitivity \times Specificity},$$

$$Macro\ avg = \frac{\sum_{i=1}^{|C|} Score(c_i)}{|C|}, \quad Micro\ avg = \frac{\sum_{i=1}^{|C|} Support(c_i) \times Score(c_i)}{|C|}$$

Ma trận nhầm lẫn (ma trận lỗi) giúp trực quan hóa hiệu suất thuật toán và chủ yếu được sử dụng trong các thuật toán học có giám sát. Ma trận nhầm lẫn thể hiện sự phân bố của các dự đoán trên tất cả các lớp một cách nhanh chóng và trực quan [20].

Hình 4.2: Ma trận nhầm lẫn - Confusion Matrix [21]

Đường cong ROC là biểu đồ thể hiện hiệu suất của mô hình phân loại ở tất cả các lớp. Đường cong này thể hiện 2 giá trị: Sensitivity (TPR - True Positive Rate) và Specificity (FPR - False Positive Rate). Trong y tế và dịch tễ học lâm sàng, phân tích ROC để định lượng các xét nghiệm chẩn đoán y tế (hoặc hệ thống) có thể phân biệt chính xác như thế nào giữa hai trạng thái: mắc bệnh và không mắc bệnh. Phần diện tích dưới đường cong ROC được gọi là AUC (Area Under Curve) là xác suất mà một đối tượng bị bệnh được chọn ngẫu nhiên được đánh giá là có nhiều khả năng bị bệnh hơn một đối tượng không bị bệnh được chọn ngẫu nhiên [22].



Hình 4.3: Ví dụ về đường cong ROC [23]

CHƯƠNG 5. ĐÁNH GIÁ THỰC NGHIỆM

5.1 Tập dữ liệu và quá trình phân tích, tiền xử lý dữ liệu

5.1.1 Tập dữ liệu và thông tin về thuộc tính trisomy 21

Dựa vào bộ dữ liệu được cung cấp từ Bệnh viện Phụ sản Trung ương (Việt Nam) bao gồm các bản ghi về thông tin của thai phụ và xét nghiệm, tôi lựa chọn các thuộc tính liên quan đến kết quả siêu âm thai kì thứ nhất, thai kì thứ hai, Double test và Triple test. Với từng loại xét nghiệm, tôi sử dụng các thuộc tính như sau:

- Siêu âm thai kì thứ nhất: *tuoime, tuoithainhi, co_khoangsangsaugay, co_nangbachhuyetvungco, mat_xuongmui, nguc_ditattim*.
- Siêu âm kì thứ hai: *tuoime, tuoithainhi, mat_xuongsongmui, co_nangbachhuyetvungco, mat_xuongmui, nguc_ditattim*.
- Double Test: *d_mom_hcgb, d_mom_pappa, d_mom_nt*.
- Triple Test: *t_mom_afp, t_mom_ue3, t_mom_hcg*.

Với sự hướng dẫn nghiệp vụ y học từ bác sĩ, tôi đã thực hiện tính toán bổ sung thêm hai thuộc tính dữ liệu được tính tại thời điểm thực hiện xét nghiệm:

i) *tuoime*: Thông tin về tuổi của thai phụ

- *tuoime1* = Ngày đăng kí khám - Ngày sinh của mẹ
- *tuoime2* = Ngày thực hiện siêu âm - Ngày sinh của mẹ
- *tuoime3* = Ngày thực hiện Double test - Ngày sinh của mẹ
- *tuoime4* = Ngày thực hiện Triple test - Ngày sinh của mẹ

ii) *tuoithainhi*: Thông tin về tuổi của thai nhi

- *tuoithai1* = 40 tuần - (Ngày dự kiến sinh - Ngày thực hiện siêu âm)
- *tuoithai2* = Ngày thực hiện siêu âm - Ngày chu kỳ kinh cuối
- *tuoithai3*: Sử dụng ánh xạ hàm trên *chieudaidaumong*
- *tuoithai4* = $10.5992189734 - 0.1682642749 \times BPD + 0.0451977088 \times HC + 0.0302344331 \times AC + 0.0575622626 \times FL + 0.002497813 \times BPD^2 + 0.0017031489 \times FL^2 + 0.0005458077 \times (BPD \times AC) - 0.0051546864 \times (BPD \times FL) - 0.000276112 \times (HC \times AC) + 0.0007657091 \times (HC \times FL) + 0.0005523225 \times (AC \times FL)$

Trong đó: BPD: đường kính lưỡng đỉnh, AC: chu vi bụng,
HC: vòng đầu, FL: chiều dài xương đùi

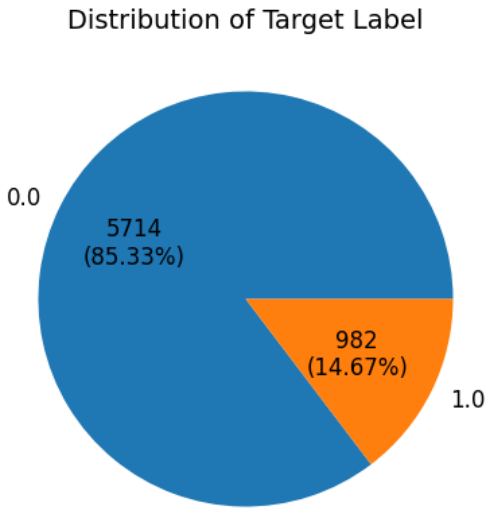
5.1.2 Nhãn dữ liệu của bệnh trisomy 21

Nhãn của các mẫu dữ liệu được gán thông qua các thuộc tính về xét nghiệm gen: *pcr_trisomy21, bobs_trisomy21 và karyotype_trisomy21*.

Tôi đã thống kê để trình bày về các thuộc tính của tập dữ liệu trisomy 21 trong bảng 5.1 và hình 5.1 dưới đây:

	trisomy 21 - Hội chứng down
Số lượng mẫu bình thường	5714
Số lượng mẫu bị bệnh	982
Tổng số lượng mẫu dữ liệu	6696

Bảng 5.1: Thống kê về số lượng mẫu và nhãn dữ liệu trismomy 21



Hình 5.1: Phân bố tỉ lệ nhãn bệnh trisomy 21

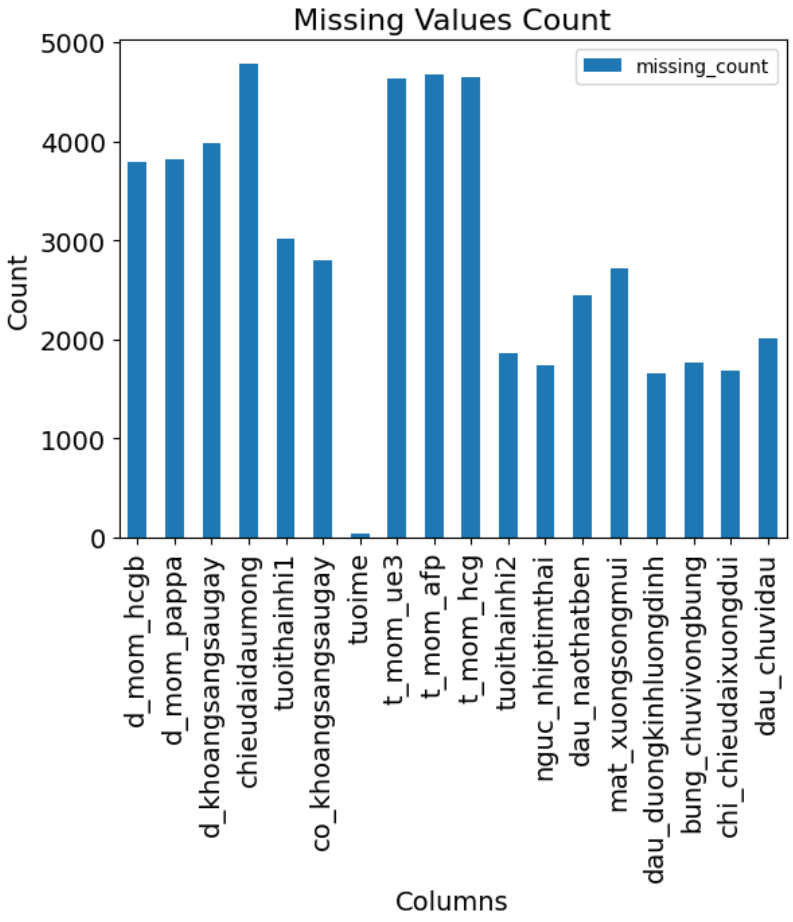
Sau khi lọc lựa chọn và tính toán các thuộc tính đồng thời gán nhãn bệnh tôi đã thu được tập dữ liệu gồm 6696 mẫu, trong đó bao gồm 5714 mẫu bình thường không mang bệnh (chiếm tỉ lệ 85.33%) và 982 mẫu bị bệnh (chiếm tỉ lệ 14.67%). Điều này có thể thấy rằng bộ dữ liệu có sự mất cân bằng nghiêm trọng giữa hai lớp: thai nhi bình thường và thai nhi mang bệnh trisomy 21.

5.1.3 Xử lý dữ liệu bị khuyết (Missing Data)

Dữ liệu bị khuyết là hiện tượng phổ biến xuất hiện trong dữ liệu thực tế, và thông thường những thuộc tính của dữ liệu bị thiếu vượt quá 30% bị coi là rất nghiêm trọng và thường được khắc phục bằng cách loại bỏ đi thuộc tính dữ liệu đó hoặc cố gắng tìm kiếm kết quả thực để điền vào đó. Tại thời điểm phân tích về bộ dữ liệu, tôi đã thống kê tình trạng khuyết dữ liệu trong từng thuộc tính với kết quả thu được như sau:

Thuộc tính	Số lượng mẫu bị khuyết	Tỉ lệ mẫu bị khuyết (%)
d_mom_hcgb	3789	56.59
d_mom_pappa	3824	57.11
d_khoangsangsaugay	3982	59.47
chieudaidaumong	4781	71.40
tuoithainhi1	3011	44.97
co_khoangsangsaugay	2798	41.79
tuoime	44	0.66
t_mom_ue3	4630	69.15
t_mom_afp	4670	69.74
t_mom_hcg	4653	69.49
tuoithainhi2	1856	27.72
nguc_nhiptimthai	1738	25.96
dau_naothatben	2450	36.59
mat_xuongsongmui	2715	40.55
dau_duongkinhluongdinh	1657	24.75
bung_chuvivongbung	1773	26.48
chi_chieudaixuongdui	1681	25.10
dau_chuvidau	2014	30.08

Bảng 5.2: Thống kê về tỉ lệ khuyết dữ liệu của từng thuộc tính trisomy 21



Hình 5.2: Biểu đồ số lượng dữ liệu bị khuyết trong từng thuộc tính trisomy 21

Từ thống kê trên bảng 5.2 và biểu đồ hình 5.2, tôi nhận thấy hầu như các thuộc tính dữ liệu đều thiếu lớn quá 30%; việc loại bỏ chúng đi không phải là sự lựa chọn hợp lý, và cũng không thể tìm được kết quả thực tế để điền bổ sung vào đó. Từ đó phân tích, tôi đã lựa chọn thuật toán K-Nearest Neighbors Imputation dự đoán giá trị bị khuyết với thông tin của các láng giềng lân cận để sinh ra những dữ liệu mới thay thế dữ liệu bị thiếu với hy vọng dữ liệu này là gần giống với dữ liệu thực tế.

Quá trình xử lý bao gồm: Đầu tiên tôi sẽ gọi mô hình KNNImputer từ thư viện scikit-learn sử dụng thuật toán K-Nearest Neighbors (KNN) và xem xét 25 láng giềng gần nhất để quyết định giá trị thích hợp cần điền bổ sung và học trên tập huấn luyện df_train, sau đó tôi sử dụng mô hình đã được huấn luyện để điền giá trị khuyết cho tập kiểm thử df_test. Kết quả thu được là hai tập dữ liệu mới, df_train_imputed và df_test_imputed đều đã được xử lý khuyết dữ liệu.

Kết quả sau khi sử dụng KNNImputation, tôi thu được như sau:

```
After using KNNImputation, the number of missing value records in train set : 0
After using KNNImputation, the number of missing value records in test set: 0
Shape of the train set : (3785, 18)
Shape of the test set : (2911, 18)
```

Hình 5.3: Kết quả sau khi xử lý khuyết dữ liệu bằng KNNImputation

Từ kết quả trên hình 5.3, tập huấn luyện và tập kiểm thử đều không có bản ghi bị khuyết dữ liệu sau khi được xử lý bằng KNNImputation. Điều này sẽ góp phần tăng sự đầy đủ và chất lượng của dữ liệu, đảm bảo tính nhất quán giữa tập huấn luyện và tập kiểm thử.

5.1.4 Chuẩn hoá dữ liệu (Data Standarization)

Trong quá trình thực hiện đồ án, tôi nhận thấy sự biến động giữa các thuộc tính trong tập dữ liệu trisomy 21. Để tối ưu hóa quá trình huấn luyện và kiểm thử dự đoán, tôi đã tiếp tục tiền xử lý dữ liệu thông qua chuẩn hóa dữ liệu sử dụng phương pháp StandardScaler từ thư viện scikit-learn.

Quá trình chuẩn hoá này bao gồm: Đầu tiên, tôi chia tách tập huấn luyện (train dataset) và kiểm thử (test dataset), và áp dụng StandardScaler để chuẩn hóa các thuộc tính số (numeric features) cho cả hai tập dữ liệu. Quá trình giúp đồng nhất đơn vị đo của các thuộc tính, đưa phân phối dữ liệu về dạng phân phối chuẩn (có giá trị trung bình bằng 0 và phương sai bằng 1).

Các mẫu trong tập huấn luyện sẽ được sử dụng để học tham số chuẩn hóa, sau đó áp dụng các tham số này để chuẩn hoá tập kiểm thử. Kết quả là hai tập dữ liệu mới (df_train_2 và df_test_2) được chuẩn hóa thuộc tính chính xác và đồng đều.

Tiếp đó, tôi cập nhật tập dữ liệu gốc từ hai tập dữ liệu mới đã được chuẩn hoá bên trên. Các thuộc tính ban đầu sẽ được loại bỏ, và thêm vào các cột thuộc tính đã được chuẩn hóa. Quá trình chuẩn hoá này sẽ giúp dữ liệu được mịn màng, giúp tăng cường khả năng học của các mô hình học máy và học sâu trong quá trình huấn luyện và dự đoán.

5.1.5 Trích chọn thuộc tính (Feature Selection)

Để tối ưu cải thiện hiệu suất mô hình sau này, tôi đã tiếp tục quá trình tiền xử lý bằng kỹ thuật trích chọn thuộc tính (Feature Selection) bằng phương pháp SelectKBest từ thư viện scikit-learn.

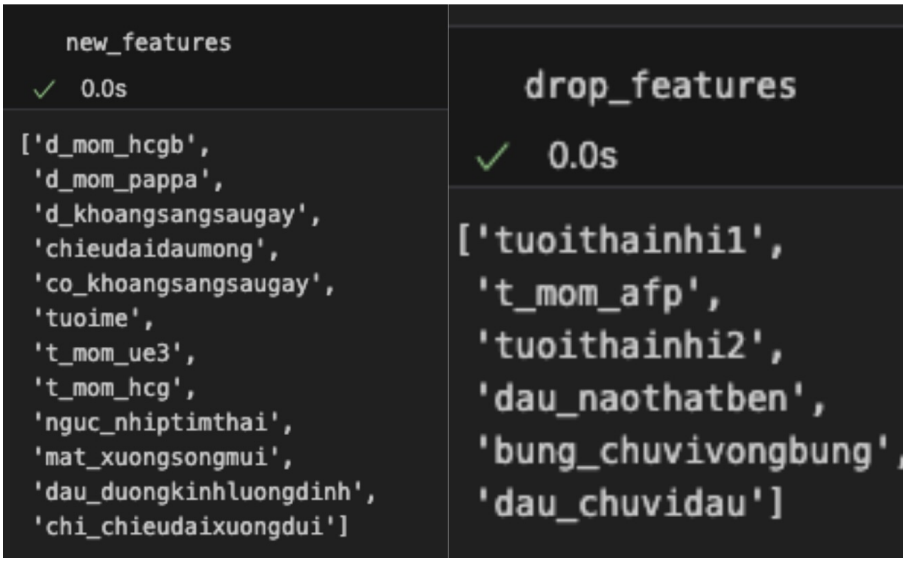
Khi xét duyệt qua từng thuộc tính dữ liệu, để đưa ra quyết định về giữ lại hay loại bỏ nó, tôi đã sử dụng phương pháp SelectKBest với hàm điểm F-score và kỹ thuật ANOVA (Analysis of Variance). Tôi đặt ra hai giá trị ngưỡng: ngưỡng threshold để xác định giá trị p-value tối đa để đánh giá thuộc tính giữ lại hay loại bỏ, và điểm ngưỡng t_score để quyết định điểm tối thiểu cho thuộc tính sử dụng hàm F-score.

Trong SelectKBest, tôi đã sử dụng hàm điểm f_classif để tính toán giá trị F-score và p-value cho mỗi thuộc tính. F-score đo lường sự tách biệt giữa các thuộc tính, và sau đó p-value sẽ kiểm định xem sự tách biệt này có ý nghĩa thống kê hay không. Các thuộc tính được giữ lại tạo thành một tập dữ liệu mới, và những thuộc tính không đạt ngưỡng được loại bỏ. Điều này giúp giảm số lượng thuộc tính ít có ý nghĩa và tập trung vào những thuộc tính có tác động lớn đối với kết quả dự đoán.

Kết quả sau khi trích chọn thuộc tính bằng SelectKBest, tôi thu được như sau:

```
Before SelectKBest = (3785, 18)
================================================
d_mom_hcgb: F-Score = 22.244, p-value = 0.000
d_mom_pappa: F-Score = 8.663, p-value = 0.003
d_khoangsangsaugay: F-Score = 102.379, p-value = 0.000
chieudaidaumong: F-Score = 20.702, p-value = 0.000
tuoithainhi1: F-Score = 2.415, p-value = 0.120
co_khoangsangsaugay: F-Score = 103.019, p-value = 0.000
tuoime: F-Score = 10.533, p-value = 0.001
t_mom_ue3: F-Score = 35.965, p-value = 0.000
t_mom_afp: F-Score = 2.167, p-value = 0.141
t_mom_hcg: F-Score = 17.776, p-value = 0.000
tuoithainhi2: F-Score = 0.488, p-value = 0.485
nguc_nhiptimthai: F-Score = 11.336, p-value = 0.001
dau_naothatben: F-Score = 0.051, p-value = 0.821
mat_xuongsongmui: F-Score = 37.094, p-value = 0.000
dau_duongkinhluongdinh: F-Score = 5.382, p-value = 0.020
bung_chuvivongbung: F-Score = 0.476, p-value = 0.490
chi_chieudaixuongdui: F-Score = 5.879, p-value = 0.015
dau_chuvidau: F-Score = 0.027, p-value = 0.868
================================================
After SelectKBest = (3785, 12)
Drop-out Features = 6
```

Hình 5.4: Kết quả sau khi xử lý trích chọn thuộc tính bằng SelectKBest



Hình 5.5: Danh sách các thuộc tính được giữ lại và bị loại bỏ

Từ kết quả nhận được trên hình 5.4 và hình 5.5, tôi nhận thấy sau khi sử dụng SelectKBest, số lượng thuộc tính dữ liệu giảm từ 18 xuống 12. Các thuộc tính không có nhiều ý nghĩa đã được loại bỏ, giảm bớt số chiều của vectơ thuộc tính và hi vọng cũng giúp cải thiện hiệu suất của mô hình.

5.1.6 Xử lý mất cân bằng dữ liệu (Imbalanced Dataset)

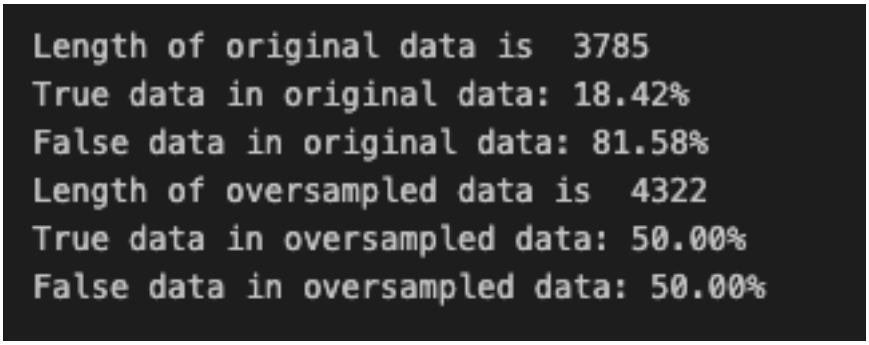
Sau khi thực hiện gán nhãn bệnh trisomy 21 trên toàn bộ tập dữ liệu, tôi nhận thấy phần lớn dữ liệu là thông tin của người bình thường, dẫn tới có sự mất cân bằng dữ liệu nghiêm trọng giữa hai lớp: thai nhi mang bệnh trisomy 21 và thai nhi bình thường.

Mất cân bằng dữ liệu là hiện tượng số mẫu của một lớp có sự chênh lệch rất lớn với các lớp còn lại. Nói cách khác, có sự thiên vị rất lớn dành cho lớp ấy với các mô hình. Rõ ràng, để tối ưu độ chính xác, mô hình sẽ có xu hướng học nhiều cho các lớp đa số mà bỏ qua các lớp thiểu số. Trong lĩnh vực y tế, những dữ liệu bất thường thường xuất hiện ít hơn những dữ liệu bình thường, do đó dẫn tới vấn đề mất cân bằng dữ liệu, và trong bài toán này, tôi cần dự đoán khả năng liệu thai nhi có mang bệnh trisomy 21 - lớp thiểu số trong dữ liệu.

Đầu tiên, tôi chia tập dữ liệu thành 70% mẫu cho tập huấn luyện và 30% mẫu cho tập kiểm thử. Để khắc phục vấn đề mất cân bằng giữa hai lớp (thai nhi mang bệnh trisomy 21 và thai nhi bình thường) trong tập huấn luyện, tôi đã lựa chọn sử dụng kỹ thuật SMOTE (Synthetic Minority Over-sampling Technique).

Ý tưởng chung của SMOTE là tạo ra dữ liệu tổng hợp giữa mỗi mẫu của lớp thiểu số từ k láng giềng gần nhất của nó (mặc định là k = 5). Sau đó giữa các cặp điểm được tạo bởi mẫu và từng láng giềng để nội suy ra dữ liệu cho các mẫu mới.

Kết quả sau khi xử lý mất cân bằng dữ liệu bằng , tôi thu được như sau:



Hình 5.6: Kết quả sau khi xử lý mất cân bằng dữ liệu bằng SMOTE

Từ kết quả trên hình 5.6, tôi nhận thấy SMOTE giúp cân bằng số lượng mẫu giữa hai lớp đồng thời tăng thêm tính đa dạng cho tập dữ liệu, mở rộng phạm vi của lớp thiểu số (lớp thai nhi mang bệnh trisomy 21).

Đến đây sau bước xử lý mất cân bằng dữ liệu, tôi đã hoàn thành kết thúc xong quá trình tiền xử lý dữ liệu. Dữ liệu mà tôi tiền xử lý đã sẵn sàng cho các quá trình huấn luyện, kiểm thử và dự đoán các mô hình học máy và học sâu sau đó.

5.2 Thử nghiệm và so sánh trên các mô hình phổ biến

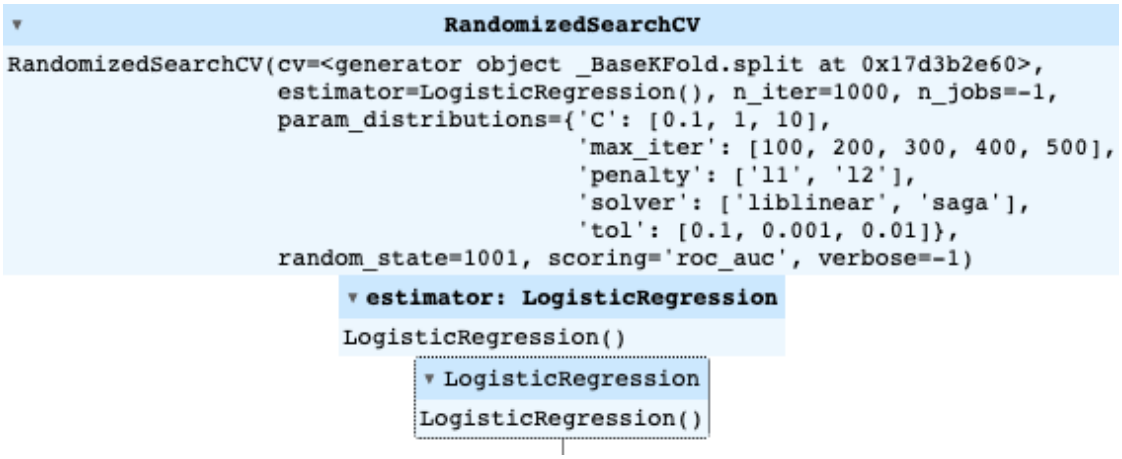
5.2.1 Mô hình hồi quy Logistic

Trong quá trình xây dựng mô hình hồi quy Logistic cho bài toán phân loại bệnh trisomy 21, tôi đã sử dụng phương pháp tìm kiếm tham số ngẫu nhiên (RandomizedSearchCV) để tối ưu hóa các tham số của mô hình.

Đầu tiên, tôi đưa ra các tham số cần được tối ưu bao gồm:

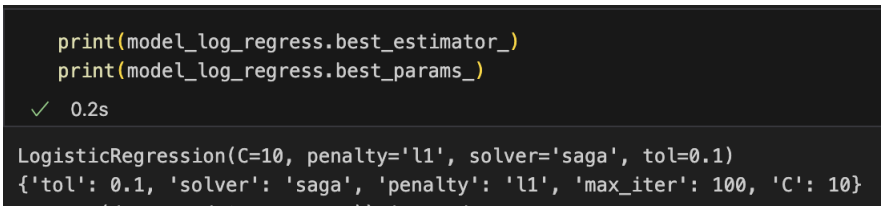
- $p\_folds = 20$: Số lượng fold, dữ liệu sẽ được chia thành 20 phần trong quá trình cross-validation.
- $p\_iter = 1000$: Số lần lặp tìm kiếm ngẫu nhiên để tối ưu hóa tham số.
- $C : [0.1, 1, 10]$: Tham số ổn định hoá, là giá trị nghịch đảo của mức độ điều chỉnh độ chệch của mô hình. Khi C giảm, mức độ điều chỉnh tăng, và mô hình sẽ đơn giản hơn với ít ảnh hưởng của nhiễu. Khi C tăng, mức độ điều chỉnh giảm, và mô hình phức tạp hơn với khả năng khai thác từ tập huấn luyện.
- $penalty : [l1, l2]$: Độ chệch trong hồi quy Logistic: l1 (Lasso) hoặc l2 (Ridge).
- $solver : [liblinear, saga]$: Thuật toán tối ưu hóa (liblinear hoặc saga).
- $max\_iter : [100, 200, 300, 400, 500]$: Số lần lặp tối đa.
- $tol : [0.1, 0.001, 0.01]$: Ngưỡng dừng, là sự sai lệch nhỏ nhất giữa các lần lặp liên tiếp khi huấn luyện.

Tiếp đó, tôi khởi tạo mô hình hồi quy Logistic từ thư viện sklearn và RandomizedSearchCV để tìm kiếm trên danh sách tham số đã khai báo bên trên. Quá trình này được huấn luyện bằng phương pháp Stratified K-Folds được sử dụng trong quá trình cross-validation trên tập huấn luyện để đảm bảo tỷ lệ phân phối dữ liệu giữa các lớp được duy trì trong từng fold.



Hình 5.7: Kiến trúc của mô hình hồi quy Logistic với RandomizedSearchCV

Sau khi huấn luyện mô hình trên tập huấn luyện, khi này tôi thu được mô hình hồi quy Logistic dựa trên độ đo ROC AUC đã được tối ưu hóa sau đó được sử dụng để dự đoán và phân loại trên tập dữ liệu kiểm thử. Kết quả tối ưu của RandomizedSearchCV được lưu vào $model\_log\_regress.best\_params\_$ và lưu trạng thái của mô hình vào file $logRegression\_model\_state.joblib$ với kết quả như hình 5.8 sau:



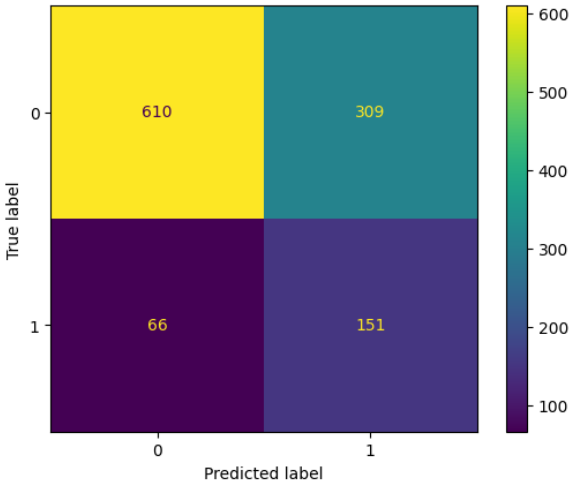
Hình 5.8: Kết quả tối ưu của mô hình Logistic Regression với RandomizedSearchCV

Sau đó, tôi sử dụng mô hình đã được tối ưu này để dự đoán kết quả trên tập kiểm thử với hai giá trị: $log\_regress\_pred\_test$ để dự đoán của mô hình trên tập kiểm thử, và $log\_regress\_proba\_test$ là xác suất dự đoán của mô hình cho lớp positive trên tập kiểm thử.

Đánh giá các phép đo hiệu suất của mô hình hồi quy Logistic như sau:

- $Accuracy = 0.66989$: Mô hình dự đoán đúng khoảng 66.99% trên tập kiểm thử.
- $Balanced\ Accuracy\ Score = 0.67981$: Mô hình có hiệu suất khá trên cả hai lớp bình thường và mang bệnh.

- $Precision = 0.32826, Recall = 0.69585, F1\ Score = 0.44609$: Mô hình có hiệu suất tương đối tốt khi dự đoán các mẫu trong lớp mang bệnh.
- $ROC\ AUC\ Score = 0.67981$: Mô hình có khả năng phân loại tốt giữa hai lớp bình thường và mang bệnh.
- Ma trận nhầm lẫn:



Hình 5.9: Ma trận nhầm lẫn của mô hình hồi quy Logistic

Từ ma trận nhầm lẫn trên hình 5.9, ta có thể thấy mô hình dự đoán 610 mẫu thai nhi bình thường được phân loại đúng vào lớp thai nhi mang bình thường, 66 mẫu thai nhi bình thường nhưng bị phân loại vào lớp thai nhi mang bệnh trisomy 21, 151 mẫu thai nhi mang bệnh trisomy 21 được phân loại đúng vào lớp thai nhi mang bệnh trisomy 21, 309 mẫu thai nhi mang bệnh trisomy 21 nhưng bị phân loại vào lớp thai nhi bình thường.

- Classification Report:

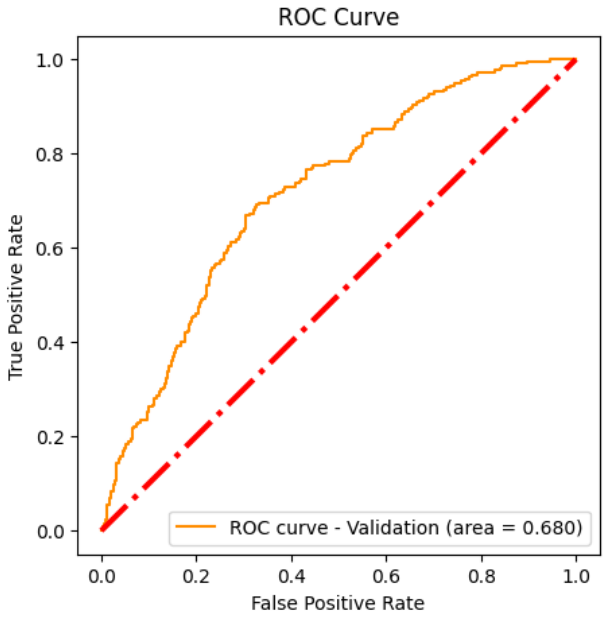
Classification_report

	precision	recall	f1-score	support
0	0.90	0.66	0.76	919
1	0.33	0.70	0.45	217
accuracy			0.67	1136
macro avg	0.62	0.68	0.61	1136
weighted avg	0.79	0.67	0.70	1136

Hình 5.10: Classification Report của mô hình hồi quy Logistic

Dựa vào Classification Report, giá trị Precision trung bình là 0.79 tức mô hình phân lớp chính xác 79% mẫu vào lớp thực sự. Nhưng giá trị Recall trung bình chỉ có 0.67, nghĩa là mô hình chỉ phân lớp được 67% các lớp thực sự mẫu.

- Đường cong ROC Curve:



Hình 5.11: Đường cong ROC Curve của mô hình hồi quy Logistic

Dựa vào đồ thị ROC curve - Validation (area = 0.680) được minh hoạ dựa trên các mẫu dữ liệu trong tập xác thực, phần diện tích dưới đường cong ROC (AUC) bằng 0.68, chứng tỏ mô hình được đánh giá có hiệu suất trung bình. Từ đó, tôi cho rằng mô hình có xu hướng phân lớp các mẫu thai nhi mang bệnh trisomy 21 là bình thường.

Tổng kết: Tôi nhận thấy mô hình hồi quy Logistic phân loại tốt trên lớp negative (thai nhi bình thường), nhưng hiệu suất trên lớp positive (thai nhi mang bệnh trisomy 21) có thể còn cần được cải thiện. Do đó mô hình cần cải thiện khả năng đưa ra dự đoán trên lớp positive và giảm thiểu số lượng False Negatives (FN).

5.2.2 Mô hình thuật toán XGBoost (Extreme Gradient Boosting)

Sau khi triển khai mô hình hồi quy Logistic với kết quả thu được trình bày trên phần 5.2.1, tôi tiếp tục triển khai mô hình thuật toán XGBoost kết hợp với sử dụng phương pháp tìm kiếm tham số ngẫu nhiên (RandomizedSearchCV) trên tập dữ liệu đã được tiền xử lý và phân chia trước để có được sự so sánh tiếp theo.

Dưới đây là danh sách các tham số tôi đưa ra cần được tối ưu mô hình bao gồm:

- $p\_folds = 20$: Số lượng fold, dữ liệu sẽ được chia thành 20 phần trong quá trình cross-validation.
- $p\_iter = 1000$: Số lần lặp tìm kiếm ngẫu nhiên để tối ưu hóa tham số.
- $p\_estimators = 1000$: Số lượng cây quyết định (Decision Tree) được sử dụng.

- $p\_learning\_rate = 0.00001$: Tốc độ học tập của mô hình.
- $max\_depth : randint(5, 10)$: Độ sâu tối đa của cây quyết định.
- $gamma : uniform(0.0, 0.5)$: Tham số kiểm soát việc bổ sung nút.
- $subsample : uniform(0.6, 1.0)$: Tỷ lệ mẫu được sử dụng để xây dựng mỗi cây quyết định.
- $colsample\_bytree : uniform(0.6, 1.0)$: Tham số đưa ra tỷ lệ các thuộc tính để xây dựng mỗi cây quyết định.
- $reg\_alpha : uniform(0.0, 1.0)$: Tham số kiểm soát sự giảm trọng số của các nút lá (Lasso regularization).
- $reg\_lambda : uniform(0.0, 1.0)$: Tham số kiểm soát sự giảm trọng số của các hệ số (Ridge regularization).
- $min\_child\_weight : randint(3, 7)$: Trọng số tối thiểu của một nút lá.
- $scale\_pos\_weight : randint(1, 10)$: Tham số cân bằng tỷ lệ giữa các lớp khi mất cân bằng dữ liệu.

Tiếp đó, tôi khởi tạo mô hình XGBClassifier từ thư viện xgboost và sử dụng RandomizedSearchCV để tinh chỉnh danh sách tham số đã khai báo bên trên. Quá trình này được huấn luyện bằng phương pháp Stratified K-Folds được sử dụng trong quá trình cross-validation.

Hình 5.12: Kiến trúc của mô hình XGBoost với RandomizedSearchCV

Sau quá trình huấn luyện, tôi đã nhận được mô hình XGBoost tối ưu nhất dựa trên độ đo ROC AUC. Kết quả tối ưu hoá được lưu vào $model\_xgb.best\_params\_$

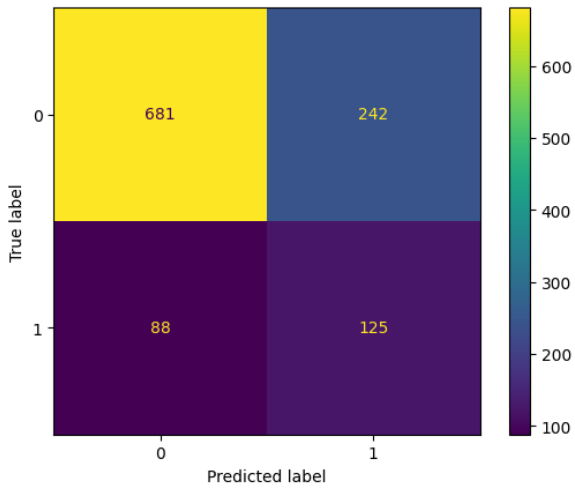
và lưu trạng thái của mô hình vào file $xgboost\_model\_state.json$ như sau:

$$colsample\_bytree : 0.6213861274463689, gamma : 0.4868988425820678,$$
$$max\_depth : 8, min\_child\_weight : 3,$$
$$reg\_alpha : 0.5924661893762875, reg\_lambda : 0.06338785318074813,$$
$$scale\_pos\_weight : 1, subsample : 0.8181751922084126.$$

Sau đó, tôi sử dụng mô hình tối ưu này để dự đoán kết quả trên tập kiểm thử với hai giá trị: xgb_pre_test là dự đoán của mô hình trên tập kiểm thử, và xgb_proba_test là xác suất dự đoán của mô hình cho lớp positive trên tập kiểm thử.

Kết quả và đánh giá với các phép đo hiệu suất của mô hình XGBoost như sau:

- $Accuracy = 0.70951$: Độ chính xác trên tập kiểm thử khá cao, tuy nhiên suy đoán chỉ với Accuracy có thể dẫn đến đánh giá lệch do dữ liệu mất cân bằng.
- $Balanced\ Accuracy\ Score = 0.66233$: Giữa độ nhạy và độ đặc hiệu trên tập kiểm thử có sự cân bằng khá tốt. Mô hình có thể có khả năng phân lớp chính xác trên cả hai lớp.
- $Precision = 0.34060, Recall = 0.58685, F1\ Score = 0.43103$: Precision khá thấp, chứng tỏ mô hình có xu hướng phân lớp về các lớp negative. Giá trị Recall trung bình, nên mô hình có khả năng phân lớp đúng một số mẫu positive khá tốt. Giá trị F1 Score ở dưới trung bình, nên mô hình cần được điều chỉnh.
- $ROC\ AUC\ Score = 0.66233$: Mô hình có khả năng phân loại trung bình giữa hai lớp bình thường và mang bệnh.
- Ma trận nhầm lẫn:



Hình 5.13: Ma trận nhầm lẫn của mô hình XGBoost

Từ ma trận nhầm lẫn trên hình 5.13, ta có thể thấy mô hình dự đoán 681 mẫu thai nhi bình thường được phân loại đúng vào lớp thai nhi mang bình thường, 88 mẫu thai nhi bình thường nhưng bị phân loại vào lớp thai nhi mang bệnh trisomy 21, 125 mẫu thai nhi mang bệnh trisomy 21 được phân loại đúng vào lớp thai nhi mang bệnh trisomy 21, 242 mẫu thai nhi mang bệnh trisomy 21 nhưng bị phân loại vào lớp thai nhi bình thường.

- Classification Report:

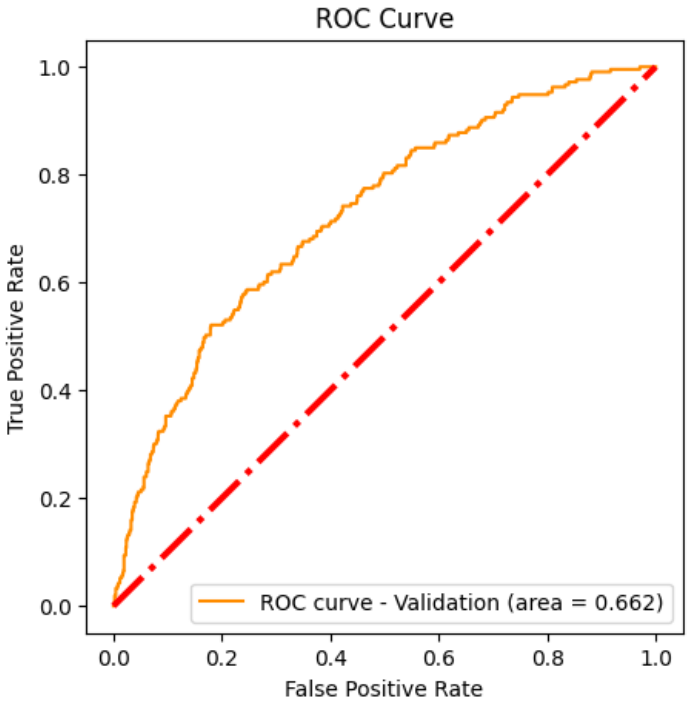
Classification_report

	precision	recall	f1-score	support
0	0.89	0.74	0.80	923
1	0.34	0.59	0.43	213
accuracy			0.71	1136
macro avg	0.61	0.66	0.62	1136
weighted avg	0.78	0.71	0.73	1136

Hình 5.14: Classification Report của mô hình XGBoost

Phân tích Classification Report trên hình 5.14, tôi thấy rằng Accuracy của mô hình đạt khoảng 0.71, nghĩa là khoảng 71% mẫu dữ liệu được phân lớp đúng. Giá trị trung bình của Precision là 0.78 và giá trị trung bình của Recall là 0.71, cả hai giá trị đều tương đối cho thấy mô hình có khả năng phân lớp khá chính xác cả hai lớp. Tuy nhiên, mô hình có độ chính xác cao hơn đối với lớp 0 (lớp thai nhi bình thường) so với lớp 1 (lớp thai nhi mang bệnh trisomy 21).

- Đường cong ROC Curve:



Hình 5.15: Đường cong ROC Curve của mô hình XGBoost

Đồ thị đường cong ROC của mô hình XGBoost cho thấy mô hình có độ chính xác tốt. Đường cong nằm ở phía trên bên trái của biểu đồ, điều này cho thấy mô hình phân lớp tốt giữa hai lớp bình thường và mang bệnh trisomy 21. Cụ thể, ở mức độ sai số dương tính giả (FPR) là 0.2, tỷ lệ dương tính thực (TPR) là 0.6, chứng tỏ ở mức độ này mô hình có thể phân biệt chính xác khoảng 60% các mẫu mang bệnh trisomy 21 thực. Tiếp tục, ta thấy phần diện tích dưới đường cong ROC (AUC) là 0.662 cho thấy mô hình có hiệu suất trung bình, vẫn có thể được cải thiện.

Tổng kết: Mô hình XGBoost mà tôi triển khai nhìn chung đã đạt được hiệu quả, nhưng sự cải thiện về độ chính xác so với mô hình hồi quy Logistic cũng không có sự cải thiện đáng kể. Tôi dự định sẽ tiếp tục triển tiếp một mô hình phức khác để nâng cao sự cải thiện về tính phân lớp trên lớp thai nhi mang bệnh.

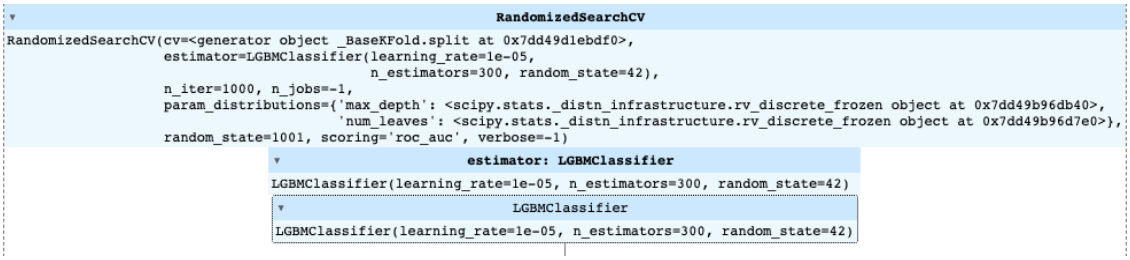
5.2.3 Mô hình thuật toán LightGBM (Light Gradient-Boosting Machine)

Sau khi đánh giá phân tích hiệu quả của mô hình XGBoost với kết quả trình bày phía trên phần 5.2.2, tôi tiếp tục xây dựng mô hình thuật toán LightGBM. Và để tối ưu hiệu suất của mô hình, tôi vẫn tiếp tục sử dụng phương pháp tìm kiếm tham số ngẫu nhiên (RandomizedSearchCV) trong quá trình huấn luyện.

Trước khi bắt đầu huấn luyện mô hình, tôi khai báo danh sách tham số để tối ưu mô hình như sau:

- $p\_folds = 20$: Số lượng fold, dữ liệu sẽ được chia thành 20 phần trong quá trình cross-validation.
- $p\_iter = 1000$: Số lần lặp thử nghiệm ngẫu nhiên khi tìm kiếm tham số tối ưu trong 1000 bộ tham số khác nhau cho mô hình.
- $p\_estimators = 300$: Số lượng cây quyết định (Decision Tree) được sử dụng.
- $p\_learning\_rate = 0.00001$: Tốc độ học tập của mô hình
- $num\_leaves : randint(5, 50)$: Số lượng nút lá tối đa mà mỗi cây con trong mô hình có thể có, để kiểm soát độ sâu của mỗi cây con.
- $max\_depth : randint(5, 10)$: Độ sâu tối đa của mỗi cây quyết định, trong mô hình này tôi đặt ngưỡng độ sâu từ 5 đến 10.

Sau khi khai báo xong danh sách tham số, tôi khởi tạo mô hình LGBMClassifier từ thư viện lightgbm và sử dụng RandomizedSearchCV để tối ưu các tham số bên trên. Bằng cách huấn luyện với phương pháp Stratified K-Folds cho quá trình cross-validation, tôi đã thu được kiến trúc của mô hình như sau:



Hình 5.16: Kiến trúc của mô hình LightGBM với RandomizedSearchCV

Sau khi xây dựng xong mô hình, tôi thực hiện huấn luyện mô hình LightGBM dựa trên độ đo ROC AUC. Tham số của mô hình sau khi được tối ưu hoá được lưu vào $model\_lgbm.best\_params\_$ và trạng thái của mô hình tối ưu được lưu vào file $LightGBM\_model\_state.joblib$ với kết quả được trình bày dưới đây:

```
print(model_lgbm.best_estimator_)
print(model_lgbm.best_params_)
```

```
LGBMClassifier(learning_rate=1e-05, max_depth=9, n_estimators=300,
               num_leaves=49, random_state=42)
{'max_depth': 9, 'num_leaves': 49}
```

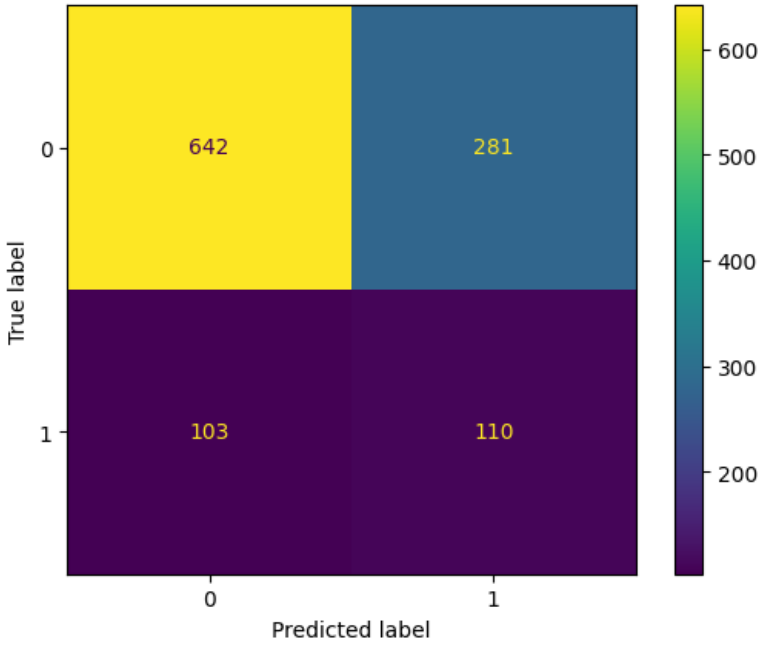
Hình 5.17: Kết quả tối ưu của mô hình LightGBM với RandomizedSearchCV

Tôi tiếp tục sử dụng mô hình tối ưu này để phân lớp trên tập kiểm thử với hai giá trị $lgbm\_pred\_test$ là dự đoán của mô hình trên tập kiểm thử, và $lgbm\_proba\_test$ là xác suất dự đoán của mô hình cho lớp positive trên tập kiểm thử.

Kết quả thu được và đánh giá của tôi với các phép đo hiệu suất của mô hình LightGBM như sau:

- $Accuracy = 0.66197$: Giá trị độ chính xác này ở trên mức trung bình, chưa đánh giá được sự cân bằng giữa việc dự đoán cả hai lớp.
- $Balanced\ Accuracy\ Score = 0.60599$: Mô hình đạt được sự cân bằng khá giữa hai lớp.
- $Precision = 0.28133, Recall = 0.51643, F1\ Score = 0.36424$: Giá trị của Precision khá thấp, chứng tỏ mô hình có độ chính xác thấp trong việc phân loại các lớp thai nhi mang bệnh. Trong khi đó giá trị Recall khoảng 51.6% và F1 Score khoảng 35.4% đều ở mức trung bình khá, cho thấy hiệu suất tổng thể vừa phải trong việc phân loại hai lớp.
- $ROC\ AUC\ Score = 0.60599$: Giá trị này cũng ở mức trung bình, chứng tỏ khả năng phân lớp của mô hình là trung bình.

- Ma trận nhầm lẫn:



Hình 5.18: Ma trận nhầm lẫn của mô hình LightGBM

Với kết quả thu được từ ma trận nhầm lẫn trên hình 5.18, mô hình đã phân lớp 642 mẫu thai nhi bình thường được phân loại đúng vào lớp thai nhi mang bình thường, 103 mẫu thai nhi bình thường nhưng bị phân loại vào lớp thai nhi mang bệnh trisomy 21, 110 mẫu thai nhi mang bệnh trisomy 21 được phân loại đúng vào lớp thai nhi mang bệnh trisomy 21, 281 mẫu thai nhi mang bệnh trisomy 21 nhưng bị phân loại vào lớp thai nhi bình thường.

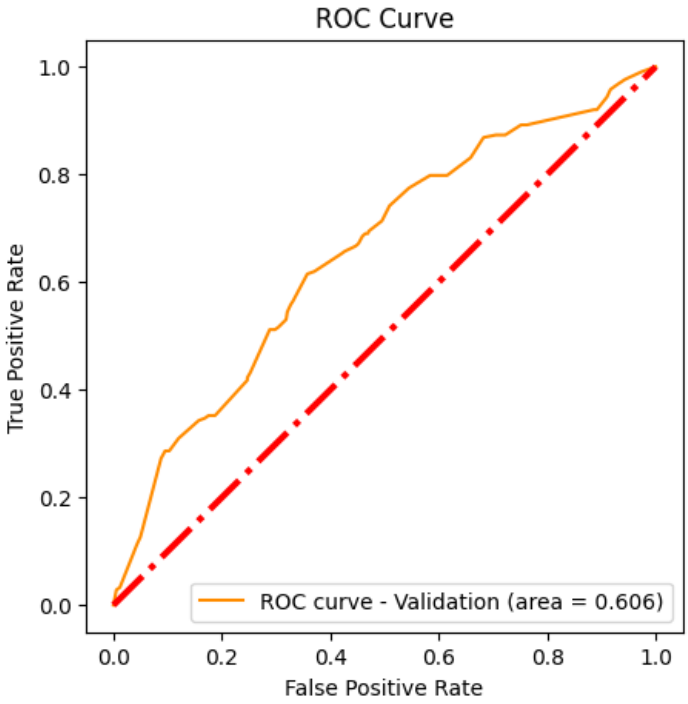
- Classification Report:

```
Classification_report
              precision    recall  f1-score   support

           0       0.86      0.70      0.77       923
           1       0.28      0.52      0.36       213

    accuracy                           0.66      1136
   macro avg       0.57      0.61      0.57      1136
weighted avg       0.75      0.66      0.69      1136
```

Hình 5.19: Classification Report của mô hình LightGBM

Ta thấy mô hình có độ chính xác cao đối với lớp 0 (lớp thai nhi bình thường) với độ chính xác 77%, trong khi độ chính xác thấp hơn với lớp 1 (lớp thai nhi mang bệnh trisomy 21) chỉ đạt độ chính xác khoảng 36%. Kết quả cho thấy mô hình có xu hướng phân lớp sai các mẫu thuộc lớp 1.

- Đường cong ROC Curve:



Hình 5.20: Đường cong ROC Curve của mô hình LightGBM

Qua tính toán, phần diện tích dưới đường cong ROC (AUC) bằng 0.606, cho thấy hiệu suất của mô hình LightGBM trong bài toán phân loại nhị phân này được đánh giá ở mức trung bình. Hơn nữa, đường cong ROC của mô hình LightGBM nằm phía trên bên trái đường thẳng $y = x$ trong đồ thị, cho thấy mô hình có độ nhạy (sensitivity) cao hơn độ đặc hiệu (specificity).

Tổng kết: Qua quá trình huấn luyện, tôi nhận thấy thời gian huấn luyện mô hình LightGBM khá lâu và mô hình tôi thu được đánh giá cũng đã có thể phân lớp tương đối chính xác các mẫu bình thường. Tuy nhiên độ chính xác và khả năng phân lớp của mô hình đối với các mẫu thai nhi mang bệnh còn khá thấp, được đánh giá chỉ trên mức trung bình. Mô hình chưa có sự phân lớp rõ rệt giữa hai lớp trong bài toán phân loại nhị phân này.

5.3 Triển khai và xây dựng mô hình FT-Transformer (Feature Tokenizer - Transformer)

Trong quá trình triển khai đồ án, tôi đã xây dựng ba mô hình khác nhau (Hồi quy Logistic, XGBoost và LightGBM) để so sánh và đánh giá hiệu suất ban đầu. Kết quả của từng mô hình tôi đã có trình bày trong phần 5.2 và đã có được cái nhìn tổng quan về khả năng phân lớp của từng mô hình.

Tuy nhiên, nhìn chung các mô hình dự đoán truyền thống này vẫn còn hạn chế

đối với việc phân lớp dữ liệu trong bài toán của tôi. Và các mô hình học sâu hiện có chuyên về xử lý dữ liệu dạng bảng liên tục đưa ra các kiến trúc vẫn còn mang tính lý thuyết và các kết quả vẫn còn mâu thuẫn nhau trên các bộ dữ liệu khác nhau, và kết quả là cả nhà nghiên cứu đều không rõ mô hình nào hoạt động tốt nhất. Trước tình trạng đó, Yury Gorishniy và Ivan Rubachev đã giới thiệu mô hình FT-Transformer — là một sự điều chỉnh đơn giản của kiến trúc Transformer cho dữ liệu dạng bảng và trở thành một giải pháp mạnh mẽ mới cho thực tế. Mô hình được một kiến trúc phổ quát hơn và hoạt động tốt trên nhiều tác vụ hơn các mô hình học sâu khác [19], tuy nhiên do mới được công bố nên vẫn còn còn hạn chế trong việc sử dụng rộng rãi.

Với hi vọng có thể khắc phục được ba mô hình bên trên, tôi quyết định triển khai mô hình Feature Tokenizer - Transformer cho bài toán này với hi vọng cung cấp một cách tiếp cận toàn diện và mạnh mẽ để đối mặt với thách thức của dữ liệu bản ghi y tế.

5.3.1 Chuẩn bị dữ liệu

Trong quá trình đầu tiên, từ các file dữ liệu đã được tiền xử lý trước đó, tôi chia bộ dữ liệu thành ba tập (train, validation và test dataset), sau đó thực hiện chuyển đổi kiểu dữ liệu từ số thực float64 thành float32 và reshape nhãn để đảm bảo tính đồng nhất và hiệu suất tối ưu.

Tiếp đó, tôi xây dựng một hàm RTDL để chuẩn bị dữ liệu cho mô hình PyTorch cần xây dựng, có nhiệm vụ chuyển thuộc tính và nhãn sang dạng Tensor của PyTorch. Tiếp tục, bằng lớp Dataset của PyTorch tôi sử dụng để dựng các đối tượng dữ liệu cho các tập train, validation và test.

Cuối cùng, tôi xây dựng lớp create_Tensor để tạo đối tượng Dataset, sau đó sử dụng lớp DataLoader để dễ dàng xử lý dữ liệu theo lô trong quá trình huấn luyện mô hình.

5.3.2 Xây dựng kiến trúc mô hình

Sau khi chuẩn bị xong dữ liệu, tôi đưa ra mô hình FT-Transformer để thực hiện bài toán phân loại dữ liệu dạng bảng. Tôi đưa ra mô hình với cấu hình như sau:

- $n\_num\_features$: Số lượng thuộc tính dữ liệu số (numeric features).
- $cat\_cardinalities = []$: Danh sách thuộc tính phân loại (categorical features).
- $d\_token = 64$: Kích thước của các vectơ nhúng token.
- $n\_blocks = 3$: Số lượng khối transformer.
- $attention\_dropout = 0.2$: Tỷ lệ dropout cho các lớp attention.

- $ffn\_d\_hidden = 128$: Kích thước lớp ẩn trong mạng feedforward.
- $ffn\_dropout = 0.1$: Tỷ lệ dropout cho các lớp mạng feedforward.
- $residual\_dropout = 0.1$: Tỷ lệ dropout cho các kết nối residual.
- d_out: Kích thước đầu ra.

Quá trình huấn luyện sau được được lặp qua $epochs$, và việc đánh giá hiệu suất đều được thực hiện trên tập huấn luyện và tập xác thực sau mỗi $epoch$. Hàm mất mát tôi chọn trong lần huấn luyện này là Binary Cross Entropy Loss (BCELoss), phù hợp với bài toán phân loại nhị phân này.

$$\mathcal{L}(y, \hat{y}) = -w\left[y * \log \hat{y} + (1 - y) * \log(1 - \hat{y})\right]$$

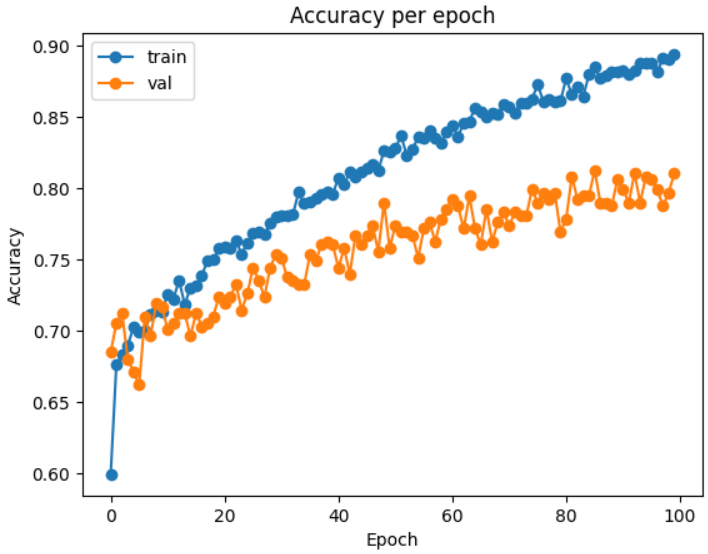
Trong đó: w: Trọng số mô hình

y: Nhãn thực tế ($label$), $\hat{y}$: Giá trị dự đoán

Trong quá trình huấn luyện trên tập huấn luyện thông qua hàm training(), giá trị của hàm mất mát ($loss$)được tính toán dựa trên giá trị dự đoán của mô hình (out) và nhãn bệnh thực tế ($label$) cho từng lô dữ liệu, sau đó giá trị $loss$ được sử dụng để cập nhật trọng số qua quá trình lan truyền ngược. Trong quá trình đưa ra dự đoán, tôi có đặt một ngưỡng 0.5 cho hàm sigmoid để phân ngưỡng dự đoán cho hai nhãn lớp. Sau mỗi epoch, tôi sẽ lưu lại trạng thái của mô hình nếu giá trị hàm mất mát trên tập xác thực giảm, để đảm bảo lưu lại trạng thái tốt nhất của mô hình.

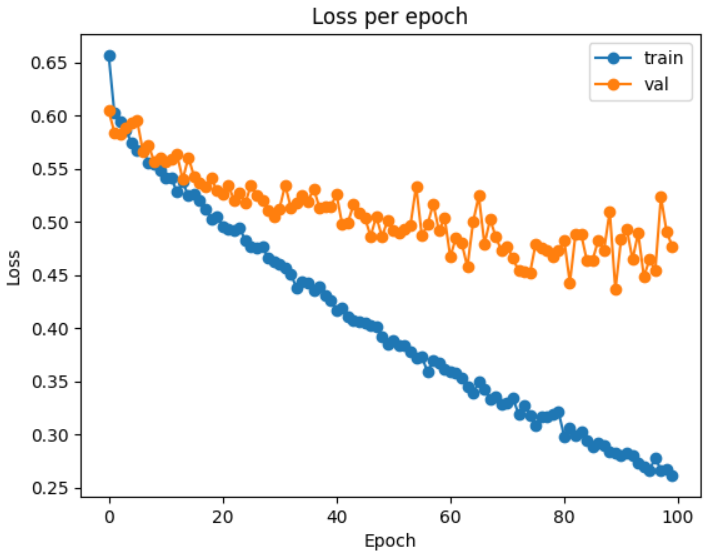
5.3.3 Đánh giá kết quả sau quá trình huấn luyện

Sau khi huấn luyện mô hình, tôi đã thu được các giá trị độ chính xác (accuracy) và độ mất mát (loss) được trực quan hoá thành biểu đồ như sau:



Hình 5.21: Độ chính xác trong quá trình huấn luyện mô hình FT-Transformer

Dựa vào biểu đồ hình 5.21, tôi nhận thấy nhìn chung độ chính xác trên tập huấn luyện tăng dần theo số epoch, độ chính xác trên tập xác thực đạt cực đại sau khoảng 60 epoch, và hầu như độ chính xác trên tập huấn luyện đều cao hơn độ chính xác trên tập xác thực. Biểu đồ cho thấy mô hình đang được huấn luyện và có sự cải thiện trên tập huấn luyện. Mô hình đã đạt được độ chính xác cao nhất trên tập xác thực sau khoảng 60 epoch, sau đó độ chính xác bắt đầu giảm, cho thấy mô hình đã bắt đầu học các thuộc tính không phù hợp với tập dữ liệu thực tế.



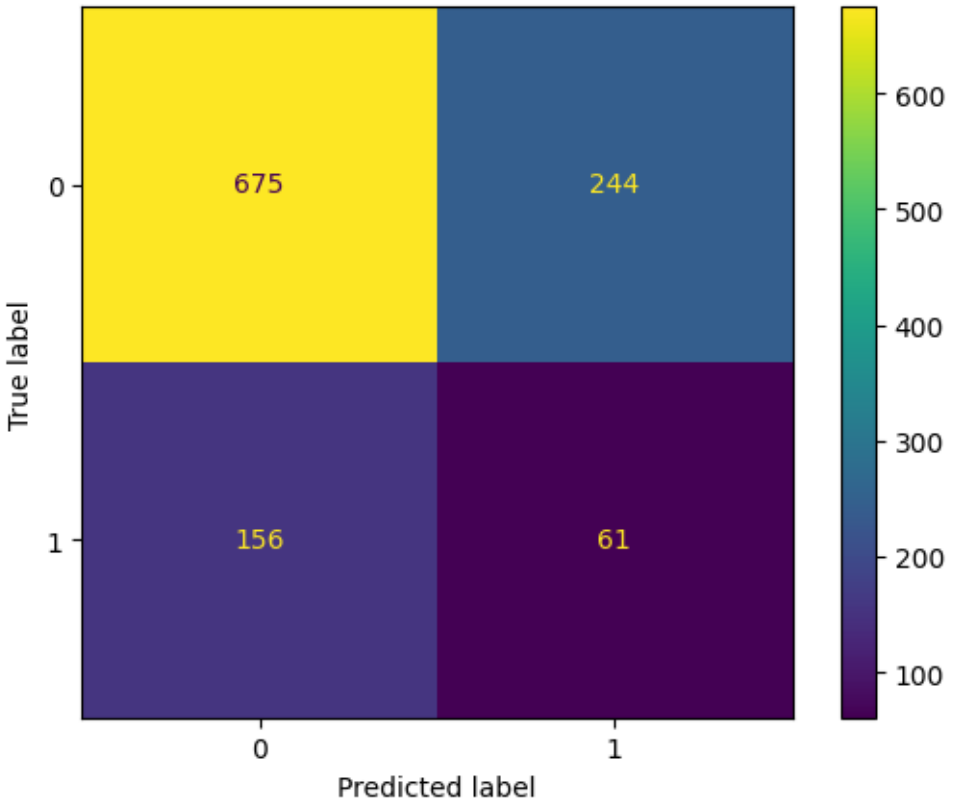
Hình 5.22: Độ mất mát trong quá trình huấn luyện mô hình FT-Transformer

Tương tự, qua phân tích biểu đồ mất mát hình 5.22, tôi nhận thấy độ mất mát trên tập huấn luyện đang giảm dần theo epoch, cụ thể là giảm từ 0.65 xuống 0.25 sau 100 epoch, cho thấy mô hình đang được huấn luyện và dần dần cải thiện độ chính xác của nó. Tuy nhiên, ban đầu độ mất mát trên tập xác thực cũng giảm dần, nhưng sau đó bắt đầu tăng lên, chứng tỏ mô hình đang bắt đầu quá khớp với dữ liệu huấn luyện, và không còn cải thiện độ chính xác trên những dữ liệu mới.

Để đánh giá độ chính xác và độ mất mát trên tập kiểm thử, tôi đã sử dụng trạng thái của mô hình trước đó đã được huấn luyện để đánh giá và thu được kết quả độ chính xác khoảng 71.65% và độ mất mát khoảng 74.65%. Cụ thể hơn về các giá trị đo hiệu suất mô hình trên tập kiểm thử tôi thu được như sau:

- $Accuracy = 0.648$: Mô hình dự đoán chính xác khoảng 64% các giá trị thực tế.
- $Balanced\ Accuracy\ Score = 0.508$: Chứng tỏ mô hình có khả năng dự đoán cả hai lớp một cách tương đối cân bằng.
- $Precision = 0.200$: Chỉ 20% trong số các dự đoán ở lớp thai nhi mang bệnh là đúng, cho thấy có thể có sự thiếu chắc chắn trong dự đoán ở lớp này.

- $Recall = 0.281$: Mô hình chỉ phát hiện được 28.1% các mẫu thực tế mang bệnh, chứng tỏ hiệu suất của mô hình giảm khi phân loại vào lớp thực sự mang bệnh.
- $F1\ Score = 0.234$: Giá trị này khá thấp, mô hình cần được điều chỉnh để cải thiện hiệu suất tổng hợp giữa precision và recall.
- $ROC\ AUC\ Score = 0.508$: Giá trị tương đối trung bình, nên mô hình không có khả năng phân loại tốt giữa các lớp hoặc có thể cần phải thực hiện điều chỉnh thêm để cải thiện hiệu suất.
- $Sensitivity = 0.281, Specificity = 0.734, G-mean = 0.454$: Giá trị specificity cao, nhưng sensitivity thấp, chứng tỏ mô hình có thể hiệu quả trong việc phân lớp các mẫu bình thường, nhưng ít hiệu quả khi phân lớp các mẫu mang bệnh.
- Ma trận nhầm lẫn:



Hình 5.23: Ma trận nhầm lẫn của mô hình FT-Transformer

Với kết quả ma trận nhầm lẫn trên hình 5.23, mô hình đã phân lớp 675 mẫu thai nhi bình thường được phân loại đúng vào lớp thai nhi mang bình thường, 156 mẫu thai nhi bình thường nhưng bị phân loại vào lớp thai nhi mang bệnh trisomy 21, 61 mẫu thai nhi mang bệnh trisomy 21 được phân loại đúng vào lớp thai nhi mang bệnh trisomy 21, 244 mẫu thai nhi mang bệnh trisomy 21 nhưng bị phân loại vào lớp thai nhi bình thường.

- Classification Report:

```
Classification_report
              precision    recall  f1-score   support

           0       0.81      0.73      0.77       919
           1       0.20      0.28      0.23       217

    accuracy                           0.65      1136
   macro avg       0.51      0.51      0.50      1136
weighted avg       0.70      0.65      0.67      1136
```

Hình 5.24: Classification Report của mô hình FT-Transformer

Nhìn chung, mô hình Feature Tokenizer Transformer có độ chính xác khá tốt, với độ chính xác macro là 0.51 và độ chính xác weighted là 0.70. Tuy nhiên, độ chính xác của mô hình vẫn có thể được cải thiện hơn nữa, đặc biệt là cần cải thiện trên lớp thai nhi mang bệnh.

5.3.4 Tinh chỉnh tham số và tối ưu hoá mô hình

Qua kết quả đánh giá mô hình thu được và đã phân tích trong phần 5.3.3, tôi tiếp tục tinh chỉnh siêu tham số của mô hình FT-Transformer để có thể đạt được hiệu suất tốt nhất hơn. Tôi đã sử dụng thư viện Optuna để tự động thực hiện quá trình tinh chỉnh này.

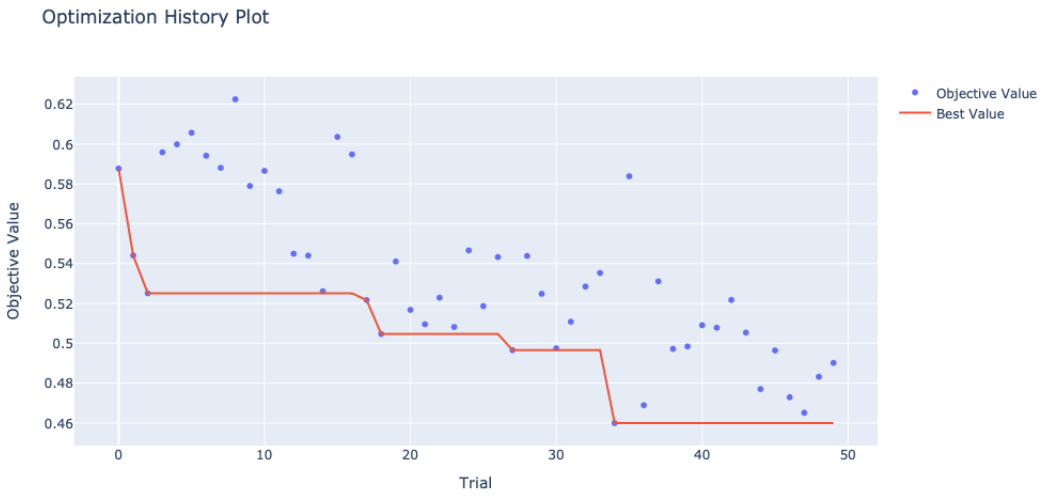
Danh sách tham số tôi đưa ra tinh chỉnh bao gồm: d_token, n_blocks, attention_dropout, ffn_d_hidden, ffn_dropout, residual_dropout, lr, weight_decay. Sau mỗi lần thử nghiệm trên từng giá trị tham số khác nhau, một mô hình mới sẽ được hình thành và được huấn luyện trên tập huấn luyện, sau đó được đánh giá trên tập kiểm thử để đo lường hiệu suất mô hình mới.

Kết quả giá trị tối ưu của tham số tôi thu được như sau:

- $d\_token = 128$: mô hình sử dụng các token có kích thước lớn hơn để biểu diễn dữ liệu đa dạng hơn.
- $n\_blocks = 4$: Sự kết hợp của 4 khối transformer là đủ để mô hình có khả năng học các mức độ biểu diễn phức tạp của dữ liệu.
- $attention\_dropout = 0.05633571882908772$: Kiểm soát quá trình học thông tin từ các token trước đó, tránh quá mức tập trung vào một số token cụ thể.
- $ffn\_d\_hidden = 529$: Có thể ảnh hưởng đến khả năng học các thuộc tính phức tạp từ dữ liệu.
- $ffn\_dropout = 0.15680906610269654$: Giảm nguy cơ overfitting.

- $residual\_dropout = 0.023306414142002692$: Có thể giúp kiểm soát sự đóng góp của residual connections trong quá trình huấn luyện.
- $lr = 0.00018315266700252362$: Tốc độ học tập, mức độ thay đổi trọng số của mô hình trong mỗi bước tối ưu hóa.
- $weight\_decay = 4.7121678184506015e - 06$: Giảm nguy cơ overfitting bằng cách kiểm soát kích thước của các trọng số.

Kết quả của quá trình tinh chỉnh này với Best Score khoảng 0.46, có thể thấy mô hình được cải thiện so với các giá trị mặc định ban đầu, và có khả năng tổng quát hóa tốt hơn trên dữ liệu mới với các giá trị mục tiêu là độ lỗi trên tập xác thực.



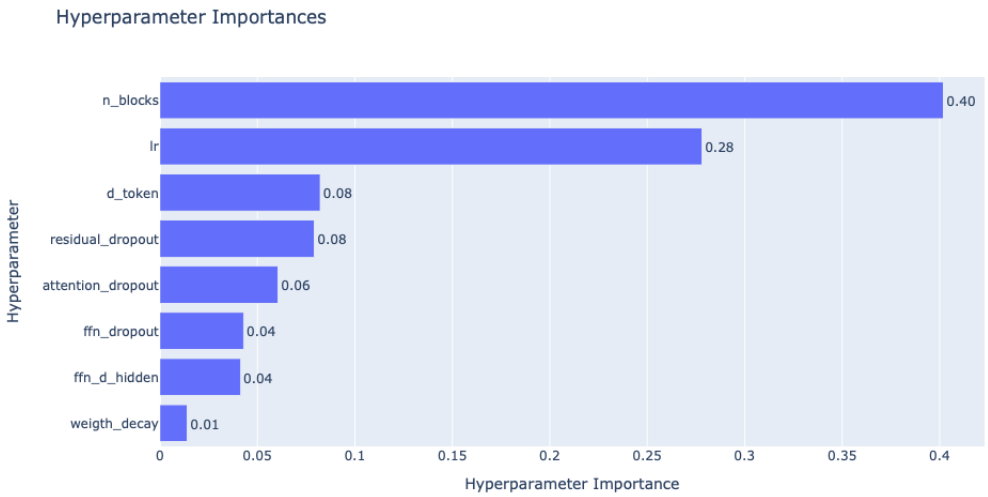
Hình 5.25: Biểu đồ lịch sử tối ưu hoá mô hình FT-Transformer của Optuna

Biểu đồ lịch sử tối ưu hóa trên hình 5.25 cho thấy Optuna đã có thể tìm thấy các giá trị siêu tham số tốt cho mô hình một cách hiệu quả.

Giá trị mục tiêu nhìn chung đều giảm theo số lần thử nghiệm tham số, chứng tỏ Optuna đang thành công khi tìm kiếm các cấu hình siêu tham số tốt hơn, do Optuna sử dụng một thuật toán tối ưu hóa tìm kiếm các giá trị tham số dẫn đến giá trị mục tiêu thấp nhất. Lần thử nghiệm tốt nhất đạt được giá trị mục tiêu là 0.46.

Phương sai của các giá trị mục tiêu tương đối nhỏ, cho thấy quá trình tối ưu hóa không quá nhạy cảm với nhiễu ngẫu nhiên, là do Optuna sử dụng kỹ thuật giảm nhiễu ngẫu nhiên như chạy nhiều lần thử nghiệm và sử dụng các kỹ thuật cắt tỉa để loại bỏ các lần thử nghiệm có khả năng không cải thiện giá trị mục tiêu.

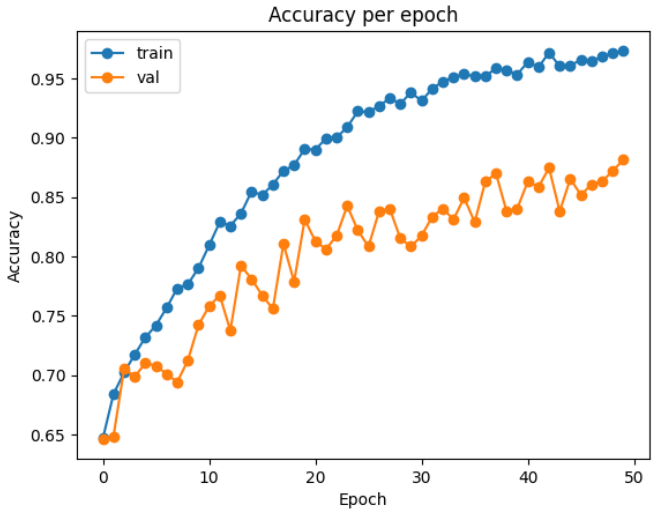
Quá trình tối ưu có một số lần thử nghiệm bị dừng sớm, do Optuna đã tìm thấy các cấu hình siêu tham số tốt hơn với số lần thử nghiệm ít hơn.



Hình 5.26: Biểu đồ đánh giá tầm quan trọng của từng siêu tham số đối với giá trị mục tiêu

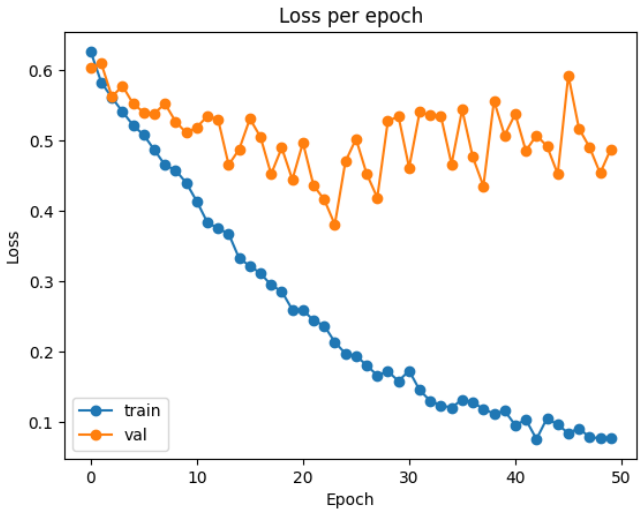
Biểu đồ hình 5.26 đã sắp xếp các siêu tham số theo thứ tự giảm dần tầm quan trọng. Các siêu tham số với độ quan trọng nhất là: $n\_blocks : 0.40, lr : 0.28$. Các siêu tham số này có tầm quan trọng cao nhất đối với giá trị mục tiêu, vì vậy chúng là những tham số quan trọng nhất cần xem xét khi tinh chỉnh mô hình. Các siêu tham số còn lại có thể vẫn có tác động đáng kể đến hiệu suất của mô hình, nhưng chúng không quan trọng như các siêu tham số quan trọng nhất.

Sau quá trình tinh chỉnh tham số, tôi khởi tạo lại mô hình FT-Transformer với các tham số được tinh chỉnh từ kết quả của quá trình tối ưu hóa trước đó, tiếp tục sử dụng hàm mất mát BCELoss cho bài toán phân loại nhị phân, và thực hiện tối ưu hóa mô hình sử dụng thuật toán AdamW với learning rate (lr) và hệ số giảm trọng lượng ($weight\_decay$) được lấy từ kết quả tinh chỉnh.



Hình 5.27: Độ chính xác trong quá trình huấn luyện mô hình với siêu tham số tinh chỉnh

Từ biểu đồ độ chính xác trên hình 5.27, có thể thấy rằng sau khi huấn luyện mô hình với bộ siêu tham số được tinh chỉnh, độ chính xác của mô hình trên tập huấn luyện và tập xác thực đều có xu hướng tăng dần theo số epoch. Độ chính xác trên tập train đạt cực đại khoảng 0.95 sau khoảng 40 epoch, và độ chính xác trên tập xác thực đạt cực đại khoảng 0.8 sau 50 epoch. Tuy nhiên, độ chính xác trên tập xác thực hầu hết thấp hơn độ chính xác trên tập huấn luyện, chứng tỏ mô hình có thể đang bị overfitting.



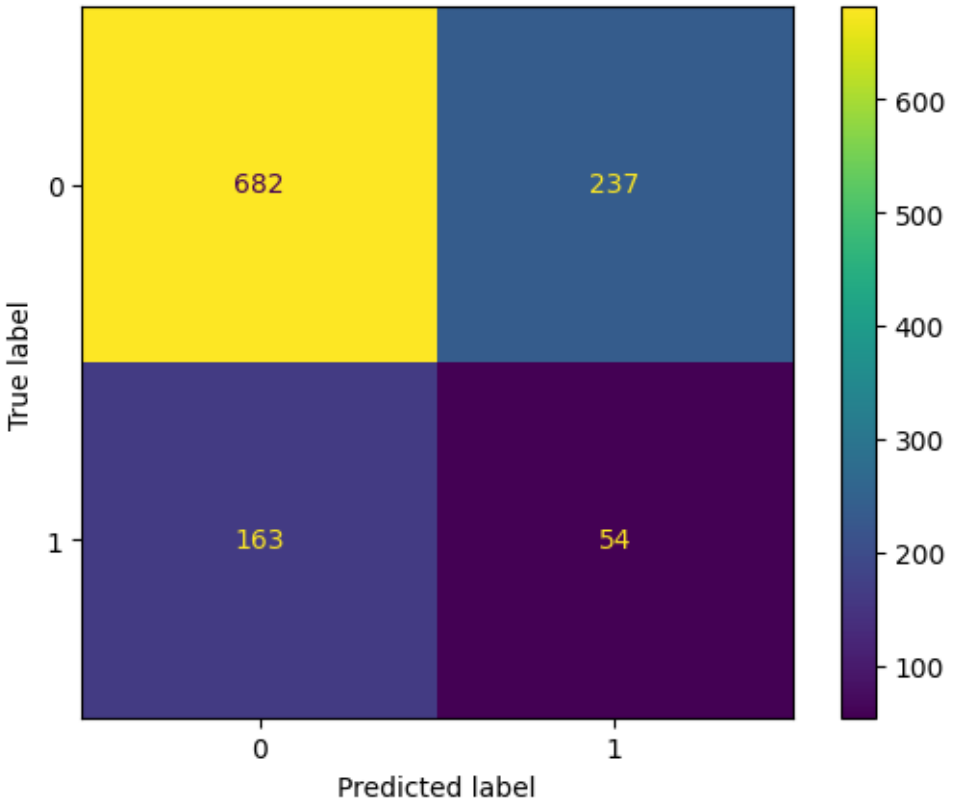
Hình 5.28: Độ mất mát trong quá trình huấn luyện mô hình với siêu tham số tinh chỉnh

Với biểu đồ độ mất mát được trực quan trên hình 5.28, khi huấn luyện mô hình mới theo số epoch tăng dần, nhìn chung độ mất mát của mô hình trên tập huấn luyện và tập xác thực đều có xu hướng giảm dần. Độ mất mát trên tập huấn luyện đạt cực tiểu khoảng 0.05 sau 42 epoch, trong khi độ mất mát trên tập xác thực đạt mức cực tiểu khoảng 0.37 sau 23 epoch.

Kết quả đánh giá của tôi với các phép đo hiệu suất của mô hình FT-Transformer với bộ siêu tham số được tinh chỉnh như sau:

- $Accuracy = 0.648, Balanced\ Accuracy\ Score = 0.495$: Giá trị Accuracy tương đối khá, nhưng giá trị Balanced Accuracy Score dưới mức ngưỡng trung bình, có thể đây là một mô hình không hoạt động hiệu quả trên cả hai lớp dự đoán
- $Precision = 0.186$: Chỉ có khoảng 18.6% trong số các dự đoán dương tính (mang bệnh trisomy 21) của mô hình là chính xác.
- $Recall = 0.249$: Mô hình chỉ phát hiện được khoảng 24.9% của tất cả các mẫu thực sự dương tính.
- $F1\ Score = 0.213$: Giá trị khá thấp, chứng tỏ mô hình không đạt được sự cân bằng tốt giữa Precision và Recall.

- $ROC\ AUC\ Score = 0.495$: Giá trị dưới ngưỡng 0.5 nên thường được coi là mô hình không có khả năng phân loại tốt hơn so với dự đoán ngẫu nhiên.
- $Sensitivity = 0.249$: Mô hình chỉ phát hiện được khoảng 24.9% trong số các mẫu thực sự dương tính. Giá trị Sensitivity thấp, có thể làm giảm khả năng của mô hình trong việc phát hiện các mẫu quan trọng.
- $Specificity = 0.742$: Mô hình có khả năng loại bỏ khoảng 74.2% các trường hợp thực sự âm tính. Giá trị này khá cao nên mô hình hoạt động tốt tốt trong việc loại bỏ các mẫu không thực sự âm tính.
- $G - mean = 0.430$: Giá trị này thấp, chứng tỏ ô hình không đạt được sự cân bằng tốt giữa Sensitivity và Specificity.
- Ma trận nhầm lẫn:



Hình 5.29: Ma trận nhầm lẫn của mô hình FT-Transformer với bộ tinh chỉnh siêu tham số

Với kết quả ma trận nhầm lẫn trên hình 5.29, mô hình đã phân lớp 682 mẫu thai nhi bình thường được phân loại đúng vào lớp thai nhi mang bình thường, 163 mẫu thai nhi bình thường nhưng bị phân loại vào lớp thai nhi mang bệnh trisomy 21, 54 mẫu thai nhi mang bệnh trisomy 21 được phân loại đúng vào lớp thai nhi mang bệnh trisomy 21, 237 mẫu thai nhi mang bệnh trisomy 21 nhưng bị phân loại vào lớp thai nhi bình thường.

- Classification Report:

```
Classification_report
              precision    recall  f1-score   support

           0       0.81      0.74      0.77       919
           1       0.19      0.25      0.21       217

    accuracy                           0.65      1136
   macro avg       0.50      0.50      0.49      1136
weighted avg       0.69      0.65      0.67      1136
```

Hình 5.30: Classification Report mô hình FT-Transformer với bộ tinh chỉnh siêu tham số

Nhìn chung, Classification Report của mô hình Feature Tokenizer Transformer sau khi tinh chỉnh là khá tốt. Độ chính xác macro là 0.5 và độ chính xác weighted là 0.69, là một con số chấp nhận được đối với các mô hình phân loại. Precision và recall của lớp 0 cũng khá cao, cho thấy mô hình có thể phân loại chính xác các mẫu thai nhi bình thường. Tuy nhiên, precision và recall của lớp 1 còn thấp, cho thấy mô hình có thể phân loại sai các mẫu thai nhi mang bệnh trisomy 21.

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Dưới sự hướng dẫn của TS. Nguyễn Hồng Quang, tôi đã lựa chọn đề tài "Dự đoán bệnh di truyền của thai nhi dựa trên dữ liệu bản ghi y tế", tập trung đặc biệt vào phân loại hội chứng down (trisomy 21) sử dụng dữ liệu bản ghi y tế làm đề tài đồ án tốt nghiệp cử nhân, tôi đã có những kết quả nghiên cứu và khám phá những khả năng mới trong ứng dụng trí tuệ nhân tạo trong lĩnh vực y tế.

Trong những tuần đầu tiên của học kì, sau khi tìm hiểu sơ bộ về nghiệp vụ y học liên quan, tôi đã tiến hành thu thập và xử lý dữ liệu bản ghi y tế. Nhờ sự hướng dẫn của TS. Nguyễn Hồng Quang và bác sĩ, tôi đã thu thập được một tập dữ liệu đa dạng và đại diện, phản ánh đầy đủ tính đa chiều của y tế thai nhi từ Bệnh viện Phụ sản Trung ương (Việt Nam). Quá trình này với tôi không chỉ là công việc kỹ thuật ban đầu của đồ án mà còn là sự tìm kiếm sự hiểu biết sâu sắc về lĩnh vực y học. Tuy nhiên, bộ dữ liệu tôi được cung cấp với số lượng bản ghi và số lượng thuộc tính dữ liệu khá lớn, thêm vào đó là tình trạng khuyết dữ liệu trầm trọng và dữ liệu mất cân bằng giữa hai lớp. Để giải quyết những tình trạng đó, tôi đã sử dụng các kỹ thuật học máy tiên tiến nhưu lọc bỏ ban ghi, trích chọn thuộc tính, KNNIMputation, oversampling SMOTE, và đạt được các kết quả với bộ dữ liệu hoàn chỉnh 100% các bản ghi không bị khuyết dữ liệu, số chiều dữ liệu được giảm đảng kể và đạt được sự cân bằng số lượng bản ghi giữa hai lớp.

Kết thúc quá trình tiền xử lý dữ liệu, tôi bắt đầu quá trình triển khai và xây dựng ba mô hình ban đầu (Logistic Regression, XGBoost và LightGBM) để có được sự so sánh và phân tích chi tiết hiệu suất của mỗi mô hình đối với bài toán này. Đầu tiên tôi lựa chọn thuật toán hồi quy Logistic bởi tính đơn giản và khả năng giải thích cao của mô hình. Trong khi đó, XGBoost và LightGBM được tôi lựa chọn với hy vọng sẽ cung cấp độ chính xác và hiệu suất tốt hơn, đặc biệt là khi đối mặt với dữ liệu lớn và phức tạp. Nhìn chung với kết quả thu được, tôi đánh giá cả ba mô hình đều đạt được hiệu suất ổn định có thể phân lớp rõ rệt với độ chính xác từ $60\% - 70\%$, đặc biệt là XGBoost và LightGBM nhờ khả năng xử lý mất cân bằng lớp và khả năng tìm kiếm tối ưu thông qua quá trình học tăng cường cây quyết định. Tuy nhiên, thách thức lớn nhất mà tôi gặp phải với hai mô hình XGBoost và LighGBM là khó kiểm soát hiện tượng quá khớp (overfitting) và tinh chỉnh tham số để đảm bảo cân bằng giữa độ chính xác và sự tổng quát của mô hình.

Sau khi có được sự so sánh cơ bản giữa ba mô hình trên, tôi đã triển khai áp dụng mô hình Feature Tokenizer Transformer vào quá trình huấn luyện. Với sự kết

hợp giữa mô hình học máy và mô hình transformer, kết quả cho tôi thu được một mô hình tối ưu hóa có thể trích xuất đặc trưng từ dữ liệu bản ghi y tế. Về tổng quát, mô hình Feature Tokenizer Transformer sau khi tinh chỉnh đạt được độ chính xác khoảng 50% và có hiệu quả tốt trong việc phân loại thai nhi bình thường, nhưng vẫn tồn tại hạn chế đối với việc phân loại các mẫu mang bệnh trisomy 21. Do đó, điều này gây lo lắng trong việc sử dụng mô hình để đưa ra quyết định chẩn đoán cho thai nhi mang bệnh trisomy 21 và xem xét kỹ lưỡng về quyết định chẩn đoán khi sử dụng mô hình này trong thực tế y học.

Kết thúc quá trình thực hiện đồ án tốt nghiệp, tôi nhận thấy đây là khoảng thời gian như được làm việc trong một dự án thực tế, không chỉ giúp tôi tổng kết lại kiến thức của mình trong những năm tháng học tại Đại học Bách khoa Hà Nội, mà còn là cơ hội cho tôi học hỏi thêm được những kỹ năng thực tế như phân tích vấn đề, giao tiếp nghiên cứu và báo cáo trình bày khoa học.

6.2 Hướng phát triển trong tương lai

Để nâng cao chất lượng cuối cùng của kết quả đầu ra đồ án, tôi dự định sẽ tiếp tục tối ưu hóa mô hình Feature Tokenizer Transformer. Quá trình tinh chỉnh sẽ được thực hiện liên tục, với việc nghiên cứu các phương pháp mới và tiên tiến trong lĩnh vực học máy, có thể cải thiện độ chính xác và khả năng dự đoán đối với lớp dữ liệu thiểu số, đồng thời đảm bảo tính ổn định của mô hình.

Hơn thế nữa, trong quá trình thực hiện đồ án, tôi nhận thấy vấn đề còn tồn đọng tốn nhiều thời gian và khó khăn nhất là khi xử lý dữ liệu khuyết. Tôi dự định sẽ tìm kiếm và triển khai các phương pháp mới nhằm giảm thiểu ảnh hưởng của giá trị khuyết đối với khả năng huấn luyện của mô hình. Điều này sẽ đảm bảo mô hình được huấn luyện trên dữ liệu đầy đủ và chính xác.

Sau đó, tôi sẽ triển khai các mô hình học sâu phức tạp hơn, ví dụ như sử dụng mô hình kết hợp để tận dụng tối đa khả năng dự đoán của các mô hình khác nhau. Việc này có thể sẽ cải thiện khả năng dự đoán và đồng thời giảm thiểu nguy cơ overfitting. Tôi hi vọng sự kết hợp linh hoạt giữa các mô hình có thể mang lại hiệu suất toàn diện và đáng tin cậy hơn.

Và trong tương lai xa hơn, tôi đặt ra mục tiêu có thể ứng dụng mô hình và kết quả trong thực tế. Kết quả này sẽ không chỉ là một nghiên cứu trong các phòng thí nghiệm hay trung tâm, mà còn là một nguồn thông tin hữu ích và hỗ trợ trong quá trình chẩn đoán bệnh di truyền của thai nhi trong lĩnh vực y học. Tôi hi vọng dự định này sẽ có thể giúp tôi tiếp tục học hỏi, chia sẻ kiến thức và đóng góp vào sự phát triển của lĩnh vực di truyền học và y học.

TÀI LIỆU THAM KHẢO

[1] F. He, B. Lin, K. Mou, L. Jin, and J. Liu, "A machine learning model for the prediction of down syndrome in second trimester antenatal screening," *Clinica Chimica Acta*, vol. 521, pp. 206–211, 2021.

[2] NOVAGEN Genetic Technology LLC, *Trisomy là gì?* [Online]. Available: `https://novagen.vn/trisomy-la-gi` (visited on 04/16/2021).

[3] A. Jamshidnezhad, S. M. Hosseini, J. Mohammadi-Asl, and M. Mahmudi, "An intelligent prenatal screening system for the prediction of trisomy-21," *Informatics in Medicine Unlocked*, vol. 24, p. 100 625, 2021.

[4] A. C. Neocleous, K. H. Nicolaides, and C. N. Schizas, "First trimester non-invasive prenatal diagnosis: A computational intelligence approach," *IEEE Journal of biomedical and Health informatics*, vol. 20, no. 5, pp. 1427–1438, 2015.

[5] A. Catic, L. Gurbeta, A. Kurtovic-Kozaric, S. Mehmedbasic, and A. Badnjevic, "Application of neural networks for classification of patau, edwards, down, turner and klinefelter syndrome based on first trimester maternal serum screening data, ultrasonographic findings and patient demographics," *BMC medical genomics*, vol. 11, no. 1, pp. 1–12, 2018.

[6] A. Durmuşoğlu, M. M. Ay, and Z. D. Unutmaz Durmuşoğlu, "A classification model for predicting fetus with down syndrome–a study from turkey," *Applied Artificial Intelligence*, vol. 34, no. 12, pp. 898–915, 2020.

[7] S. Ramanathan, M Sangeetha, S. Talwai, and S Natarajan, "Probabilistic determination of down's syndrome using machine learning techniques," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2018, pp. 126–132.

[8] H.-G. Zhang, Y.-T. Jiang, S.-D. Dai, L. Li, X.-N. Hu, and R.-Z. Liu, "Application of intelligent algorithms in down syndrome screening during second trimester pregnancy," *World Journal of Clinical Cases*, vol. 9, no. 18, p. 4573, 2021.

[9] A. Neocleous, A Syngelaki, K. Nicolaides, and C. N. Schizas, "Two-stage approach for risk estimation of fetal trisomy 21 and other aneuploidies using computational intelligence systems," *Ultrasound in Obstetrics & Gynecology*, vol. 51, no. 4, pp. 503–508, 2018.

[10] J. Marttala, S. Peuhkurinen, J. K. Ranta, *et al.*, "Screening and outcome of chromosomal abnormalities other than trisomy 21 in northern finland," *Acta obstetricia et gynecologica scandinavica*, vol. 90, no. 8, pp. 885–889, 2011.

[11] M. T. Khattak, E. Supriyanto, M. N. Aman, and R. H. Al-Ashwal, "Predicting down syndrome and neural tube defects using basic risk factors," *Medical & biological engineering & computing*, vol. 57, pp. 1417–1424, 2019.

[12] A. M. Abed, S. A. Gitaffa, and A. H. Issa, "Quadratic support vector machine and k-nearest neighbor based robust sensor fault detection and isolation," *Eng. Technol. J*, vol. 39, no. 5A, pp. 859–869, 2021.

[13] D. Elreedy, A. F. Atiya, and F. Kamalov, "A theoretical distribution analysis of synthetic minority oversampling technique (smote) for imbalanced learning," *Machine Learning*, pp. 1–21, 2023.

[14] H. Singh, *Basic ensemble techniques in machine learning*. [Online]. Available: `https://www.analyticsvidhya.com/blog/2021/03/basic-ensemble-technique-in-machine-learning/` (visited on 03/10/2021).

[15] M. Shoaran, B. A. Haghi, M. Taghavi, M. Farivar, and A. Emami-Neyestanak, "Energy-efficient classification for resource-constrained biomedical applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 693–707, 2018.

[16] X. Yao, X. Fu, and C. Zong, "Short-term load forecasting method based on feature preference strategy and lightgbm-xgboost," *IEEE Access*, vol. 10, pp. 75 257–75 268, 2022.

[17] G. Ke, Q. Meng, T. Finley, *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[18] T. Næs, K. Kvaal, T. Isaksson, and C. Miller, "Artificial neural networks in multivariate calibration," *Journal of Near Infrared Spectroscopy*, vol. 1, no. 1, pp. 1–11, 1993.

[19] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021.

[20] S. Haghighi, M. Jasemi, S. Hessabi, and A. Zolanvari, "Pycm: Multiclass confusion matrix library in python," *Journal of Open Source Software*, vol. 3, no. 25, p. 729, 2018.

[21] Z. Karimi, "Confusion matrix," *Encycl. Mach. Learn. Data Min., no. October*, pp. 260–260, 2021.

[22] K. Hajian-Tilaki, "Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation," *Caspian journal of internal medicine*, vol. 4, no. 2, p. 627, 2013.

[23] M. Rodríguez-Hernández, R. Pruneda, and J. Rodríguez-Díaz, “Statistical analysis of the evolutive effects of language development in the resolution of mathematical problems in primary school education,” *Mathematics*, vol. 9, no. 10, p. 1081, 2021.