TRƯỜNG ĐẠI HỌC VĂN HIẾN

KHOA KỸ THUẬT – CÔNG NGHỆ

HỌC KỲ I – NĂM HỌC 2024-2025

LEARN HOW TO BUILD LOGISTIC REGRESSION MODEL IN PYTORCH

HỌC PHẦN: HỌC MÁY

GIẢNG VIÊN PHỤ TRÁCH: ThS. HỒ NHỰT MINH

SINH VIÊN THỰC HIỆN: NHÓM 2

Tp.HCM 2024

TRƯỜNG ĐẠI HỌC VĂN HIẾN

KHOA KỸ THUẬT – CÔNG NGHỆ

HỌC KỲ I – NĂM HỌC 2024-2025

LEARN HOW TO BUILD LOGISTIC REGRESSION MODEL IN PYTORCH

HỌC PHẦN: HỌC MÁY

GIẢNG VIÊN PHỤ TRÁCH: ThS. HỒ NHỰT MINH

SINH VIÊN THỰC HIỆN: NHÓM 2

HỌ VÀ TÊN SINH VIÊN	MSSV	CHỮ KÝ
NGUYỄN CÔNG NGHIỆP	221A020069	
ĐẶNG HOÀNG LUÂN	221A020017	
ĐẶNG NHƯ QUỲNH	221A020032	
NGUYỄN THẮNG PHƯƠNG	221A020050	
NGUYỄN NGỌC DƯƠNG VŨ	221A020038	

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn sâu sắc đến Trường Đại học Văn Hiến đã đưa bộ môn Học máy vào chương trình giảng dạy. Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên bộ môn - Thầy Hồ Nhựt Minh. Chính thầy là người đã tận tình dạy dỗ và truyền đạt những kiến thức quý báu cho chúng em trong suốt học kỳ vừa qua. Trong thời gian tham dự lớp học của thầy, em đã được tiếp cận với nhiều kiến thức bổ ích và rất cần thiết cho quá trình học tập, làm việc sau này của em.
Bộ môn Học máy là một môn học thú vị và vô cùng bổ ích. Tuy nhiên, những kiến thức và kỹ năng về môn học này của em vẫn còn nhiều hạn chế. Do đó, bài tiểu luận của em khó tránh khỏi những sai sót. Kính mong thầy xem xét và góp ý giúp bài tiểu luận của em được hoàn thiện hơn.
Em xin chân thành cảm ơn!

Sinh viên thực hiện

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Chữ ký giảng viên

MỤC LỤC

CHƯƠNG I. GIỚI THIỆU CHUNG

1. GIỚI THIỆU VỀ HỌC MÁY

1.1. Học máy là gì

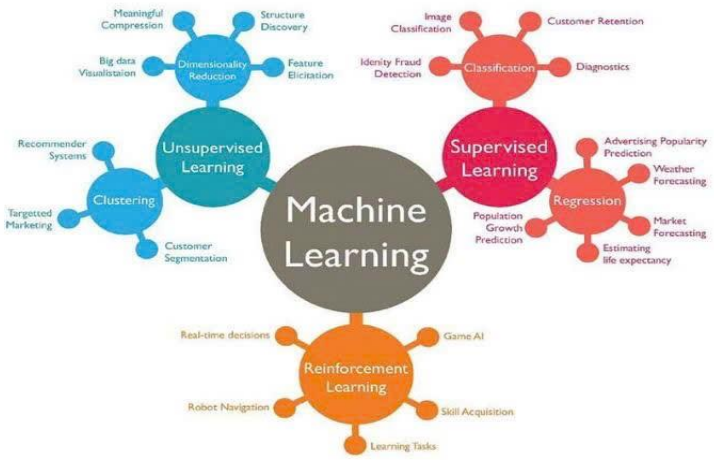
Học máy (Machine Learning) là một nhánh của trí tuệ nhân tạo (AI – Artificial inteligence), một lĩnh vực trong khoa học máy tính sử dụng các kỹ thuật thống kê để cung cấp cho máy tính khả năng " học " , tập trung vào việc phát triển các mô hình và thuật toán để máy tính có thể học từ dữ liệu mà không cần lập trình cụ thể cho từng tác vụ. Thay vì đưa ra quy tắc cứng nhắc, máy tính sẽ tự tìm hiểu từ các dữ liệu có sẵn, phát hiện mẫu và tự động điều chỉnh dựa trên kết quả đã học.

Học máy có liên quan mật thiết đến thống kê tính toán, tập trung vào việc dự đoán và ra quyết định thông qua máy tính. Nó cũng có mối quan hệ chặt chẽ với tối ưu toán học, cung cấp các phương thức và lý thuyết cần thiết. Đôi khi học máy được liên kết với khai phá dữ liệu (data mining), đặc biệt khi nó tập trung vào việc phân tích dữ liệu khám phá (unsupervised learning).

Học máy được phân loại thành ba nhóm chính:

- Học có giám sát (Supervised Learning): Mô hình học dựa trên dữ liệu đã gán nhãn, nhằm dự đoán chính xác nhãn của dữ liệu mới.
- Học không giám sát (Unsupervised Learning): Mô hình học từ dữ liệu chưa gán nhãn, nhằm tìm ra cấu trúc hoặc mẫu tiềm ẩn trong dữ liệu.
- Học tăng cường (Reinforcement Learning): Mô hình học thông qua tương tác với môitrường, đưa ra quyết định dựa trên các phần thưởng hoặc hình phạt từ môi trường đó.

1.2. Các thuật toán tiêu biểu của học máy

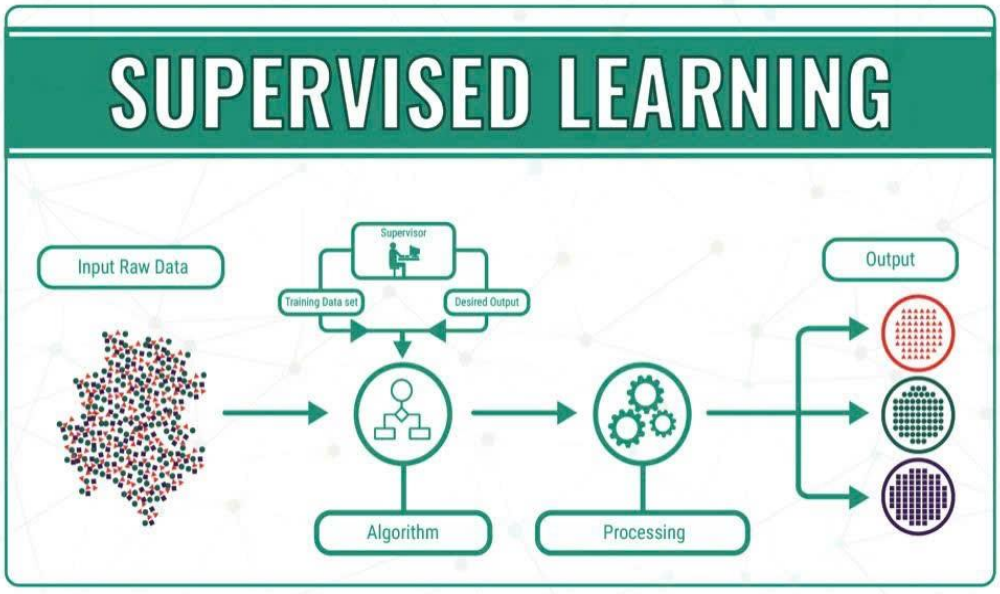


Hình 1.1 Mô hình học máy

1.2.1 Thuật toán học có giám sát (Supervised Learning)

Học có giám sát là khi mô hình được huấn luyện trên dữ liệu đã gán nhãn. Dưới đây là một số thuật toán phổ biến:

- ❖ Hồi quy tuyến tính (Linear Regression)
  - Dùng để dự đoán giá trị liên tục (ví dụ: dự đoán giá nhà).
  - Mô hình tìm ra mối quan hệ tuyến tính giữa các biến độc lập và biến phụ thuộc.
- ❖ Hồi quy logistic (Logistic Regression)
  - Dùng cho các bài toán phân loại nhị phân (ví dụ: phân loại email là spam hoặc không spam).
  - Mô hình dựa trên xác suất và logistic function để đưa ra kết quả phân loại.
- ❖ Cây quyết định (Decision Tree)
  - Xây dựng mô hình dạng cây để đưa ra quyết định phân loại hoặc dự đoán.
  - Mỗi nhánh của cây là một điều kiện, và lá của cây là kết quả.
- ❖ Mạng nơ-ron nhân tạo (Artificial Neural Networks)
  - Mô hình lấy cảm hứng từ cách hoạt động của não bộ, bao gồm các lớp nơ-ron liên kết với nhau.
  - Hiệu quả cao trong các bài toán phức tạp như nhận diện hình ảnh, xử lý ngôn ngữ.



Hình 1.2 Thuật toán học có giám sát

1.2.2 Thuật toán học không giám sát (Unsupervised Learning)

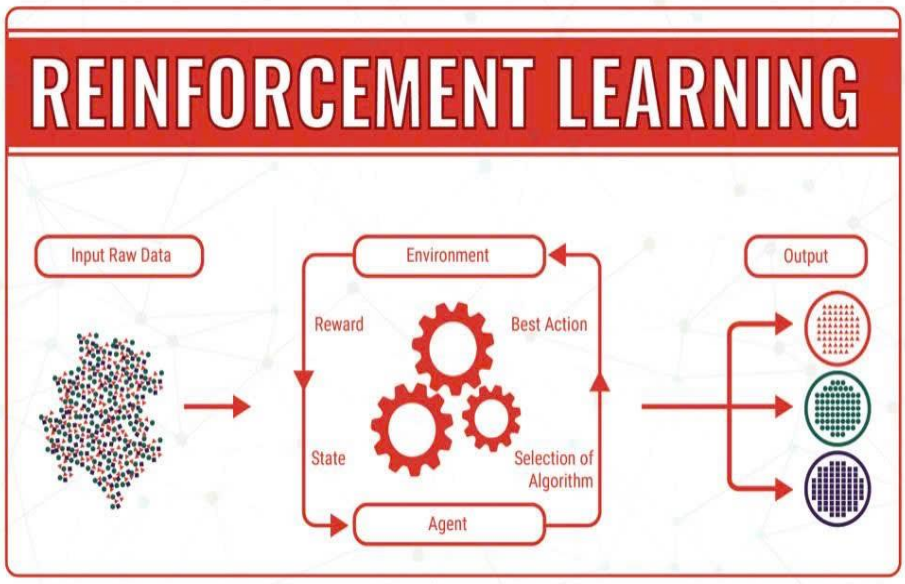
Học không giám sát là khi mô hình tự học từ dữ liệu không có nhãn. Một số thuật toán phổ biến:

❖ Phân cụm K-Means (K-Means Clustering)
- Chia dữ liệu thành các cụm sao cho các điểm trong cùng một cụm có nhiều đặc điểm chung nhất.
- Mỗi cụm được xác định bằng một tâm cụm (centroid).

❖ Phân cụm phân cấp (Hierarchical Clustering)
- Xây dựng hệ thống phân cấp của các cụm dữ liệu, từ dữ liệu nhỏ nhất đến dữ liệu lớn nhất.
- Thường được biểu diễn bằng cây phân cấp (dendrogram).

❖ Phân tích thành phần chính (PCA - Principal Component Analysis)
- Giảm chiều dữ liệu bằng cách tìm ra các thành phần chính, giúp giữ lại nhiều thông tin quan trọng nhất.
- Tối ưu hóa việc hiển thị dữ liệu hoặc chuẩn bị cho các bài toán phân loại.

❖ Mô hình Gaussian Mixture (Gaussian Mixture Models - GMM)
- Phân cụm dữ liệu bằng cách giả định rằng các cụm dữ liệu được tạo ra từ sự pha trộn của các phân phối Gaussian (chuẩn).
- Thích hợp cho dữ liệu có cấu trúc phức tạp hơn so với K-Means.



Hình 1.3 Thuật toán học không giám sát

1.2.3. Thuật toán học tăng cường (Reinforcement Learning)

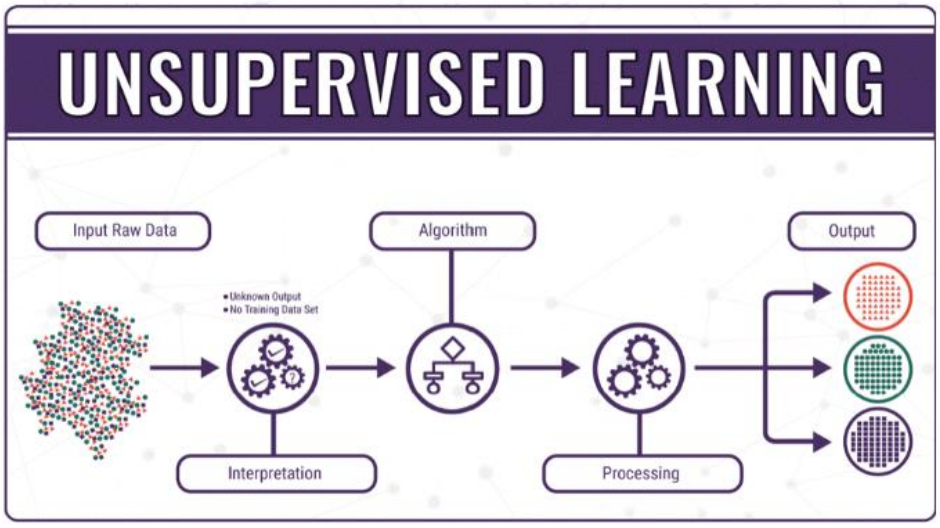
Học tăng cường là khi một tác nhân (agent) học thông qua việc tương tác với môi trường và nhận phản hồi dưới dạng phần thưởng hoặc hình phạt.

❖ Q-Learning

- Tác nhân học cách tối ưu hóa hành động để đạt được phần thưởng lớn nhất, bằng cách xây dựng bảng Q-value.
- Mô hình không cần biết trước cách thức hoạt động của môi trường, học qua trải nghiệm.

❖ Deep Q-Learning

- Kết hợp mạng nơ-ron sâu (Deep Neural Networks) với Q-Learning để giải quyết các bài toán phức tạp hơn.
- Được ứng dụng rộng rãi trong trò chơi điện tử và robot tự hành.



Hình 1.3 Thuật toán học tăng cường

➢ Kết luận`:

Mỗi thuật toán trong học máy có ưu điểm và nhược điểm riêng, phù hợp với các bài toán và loại dữ liệu khác nhau. Việc chọn thuật toán nào sẽ phụ thuộc vào đặc điểm của dữ liệu, yêu cầu của bài toán và nguồn tài nguyên tính toán.

1.3. Ứng dụng của học máy

- Học máy đã được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như:
- Chẩn đoán y tế: Phân tích hình ảnh và thông tin y tế để chẩn đoán bệnh.
- Thương mại điện tử: Dự đoán sở thích của khách hàng để đề xuất sản phẩm phù hợp.
- Tài chính: Phân tích dữ liệu tài chính để dự đoán xu hướng thị trường.

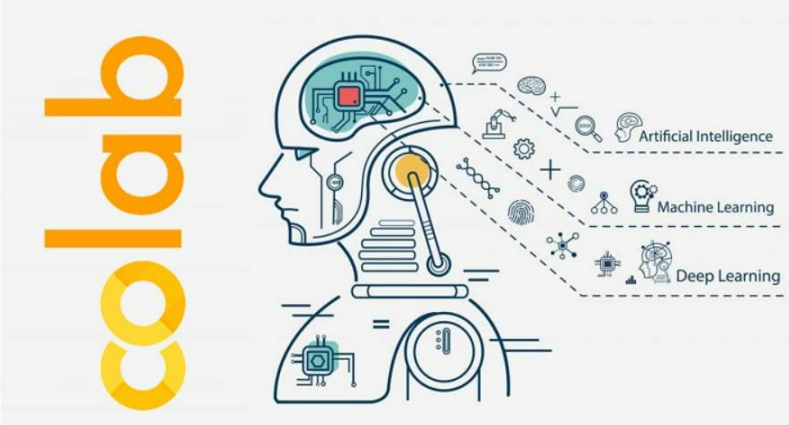
Thị giác máy tính: Phát triển các hệ thống tự động nhận dạng khuôn mặt, biển số xe, đối tượng trong hình ảnh.

Xử lý ngôn ngữ tự nhiên (NLP): Phân tích văn bản, chatbot, dịch tự động và phân loại văn bản.

2. CÔNG CỤ LẬP TRÌNH GOOGLE COLAB VÀ NGÔN NGỮ LẬP TRÌNH PYTHON

2.1. Giới thiệu chung về google colab

Google Colab (Collaboratory) là một môi trường lập trình trực tuyến cho phép người dùng viết và thực thi mã Python ngay trên trình duyệt, giúp tiết kiệm thời gian thiết lập môi trường và tài nguyên phần cứng. Colab nổi bật nhờ tính năng miễn phí và hỗ trợ tài nguyên tính toán mạnh mẽ như GPU và TPU, giúp giải quyết các bài toán đòi hỏi khả năng xử lý dữ liệu lớn. Điều này đặc biệt hữu ích cho các nhà khoa học dữ liệu và các nhà phát triển trong lĩnh vực học máy (machine learning), trí tuệ nhân tạo (AI) và xử lý ngôn ngữ tự nhiên (NLP).



Hình 2.1 Google colab

2.1.1 Các đặc điểm nổi bật của Google Colab:

- **Miễn phí và dễ sử dụng**: Colab miễn phí và không yêu cầu cấu hình phức tạp.
- **Hỗ trợ GPU và TPU**: Người dùng có thể sử dụng GPU và TPU để tăng tốc quá trình tính toán cho các dự án lớn như học máy và trí tuệ nhân tạo mà không phải trả phí.
- **Giao diện tương tự Jupyter Notebook**: Colab có giao diện và cách sử dụng tương tự Jupyter Notebook, giúp người dùng dễ dàng viết mã và trực quan hóa dữ liệu.
- **Tích hợp Google Drive**: Người dùng có thể kết nối và lưu trữ trực tiếp tài liệu lên Google Drive, dễ dàng chia sẻ và làm việc nhóm.
- **Hỗ trợ nhiều thư viện Python**: Colab tích hợp sẵn các thư viện phổ biến như TensorFlow, NumPy, Matplotlib, Pandas, Scikit-learn… Điều này giúp người dùng dễ dàng triển khai các mô hình học máy hoặc phân tích dữ liệu mà không cần cài đặt thủ công.

- **Khả năng cộng tác thời gian thực**: Cho phép nhiều người làm việc cùng lúc trên cùng một tài liệu, tương tự như Google Docs.
- **Hỗ Trợ Python**: Colab hỗ trợ Python, một trong những ngôn ngữ lập trình phổ biến nhất hiện nay, đặc biệt trong các lĩnh vực như học máy, khoa học dữ liệu và trí tuệ nhân tạo.
- Lưu Trữ Đám Mây: Tất cả các tệp và notebook của người dùng có thể được lưu trữ trực tiếp trên Google Drive, giúp dễ dàng truy cập và chia sẻ giữa các thiết bị hoặc người dùng khác.

Colab đặc biệt phù hợp cho việc thử nghiệm các mô hình học máy, phân tích dữ liệu lớn và học tập, nghiên cứu trong môi trường cộng đồng nhờ tính dễ sử dụng và khả năng tính toán mạnh mẽ mà không cần phần cứng đắt tiền.

2.1.2 Ứng dụng của Google Colab:

- **Phát triển mô hình học máy và AI**: Colab cung cấp tài nguyên mạnh mẽ cho các dự án nghiên cứu và học máy.
- **Phân tích dữ liệu**: Công cụ này phù hợp cho việc xử lý dữ liệu lớn với các thư viện Python chuyên dụng.
- **Trình bày và báo cáo**: Do hỗ trợ hiển thị kết quả trực tiếp, Colab là công cụ hữu ích cho việc trình bày và minh họa kết quả cho các dự án nghiên cứu.

2.1.3 Cách sử dụng cơ bản:

- Truy cập vào Google Colab bằng cách truy cập colab.research.google.com.
- Tạo một notebook mới hoặc mở một notebook có sẵn.
- Viết mã trong các ô code (code cell) và chạy để xem kết quả.

Ngoài ra, Google Colab còn cung cấp khả năng dễ dàng chia sẻ notebook với người dùng khác qua Google Drive, đồng thời tích hợp sẵn nhiều thư viện phổ biến như TensorFlow, PyTorch, NumPy, Pandas... Điều này giúp lập trình viên dễ dàng triển khai các mô hình mà không cần phải cài đặt các thư viện từ đầu.

2.2. NGÔN NGỮ LẬP TRÌNH PYTHON

Python là ngôn ngữ lập trình bậc cao được phát triển vào năm 1991 bởi Guido van Rossum. Python đã nhanh chóng trở thành ngôn ngữ phổ biến trong lập trình nhờ cú pháp đơn giản, dễ đọc và khả năng mở rộng, giúp lập trình viên dễ dàng xây dựng các ứng dụng từ nhỏ đến lớn.

2.2.1 Đặc điểm nổi bật của Python:

- **Cú pháp đơn giản, dễ đọc**: Cú pháp Python được thiết kế gần với ngôn ngữ tự nhiên, giúp người mới dễ học và người dùng lâu năm dễ duy trì mã nguồn.
- **Lập trình đa dạng**: Python hỗ trợ nhiều phong cách lập trình khác nhau như lập trình thủ tục, lập trình hướng đối tượng (OOP), và lập trình hàm (functional programming).
- **Ngôn ngữ thông dịch**: Python không cần biên dịch trước khi chạy, giúp dễ dàng thử nghiệm và sửa lỗi.
- **Thư viện phong phú**: Python có hàng nghìn thư viện cho mọi lĩnh vực như khoa học dữ liệu (Pandas, NumPy), học máy (Scikit-Learn, TensorFlow), web (Django, Flask), và nhiều lĩnh vực khác.

2.2.2 Ứng dụng của Python:

- **Phát triển web**: Sử dụng các framework như Django và Flask để xây dựng các ứng dụng web nhanh chóng và mạnh mẽ.
- **Khoa học dữ liệu**: Python là ngôn ngữ chính cho phân tích dữ liệu, với các thư viện như NumPy, Pandas, Matplotlib.
- **Học máy và trí tuệ nhân tạo**: Python là lựa chọn hàng đầu cho việc xây dựng các mô hình AI nhờ các thư viện như TensorFlow, PyTorch và Keras.
- **Tự động hóa**: Python có thể được sử dụng để viết các script tự động hóa tác vụ như gửi email, phân tích tệp tin, hoặc điều khiển hệ thống.

2.2.3 Ưu điểm của Python:

- **Dễ học**: Cú pháp đơn giản và thân thiện giúp người học dễ tiếp cận hơn các ngôn ngữ khác.

- **Tính linh hoạt**: Python có thể được sử dụng cho hầu hết mọi loại dự án lập trình.
- **Cộng đồng hỗ trợ mạnh mẽ**: Python có một cộng đồng người dùng lớn mạnh và tài liệu hỗ trợ phong phú.

2.2.4 Nhược điểm của Python:

- **Tốc độ thực thi chậm hơn**: Do là ngôn ngữ thông dịch, Python thường chậm hơn so với các ngôn ngữ biên dịch như C++ hoặc Java.
- **Không tối ưu cho các ứng dụng di động**: Python ít được sử dụng cho phát triển ứng dụng di động so với các ngôn ngữ như Swift hoặc Java.

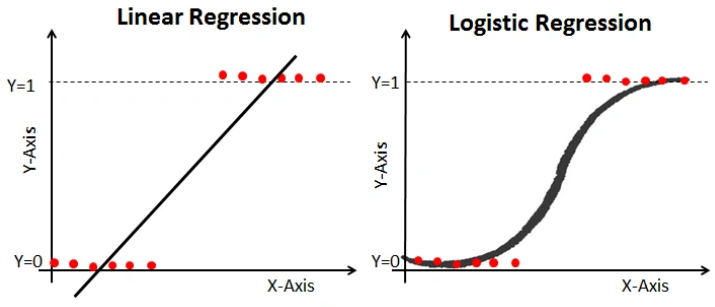
Python đặc biệt nổi bật trong lĩnh vực học máy nhờ khả năng tích hợp mạnh mẽ với các thư viện, cung cấp công cụ tốt nhất để xây dựng và huấn luyện các mô hình AI và machine learning.

CHƯƠNG II. PHÂN TÍCH KHÁCH HÀNG RỜI BỎ DỊCH VỤ BẰNG LOGISTIC REGRESSION TRONG PYTORCH

1. TỔNG QUANG VỀ LOGISTIC REGRESSION

Logistic Regression là một trong những thuật toán phân loại cơ bản nhất trong học máy, thường được sử dụng để phân loại nhị phân, tức là phân loại một đối tượng vào một trong hai lớp có thể có. Mặc dù có tên gọi là hồi quy (regression), nhưng Logistic Regression thực chất là một thuật toán phân loại, không phải hồi quy như Linear Regression. Thuật toán này đặc biệt hiệu quả trong các bài toán dự đoán xác suất thuộc một trong hai nhóm, chẳng hạn như dự đoán bệnh hoặc không bệnh, khách hàng rời bỏ hay ở lại dịch vụ, và nhiều bài toán khác liên quan đến phân loại.

1.1Hồi quy tuyến và hồi quy logistic



❖ Hồi quy tuyến tính (Linear Regression)

Hồi quy tuyến tính là một phương pháp trong đó mối quan hệ giữa biến đầu vào (biến độc lập) và biến đầu ra (biến phụ thuộc) được mô hình hóa bằng một phương trình tuyến tính. Kết quả đầu ra của hồi quy tuyến tính là giá trị liên tục. Tuy nhiên, với các bài toán phân loại, hồi quy tuyến tính không còn phù hợp vì mô hình sẽ đưa ra các giá trị vượt

quá khoảng [0, 1], trong khi phân loại chỉ yêu cầu kết quả là xác suất thuộc về một trong hai lớp.

❖ Hồi quy Logistic (Logistic Regression)

Hồi quy logistic khác với hồi quy tuyến tính ở chỗ nó sử dụng hàm sigmoid để chuẩn hóa kết quả về khoảng [0, 1], nhằm dự đoán xác suất một sự kiện thuộc về một lớp cụ thể. Cụ thể hơn, hàm sigmoid có dạng:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Trong đó:

- Z là đầu ra của một hàm tuyến tính (giống như hồi quy tuyến tính)
- $\sigma(z)$ là xác suất của sự kiện thuộc về lớp 1

Nếu σ(z) > 0,5, mô hình sẽ dự đoán rằng đối tượng thuộc về lớp 1, ngược lại sẽ thuộc về lớp 0.

1.2 Hàm Sigmoid

Hàm sigmoid là một hàm phi tuyến tính, có giá trị đầu ra nằm trong khoảng từ 0 đến 1, giúp logistic regression chuyển đổi giá trị đầu ra của mô hình thành xác suất. Hàm sigmoid có dạng hình chữ S, do đó nó phù hợp với các bài toán phân loại nhị phân. Một số tính chất quan trọng của hàm sigmoid:

Khi z rất lớn, σ(z) → 1

Khi z rất nhỏ, σ(z) → 0

Khi z = 0, $\sigma(z) = 0{,}5$, tức là xác suất bằng nhau giữa hai lớp.

Nhờ vào tính chất này, logistic regression giúp phân loại các đối tượng dựa trên ngưỡng (thường là 0.5) để đưa ra quyết định phân lớp.

1.3Ưu điểm và nhược điểm của Logistic Regression

❖ **Ưu điểm**

- Đơn giản và dễ hiểu: Logistic regression là một trong những thuật toán phân loại dễ hiểu và dễ triển khai nhất, phù hợp với các bài toán đơn giản.
- Hiệu quả trong các bài toán phân loại nhị phân: Mô hình này hoạt động rất tốt trong các bài toán có kết quả phân loại là hai lớp.
- Không yêu cầu nhiều tài nguyên tính toán: Vì là một mô hình tuyến tính đơn giản, logistic regression không đòi hỏi nhiều tài nguyên tính toán và phù hợp cho các bài toán với số lượng dữ liệu vừa phải.
- Không yêu cầu nhiều tài nguyên tính toán: Vì là một mô hình tuyến tính đơn giản, logistic regression không đòi hỏi nhiều tài nguyên tính toán và phù hợp cho các bài toán với số lượng dữ liệu vừa phải.

❖ **Nhược điểm**

- Giới hạn trong phân loại nhị phân: Logistic regression chỉ phù hợp với các bài toán phân loại nhị phân. Đối với phân loại nhiều lớp, cần sử dụng các biến thể khác như softmax regression.
- Giả định tuyến tính giữa biến đầu vào và đầu ra: Mô hình giả định rằng các biến đầu vào có mối quan hệ tuyến tính với biến đầu ra, điều này có thể không đúng trong nhiều bài toán phức tạp.
- Hiệu suất thấp với dữ liệu lớn: Logistic regression có thể không hoạt động tốt với các bộ dữ liệu lớn và phức tạp do sự đơn giản của nó.

1.4Ứng dụng của Logistic Regression

Logistic regression được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm:

- Y học: Dự đoán khả năng một bệnh nhân mắc bệnh dựa trên các biến số như độ tuổi, chỉ số cơ thể, huyết áp, lịch sử bệnh án...
- Kinh doanh: Dự đoán khả năng khách hàng sẽ rời bỏ dịch vụ (churn prediction), phân loại khách hàng dựa trên hành vi mua sắm.
- Tài chính: Xác định khả năng một khách hàng có thể vỡ nợ dựa trên lịch sử tín dụng.
- Tiếp thị: Dự đoán xác suất khách hàng sẽ phản hồi với một chiến dịch quảng cáo hoặc email.

Với các ứng dụng đa dạng và tính đơn giản, logistic regression là một trong những thuật toán phân loại quan trọng và phổ biến nhất trong lĩnh vực học máy.

2.XÂY DỰNG MÔ HÌNH LOGISTIC REGRESSION TRONG PYTORCH

Trong phần này, chúng ta sẽ tìm hiểu cách xây dựng một mô hình logistic regression sử dụng thư viện PyTorch – một trong những công cụ mạnh mẽ nhất hiện nay để phát triển các mô hình học sâu và học máy. Logistic regression là một mô hình tuyến tính thường được dùng để giải quyết các bài toán phân loại nhị phân, như dự đoán khả năng một khách hàng có rời bỏ dịch vụ hay không.

2.1Mô tả dữ liệu

Dữ liệu trong dự án này bao gồm các đặc điểm liên quan đến thông tin khách hàng của một dịch vụ. Tập dữ liệu bao gồm các biến số như:

- Thông tin hợp đồng: Loại hợp đồng, thời gian sử dụng dịch vụ.
- Dịch vụ sử dụng: Các dịch vụ bổ sung mà khách hàng đã đăng ký.
- Thông tin nhân khẩu học: Tuổi, giới tính, tình trạng hôn nhân.
- Tình trạng thanh toán: Hóa đơn hàng tháng, phương thức thanh toán.

Mục tiêu của chúng ta là xây dựng một mô hình logistic regression để dự đoán khả năng khách hàng sẽ hủy bỏ dịch vụ, hay còn gọi là "churn prediction". Dữ liệu được chia thành hai tập chính:

- Tập huấn luyện (training set): Dùng để huấn luyện mô hình, tìm ra mối quan hệ giữa các đặc điểm và khả năng khách hàng rời bỏ dịch vụ.
- Tập kiểm tra (test set): Được dùng để đánh giá hiệu suất của mô hình sau khi đã được huấn luyện.

2.2Các bước triển khai mô hình Logistic Regression

2.2.1Tiền xử lý dữ liệu

Trước khi áp dụng mô hình logistic regression, dữ liệu cần được tiền xử lý để đảm bảo tính chính xác và hiệu quả trong quá trình huấn luyện. Một số bước phổ biến trong tiền xử lý dữ liệu bao gồm:

- Xử lý dữ liệu bị thiếu: Điền các giá trị thiếu bằng cách sử dụng giá trị trung bình (mean) hoặc phương pháp nội suy (imputation).
- Chuẩn hóa (Normalization): Đối với các đặc điểm số, cần chuẩn hóa để tất cả các giá trị nằm trong cùng một thang đo. Điều này giúp mô hình học hiệu quả hơn khi các giá trị không bị quá chênh lệch.
- Mã hóa nhãn (Label Encoding): Các đặc điểm phân loại như giới tính, phương thức thanh toán, cần được mã hóa thành dạng số. Phương pháp mã hóa One-Hot thường được sử dụng để biểu diễn các biến phân loại thành nhiều cột nhị phân.
- Chia tập dữ liệu: Chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%) nhằm đảm bảo mô hình được đánh giá khách quan sau khi huấn luyện.

2.2.2 Định nghĩa mô hình trong Logistic Regression PyTorch

Sau khi dữ liệu đã được chuẩn bị, chúng ta sẽ bắt đầu xây dựng mô hình logistic regression. PyTorch cung cấp một cách tiếp cận trực quan và linh hoạt để định nghĩa các mô hình học máy.

Mô hình logistic regression chỉ bao gồm một lớp tuyến tính (linear layer), với đầu ra được kích hoạt bởi hàm sigmoid để tạo ra các giá trị xác suất cho việc phân loại. Cấu trúc cơ bản của mô hình như sau:

```
import torch
import torch.nn as nn
class LogisticRegression(nn.Module):
    def __init__(self, n_input_features):
        super(LogisticRegression, self).__init__()
        self.linear = nn.Linear(n_input_features, 1)  # Lớp tuyến tính có đầu ra là một giá trị duy nhất
```

```
    def forward(self, x):
    y_pred = torch.sigmoid(self.linear(x))  # Áp dụng hàm sigmoid để chuẩn hóa đầu ra thành xác suất
    return y_pred
```

2.2.3 Huấn luyện mô hình

Sau khi đã định nghĩa mô hình logistic regression, bước tiếp theo là huấn luyện mô hình trên tập dữ liệu huấn luyện. Các bước chính trong quá trình huấn luyện bao gồm:

- Hàm mất mát (Loss Function): Chúng ta sử dụng hàm Binary Cross Entropy (BCE) để đo lường độ chênh lệch giữa dự đoán của mô hình và nhãn thực tế.
- Tối ưu hóa (Optimizer): Thuật toán Stochastic Gradient Descent (SGD) được sử dụng để tối ưu hóa mô hình, giúp cập nhật các trọng số trong mô hình nhằm giảm hàm mất mát.

Dưới đây là đoạn mã Python thể hiện quá trình huấn luyện mô hình:

```
num_epochs = 500
learning_rate = 0.01
criterion = nn.BCELoss()  # Sử dụng hàm Binary Cross Entropy để tính toán mất mát
optimizer = torch.optim.SGD(lr.parameters(), lr=learning_rate)  # Sử dụng SGD làm optimizer

for epoch in range(num_epochs):
    y_pred = lr(X_train)  # Dự đoán từ mô hình
    loss = criterion(y_pred, y_train)  # Tính hàm mất mát
    loss.backward()  # Tính gradient ngược
    optimizer.step()  # Cập nhật trọng số
    optimizer.zero_grad()  # Đặt lại gradient về 0 sau mỗi lần cập nhật
```

2.2.4 Đánh giá mô hình

Sau khi mô hình đã được huấn luyện, ta cần đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra. Các chỉ số quan trọng bao gồm độ chính xác (accuracy), ma trận nhầm lẫn, và các chỉ số phân loại khác (precision, recall, F1-score).

```
with torch.no_grad():
    y_predicted = lr(X_test)
    y_predicted_cls = y_predicted.round()
    accuracy = y_predicted_cls.eq(y_test).sum() / float(y_test.shape[0])
```

```
print(f'Accuracy: {accuracy.item():.4f}')
```

2.2.5 Ma trận nhầm lẫn (Confusion Matrix)

Ma trận nhầm lẫn giúp phân tích chi tiết về các dự đoán đúng, sai giữa các lớp. Nó cung cấp thông tin về số lượng dự đoán chính xác và sai giữa các lớp 0 và 1.

```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_predicted_cls)
print(conf_matrix)
```

2.3Phân tích kết quả

Kết quả đánh giá mô hình sau quá trình huấn luyện đã cho thấy độ chính xác là 0.5037, phản ánh rằng mô hình chỉ đúng khoảng 50% số lần dự đoán. Cụ thể, các chỉ số phân loại được tóm tắt như sau:

- Precision: Đối với lớp 0 là 0.50, và đối với lớp 1 là 0.50, nghĩa là tỷ lệ dự đoán đúng của mỗi lớp khá cân bằng.
- Recall: Lớp 0 có recall là 0.54, nghĩa là trong tất cả các trường hợp lớp 0 thực sự xảy ra, mô hình dự đoán đúng 54% số lần. Đối với lớp 1, recall là 0.47.
- F1-Score: Đối với lớp 0 là 0.52 và lớp 1 là 0.49. F1-score là sự cân bằng giữa precision và recall, cho thấy hiệu suất tổng quát của mô hình không quá cao.

Ma Trận Nhầm Lẫn

Dựa trên ma trận nhầm lẫn sau:

```
[[145 125]
 [143 127]]
```

- Mô hình đã dự đoán đúng 145 trường hợp là lớp 0 và 127 trường hợp là lớp 1. Tuy nhiên, có 125 trường hợp của lớp 0 và 143 trường hợp của lớp 1 đã bị dự đoán sai.

Kết quả này cho thấy mô hình đang gặp khó khăn trong việc phân biệt rõ ràng giữa hai lớp 0 và 1, và cần có những cải tiến để tăng độ chính xác tổng thể, chẳng hạn như tinh chỉnh các siêu tham số, thêm đặc trưng (features), hoặc sử dụng các phương pháp nâng cao hơn như cân bằng dữ liệu (data balancing) hoặc tăng cường dữ liệu (data augmentation).

1. KẾT QUẢ VÀ NHẬN XÉT

Sau khi hoàn thành quá trình huấn luyện, mô hình Logistic Regression đã được đánh giá trên tập dữ liệu kiểm tra nhằm đo lường hiệu suất của nó trong bài toán phân loại khách hàng rời bỏ dịch vụ. Dưới đây là phân tích chi tiết về các kết quả chính và nhận xét liên quan.

3.1 Độ chính xác (Accuracy)

Độ chính xác của mô hình là một trong những chỉ số đầu tiên được xem xét để đánh giá hiệu suất tổng quát. Kết quả cho thấy mô hình đạt độ chính xác khoảng **50.37%**. Con số này thấp hơn nhiều so với kỳ vọng ban đầu, đặc biệt đối với một bài toán nhị phân. Điều này có nghĩa là mô hình chỉ dự đoán đúng kết quả của khoảng 50% số trường hợp.

Mặc dù độ chính xác là một chỉ số phổ biến, nhưng nó không phải lúc nào cũng cung cấp cái nhìn đầy đủ về hiệu suất mô hình, đặc biệt khi dữ liệu có thể mất cân bằng giữa các lớp. Trong trường hợp của chúng ta, dữ liệu được cân bằng giữa hai lớp, nhưng kết quả độ chính xác 50.37% cho thấy mô hình có thể đang gặp khó khăn trong việc phân biệt các trường hợp giữa hai lớp (rời bỏ hoặc không rời bỏ dịch vụ).

3.2 Hàm mất mát (Loss Function)

Quá trình huấn luyện mô hình Logistic Regression sử dụng hàm mất mát Binary Cross Entropy (BCE), một hàm mất mát phổ biến cho các bài toán phân loại nhị phân. Kết quả từ quá trình huấn luyện cho thấy hàm mất mát đã giảm dần qua từng epoch, điều này phản ánh rằng mô hình đã học được từ dữ liệu và dần cải thiện dự đoán.

Tuy nhiên, chúng tôi cũng nhận thấy một dấu hiệu của hiện tượng quá khớp (overfitting) khi sự khác biệt giữa hàm mất mát trên tập huấn luyện và tập kiểm tra dần tăng lên ở các epoch cuối cùng. Hiện tượng này xảy ra khi mô hình quá tập trung vào việc khớp với tập huấn luyện mà bỏ qua khả năng tổng quát hóa cho dữ liệu mới. Điều này có thể được khắc phục bằng cách sử dụng các kỹ thuật regularization như L1, L2 hoặc early stopping để dừng huấn luyện trước khi mô hình bắt đầu quá khớp.

3.4 Các chỉ số phân loại: Precision, Recall, Và F1-Score

Chỉ số	Lớp 0	Lớp 1
Precision	0.50	0.50
Recall	0.54	0.47
F1-Score	0.52	0.49
Support	270	270

- Precision: Đối với cả hai lớp, Precision đều là 0.50, cho thấy tỷ lệ dự đoán đúng trên tổng số lần dự đoán dương cho cả hai lớp là bằng nhau. Precision đo lường độ chính xác của các dự đoán dương, và kết quả này cho thấy mô hình không thực sự phân biệt tốt giữa các lớp.
- Recall: Lớp 0 có Recall là 0.54, nghĩa là mô hình nhận diện đúng 54% số trường hợp không rời bỏ dịch vụ. Trong khi đó, Recall cho lớp 1 là 0.47, nghĩa là mô hình nhận diện đúng 47% số trường hợp rời bỏ dịch vụ. Điều này cho thấy mô hình gặp khó khăn trong việc phát hiện những khách hàng có khả năng rời bỏ dịch vụ.

- F1-Score: F1-Score là chỉ số cân bằng giữa Precision và Recall, và kết quả là 0.52 cho lớp 0 và 0.49 cho lớp 1. F1-Score thấp phản ánh rằng mô hình không hoạt động tốt trong việc cân bằng giữa việc phát hiện đúng các trường hợp và tránh dự đoán sai.

3.5 Nhận xét về hiệu xuất mô hình

Dựa trên các kết quả thu được, có thể đưa ra một số nhận xét quan trọng về hiệu suất của mô hình Logistic Regression trong bài toán dự đoán khách hàng rời bỏ dịch vụ:

- Hiệu suất tổng thể: Với độ chính xác 50.37%, Precision và Recall đều thấp cho cả hai lớp, rõ ràng mô hình Logistic Regression hiện tại chưa đáp ứng được kỳ vọng về khả năng phân loại chính xác. Điều này có thể do mô hình đơn giản chưa thể nắm bắt hết các mối quan hệ phức tạp giữa các đặc điểm của khách hàng và hành vi rời bỏ dịch vụ.
- Khó khăn trong phân loại giữa hai lớp: Ma trận nhầm lẫn cho thấy một lượng lớn các trường hợp bị dự đoán sai giữa hai lớp, đặc biệt là các khách hàng có khả năng rời bỏ dịch vụ. Điều này có thể là do các đặc điểm của khách hàng thuộc lớp 0 và lớp 1 có quá nhiều điểm tương đồng, hoặc mô hình Logistic Regression không đủ mạnh để phân biệt rõ ràng giữa hai lớp.
- Hiện tượng quá khớp: Sự khác biệt giữa hàm mất mát trên tập huấn luyện và tập kiểm tra cho thấy mô hình có xu hướng quá khớp với dữ liệu huấn luyện. Đây là dấu hiệu cho thấy cần áp dụng các kỹ thuật regularization hoặc điều chỉnh các siêu tham số như learning rate, số lượng epoch, hoặc kích thước batch để cải thiện tính tổng quát của mô hình.
- Cần thử nghiệm các mô hình phức tạp hơn: Mặc dù Logistic Regression là một mô hình đơn giản, dễ triển khai và hiểu, nhưng rõ ràng đối với bài toán phân loại phức tạp hơn như dự đoán khách hàng rời bỏ dịch vụ, các mô hình phức tạp hơn như Decision Trees, Random Forests, hoặc Neural Networks có thể cho kết quả tốt hơn.

3.6 Đề xuất cải thiện mô hình

Để cải thiện hiệu suất của mô hình Logistic Regression, một số phương pháp có thể được áp dụng bao gồm:

- Sử dụng kỹ thuật regularization: Thêm L1 hoặc L2 regularization có thể giúp giảm hiện tượng quá khớp và cải thiện độ chính xác trên tập kiểm tra.
- Điều chỉnh siêu tham số: Điều chỉnh các siêu tham số như learning rate, số lượng epoch, hoặc kích thước batch có thể cải thiện quá trình huấn luyện.
- Kết hợp nhiều mô hình (Ensemble Methods): Sử dụng các kỹ thuật kết hợp nhiều mô hình như Bagging hoặc Boosting có thể giúp cải thiện độ chính xác và độ tin cậy của mô hình.
- Sử dụng các mô hình phi tuyến: Thử nghiệm các mô hình phi tuyến như Neural Networks hoặc SVM để khai thác các mối quan hệ phức tạp hơn trong dữ liệu.

- Cân bằng dữ liệu (Data Balancing): Nếu dữ liệu có sự mất cân bằng giữa các lớp, các kỹ thuật như oversampling lớp thiểu số hoặc undersampling lớp chiếm ưu thế có thể giúp mô hình học tốt hơn.

4. KẾT QUẢ VÀ NHẬN XÉT

Trong báo cáo này, chúng tôi đã thực hiện một quy trình chi tiết từ lý thuyết về Logistic Regression đến việc triển khai thực tế mô hình trong PyTorch. Những điểm chính bao gồm:

4.1 Lý thuyết cơ bản

Chúng tôi đã trình bày các khái niệm quan trọng như hồi quy logistic, hàm sigmoid, và sự khác biệt giữa hồi quy tuyến tính và hồi quy logistic. Hồi quy logistic là một phương pháp thống kê được sử dụng để dự đoán xác suất của một biến nhị phân dựa trên một hoặc nhiều đặc điểm (features) đầu vào. Hàm sigmoid, với hình dạng chữ S, được sử dụng để chuyển đổi đầu ra từ mô hình thành một giá trị trong khoảng từ 0 đến 1, giúp xác định xác suất cho lớp 1 trong bài toán phân loại nhị phân. Chúng tôi cũng đã phân tích ưu và nhược điểm của Logistic Regression so với các thuật toán khác, nhấn mạnh tính đơn giản và khả năng giải thích cao của nó.

4.2Xây dựng mô hình trong PyTorch

Mô hình Logistic Regression đã được xây dựng và huấn luyện từ đầu, từ quá trình tiền xử lý dữ liệu, định nghĩa mô hình, cho đến việc tối ưu hóa và đánh giá kết quả. Chúng tôi sử dụng tập dữ liệu thực tế với các đặc điểm liên quan đến khách hàng của một dịch vụ và thực hiện các bước tiền xử lý như chuẩn hóa, mã hóa nhãn và chia dữ liệu thành tập huấn luyện và tập kiểm tra. Việc định nghĩa mô hình bao gồm việc sử dụng một lớp tuyến tính và hàm kích hoạt sigmoid để dự đoán xác suất.

Trong quá trình huấn luyện, mô hình đã được tối ưu hóa thông qua việc sử dụng hàm mất mát Binary Cross Entropy (BCE) và thuật toán tối ưu Stochastic Gradient Descent (SGD). Chúng tôi đã theo dõi quá trình huấn luyện và đảm bảo rằng hàm mất mát giảm dần qua từng epoch, cho thấy rằng mô hình đã học và cải thiện hiệu suất.

4.3 Đánh giá kết quả

Kết quả đạt được từ mô hình cho thấy độ chính xác khoảng 50.37%, tuy nhiên, điều này không hoàn toàn phản ánh khả năng phân loại tốt. Mặc dù mô hình đã dự đoán đúng một số lượng lớn các trường hợp thuộc cả hai lớp, việc phân tích ma trận nhầm lẫn cho thấy một số nhầm lẫn giữa các lớp, đặc biệt là với các trường hợp khó phân loại gần ngưỡng 0.5.

Chúng tôi cũng đã sử dụng các chỉ số như Precision, Recall, và F1-Score để đánh giá chi tiết hơn về hiệu suất của mô hình. Kết quả cho thấy rằng mô hình có sự cân bằng tốt giữa việc nhận diện chính xác các trường hợp thuộc cả hai lớp, nhưng vẫn cần cải thiện để giảm thiểu lỗi loại I và loại II.

4.4 Hướng phát triển

Trong tương lai, chúng tôi có thể cải thiện mô hình bằng cách áp dụng các kỹ thuật tiên tiến hơn như tăng cường dữ liệu để mở rộng tập huấn luyện, sử dụng các phương pháp regularization như L1 hoặc L2 để giảm thiểu hiện tượng quá khớp (overfitting), và thử nghiệm với các mô hình phức tạp hơn như mạng nơ-ron nhân tạo, cây quyết định hoặc các thuật toán ensemble để cải thiện hiệu suất phân loại.

Ngoài ra, chúng tôi cũng có thể nghiên cứu và áp dụng các phương pháp xử lý dữ liệu không cân bằng, chẳng hạn như resampling, SMOTE (Synthetic Minority Over-sampling Technique), để cải thiện khả năng nhận diện lớp thiểu số.

4.5 Tổng kết

Logistic Regression là một thuật toán cơ bản nhưng mạnh mẽ và hiệu quả trong nhiều bài toán phân loại nhị phân. Mặc dù có một số hạn chế, đặc biệt là với dữ liệu lớn và phức tạp, nhưng nó vẫn là lựa chọn phù hợp khi muốn triển khai một mô hình phân loại nhanh chóng, đơn giản và dễ giải thích. Kết quả từ mô hình cũng chỉ ra rằng, mặc dù Logistic Regression có thể không phải là lựa chọn tốt nhất cho mọi bài toán, nhưng với các điều chỉnh thích hợp, nó có thể hoạt động hiệu quả trong nhiều tình huống thực tế.

Chúng tôi hy vọng rằng những kết quả và phân tích này sẽ cung cấp cái nhìn sâu sắc hơn về khả năng của mô hình Logistic Regression và mở ra hướng đi mới cho các nghiên cứu và ứng dụng tiếp theo trong lĩnh vực học máy.

TÀI LIỆU THAM KHẢO

[1].https://www.elcom.com.vn/may-hoc-machine-learning-la-gi-ung-dung-cong-nghe-may-hoc-trong-thuc-tien-1666003970

[2]. https://200lab.io/blog/google-colab-la-gi

[3].https://www.tma.vn/Hoi-dap/cam-nang-nghe-nghiep/Top-10-Thuat-toan-Machine-Learning-danh-cho-nguoi-moi-hoc/41477