Đánh giá các mô hình máy học trên bài toán phân loại bệnh nhân tiểu đường

Hai Le
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20520481@gm.uit.edu.vn

Truong Nguyen
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20522087@gm.uit.edu.vn

Thien Lai
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20520309@gm.uit.edu.vn

Vy Le
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20520355@gm.uit.edu.vn

***Abstract*—Ngày ngay trên thế toàn giới có số lượng lớn người bị bệnh tiểu đường lớn, nhằm mục đích giảm tải sức ép lên bộ máy y tế, nhóm em sử dụng các mô hình phân loại: Logistic Regression, kNN, Decision Tree và SVM (Support Vector Machine) trên bộ dữ liệu Predict Diabetes giúp phát hiện sớm bệnh nhân có bị bệnh tiểu đường hay không. Kết quả thu được cho thấy khi sử dụng Train/Test Split SVM là phương pháp có kết quả tốt nhất (F1 score: 0.673, Accuracy score: 0.807, Log loss: 6.656 khi sử dụng Stander Scaler) và kết quả phương pháp thấp nhất thuộc về KNN (F1 score: 0.54, Accuracy score: 0.698, Log loss: 10.434 khi không sử dụng Stander Scaler). Về phía KFold là phương pháp có kết quả tốt nhất là Logistic Regression (F1 score: 0.735, Accuracy score: 0.773, Log loss: 0.481 khi sử dụng Stander Scaler) và các phương pháp có kết quả thấp nhất gồm: KNN (Accuracy score: 0.688, Log loss: 3.94 khi không sử dụng Stander Scale) và SVM (F1 score: 0.631 khi không sử dụng Stander Scale).**

I. Introduction

Tiểu đường, là bệnh rối loạn chuyển hóa đặc trưng với biểu hiện lượng đường ở trong máu luôn ở mức cao hơn so với bình thường do cơ thể thiếu hụt về tiết insulin hoặc đề kháng với insulin hoặc cả 2, dẫn đến rối loạn quan trọng về chuyển hóa đường, đạm, mỡ, chất khoáng. Theo thống kê của hội tiểu đường thế giới (IDF) vào năm 2021 trên thế giới có gần 537 người lớn (từ 20 đến 79 tuổi) sống với bệnh tiểu đường và theo dự đoán con số này sẽ tăng lên 634 triệu vào năm 2030 và 783 triệu vào năm 2045. Số lượng người bị tiểu đường lớn gây áp lực không nhỏ lên hệ thống y tế ở nhiều quốc gia. Ngày nay nhờ những tiến bộ về thống kê và máy học chúng ta có thể giảm bớt gánh nặng này nhờ việc áp dụng các mô hình máy học để dự đoán bệnh tiểu đường thông qua các số liệu về sức khỏe của bệnh nhân.

Trong đồ án này, nhóm em sử dụng 4 mô hình phân loại (Classification), bài toán có nhiệm vụ giúp máy tính phân loại các lớp cho điểm dữ liệu bằng cách sử dụng các thuật toán học máy: Logistic Regression, kNN, Decision Tree và SVM (Support Vector Machine) trên bộ dữ liệu Predict Diabetes được puslish ở kaggle vào ngày 09/11/2022 để dự đoán người đó có mắc bệnh hay không. Bộ dữ liệu gồm có các trường: Pregnancies(số lần mang thai), Glucose (chỉ số đường huyết (mg/dL)), BloodPressure (huyết áp (mmHg)), SkinThickness (độ dày da (μm)), Insulin (chỉ số insulin (μmol/L)), BMI (chỉ số khối cơ thể được tính bằng cân nặng (kg) chia cho bình phương của chiều cao (m)), DiabetesPedigreeFunction (khả năng mắc bệnh tiểu đường dựa trên lịch sử gia đình), Age (tuổi) và Outcome (kết quả người đó bị tiểu đường (1) hoặc không bị tiểu đường (0)).

II. Method

A. Logistic Regression

Model thống kê Logictic Regression thường được sử dụng để phân loại và phân tích dự đoán. Trong bài toán phân loại, hồi quy Logistic ước tính xác suất của một sự kiện xảy ra (ví dụ như người xem xác suất quảng cáo sẽ sử dụng sản phẩm) dựa trên một bộ dữ liệu nhất định của các biến độc lập, kết quả thu được là xác suất nằm trong khoảng từ 0 đến 1. Trong hồi quy logistic, một biến đổi logit được áp dụng trên tỷ lệ, đó là xác suất thành công chia cho xác suất thất bại. Điều này cũng thường được gọi là tỷ lệ log hoặc logarit tự nhiên của tỷ lệ và hàm logistic này được biểu thị bằng các công thức sau:

$$logit(pi) = \frac{1}{1 + exp(-p_i)} \tag{1}$$

$$ln(\frac{p_i}{1 - p_i}) = Beta_0 + Beta_1 X_1 + ... + Beta_k X_k \tag{2}$$

Trong phương trình hồi quy logistic này, logit (PI) là biến phụ thuộc x là biến độc lập. Trong mô hình này tham số beta thường được ước tính thông qua ước tính khả năng tối đa (MLE). Phương pháp này kiểm tra các giá trị khác nhau của beta thông qua nhiều lần lặp để tối ưu hóa để phù hợp nhất với tỷ lệ log. Tất cả các lần lặp này tạo ra hàm log likelihood và hồi quy logistic tìm cách tối đa hóa chức năng này để tìm ra tham số tốt nhất. Khi hệ số tối ưu (hoặc các hệ số nếu có

nhiều hơn một biến độc lập), xác suất có điều kiện cho mỗi quan sát có thể được tính toán, ghi lại và tổng hợp lại với nhau để mang lại xác suất dự đoán. Để phân loại nhị phân, xác suất nhỏ hơn 0,5 sẽ dự đoán 0 trong khi xác suất lớn hơn 0,5 sẽ dự đoán 1. Tương tự thế, trong trường hợp dự đoán nhiều lớp khác nhau mỗi lớp sẽ được dự đoán một xác suất khác nhau dự đoán cuối cùng là class có xác suất lớn nhất.

1) Ưu điểm của Logistic Regression:

- Hồi quy logistic dễ thực hiện, giải thích và thường đạt kết quả tốt.
- Phân loại các điểm dữ liệu không xác định rất nhanh
- Hồi quy logistic ít over-fitting.

2) Nhược của Logistic Regression:

- Hồi quy logistic cần các biến độc lập có liên quan tuyến tính với tỷ lệ ln($\frac{p_i}{1-p_i}$).
- Các vấn đề phi tuyến tính không thể được giải quyết bằng hồi quy logistic.
- Hồi quy logistic có thể bị over-fitting nếu bộ dữ liệu quá nhiều chiều (L1 và L2 có thể sử dụng để tránh các trường hợp này).

B. kNN - K Nearest Neighbors

K-nearest neighbor là một trong những thuật toán học có giám sát đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Máy Học. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán học có giám sát là Phân lớp và Hồi quy. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning. Vì bài toán mà nhóm em quan tâm là Phân lớp nên sẽ không đề cập đến những tính chất cảu bài toán hồi quy trong thuật toán.

Với KNN, trong bài toán Phân lớp, nhãn của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong tập huấn luyện (training set). Nhãn của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra nhãn.

Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong tập training gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu. Để biết được rằng điểm đó gần hay không gần với những điểm dữ liệu trong tập training thì ta phải hiểu được khoảng cách trong không gian vector.

Có một điều đáng lưu ý là KNN phải **nhớ** tất cả các điểm dữ liệu training, việc này không được lợi về cả bộ nhớ và thời gian tính toán.

1) Khoảng cách trong không gian vector: Trong không gian một chiều, khoảng cách giữa hai điểm là trị tuyệt đối giữa hiệu giá trị của hai điểm đó. Trong không gian nhiều chiều, khoảng cách giữa hai điểm có thể được định nghĩa bằng nhiều hàm số khác nhau, trong đó độ dài đường thẳng nối hai điểm chỉ là một trường hợp đặc biệt trong đó.

2) Ưu điểm của kNN:

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

3) Nhược điểm của kNN:

- KNN rất nhạy cảm với nhiễu khi K nhỏ.
- Như đã nói, KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới từng điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu. Với K càng lớn thì độ phức tạp cũng sẽ tăng lên. Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.

C. Decision Tree

Decision tree là một mô hình học có giám sát, có thể được áp dụng vào cả hai bài toán phân lớp và hồi quy. Việc xây dựng một decision tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của decision tree là nó có thể làm việc với các đặc trưng (trong các tài liệu về decision tree, các đặc trưng thường được gọi là thuộc tính – attribute) dạng categorical, thường là rời rạc và không có thứ tự. Ví dụ, mưa, nắng hay xanh, đỏ, v.v. Decision tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng categorical và liên tục (numeric). Một điểm đáng lưu ý nữa là decision tree ít yêu cầu việc chuẩn hoá dữ liệu.

Trong cây Quyết định, có hai nút, đó là Nút Quyết định (Decision Node) và Nút Lá (Decision Node). Các nút quyết định được sử dụng để đưa ra bất kỳ quyết định nào và có nhiều nhánh, trong khi các nút Lá là đầu ra của các quyết định đó và không chứa bất kỳ nhánh nào nữa.

Nó là một biểu diễn đồ họa để có được tất cả các giải pháp khả thi cho một vấn đề/quyết định dựa trên các điều kiện nhất định.

Nó được gọi là cây quyết định bởi vì, tương tự như cây, nó bắt đầu với nút gốc, mở rộng trên các nhánh tiếp theo và xây dựng cấu trúc giống như cây.

Để xây dựng một cây, chúng em sử dụng thuật toán CART algorithm, viết tắt của Classification and Regression Tree algorithm.

Một cây quyết định chỉ cần đặt một câu hỏi và dựa trên câu trả lời (Có/Không), nó tiếp tục chia cây thành các cây con.

Vậy tại sao phải sử dụng Decision Tree?

Có nhiều thuật toán khác nhau trong Machine Learning, vì vậy, việc chọn thuật toán tốt nhất cho tập dữ liệu là một điểm mấu chốt cần nhớ khi tạo mô hình machine learning. Dưới đây là hai lý do chúng em sử dụng cây Quyết định:

- Cây quyết định thường bắt chước khả năng tư duy của con người khi đưa ra quyết định nên rất dễ hiểu.
- Có thể dễ dàng hiểu logic đằng sau cây quyết định vì nó cho thấy cấu trúc giống như cây.

Một thuật toán xây dựng decision tree ra đời rất sớm và phổ biến là Iterative Dichotomiser 3 (ID3). Decision tree cũng ứng

dụng một hàm rất phổ biến trong lý thuyết thông tin là hàm entropy.

1) Hàm entropy: Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau $x_1, x_2, ..., x_n$.

Giả sử rằng xác suất để x nhận các giá trị này là $p_i = p(x = x_i)$ với $0 \leqslant p_i \leqslant 1, \sum_{i=1}^{n} p_i = 1$. Ký hiệu phân phối này là $p = (p_1, p_2, \ldots, p_n)$. Entropy của phân phối này được định nghĩa là

$$H(p) = -\sum_{i=1}^{n} p_i \log(p_i) \tag{3}$$

trong đó, log là logarit cơ số 2. Với những tính chất của hàm entropy cũng được sử dụng trong ID3, do đó ID3 còn có tên gọi khác là entropy-based decision tree.

2) ID3: Trong ID3, tổng có trọng số của entropy tại các leaf-node sau khi xây dựng decision tree được coi là hàm loss của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Như đã đề cập, việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một non-leaf node. Chúng ta sẽ xây dựng phương pháp tính toán dựa trên mỗi node này.

Xét một bài toán với C class khác nhau. Giả sử ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập S với số phần tử là $|S| = N$. Giả sử thêm rằng trong số N điểm dữ liệu này, $N_c, c = 1, 2, \ldots, C$ điểm thuộc vào class c. Xác suất để mỗi điểm dữ liệu rơi vào một class c được xấp xỉ bằng $\frac{N_c}{N}$ (maximum likelihood estimation). Như vậy, entropy tại node này được tính bởi:

$$H(S) = -\sum_{c=1}^{C} \frac{N_c}{N} \log(\frac{N_c}{N}) \tag{4}$$

Tiếp theo, giả sử thuộc tính được chọn là x. Dựa trên x, các điểm dữ liệu trong S được phân ra thành K child node $S_1, S_2, \ldots, S_K$ với số điểm trong mỗi child node lần lượt là $m_1, m_2, \ldots, m_K$. Ta định nghĩa:

$$H(x, S) = -\sum_{k=1}^{K} \frac{m_k}{N} H(S_k) \tag{5}$$

là tổng có trọng số entroy của mỗi child node–được tính tương tự như (4). Việc lấy trọng số này là quan trọng vì các node thường có số lượng điểm khác nhau.

Tiếp theo, ta định nghĩa information gain dựa trên thuộc tính x:

$$G(x, S) = H(S) - H(x, S) \tag{6}$$

Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên thuộc tính khiến cho information gain đạt giá trị lớn nhất là:

$$x^* = \underset{x}{argmax} G(x, S) = \underset{x}{argmin} H(x, S) \tag{7}$$

3) Ưu điểm của Decision Tree:

- Đơn giản để hiểu, giải thích và hình dung.
- Cây quyết định ngầm làm việc với các đặc trưng.
- Có thể xử lý cả dữ liệu số và dữ liệu phân loại. Cũng có thể xử lý các vấn đề về multi-output.
- Cây quyết định đòi hỏi người dùng tương đối ít sức lực để chuẩn bị dữ liệu.
- Mối quan hệ phi tuyến tính giữa các tham số không ảnh hưởng đến hiệu suất của cây.

4) Nhược điểm của Decision Tree:

- Người học cây quyết định có thể tạo các cây quá phức tạp không tổng quát hóa tốt dữ liệu, tức là chúng có thể dễ dàng dẫn đến việc khớp dữ liệu quá mức (overfitting).
- Cây quyết định có thể không ổn định vì các biến thể nhỏ trong dữ liệu có thể dẫn đến việc tạo ra một cây hoàn toàn khác. Điều này được gọi là phương sai, cần được giảm xuống bằng các phương pháp như đóng gói (bagging) và tăng tốc (boosting).
- Các thuật toán tham lam không thể đảm bảo trả về cây quyết định tối ưu toàn cục. Điều này có thể được giảm thiểu bằng cách đào tạo (training) nhiều cây, trong đó các feature và sample được lấy ngẫu nhiên cùng với sự thay thế.
- Người học cây quyết định tạo cây thiên vị (bias) nếu một số lớp chiếm ưu thế. Do đó, nên cân bằng tập dữ liệu trước khi khớp (fit) với cây quyết định.

D. SVM - Support Vector Machine

SVM - Support Vector Machine là một trong những thuật toán phổ biến và được sử dụng nhiều nhất trong học máy trước khi mạng neural nhân tạo (artificial neural network) trở nên bùng nổ với các mô hình học sâu. Mục tiêu chính của thuật toán này là tìm ra một siêu phẳng trong không gian đa chiều (với không gian N chiều sẽ ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Có rất nhiều siêu phẳng có thể phân chia được hai lớp dữ liệu, ta cần tìm ra siêu phẳng có lề rộng nhất (khoảng cách tới các điểm của hai lớp là lớn nhất).

Khoảng cách từ một điểm (vector) có tọa độ x_0 tới siêu phẳng có phương trình $w^T x_0 + b = 0$ được xác định bởi:

$$\frac{|w^T x_0 + b|}{||w||_2} \tag{8}$$

Với $||w||_2 = \sqrt{\Sigma_{i=1}^{d} w_i^2}$ với d là số chiều của không gian.

Một điểm trong không gian véc tơ có thể được coi là một véc tơ từ gốc tọa độ tới điểm đó. Các điểm dữ liệu nằm trên hoặc gần nhất với siêu phẳng được gọi là véc tơ hỗ trợ, chúng ảnh hưởng đến vị trí và hướng của siêu phẳng. Các véc tơ này được sử dụng để tối ưu hóa lề và nếu xóa các điểm này, vị trí của siêu phẳng sẽ thay đổi. Một điểm lưu ý nữa đó là các véc tơ hỗ trợ phải cách đều siêu phẳng. Một kernel là một hàm ánh xạ dữ liệu từ không gian ít nhiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu. Một cách trực quan, kỹ thuật này giống như việc bạn gập tờ giấy lại để có thể dùng kéo cắt một lỗ tròn trên nó.

Các loại kernel:

- Linear.
- Polynomial.
- RBF.
- Sigmoid.

III. Experiment

A. Experiment Settings

Dataset: Nhóm em tiến hành đánh giá hiệu suất của 4 phương pháp này trên bộ dữ liệu Diabetes[1] được cung cấp trên Kaggle. Bộ dữ liệu gồm có 768 điểm dữ liệu. Đối với phương pháp đánh giá Train/Test Split thì dữ liệu được chia với tỉ lệ 3/1 tương ứng với tỉ lệ tập Training/Testing. Đối với đánh giá trên KFold, nhóm em chọn k = 4.

Configuration: Nhóm em thực nghiệm trên colab sử dụng thư viện phân tích dữ liệu mã nguồn mở scikit-learn.

B. Evaluation Metrics

1) Accuracy Score: Khi xây dựng mô hình phân loại chúng ta sẽ muốn biết một cách khái quát tỷ lệ các trường hợp được dự báo đúng trên tổng số các trường hợp là bao nhiêu. Tỷ lệ đó được gọi là độ chính xác. Độ chính xác giúp ta đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình của chúng ta càng chuẩn xác. Khi một ai đó nói mô hình của họ dự báo chính xác 90.5% thì chúng ta hiểu rằng họ đang đề cập tới độ chính xác được tính theo công thức :

$$Accuracy = \frac{TP + TN}{totalsample} \tag{9}$$

2) F1 Score: F1 Score là trung bình điều hòa giữa precision và recall. Do đó nó đại diện hơn trong việc đánh gía độ chính xác trên đồng thời precision và recall.

$$F1 = \frac{2}{precision^{-1} + recall^{-1}} \tag{10}$$

Trong trường hợp precision = 0 hoặc recall = 0 ta qui ước F1 = 0

Tại sao F1 Score không là trung bình cộng precision và recall? Lấy ví dụ trực quan trong trường hợp mô hình của bạn có precision quá thấp và recall quá cao, chẳng hạn precision=0.01 và recall=1.0.

Nhìn vào biểu đồ trade off giữa precision và recall thì đây có thể được xem như một mô hình thiết lập threshold thấp. Nó tương đương với việc dự đoán ngẫu nhiên toàn bộ là positive. Do đó không thể xem đó là một mô hình tốt. Nếu sử dụng công thức trung bình thì:

$$F_1 = \frac{precision + recall}{2} = 0.5005 \tag{11}$$

giá trị này cho thấy đây là một mô hình ở mức trung bình. Trong khi sử dụng công thức trung bình điều hòa thì:

$$F_1 = \frac{2 * precision * recall}{precision + recall} \approx 0 \tag{12}$$

[1]https://www.kaggle.com/datasets/whenamancodes/predict-diabities

giá trị này giúp nhận diện được mô hình không tốt.

Tóm lại sử dụng trung bình điều hòa sẽ phạt nặng hơn những trường hợp mô hình có precision thấp, recall cao hoặc precision cao, recall thấp. Đây là những trường hợp tương đương với dựbáo thiên về một nhóm là positive hoặc negative nên không phải là mô hình tốt. Điểm số từ trung bình điều hòa sẽ giúp ta nhận biết được những trường hợp không tốt như vậy.

3) Log loss: Đây là hàm mất mát được dùng trong hồi quy logistic và các phiên bản mở rộng của nó như neural networks, được định nghĩa là negative log-likelihood của một mô hình logistic trả về các xác suất của các lớp từ bộ dữ liệu train. Log loss chỉ được định nghĩa cho hai hoặc nhiều nhãn. Với một mẫu dữ liệu với nhãn thật là $y \in \{0, 1\}$ và xác suất $p = Pr(y = 1)$, log loss được tính:

$$L_{log}(y, p) = -(ylog(p) + (1 - y)log(1 - p)) \tag{13}$$

C. KFold Cross-validation

Cross validation là một phương pháp thống kê được sử dụng để ước lượng hiệu quả của các mô hình học máy. Nó thường được sử dụng để so sánh và chọn ra mô hình tốt nhất cho một bài toán. Kỹ thuật này dễ hiểu, dễ thực hiện và cho ra các ước lượng tin cậy hơn so với các phương pháp khác.

Cross validation là một kỹ thuật lấy mẫu để đánh giá mô hình học máy trong trường hợp dữ liệu không được dồi dào cho lắm.

Tham số quan trọng trong kỹ thuật này là k, đại diện cho số nhóm mà dữ liệu sẽ được chia ra. Vì lý do đó, nó được mang tên k-fold cross-validation. Khi giá trị của k được lựa chọn, người ta sử dụng trực tiếp giá trị đó trong tên của phương pháp đánh giá. Ví dụ với k=10, phương pháp sẽ mang tên 10-fold cross-validation.

Kỹ thuật này thường bao gồm các bước như sau:

1. Xáo trộn dataset một cách ngẫu nhiên
2. Chia dataset thành k nhóm
3. Với mỗi nhóm:
    1. Sử dụng nhóm hiện tại để đánh giá hiệu quả mô hình
    2. Các nhóm còn lại được sử dụng để huấn luyện mô hình
    3. Huấn luyện mô hình
    4. Đánh giá và sau đó hủy mô hình
4. Tổng hợp hiệu quả của mô hình dựa từ các số liệu đánh giá

Kết quả của quá trình thực nghiệm bằng KFold Cross-validation được trực quan trên TABLE I. Trong TABLE I, Logistic Regression có kết quả tốt nhất với phần còn lại với mean F1 score và mean Accuracy score đạt lần lượt là 73.5% và 77.3%. Hơn nữa, mean Log loss cũng đạt giá trị thấp nhất với 0.481. Ngược lại, SVM tuy là một mô hình rất chất lượng với các thuật toán phức tạp, được đánh giá cao hơn so với phần còn lại nhưng nó lại góp một phần vào những kết quả tệ nhất với mean F1 score chỉ đạt 63.1% với trường hợp không ứng dụng Standard Scaler, tuy nhiên dù có Standard Scaler thì cũng không cải thiện được quá nhiều. Bên cạnh đó, không quá ngạc nhiên khi phần còn lại của kết quả tệ nhất thuộc về kNN.

TABLE I: Kết quả đánh giá bằng phương pháp KFold với k = 4 trên bộ dữ liệu Diabetes (Kết quả tốt nhất được in đậm màu đỏ, ngược lại là in đậm màu đen)

Methods		mean F1 score	mean Accuracy score	mean Log loss
Logistic Regression	w/o Standard Scaler	0.73	0.768	0.487
	with Standard Scaler	**0.735**	**0.773**	**0.481**
kNN (n = 3)	w/o Standard Scaler	0.655	**0.688**	**3.94**
	with Standard Scaler	0.7	0.736	3.547
Decision Tree (max_depth = 5)	w/o Standard Scaler	0.69	0.734	1.609
	with Standard Scaler	0.698	0.738	1.559
SVM (kernel = linear)	w/o Standard Scaler	**0.631**	0.768	0.482
	with Standard Scaler	0.633	0.771	0.482

kNN vốn dĩ là một thuật toán học đơn giản, phương pháp cũng rất đơn giản, ý tưởng cũng không mang tính chuyên sâu. Điều đó làm kNN trở nên lép vế hơn so với mặt bằng chung của các phương pháp máy học mà nhóm em khảo sát.

Tuy nhiên, từ bảng kết quả ta có thể nhìn nhận theo một góc độ khác để thấy được tầm quan trọng của việc tiền xử lý dữ liệu. Chỉ với áp dụng Standard Scaler, kết quả trên từng phương pháp được cải thiện rất đáng kể. Điều này giúp ta nhận ra được mấu chốt để có một kết quả tốt không chỉ nằm trong việc chọn lọc hay thiết kế mô hình như thế nào mà còn là việc dữ liệu cảu chúng ta có sạch hay đã được chuẩn hóa hay chưa. Để có thể chắn chắn hơn về luận điểm này thì nhóm em còn cung cấp thêm một bảng kết quả nữa khi sử dụng Train/Test Split để chia dữ liệu. Một ví dụ mà ta có thể thấy về tính hiệu quả của Standard Scaler là phương pháp kNN, nhờ có Standard Scaler, kết quả được cải thiện từ 65.5% lên đến 70.0%, tức là khoảng 4.5%. Đây là một sự chênh lệch không nhỏ để chứng minh được tầm ảnh hưởng của các phương pháp tiền xử lý dữ liệu.

Nhìn chung, Logistic Regression là phương pháp tốt nhất với việc ứng dụng KFold để đánh giá, sau đó là SVM, Decision Tree và kNN. Dù mean F1 score của SVM khá thấp nhưng hai độ đo còn lại rất tích cực với 77.1% ở độ đo mean Accuracy Score và 0.482 ở Log loss.

D. Train/Test Split

Kết quả của quá trình thực nghiệm bằng Train/Test Split được trực quan trên TABLE II. Trong TABLE II, kết quả có chút khác biệt, như nhóm em dự đoán thì SVM đạt kết quả tốt nhất về mọi mặt với phương pháp huấn luyện mô hình bằng Train/Test Split. SVM đạt 67.3% và 80.7% với lần lượt là F1 score và Accuracy Score. Từ đây ta có thể nhìn nhận được là SVM hoạt động tốt trên dữ liệu dồi dào và không bị chia nhỏ, nói cách khác là dữ liệu được chia theo truyền thống. Kết quả của SVM vượt trội hơn so với phần còn lại. Trong khi đó, kNN tiếp tục là phương pháp có kết quả thấp nhất khi chỉ đạt 54% F1 score và 69.8% Accuracy score. Vấn đề của kNN thì vẫn như cũ, là quá đơn giản. Ngược lại, SVM được biết đến là một phương pháp cực kì chất lượng, có nhiều thuật toán phức tạp và nhiều siêu tham số để điều chỉnh. Điều đó giúp SVM chiếm vị trí đầu bảng.

Một lần nữa, tầm ảnh hưởng của Standard Scaler nói riêng mà phương pháp tiền xử lí dữ liệu nói chung được trực quan rõ ràng. Tất cả những phương pháp trong TABLE II sau khi áp dụng Standard Scaler đều có kết quả được cải thiện cục kì tốt, ví dụ như kNN cải thiện được tới 8.6% sau khi áp dụng Standard Scaler, và cũng dựa trên kết quả của TABLE I thì ta có thể rút ra được rằng Standard Scaler hoạt động rất tốt với phương pháp kNN khi sự chênh lệch của kết quả trước và sau khi áp dụng phương pháp trên là không hề nhỏ. Với việc chuẩn hóa dữ liệu, kNN có thể trực quan được không gian của dữ liệu một cách đơn giản hơn và tính toán khoảng cách không gian ít tốn chi phí hơn mà hiệu quả cũng được cải thiện không ít.

Xếp sau SVM là Logistic Regression và Decision Tree, hai phương pháp này có vẻ là ngang nhau, tuy nhiên còn phụ thuộc nhiều vào cách ta chọn siêu tham số hay xử lí dữ liệu đầu vào như thế nào. Như đã đề cập thì một mô hình tốt là chưa đủ để có một kết quả tốt mà còn phải xử lí dữ liệu như thế nào cho chuẩn và phù hợp nữa.

IV. Conclusion

Sau cùng thì, qua quá trình thực nghiệm, thì việc chọn lọc một phương pháp để tiền xử lí dữ liệu đóng một vai trò không nhỏ trong việc đạt kết quả tốt hoặc cải thiện hiệu suất của mô hình trong bài toán mà chúng ta quan tâm. Vẫn còn nhiều phương pháp tiền xử lí dữ liệu khác mà nhóm em chưa biết đến và chưa sử dụng qua bao giờ, nhóm em sẽ thử nghiệm và đánh giá khách quan hơn trong tương lai. Về kết quả đạt được thì nhóm em vẫn tin rằng SVM vẫn là một phương pháp hoạt động tốt với bài toán Phân lớp. Tuy nhiên, nếu ta cần một thuật toán đơn giản và cũng hiệu quả không kém cạnh thì Logistic Regression cũng tạo ra được những điểm mạnh nhất định trong bài toán Phân lớp, đặc biệt là Phân lớp nhị phân. Hơn nữa, tùy vào từng loại phương pháp tiền xử lí dữ liệu mà nó sẽ thích hợp với phương pháp khác nhau, ví dụ như kNN có vẻ hoạt động rất tốt với Standard Scaler, tốt hơn so với những phương pháp khác. Và chắc chắn rằng SVM hay Decision Tree hay cả Logistic Regression sẽ còn được cải thiện nhiều hơn nữa nếu chọn phương pháp tiền xử lí thích hợp.

Acknowledgment

Đồ án này được thực hiện với sự hướng dẫn của thầy Nguyễn Vinh Tiệp. Sau một học kì vô cùng thú vị và bổ ích, chúng em muốn dành một lời cảm ơn sâu sắc đến thầy, những chia sẻ và kinh nghiệm vô giá mà thầy đã mang đến trong học kì vừa qua

TABLE II: Kết quả đánh giá bằng phương pháp Train/Test Split trên bộ dữ liệu Diabetes (Kết quả tốt nhất được in đậm màu đỏ, ngược lại là in đậm màu đen)

Methods		F1 Score	Accuracy Score	Log loss
Logistic Regression	w/o Standard Scaler	0.649	0.792	7.196
	with Standard Scaler	0.649	0.797	7.016
kNN (n = 3)	w/o Standard Scaler	**0.54**	**0.698**	**10.434**
	with Standard Scaler	0.626	0.776	7.735
Decision Tree (max_depth = 5)	w/o Standard Scaler	0.625	0.781	7.56
	with Standard Scaler	0.638	0.781	7.56
SVM (kernel = linear)	w/o Standard Scaler	0.661	0.802	6.84
	with Standard Scaler	**0.673**	**0.807**	**6.656**

sẽ giúp chúng em chuẩn bị tốt hơn những hành trang để tiếp tục trên con đường học tập cũng như nghiên cứu. Em là Lê Trương Ngọc Hải, là trưởng nhóm, xin gửi lời cảm ơn đến các bạn trong nhóm đã hoàn thành tốt nhiệm vụ của mình cũng như sẵn lòng hỗ trợ các bạn còn lại, cảm ơn Lê Thị Phương Vy đã hoàn thành phương pháp Logistic Regression, cảm ơn Nguyễn Nhật Trường đã hoàn thành phương pháp Decision Tree, cảm ơn bạn Lại Chí Thiện đã hoàn thành phương pháp Support Vector Machine, và cả em đã hoàn thành phương pháp K Nearest Neighbors. Nhóm chúng em chúc thầy luôn có nhiều sức khỏe và niềm vui trong cuộc sống, đầy năng lượng để tiếp tục truyền những ngọn lửa đam mê đến những sinh viên của trường ta.