ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO BÀI TẬP LỚN

SONG SONG HÓA CÁC THUẬT TOÁN HỌC MÁY CHO BÀI TOÁN DỰ ĐOÁN VỠ NỢ TÍN DỤNG TRÊN BỘ DỮ LIỆU LỚN

Thực hiện

Đặng Quý Anh

Trần Thanh Tùng

Vũ Thị Thùy Dung

Giảng viên hướng dẫn

TS. Cao Văn Chung

HÀ NỘI - 2022

Lời giới thiệu

Học máy là một thuật ngữ chung cho một tập hợp các kỹ thuật và công cụ giúp máy tính tự học và thích nghi. Các thuật toán học máy giúp AI học mà không được lập trình rõ ràng để thực hiện hành động mong muốn. Bằng cách học một mẫu từ đầu vào mẫu, thuật toán máy học dự đoán và thực hiện các tác vụ chỉ dựa trên mẫu đã học chứ không phải hướng dẫn chương trình được xác định trước. Học máy là một vị cứu tinh trong một số trường hợp không thể áp dụng các thuật toán truyền thống. Nó sẽ học quy trình mới từ các mẫu trước đó và đưa ra một mô hình cho việc dự đoán trong tương lai.

Tuy nhiên cùng với việc phát triển của khoa học công nghệ, dữ liệu được lưu trữ cũng ngày được tăng lên với cấp số nhân. Vì thế ngoài việc áp dụng các thuật toán và kỹ thuật học máy đơn thuần, thì MapReduce là một giải pháp hiệu quả để song song hóa quá trình huấn luyện nhằm làm tăng tốc độ mô hình. Ở bài báo cáo, nhóm sẽ thực hiện dự đoán khả năng vỡ nợ tín dụng từ bộ dữ liệu lớn do American Express cung cập với 3 thuật toán học máy chính gồm K-means, Decision Tree và Support Vector Machine.

Lời cảm ơn

Chúng em xin bày tỏ lòng biết ơn chân thành và sâu sắc đến thầy **Cao Văn Chung** đã dày công truyền đạt kiến thức và tận tình hướng dẫn, chỉ bảo chúng em trong suốt quá trình học tập và hoàn thành tiểu luận.

Tuy nhiên, do kiến thức và thời gian hoàn thành có hạn nên bài tiểu luận của chúng em không tránh khỏi những sai sót. Do đó, em rất mong nhận được sự góp ý của thầy để chúng em có điều kiện hoàn thiện hơn kiến thức của mình.

Chúng em xin chân thành cảm ơn!

Đặng Quý Anh
Trần Thanh Tùng
Vũ Thị Thùy Dung

Mục lục

1 Lý do chọn đề tài **4**

2 Dữ liệu **4**

3 MapReduce **5**
3.1 MapReduce là gì? 5
3.2 Các hàm chính của MapReduce 5

4 Một số thuật toán học máy cho dữ liệu lớn **6**
4.1 Thuật toán k-means 6
4.1.1 Ý tưởng thuật toán 6
4.1.2 k-means với MapReduce 7
4.1.3 Ưu điểm và hạn chế 8
4.2 Thuật toán Decision Tree 8
4.2.1 Một số thuật ngữ quan trọng liên quan đến Decision Tree 8
4.2.2 Thuật toán ID3 cho Decision Tree 9
4.2.3 Decision Tree với MapReduce 10
4.2.4 Ưu điểm và một số hạn chế 11
4.3 Thuật toán Support Vector Machines 12
4.3.1 Ý tưởng thuật toán 12
4.3.2 SVMs với Gradient Descent 16
4.3.3 SVMs với MapReduce 17

5 Thực nghiệm **17**
5.1 Tiền xử lý dữ liệu 17
5.1.1 Giảm kích thước dữ liệu 17
5.1.2 Xử lý dữ liệu bị thiếu 18
5.2 Khai phá dữ liệu 19
5.3 Xây dựng mô hình huấn luyện 20
5.3.1 Một số metric để đánh giá mô hình 20
5.3.2 Song song hóa với pyspark 21
5.3.3 Mô hình K-means 22
5.3.4 Mô hình Decision Tree 22
5.3.5 Mô hình SVM 23

6 Tổng kết **24**

7 Tài liệu tham khảo **25**

1 Lý do chọn đề tài

Cho dù đi ăn nhà hàng hay mua vé xem hòa nhạc, cuộc sống hiện đại đều dựa vào sự tiện lợi của thẻ tín dụng để mua hàng hàng ngày. Nó giúp chúng ta không phải mang theo một lượng lớn tiền mặt và cũng có thể tạm ứng toàn bộ giao dịch mua có thể được thanh toán theo thời gian. Làm thế nào để các tổ chức phát hành thẻ biết chúng ta sẽ trả lại những gì được thanh toán? Đó là một vấn đề phức tạp với nhiều giải pháp hiện có và thậm chí nhiều cải tiến tiềm năng hơn sẽ được khám phá lĩnh vực này.

Dự đoán vỡ nợ tín dụng (Credit Default Prediction) là trung tâm để quản lý rủi ro trong kinh doanh cho vay tiêu dùng. Dự đoán vỡ nợ tín dụng cho phép người cho vay tối ưu hóa các quyết định cho vay, dẫn đến trải nghiệm khách hàng tốt hơn và kinh doanh hợp lý. Các mô hình hiện tại tồn tại để giúp quản lý rủi ro. Nhưng có thể tạo ra những mô hình tốt hơn có thể hoạt động tốt hơn những mô hình hiện đang được sử dụng.

Trong phạm vi bài báo cáo này, nhóm sẽ áp dụng các thuật toán học máy (Machine Learning) để dự đoán tình trạng vỡ nợ tín dụng. Cụ thể, nhóm sẽ tận dụng tập dữ liệu quy mô công nghiệp để xây dựng mô hình học máy thách thức mô hình hiện tại trong sản xuất. Bộ dữ liệu đào tạo, xác thực và thử nghiệm bao gồm dữ liệu hành vi chuỗi thời gian và thông tin hồ sơ khách hàng ẩn danh. Bạn có thể tự do khám phá bất kỳ kỹ thuật nào để tạo mô hình hiệu quả nhất, từ việc tạo các tính năng đến sử dụng dữ liệu theo cách hữu cơ hơn trong mô hình. Mô hình được xây dựng thành công sẽ tạo ra trải nghiệm khách hàng tốt hơn cho chủ thẻ bằng cách giúp việc phê duyệt thẻ tín dụng dễ dàng hơn. Mục tiêu của bài tập lớn này là dự đoán khả năng khách hàng không trả lại số dư thẻ tín dụng của họ trong tương lai dựa trên hồ sơ khách hàng hàng tháng của họ. Biến nhị phân mục tiêu được tính toán bằng cách quan sát cửa sổ hiệu suất 18 tháng sau ngày sao kê thẻ tín dụng mới nhất và nếu khách hàng không thanh toán số tiền đến hạn trong 120 ngày sau ngày sao kê gần nhất thì đó được coi là sự kiện vỡ nợ.

2 Dữ liệu

American Express là một công ty thanh toán tích hợp toàn cầu. Là nhà phát hành thẻ thanh toán lớn nhất trên thế giới, họ cung cấp cho khách hàng quyền truy cập vào các sản phẩm, thông tin chi tiết và trải nghiệm giúp làm phong phú cuộc sống và xây dựng thành công trong kinh doanh. Bộ dữ liệu được American Express chứa các đặc trưng hồ sơ tổng hợp cho từng khách hàng tại mỗi ngày sao kê. Các đặc trưng được ẩn danh và chuẩn hóa, đồng thời thuộc các danh mục chung sau:

- D_* = Delinquency variables
- S_* = Spend variables
- P_* = Payment variables
- B_* = Balance variables
- R_* = Risk variables

Nhiệm vụ của bài toán là dự đoán, đối với mỗi customer_ID, xác suất không thanh toán được trong tương lai (mục tiêu = 1). Về kích thước, bộ dữ liệu là bộ dữ liệu lớn với khoảng 50.31 GB được chứa trong các file chủ yếu

- **train_data.csv**: Dữ liệu đào tạo với nhiều ngày sao kê cho mỗi **customer_ID**.

- **train_labels.csv**: Nhãn mục tiêu cho các **customer_ID** tương ứng.
- **test_data.csv**: Dữ liệu kiểm thử để dự đoán nhãn mục tiêu cho từng **customer_ID**

Hiện nay, bộ dữ liệu này cũng đã được chia sẻ công khai và từng được tổ chức dưới dạng cuộc thi *American Express - Default Prediction* trên *Kaggle* tại đường dẫn liên kết *https://www.kaggle.com/competitions/amex-default-prediction/data*

3 MapReduce

3.1 MapReduce là gì?

MapReduce là mô hình được thiết kế độc quyền bởi Google, nó có khả năng lập trình xử lý các tập dữ liệu lớn song song và phân tán thuật toán trên 1 cụm máy tính. MapReduce trở thành một trong những thành ngữ tổng quát hóa trong thời gian gần đây.

MapReduce sẽ bao gồm những thủ tục sau: thủ tục 1 Map() và 1 Reduce(). Thủ tục Map() bao gồm lọc (filter) và phân loại (sort) trên dữ liệu khi thủ tục khi thủ tục Reduce() thực hiện quá trình tổng hợp dữ liệu. Đây là mô hình dựa vào các khái niệm biến đối của bản đồ và reduce những chức năng lập trình theo hướng chức năng. Thư viện của thủ tục Map() và Reduce() sẽ được viết bằng nhiều loại ngôn ngữ khác nhau. Thủ tục được cài đặt miễn phí và được sử dụng phổ biến nhất là Apache Hadoop hay Apache Spark.

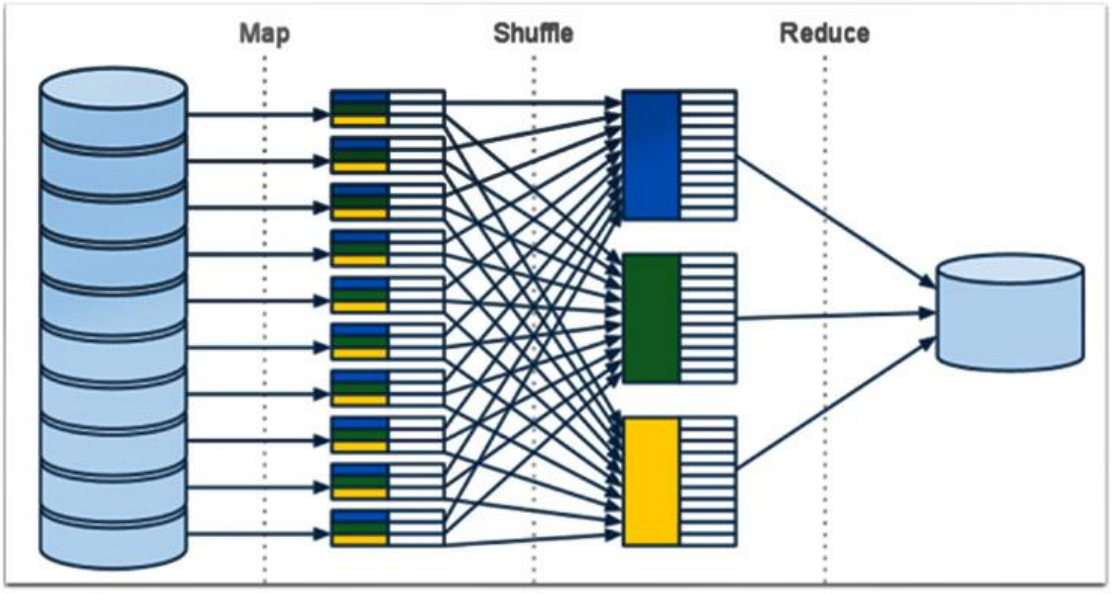
3.2 Các hàm chính của MapReduce

MapReduce có 2 hàm chính là Map() và Reduce(), đây là 2 hàm đã được định nghĩa bởi người dùng và nó cũng chính là 2 giai đoạn liên tiếp trong quá trình xử lý dữ liệu của MapReduce. Nhiệm vụ cụ thể của từng hàm như sau

- Hàm Map(): có nhiệm vụ nhận Input cho các cặp giá trị/khóa và output chính là tập những cặp giá trị/khóa trung gian. Sau đó, chỉ cần ghi xuống đĩa cứng và tiến hành thông báo cho các hàm Reduce() để trực tiếp nhận dữ liệu.

- Hàm Reduce(): có nhiệm vụ tiếp nhận từ khóa trung gian và những giá trị tương ứng với lượng từ khóa đó. Sau đó, tiến hành ghép chúng lại để có thể tạo thành một tập khóa khác nhau. Các cặp khóa/giá trị này thường sẽ thông qua một con trỏ vị trí để đưa vào các hàm reduce. Quá trình này sẽ giúp cho lập trình viên quản lý dễ dàng hơn một lượng danh sách cũng như phân bổ giá trị sao cho phù hợp nhất với bộ nhớ hệ thống.

- Ở giữa Map và Reduce thì còn 1 bước trung gian đó chính là Shuffle. Sau khi Map hoàn thành xong công việc của mình thì Shuffle sẽ làm nhiệm vụ chính là thu thập cũng như tổng hợp từ khóa/giá trị trung gian đã được map sinh ra trước đó rồi chuyển qua cho Reduce tiếp tục xử lý.



Hình 1: Mô hình MapReduce

4 Một số thuật toán học máy cho dữ liệu lớn

4.1 Thuật toán k-means

4.1.1 Ý tưởng thuật toán

Phân cụm là một tập hợp các kỹ thuật được sử dụng để phân vùng dữ liệu thành các nhóm hoặc cụm. Các cụm được định nghĩa là các nhóm đối tượng dữ liệu giống với các đối tượng khác trong cụm của chúng hơn là với các đối tượng dữ liệu trong các cụm khác. Phương pháp phân cụm **k-means** là một kỹ thuật học máy không giám sát (unsupervised learning) được sử dụng để xác định các cụm đối tượng dữ liệu trong tập dữ liệu và là một thuật toán phân cụm phân vùng chia các đối tượng dữ liệu thành các nhóm không chồng chéo. Nói cách khác, không có đối tượng nào có thể là thành viên của nhiều hơn một cụm và mỗi cụm phải có ít nhất một đối tượng. Các kỹ thuật này yêu cầu người dùng chỉ định số cụm, được biểu thị bằng biến k. Nhiều thuật toán phân cụm phân vùng hoạt động thông qua một quy trình lặp đi lặp lại để gán các tập hợp con của các điểm dữ liệu thành k cụm. Có nhiều loại phương pháp phân cụm khác nhau, nhưng k-means là một trong những phương pháp lâu đời nhất và dễ tiếp cận nhất.

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

Hình 2: Thuật toán k-means

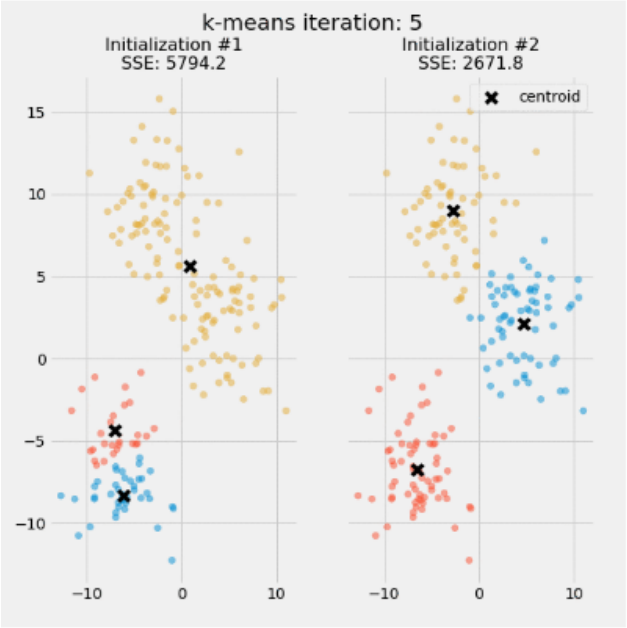
Thuật toán k-means thông thường chỉ cần một vài bước. Bước đầu tiên là chọn ngẫu

nhiên k tâm, trong đó k bằng số cụm chọn. **Centroid** là các điểm dữ liệu đại diện cho tâm của một cụm. Yếu tố chính của thuật toán hoạt động theo quy trình gồm hai bước được gọi là **tối đa hóa kỳ vọng (expectation maximization)**. Bước kỳ vọng (**expectation**) gán từng điểm dữ liệu cho trọng tâm gần nhất của nó. Sau đó, bước tối đa hóa (**maximization**) tính toán giá trị trung bình của tất cả các điểm cho từng cụm và đặt trọng tâm mới. Hai bước này lặp đi lặp lại cho tới khi không có sự thay đổi giữa các tâm cụm. Và bằng toán học, người ta cũng đã chứng minh phương pháp này là hội tụ, tức tới một bước lặp nhất định, các *centroid* không đổi. Điều này là quan trọng để đảm bảo tính dừng của thuật toán.

Chất lượng của các phép gán cụm trên được đo bằng cách tính tổng bình phương sai số (Sum of Square Error - SSE) sau khi các trọng tâm hội tụ hoặc khớp với phép gán của lần lặp trước đó. SSE được định nghĩa là tổng các khoảng cách Euclide bình phương của mỗi điểm đến trọng tâm gần nhất của nó. Vì đây là thước đo lỗi nên mục tiêu của phương pháp k-means là cố gắng giảm thiểu giá trị này. Cụ thể, giả sử $X = \{x_1, x_2, ..., x_n\}$ là một tập n quan sát trong không gian d chiều. Thuật toán k-means đi tìm một tập k phần tử $C = \{\mu_1, \mu_2, ..., \mu_k\}$ để tối thiểu hóa hàm

$$J(C) = \sum_{x \in X} \min_{\mu \in C} \|x - \mu\|^2$$

Nói cách khác, chúng ta muốn chọn k tâm để giảm thiểu tổng của khoảng cách bình phương giữa mỗi điểm trong tập dữ liệu và giá trị trung bình gần nhất với điểm đó.



Hình 3: SSE được cập nhật qua 5 vòng lặp với khởi tạo khác nhau trên cùng một tập dữ liệu

4.1.2 k-means với MapReduce

Nhớ lại rằng mỗi lần lặp lại phương pháp k-means có thể được chia thành hai giai đoạn, giai đoạn đầu tiên tính toán các tập hợp S_i của các điểm gần nhất với giá trị trung bình μ_i và giai đoạn thứ hai tính toán các tâm cụm mới theo các S_i. Hai giai đoạn này ứng với Map và Reduce trong thuật toán MapReduce.

Ở bước Map, mỗi điểm dữ liệu x trong tập dữ liệu, ta tính toán bình phương khoảng cách tới mỗi tâm cụm để tìm ra μ_i gần nó nhất lấy chỉ số làm khóa (key). Còn giá trị (value) là

cặp khoảng cách cùng chỉ số của quan sát đó trong tập dữ liệu (được ký hiệu là id_x). Đến bước Reduce, ta chỉ việc nhóm các cặp khóa - gía trị theo chỉ số tâm cụm đồng thời tính tổng khoảng cách ứng với mỗi tâm và cập nhật tập S_i.

- Map

$$(\text{argmin}_i \|x - \mu_i\|^2, (d(x, \mu_i), id_x))$$

- Reduce

$$(i, [(d(x, \mu_i), id_x), (d(y, \mu_i), id_y)]) \Rightarrow (i, ((d(x, \mu_i) + (d(y, \mu_i), id_x \cup id_y))$$

MapReduce được đặc trưng bởi hai quá trình này tạo ra một bộ k các giá trị ứng với k tâm cụm cần tìm.

$$(i, (\sum_{x \in S_i} d(x, \mu_i), S_i))$$

trong đó S_i biểu thị tập hợp các điểm gần nhất với giá trị trung bình μ_i. Từ đó chúng ta có thể cập nhật tâm cụm mới của S_i là

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} d(x, \mu_i)$$

4.1.3 Ưu điểm và hạn chế

Phương pháp k-means nói riêng và các thuật toán phân cụm phân vùng khác nói chung đều không có tính xác định, nghĩa là chúng có thể tạo ra các kết quả khác nhau từ hai lần chạy riêng biệt ngay cả khi các lần chạy dựa trên cùng một đầu vào. Ưu điểm là chúng hoạt động tốt khi các cụm có dạng hình cầu và có thể mở rộng (scale) tùy theo độ phức tạp của thuật toán. Còn về hạn chế thì các phương pháp này không phù hợp với các cụm có hình dạng phức tạp và kích cỡ khác nhau và thường bị "lỗi" khi được sử dụng với các cụm có mật độ khác nhau.

4.2 Thuật toán Decision Tree

Thuật toán Decision Tree thuộc họ thuật toán học có giám sát. Không giống như các thuật toán học có giám sát khác, thuật toán Decision Tree cũng có thể được sử dụng để giải các bài toán hồi quy và phân loại. Mục tiêu là tạo ra một mô hình dự đoán giá trị của biến mục tiêu bằng cách học các quy tắc quyết định đơn giản được suy ra từ các tính năng dữ liệu. Một cây có thể được coi là một xấp xỉ hằng số từng phần. Trong Decision Tree, để dự đoán nhãn lớp cho các quan sát, chúng ta bắt đầu từ gốc của cây, so sánh các giá trị của thuộc tính gốc với thuộc tính của các quan sát. Trên cơ sở so sánh, ta đi theo nhánh tương ứng với giá trị đó và nhảy sang nút tiếp theo.

4.2.1 Một số thuật ngữ quan trọng liên quan đến Decision Tree

Để hiểu về thuật toán Decision Tree, trước hết cần hiểu về các thành phần cấu tạo lên cây (tree). Cụ thể

- Nút gốc (Root node): Đại diện cho toàn bộ mẫu, từ nút này ta có thể đi đến các nút khác trong cây.
- Quá trình chia (Splitting): Quá trình chia một nút thành hai hoặc nhiều nút con.

- Nút quyết định (Decision node): Khi một nút con tách thành các nút con khác, thì nó được gọi là nút quyết định.
- Nút lá (Leaf hay Terminal node): Các nút không phân tách được (không có nút con).
- Cắt tỉa (Pruning): Khi chúng ta loại bỏ các nút con của một nút quyết định, quá trình này được gọi là cắt tỉa.
- Nhánh/Cây con (Branch/Subtree): Một phần con của toàn bộ cây được gọi là nhánh hoặc cây con.
- Nút cha và nút con (parent and child node): Một nút được chia thành các nút con được gọi là nút cha của các nút con.

4.2.2 Thuật toán ID3 cho Decision Tree

Quyết định phân chia là chiến lược ảnh hưởng rất nhiều đến độ chính xác của cây. Các tiêu chí quyết định là khác nhau đối với cây phân loại và cây hồi quy. Decision Tree sử dụng nhiều thuật toán để quyết định chia một nút thành hai hoặc nhiều nút con. Việc tạo các nút con làm tăng tính đồng nhất của các nút phụ kết quả. Nói cách khác, chúng ta có thể nói rằng độ tinh khiết của nút tăng lên đối với biến mục tiêu. Decision Tree phân tách các nút trên tất cả các biến có sẵn và sau đó chọn cách phân tách dẫn đến hầu hết các nút con đồng nhất. Một trong những thuật toán dễ hiểu nhất trong nhóm Decision Tree là thuật toán ID3 (Iterative Dichotomiser 3).

Thuật toán ID3 bắt đầu với tập gốc S là nút gốc. Trên mỗi lần lặp của thuật toán, nó lặp qua mọi thuộc tính không được sử dụng của tập S và tính toán entropy $H(S)$ hoặc thông tin thu được $IG(S)$ của thuộc tính đó. Sau đó chọn thuộc tính có giá trị entropy nhỏ nhất (hoặc mức tăng thông tin lớn nhất). Tập hợp S sau đó được phân chia hoặc phân vùng bởi thuộc tính đã chọn để tạo ra các tập hợp con của dữ liệu (Ví dụ: một nút có thể được chia thành các nút con dựa trên các tập hợp con của dân số có độ tuổi dưới 50, từ 50 đến 100 và lớn hơn 100). Thuật toán tiếp tục lặp lại trên mỗi tập hợp con, chỉ xem xét các thuộc tính chưa từng được chọn trước đó. Như vậy có thể tóm tắt thuật toán ID3 qua các bước sau

- Tính entropy của mọi thuộc tính a của tập dữ liệu S
- Chia tập S thành các tập con theo thuộc tính mà entropy thu được sau khi chia nhánh là nhỏ nhất hoặc thông tin thu được là lớn nhất.
- Tạo nút quyết định chứa thuộc tính đó.
- Tiếp tục trên các tập con sử dụng các thuộc tính còn lại.

Việc đệ quy hay tiếp tục trên một tập hợp con có thể dừng lại ở một trong các trường

- Mọi phần tử trong tập hợp con đều thuộc về cùng một lớp; trong trường hợp đó, nút được biến thành nút lá và được gắn nhãn với lớp của các quan sát.
- Không còn thuộc tính nào được chọn, nhưng các quan sát vẫn không thuộc cùng một lớp. Trong trường hợp này, nút được tạo thành một nút lá và được gắn nhãn với lớp phổ biến nhất của các quan sát trong tập hợp con.

- Không có quan sát nào trong tập hợp con, điều này xảy ra khi không tìm thấy mẫu nào trong tập hợp gốc khớp với một giá trị cụ thể của thuộc tính đã chọn. Một ví dụ có thể là sự vắng mặt của một người trong dân số trên 100 tuổi. Sau đó, một nút lá được tạo và gắn nhãn với lớp phổ biến nhất của các ví dụ trong tập hợp của nút cha.

Entropy
Entropy là thước đo tính ngẫu nhiên trong thông tin đang được xử lý. Entropy càng cao thì càng khó rút ra bất kỳ kết luận nào từ thông tin đó. Nói cách khác, entropy $H(S)$ là thước đo mức độ không chắc chắn trong tập dữ liệu S.

$$H(S) = \sum_{y \in Y} -p(y) \log_2 p(y)$$

trong đó

- S là tập dữ liệu hiện tại mà entropy đang được tính toán
- Y là tập hợp các lớp (nhãn) trong S
- $p(y)$ là tỉ lệ giữa số phần tử của lớp y với số phần tử của tập hợp S

Khi $H(S) = 0$, tập S được phân loại hoàn hảo (nghĩa là tất cả các phần tử trong S đều thuộc cùng một lớp). Trong ID3, entropy được tính cho từng thuộc tính còn lại, thuộc tính có entropy nhỏ nhất được sử dụng để phân chia tập hợp S trong lần lặp này.

Information Gain (Thông tin thu được)
Thông tin thu được $IG(A)$ là thước đo sự khác biệt về entropy từ trước đến sau khi tập hợp S được tách ra trên thuộc tính A. Nói cách khác, mức độ không chắc chắn trong S đã giảm sau khi tách tập hợp S trên thuộc tính A.

$$IG(S, A) = H(S) - \sum_{t \in T} p(t)H(t) = H(S) - H(S|A)$$

trong đó

- $H(S)$: Entropy của S
- T: Các tập con được tạo ra từ việc tách tập S theo thuộc tính A sao cho $S = \cup_{t \in T} t$
- $p(t)$: Tỉ lệ giữa số phần tử của t với số phần tử của tập hợp S
- $H(t)$: Entropy của tập hợp con t

4.2.3 Decision Tree với MapReduce

Ở thuật toán ID3 cũng như các thuật toán Decision Tree, việc tính toán thông tin thu được (Information Gain) để quyết định chọn thuộc tính có giá trị này lớn nhất làm cơ sở chia một nút thành các nút con luôn được thực hiện đồng thời trên tất cả các thuộc tính còn lại của cây. Do đó ở bước Map, khóa là $(att, threshold)$ trong đó att là thuộc tính đang được xem xét, $threshold$ là ngưỡng chọn để chia nhánh và giá trị là nhãn y_i tương ứng với quan sát x_i. Đến bước Reduce thì việc đơn giản là thực hiện nhóm các nhãn thành một tập hợp. Một cách tổng quát hóa

- Map: cặp khóa - giá trị là $((att, threshold), y_i)$
- Reduce:

$$((att, threshold), y_i) \text{ và } ((att, threshold), y_j) \Rightarrow ((att, threshold), \{y_i, y_j\})$$

Lưu ý rằng ta chọn y_i làm giá trị và dùng cho việc tính toán Information Gain và Entropy để chọn ra att tốt nhất cũng như ngưỡng $threshold$ quyết định.

4.2.4 Ưu điểm và một số hạn chế

Dù được triển khai bằng thuật toán nào thì Decision Tree cũng có những ưu điểm và nhược điểm sau.

Một số ưu điểm

- Đơn giản để hiểu và giải thích. Cây có thể được trực quan hóa.
- Yêu cầu xử ký dữ liệu ít. Các kỹ thuật khác thường yêu cầu chuẩn hóa dữ liệu, cần tạo các biến giả và xóa các giá trị trống.
- Chi phí thuật toán để dự đoán là logarit theo số lượng điểm dữ liệu được sử dụng để huấn luyện cây.
- Có thể xử lý cả dữ liệu số và phân loại và xử lý biến đa mục tiêu.
- Sử dụng mô hình "white box". Nếu một tình huống nhất định có thể quan sát được trong mô hình, thì lời giải thích cho điều kiện đó dễ dàng được giải thích bằng câu trả lời có hoặc không.
- Có thể kiểm đinh một mô hình bằng cách sử dụng thống kê. Điều đó làm nâng cao đến độ tin cậy của mô hình.

Một số hạn chế

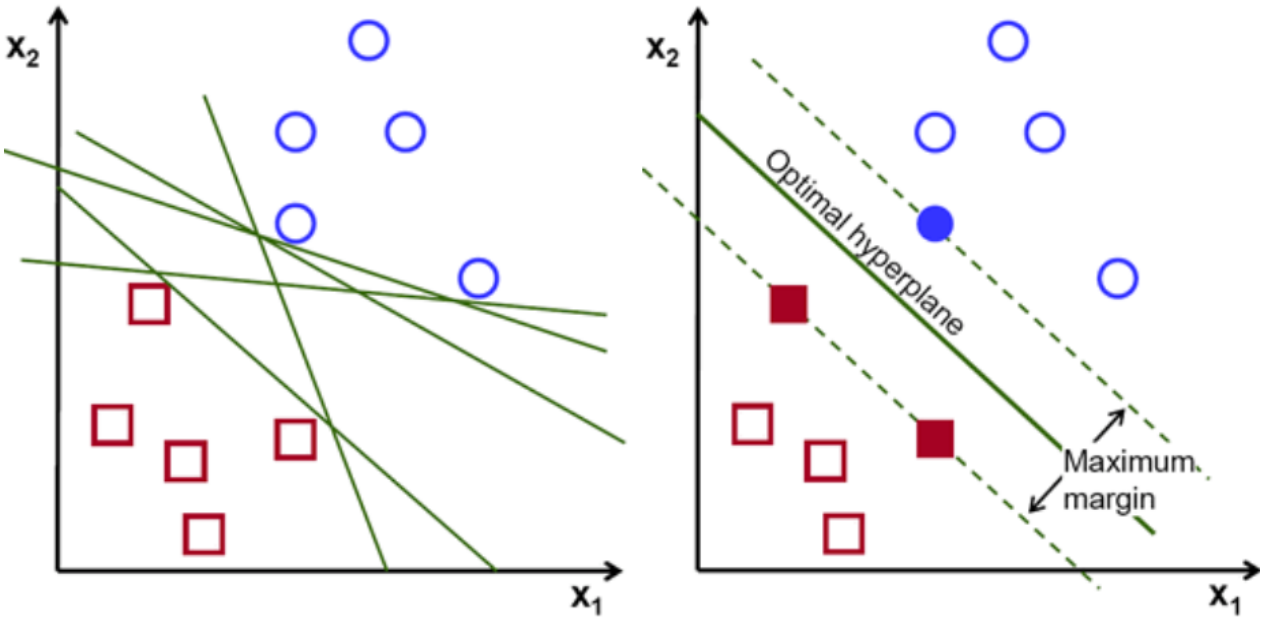
- Mô hình có thể tạo ra các cây quá phức tạp không tổng quát hóa dữ liệu. Điều này được gọi là mô hình quá khớp (overfitting). Các cơ chế như cắt tỉa (pruning), đặt số lượng mẫu tối thiểu cần thiết tại một nút lá hoặc đặt độ sâu tối đa của cây là cần thiết để tránh vấn đề này.
- Có thể không ổn định vì các biến thể chiếm tỉ lệ nhỏ trong dữ liệu có thể dẫn đến việc tạo ra một cây hoàn toàn khác. Vấn đề này được giảm thiểu bằng cách sử dụng các cây quyết định trong một tập hợp.
- Dự đoán của cây quyết định không trơn tru cũng không liên tục, mà là xấp xỉ hằng số từng phần. Do đó không tốt cho quá trình ngoại suy.
- Có những khái niệm khó học vì cây quyết định không diễn đạt chúng dễ dàng, chẳng hạn như XOR, bài toán chẵn lẻ hoặc bộ ghép kênh.
- Decision Tree tạo cây lệch phía nếu một số lớp chiếm ưu thế. Do đó, nên cân bằng tập dữ liệu trước khi huấn luyện.

4.3 Thuật toán Support Vector Machines

Support Vector Machines (SVMs) là một mô hình học máy (Machine Learning) rất mạnh mẽ và linh hoạt, có khả năng thực hiện phân loại tuyến tính hoặc phi tuyến tính, hồi quy và thậm chí phát hiện ngoại lệ. SVMs đặc biệt phù hợp để phân loại các bộ dữ liệu phức tạp nhưng có kích thước nhỏ hoặc trung bình.

4.3.1 Ý tưởng thuật toán

Giả sử chúng ta có một tập dữ liệu có thể phân tách tuyến tính với từng cặp điểm, $(x^{(1)}, y_1), (x^{(2)}, y_2), (x^{(3)}, y_3), ..., (x^{(n)}, y_n)$, trong đó n là số quan sát, $x^{(i)} = (x_1, x_2, ..., x_d)$ và d là chiều của lần quan sát thứ i. Như hình bên dưới, chúng ta có vô số siêu phẳng ứng cử viên, và do đó có các bộ phân loại, giải quyết vấn đề phân loại mà không có bất kỳ lỗi huấn luyện nào. Tuy nhiên để tìm một giải pháp duy nhất, một ý tưởng là chọn siêu phẳng phân cách sao cho cực đại biên (margin) giữa hai nhãn. Nói cách khác, ta muốn muốn mỗi nhãn được phân tách bằng lề lớn nhất.



Hình 4: Các siêu phẳng ứng viên (bên trái) và Siêu phẳng tối ưu (bên phải)

Khái niệm về lề rất đơn giản về mặt trực quan: Đó là khoảng cách của siêu phẳng phân tách đến các quan sát gần nhất trong tập dữ liệu, giả sử rằng tập dữ liệu có thể phân tách tuyến tính.

Như đã đề cập, chúng ta cần tìm siêu phẳng tối ưu với lề lớn nhất. Giả sử $\mathbf{x} \in \mathbb{R}^d$ là một phần tử của không gian dữ liệu. Xét hàm

$$f : \mathbb{R}^d \to \mathbb{R}$$

$$\mathbf{x} \mapsto f(\mathbf{x}) := \mathbf{w}^\mathbf{T}\mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

được tham số hóa bởi $\mathbf{w} \in \mathbb{R}^d$ và $b \in \mathbb{R}$. Nhớ lại rằng các siêu phẳng là các không gian con affine. Do đó, ta xác định siêu phẳng phân tách hai lớp trong bài toán phân loại nhị phân là

$$\{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) = 0\}$$

Khi huấn luyện, chúng ta muốn đảm bảo rằng các quan sát có nhãn "positive" nằm ở phía "positive" của siêu phẳng

$$\mathbf{w^T x} + b \geq 0 \text{ when } y = +1$$

và các quan có nhãn "negative" nằm ở phía "negative" của siêu phẳng

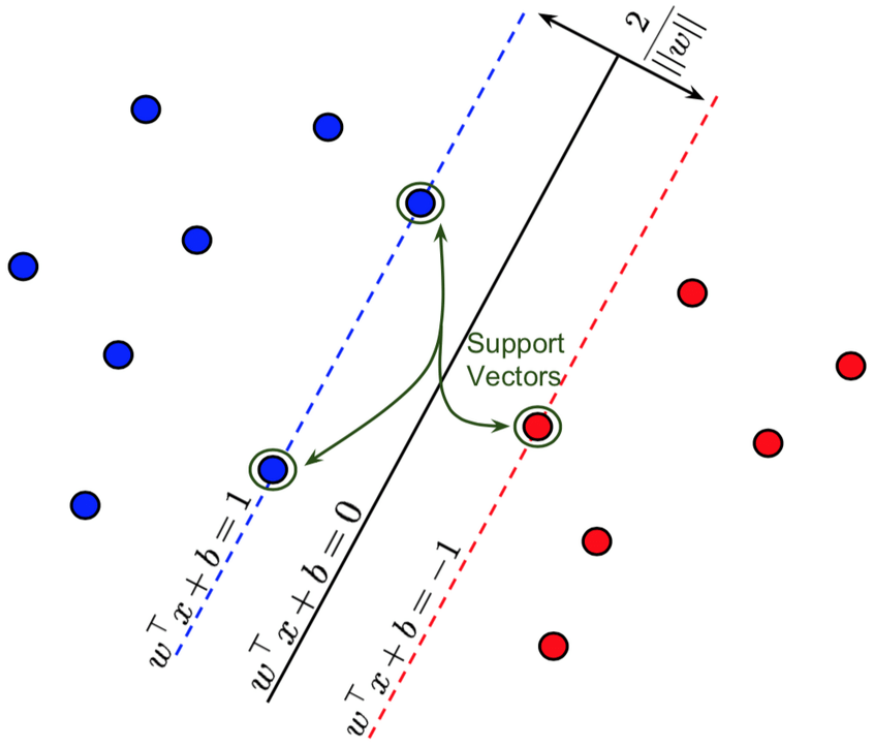
$$\mathbf{w^T x} + b < 0 \text{ when } y = -1$$

Hard Margin

Một cách tiếp cận khác không làm thay đổi các thuộc tính của siêu phẳng tối ưu là

$\mathbf{w^T x} + b = 1$ (bất cứ điểm nào nằm bên trên đường phân chia đều được gãn nhãn 1)

$\mathbf{w^T x} + b = -1$ (bất cứ điểm nào nằm bên dưới đường phân chia đều được gãn nhãn -1)



Hình 5: Hard Margin

Chúng ta muốn các quan sát "positive" và "negative" cách xa siêu phẳng phân cách ít nhất 1, điều này dẫn đến điều kiện

$$y_i(\mathbf{w^T x^{(i)}} + b) \geq 1$$

Kết hợp với việc tối đa hóa lề với thực tế là các quan sát cần phải ở đúng vị trí qua siêu phẳng (dựa trên nhãn của chúng) thì cuối cùng ta được một bài toán tối ưu hóa tổng quát với ràng buộc

$$\max_{w,b} \frac{1}{\|\mathbf{w}\|}$$

s.t

$$y_i(\mathbf{w^T x^{(i)}} + b) \geq 1 \quad \text{với mọi } i = 1, 2, ..., n$$

Thay vì cực đại nghịch đảo của chuẩn như trên, ta thường muốn cực tiểu hóa chuẩn bình phương của chuẩn. Thêm hằng số $\frac{1}{2}$ không ảnh hưởng đến tìm w, b tối ưu nhưng mang lại thuận lợi tính toán khi dùng thuật toán "Gradient Descent". Cuối cùng, ta được

$$\min_{w,b} \frac{1}{2}\|\mathbf{w}\|^2$$

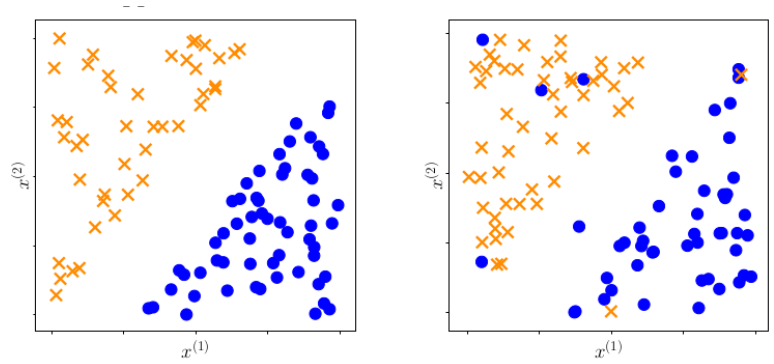
s.t

$$y_i(\mathbf{w^T x^{(i)}} + b) \geq 1 \quad \text{với mọi } i = 1, 2, ..., n$$

Phương trình này được gọi là hard margin cho SVM. Lý do cho cụm từ "hard" là vì không cho phép bất kỳ hành vi phạm điều kiện các quan sát ở sai vị trí nào.

Soft Margin

Trong trường hợp dữ liệu không thể phân tách tuyến tính, chúng ta có thể muốn cho phép một số quan sát nằm trong vùng lề hoặc thậm chí nằm ở sai phía của siêu phẳng như minh họa bên dưới

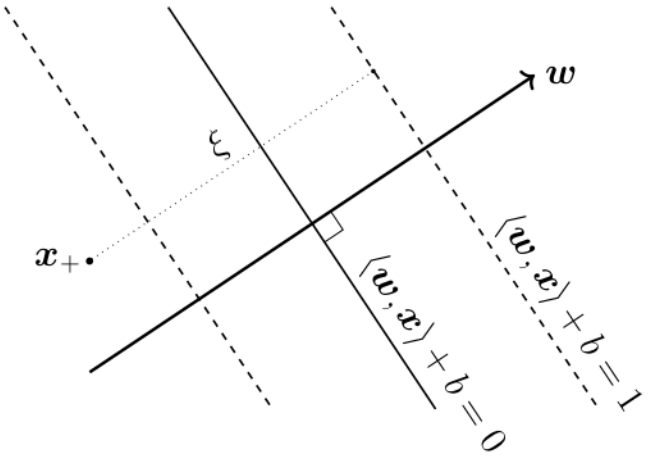


(a) Linearly separable data, with a large margin

(b) Non-linearly separable data

Hình 6: Soft Margin

Mô hình cho phép một số lỗi phân loại được gọi là SVM "soft-margin".



Hình 7: Soft Margin

Ý tưởng hình học chính là giới thiệu một biến ξ_i tương ứng với từng cặp quan sát và nhãn $(x^{(i)}, y_i)$ cho phép một quan sát cụ thể nằm trong lề hoặc thậm chí ở sai phía của siêu phẳng. Biến ξ_i đo khoảng cách của một quan sát "positive" x_+ đến siêu phẳng biên $\mathbf{w^T x} + b = 1$ khi x_+ ở phía sai. Hàm mục tiêu mới

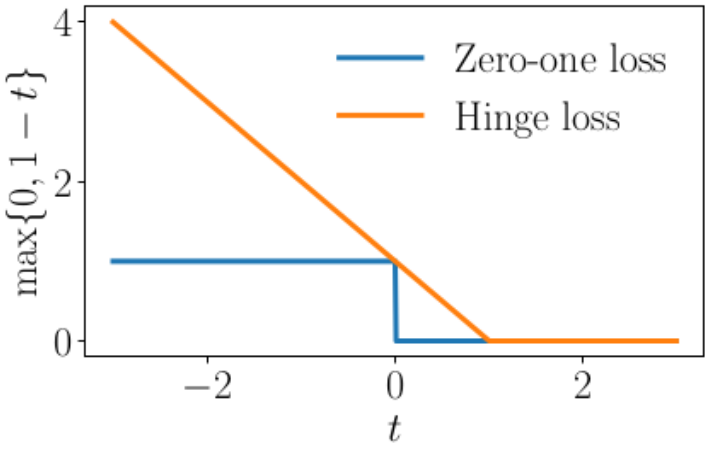
$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

s.t

$$y_i(\mathbf{w^T x^{(i)}} + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad \text{với mọi } i = 1, 2, ..., n$$

Hàm Hinge Loss



Hình 8: Hinge loss là hàm lồi trên hàm zero-one loss

Trong các SVM, chúng ta sử dụng hàm **hinge loss** để biểu thị cho sai số ξ

$$l(t) = \max\{0, 1 - t\}$$

trong đó $t = yf(\mathbf{x}) = y(\mathbf{w^T x} + b)$. Tuy nhiên, chúng ta có thể thay thế tương đương việc cực tiểu hóa hàm hinge loss trên t bằng việc cực tiểu biến ξ với hai ràng buộc. Cụ thể

$$\min_t max\{0, 1 - t\}$$

tương đương với bài toán

$$\min_{\xi,t} \xi$$

s.t

$$\xi \geq 0$$

$$\xi \geq 1 - t$$

Do đó, hàm mục tiêu có thể được viết

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b)\}$$

hay

$$\min_{\mathbf{w},b} \underbrace{\frac{1}{2}\lambda\|\mathbf{w}\|^2}_{\text{regularizer}} + \underbrace{\sum_{i=1}^{n} \max\{0, 1 - y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b)\}}_{\text{error term}}$$

với $\lambda = \frac{1}{C}$.

4.3.2 SVMs với Gradient Descent

Việc sử dụng hàm hinge loss mang lại một bài toán tối ưu hóa không ràng buộc, vì vậy chúng ta có thể sử dụng Gradient Descent để tìm giá trị tối ưu.

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} - \alpha\nabla J(\mathbf{w}^{(j)}) \quad \text{(Công thức Gradient Descent)}$$

trong đó

- $\mathbf{w}^{(j)}$ là trọng số tại lần lặp thứ j
- α là hệ số học
- $\nabla J(\mathbf{w}^{(j)})$ là gradient của J tại $\mathbf{w}^{(j)}$

và

$$J(\mathbf{w}) = \frac{1}{2}\lambda\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max\{0, 1 - y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b)\}$$

Trường hợp 1: $y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b) < 1$

$$\Rightarrow 1 - y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b) > 0 \Rightarrow J(\mathbf{w}) = \frac{1}{2}\lambda\|\mathbf{w}\|^2 + \sum_i \big(1 - y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b)\big)$$

thì

- $\nabla\frac{\partial J}{\partial \mathbf{w}_i} = \lambda\mathbf{w}_i - y_i\mathbf{x}^\mathbf{(i)}$
- $\nabla\frac{\partial J}{\partial b} = y_i$

Trường hợp 2: $y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b) \geq 1$

$$\Rightarrow 1 - y_i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{(i)} + b) \leq 0 \Rightarrow J(\mathbf{w}) = \frac{1}{2}\lambda\|\mathbf{w}\|^2$$

thì

- $\nabla\frac{\partial J}{\partial \mathbf{w}_i} = \lambda\mathbf{w}_i$
- $\nabla\frac{\partial J}{\partial b} = 0$

trong điều kiện

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^\mathbf{(1)} \\ \mathbf{x}^\mathbf{(2)} \\ \vdots \\ \mathbf{x}^\mathbf{(n)} \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \text{ và } \mathbf{x}^\mathbf{(i)} \text{ là một vectơ với số chiều } (1, d)$$

4.3.3 SVMs với MapReduce

Nhận thấy ở bước cập nhật Gradient Descent, các thành phần w_i là hoàn toàn độc lập nên có thể chia các phần tử của trọng số $\mathbf{w}$ thành k tập con S_m sao cho $\cup_{m=1}^{k} S_m = \{w_1, w_2, ..., w_d\}$ để áp dụng kỹ thuật MapReduce

- Map: Cặp khóa và giá trị là (m, w_i)
- Reduce: Cập nhật qua hết các vòng lặp thì tại bước Reduce, các trọng số w_i sẽ được nhóm theo khóa m đồng thời được sắp xếp theo chỉ số i.

$$(m, w_i) \text{ và } (m, w_{i+1})) \Rightarrow (m, \{w_i, w_{i+1}\})$$

Cuối cùng, ta chỉ sắp xếp lại một lần nữa trọng số trong tất cả S_m để nhận đươc vectơ $\mathbf{w}$ theo đúng thứ tự.

5 Thực nghiệm

5.1 Tiền xử lý dữ liệu

5.1.1 Giảm kích thước dữ liệu

Như đã nói ở phần dữ liệu, bộ dữ liệu nhóm dùng trong báo cáo này có kích thước lên tới 50 GB - quá lớn để các phần mềm và thư viện hiện tại có thể xử lý nhanh trong quá trình huấn luyện mô hình. Một trong những nguyên nhân dẫn việc lưu trữ lớn đến thế là tập tin đang được lưu dưới dạng csv, các kiểu dữ liệu trong file sẽ được mặc định là float64 (dùng 64 bit để biểu diễn diễn số thực), và int32 (dùng 32 bit để biểu diễn các trường số nguyên), trong khi không phải đặc trưng nào của dữ liệu cũng cần nhiều bit đến thế, dẫn đến việc quá lãng phí trong việc lưu trữ, từ đó quá trình đọc và xử lý tập tin cũng bị chậm rất nhiều. Và cách xử lý của nhóm là

- Với các trường liên tục thay đổi từ kiểu float64 về float32
- Với các trường rời rạc (các trường này chỉ mức độ, nhưng kiểu định dạng là int32) sẽ được chuyển về kiểu int16 hoặc int8
- Thay đổi định dạng tập tin, thay vì chọn định dạng csv - một định dạng chỉ phù hợp với dữ liệu nhỏ và vừa thì chọn định dạng parquet - định dạng phù hợp cho dữ liệu lớn và cũng được hỗ trợ bởi rất nhiều thư viện.

Định dạng csv: Định dạng tập tin CSV (Comma Separated Values) là một định dạng tập tin văn bản thuần túy để lưu trữ dữ liệu bảng (số và văn bản) dưới dạng văn bản thuần. Đây là một định dạng tập tin rất đơn giản được hỗ trợ bởi nhiều ứng dụng khác nhau, bao gồm các chương trình bảng tính như Microsoft Excel và Google Sheets và các công cụ phân tích dữ liệu như thư viện Pandas của Python.

Định dạng parquet: Parquet là một định dạng lưu trữ cột để lưu trữ dữ liệu bảng. Nó được thiết kế để hiệu quả và linh hoạt, và được sử dụng rộng rãi trong hệ sinh thái dữ liệu lớn. Một trong những lợi ích chính của việc sử dụng Parquet là nó được tối ưu hóa để đọc và ghi các tập dữ liệu lớn, và nó có thể được nén để chiếm ít không gian hơn trên đĩa.

Có một số khác biệt chính giữa CSV và Parquet

- Cấu trúc: CSV là một tập tin văn bản thuần với các hàng dữ liệu, trong đó mỗi hàng là một bản ghi và các cột được phân tách bởi một ký tự phân cách (thường là dấu phẩy). Parquet, khác, là một định dạng tập tin nhị phân để lưu trữ dữ liệu theo cột.

- Hiệu suất: Parquet thường hiệu quả hơn CSV để lưu trữ và đọc các tập dữ liệu lớn. Nó được tối ưu hóa để lưu trữ theo cột và có thể được nén để chiếm ít không gian hơn trên đĩa. CSV, khác, là một định dạng văn bản thuần và không hỗ trợ nén.

- Tương thích: CSV là một định dạng tập tin rất đơn giản và được hỗ trợ rộng rãi, và nó có thể được mở và chỉnh sửa bằng nhiều ứng dụng khác nhau, bao gồm các chương trình bảng tính và trình soạn thảo văn bản. Parquet là một định dạng tập tin chuyên biệt hơn và có thể không được hỗ trợ bởi tất cả các ứng dụng.

Nhìn chung, CSV là một lựa chọn tốt cho tập dữ liệu cỡ nhỏ đến trung bình và cho các tình huống mà khả năng tương thích với nhiều loại ứng dụng là quan trọng. Parquet là lựa chọn tốt hơn cho các bộ dữ liệu lớn và khi hiệu quả và hiệu suất là rất quan trọng. Và dưới đây là chi tiết về kích thước trước và sau khi tiền xử lý với 2 tập tin là train_data.csv và test_data.csv theo hướng đã bàn ở bên trên. Dễ thấy dữ liệu sau khi được tiền xử lý đã giảm xuống còn 1/10 so với không gian cần lưu trữ ban đầu.

Tên file	Trước khi xử lý	Sau khi xử lý
train_data	16.39 GB	1.53 GB
test_data	33.82 GB	3.07 GB

Lưu ý rằng tập tin train_labels.csv không được xử lý vì nó chỉ chiếm một phần nhỏ dung lượng bộ nhớ và chỉ có 2 trường dữ liệu là customer_ID và target.

5.1.2 Xử lý dữ liệu bị thiếu

Sau khi dữ liệu đã được giảm kích thước, việc tiếp theo của nhóm sẽ là xử lý đến các trường dữ liệu bị thiếu. Dữ liệu đã cho có tổng cộng 190 cột, trong đó 1 cột về id khách hàng (customer_ID), 1 cột chỉ ngày giao dịch (S_2), và trong số các cột còn lại thì có tới 61 cột là có giá trị null (dữ liệu bị thiếu, không xác định). Trên tổng số 61 cột này, có những cột giá trị null chiếm đến hơn nửa, thậm chí là gần 90%, vì thế nhóm đưa ra cách tiếp cận

- Với các cột có giá trị null chiếm dưới 60%, các giá trị null này sẽ được thay bằng giá trị trung bình nếu biến đó là biến liên tục, được thay bằng giá trị mode nếu biến đó là biến rời rạc.

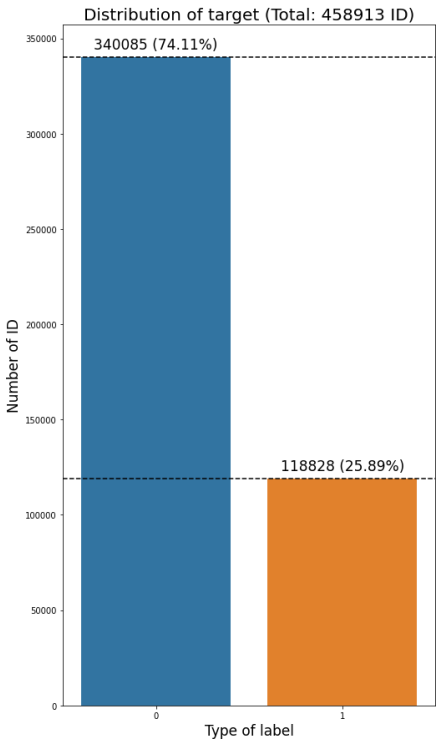
- Với các cột có giá trị null chiếm tới hơn 60%, các cột này sẽ bị loại bỏ, vì khả năng giữ lại cũng sẽ không mang đến được nhiều giá trị trong quá trình xây dựng mô hình.

	Quantity	Percentage	Type
D_88	5525447	99.89	float32
D_110	5500117	99.43	float32
B_39	5497819	99.39	float32
D_73	5475595	98.99	float32
B_42	5459973	98.71	float32
D_134	5336752	96.48	float32
B_29	5150035	93.10	float32
D_132	4988874	90.19	float32
D_76	4908954	88.75	float32
D_42	4740137	85.69	float32
D_142	4587043	82.93	float32
D_53	4084585	73.84	float32

Hình 9: Chi tiết các trường dữ liệu có giá trị null chiếm hơn 60%

5.2 Khai phá dữ liệu

Dữ liệu được nhóm chọn để huấn luyện và đánh giá mô hình là dữ liệu chứa trong tập tin train_data.parquet và train_label.csv. Theo thống kê, dữ liệu có tổng cộng 190 trường dữ liệu đặc trưng và 1 trường dữ liệu biến mục tiêu, tổng cộng có 191 trường. Trong đó, với biến mục tiêu chỉ nhận 2 giá trị là 0 và 1, 0 đại diện cho điểm tín dụng cao, 1 đại diện cho điểm tín dụng thấp, tức là không có khả năng hoàn thành thanh toán đúng kỳ hạn nêu được cho vay. Nhãn 0 chiếm hầu hết trong dữ liệu, gấp 3 lần nhãn 1 (khoảng 75% so với gần 25%).



Hình 10: Phân bố của nhãn phân lớp

Còn lại 190 trường dữ liệu đã số đã bị mã hóa để ẩn đi thông tin khách hàng và thông tin giao dịch gồm: 11 trường rời rạc (categorical), 1 trường ID khách hàng, 1 trường về ngày giao dịch và 177 trường liên tục (numeric). Trên 190 trường này, có 61 trường chứa giá trị null, 12/61 trường có giá trị null chiếm trên 60% sẽ bị loại bỏ.

Tổng số lượng giao dịch là 5531451, của 458913 khách hàng được lưu trữ theo ngày. Tuy nhiên, thông tin về ngày nhóm không tận dụng nên sẽ loại bỏ trường này, nhóm các giao dịch theo ID khách hàng với tiêu chí

- Lấy tổng trên các trường liên tục
- Lấy max trên các trường rời rạc

Sau đó, dữ liệu trên được chuẩn hóa rồi lưu lại vào một tập tin định dạng parquet, phục vụ cho quá trình huấn luyện trên pyspark ở phần sau.

5.3 Xây dựng mô hình huấn luyện

5.3.1 Một số metric để đánh giá mô hình

Trước khi đi vào tìm hiểu các metric để đánh giá một mô hình, nhóm xin giới thiệu về một khái niệm là **ma trận nhầm lẫn (confusion matrix)**. Ma trận nhầm lẫn là một bảng thường được dùng để mô tả hiệu suất của một thuật toán phân loại. Nó cho phép bạn xem thuật toán của bạn đang làm việc tốt hay không trong việc dự đoán đúng các lớp khác nhau. Ma trận này là một bảng có hai hàng và hai cột chứa bốn mục sau:

- True Positives (TP): Đây là các trường hợp mà mô hình dự đoán lớp poitive và lớp thực tế cũng là positive.
- True Negatives (TN): Đây là các trường hợp mà mô hình dự đoán lớp negative và lớp thực tế cũng là negative.
- False Positives (FP): Đây là các trường hợp mà mô hình dự đoán lớp positive nhưng lớp thực tế là negative.
- False Negatives (FN): Đây là các trường hợp mà mô hình dự đoán lớp negative nhưng lớp thực tế là poitive.

	Actual Positive	**Actual Negative**
Predicted Positive	True Positives (TP)	False Positives (FP)
Predicted Negative	False Negatives (FN)	True Negatives (TN)

Từ ma trận nhầm lẫn trên, người ta có thể đưa ra một số metric để đánh giá một mô hình phân loại thế nào là "đủ" tốt. Các metric có thể kể đến là

- **Accuracy**: Đó là tỷ lệ số lượng dự đoán đúng đến tổng số lượng dự đoán, được tính theo công thức
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision**: Đo lường khả năng của mô hình tránh sai lệch khi gán nhãn cho một mục có thể là "positive". Nó được tính bằng cách lấy tổng số lượng dự đoán đúng trên tổng số lượng các dự đoán đã được gán nhãn là "positive", được tính theo công thức
$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: Đo lường khả năng của mô hình tìm ra tất cả các mục poitive trong dữ liệu. Nó được tính bằng cách lấy tổng số true positives chia cho tổng số lượng mục poitive thực tế trong dữ liệu, được tính theo công thức

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1**: Là sử kết hợp hài hòa của Precision và Recall, thường được dùng khi dữ liệu bị mất cân bằng

$$\text{F1 Score} = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

5.3.2 Song song hóa với pyspark

Apache Spark là một nền tảng xử lý dữ liệu phân tán mã nguồn mở được sử dụng rộng rãi cho việc xử lý dữ liệu lớn, học máy và phân tích dữ liệu thời gian thực. Nó được thiết kế để nhanh và dễ sử dụng, và cho phép lập trình viên viết ứng dụng bằng nhiều ngôn ngữ lập trình khác nhau, bao gồm Python, Java và R. pyspark là giao diện lập trình Python cho Spark. Nó cho phép lập trình viên tương tác với Spark bằng ngôn ngữ lập trình Python và sử dụng cơ hội của hệ thống xử lý phân tán của Spark.

RDD (Resilient Distributed Datasets) là cấu trúc dữ liệu cơ bản của Spark và là nguồn dữ liệu phân tán chính trong ứng dụng Spark. Chúng là các tập hợp đối tượng không thay đổi (immutable) có thể được xử lý song song và chúng có khả năng phục hồi vì chúng có thể được xây dựng lại nếu một phần lưu trữ nút của RDD bị lỗi hoặc nếu tập dữ liệu cần được tính toán lại vì bất kỳ lý do gì. RDD rất quan trọng trong Spark vì chúng cung cấp nền tảng xử lý dữ liệu song song, chịu lỗi và phân tán cho phần còn lại của hệ sinh thái Spark. Chúng là cơ sở cho các khái niệm trừu tượng cấp cao hơn như DataFrames, Datasets và SparkSQL, đồng thời chúng là cách chính mà dữ liệu được phân phối và xử lý trong ứng dụng Spark. Tóm lại, RDD có một số tính chất sau

- **Immutable**: Sau khi RDD được tạo, nó không thể sửa đổi được.
- **Distributed**: RDD được phân phối trên một cụm máy, cho phép chúng được xử lý song song.
- **Fault-tolerant**: RDD có thể được tính toán lại nếu một phần lưu trữ nút của RDD bị lỗi hoặc nếu tập dữ liệu cần được tính toán lại vì bất kỳ lý do gì.
- **Parallel**: RDD có thể được xử lý song song, điều này làm cho chúng hiệu quả để xử lý dữ liệu quy mô lớn.

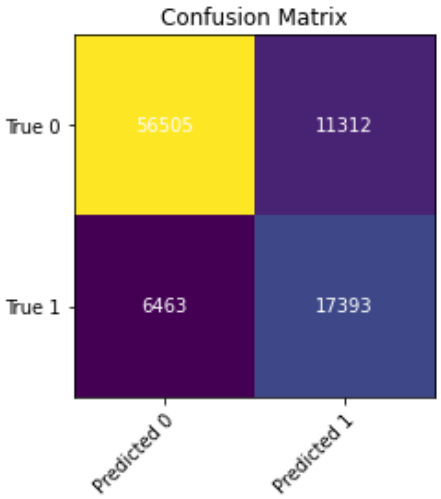
Sử dụng RDD trong pyspark, nhóm chia dữ liệu thành hai phần: một phần để huấn luyện (training set) chiếm 80%, một phần để đánh giá (validation set) chiếm 20%. Dữ liệu sau khi được tách ra sẽ đưa về các dạng chuẩn hóa đầu vào để có thể chạy trên các mô hình học máy được tính toán song song cung cấp bởi module mllib. Tuy nhiên, do phạm vi tìm hiểu và để bài cáo được mạch lạc, nhóm chỉ chọn 3 thuật toán đã trình bày ở phần lý thuyết để huấn luyện mô hình từ module này gồm: Kmean, Decision Tree và Support Vector Machines

5.3.3 Mô hình K-means

Trong PySpark, thuật toán phân cụm KMeans được triển khai như một phần của thư viện MLlib và được áp dụng cho dữ liệu được lưu trữ trong Resilient Distributed Dataset (RDD). Sau đây là một số tham số mà có thể sử dụng khi áp dụng KMeans cho RDD bằng PySpark:

- **k**: Chỉ định số lượng cụm sẽ tạo.
- **maxIterations**: Xác định số lần lặp lại tối đa mà thuật toán sẽ chạy, mặc định là 100.
- **initializationMode**: Chỉ định phương thức khởi tạo các cụm. Các tùy chọn là "random" (chọn k điểm ngẫu nhiên làm tâm ban đầu) hoặc "k-means||" (khởi tạo song song bằng thuật toán k-means||).

ngoài ra còn một số tham số khác. Khi huấn luyên mô hình, nhóm chọn $k = 2$ vì bài toán chỉ có 2 nhãn 0 và 1, $maxIterations = 50$ cùng $initializationMode = "random"$. Và mô hình trả về kết quả



	precision	recall	f1-score	support
0.0	0.90	0.83	0.86	67817
1.0	0.61	0.73	0.66	23856
accuracy			0.81	91673
macro avg	0.75	0.78	0.76	91673
weighted avg	0.82	0.81	0.81	91673

Hình 11: Confusion matrix và một số metric đánh giá

độ chính xác đạt được khá cao với 81%, điểm f1_score cũng như precision và recall của nhãn 1 thì lại tương đối thấp, tương ứng là 66%, 61% và 73%.

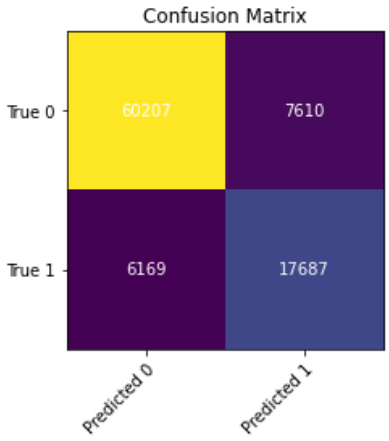
5.3.4 Mô hình Decision Tree

Trong PySpark, thuật toán Decision Tree được triển khai như một phần của thư viện MLlib và được áp dụng cho dữ liệu được lưu trữ trong Resilient Distributed Dataset (RDD). Sau đây là một số tham số mà bạn có thể sử dụng khi huấn luyện Decision Tree bằng PySpark:

- **numClasses**: Số lớp để phân loại.

- **impurity**: Tiêu chí được sử dụng để tính toán mức thông tin thu được khi cây chia nhánh . Các giá trị được hỗ trợ: "gini" hoặc "entropy".
- **maxDepth**: Chỉ định chiều cao tối đa của cây. Việc tăng chiều cao tối đa cho phép cây đưa ra các quyết định phức tạp hơn, nhưng cũng có thể làm tăng nguy cơ quá khớp của mô hình (overfitting).
- **maxBins**: Chỉ định số lượng nhóm tối đa được sử dụng khi rời rạc hóa các đặc trưng liên tục. Việc tăng giá trị này có thể cải thiện độ chính xác của cây, nhưng cũng có thể làm tăng nguy cơ overfitting.
- ...

Nhóm sử dụng các tham số $numClasses = 2, impurity = "gini", maxDepth = 4$ và $maxBins = 32$, mô hình đạt độ chính xác 85% và các metric còn lại đều trên 70% với nhãn 1 như hình bên dưới



	precision	recall	f1-score	support
0.0	0.91	0.89	0.90	67817
1.0	0.70	0.74	0.72	23856
accuracy			0.85	91673
macro avg	0.80	0.81	0.81	91673
weighted avg	0.85	0.85	0.85	91673

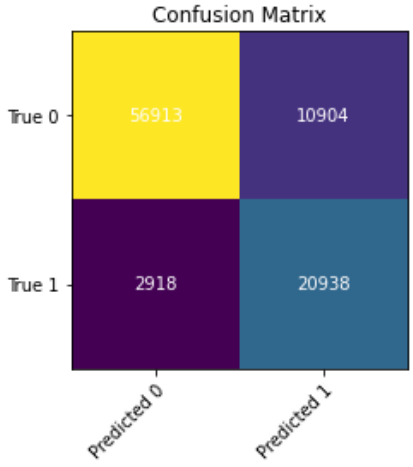
Hình 12: Confusion matrix và một số metric đánh giá

5.3.5 Mô hình SVM

Trong PySpark, thuật toán SVM được triển khai như một phần của thư viện MLlib dưới tên SVMWithSGD sử dụng thuật toán Stochastic Gradient Descent (SGD) để tối ưu hóa các tham số. Sau đây là một số tham số mà bạn có thể sử dụng khi huấn luyện mô hình:

- **iterations**: Số lần lặp lại tối đa mà thuật toán sẽ chạy.
- **step**: hệ số học được sử dụng trong thuật toán SGD
- **regParam**: Xác định tham số hiệu chỉnh, tham số này kiểm soát mức phạt áp dụng cho điểm nằm sai vị trí trong hàm mục tiêu tối ưu hóa.
- ...

Nhóm sử dụng các tham số $iterations = 100, step = 0.1 \text{và} regParam = 0.1$, mô hình đạt độ chính xác 85% và các metric còn lại đều trên 70%, riêng điểm recall của nhãn 1 đạt khoảng 88% như hình bên dưới



	precision	recall	f1-score	support
0.0	0.95	0.84	0.89	67817
1.0	0.66	0.88	0.75	23856
accuracy			0.85	91673
macro avg	0.80	0.86	0.82	91673
weighted avg	0.87	0.85	0.86	91673

Hình 13: Confusion matrix và một số metric đánh giá

6 Tổng kết

Như vậy, qua bài báo cáo này nhóm xin tổng kết một số việc đã làm được

- Tìm hiểu về nội dung lý thuyết cũng như cách song song hóa các thuật toán K-means, Decision Tree và SVM
- Triển khai và huấn luyện mô hình các thuật toán đã được song song trên qua pyspark
- Hiểu thêm về cách giảm kích bộ nhớ lưu trữ với dữ liệu lớn như thay đổi định dạng file, thay đổi kiểu của trường dữ liệu
- Xây dựng được luồng thực hiện tiền xử lý dữ liệu khi bị thiếu
- Khai phá được một số insight liên quan đến cấu trúc dữ liệu cũng như nhãn
- Tiềm hiểu thêm được về cấu trúc cở bản RDD được dùng để song song hóa trong pyspark
- Tìm hiểu thêm được một số metric để đánh giá các mô hình
- Thực nghiệm được các mô hình song song

Tuy nhiên, nhóm vẫn chưa thực hiện được nhiều về khai phá dữ liệu cũng như trực quan hóa, hay giảm chiều dữ liệu. Việc giảm chiều dữ liệu qua định dạng tập tin và thay đổi trường dữ liệu do hạn chế về điều kiện máy tính, nhóm chưa thể trực tiếp thực hiện mà nhận dữ liệu này thông qua một số ứng viên từ cuộc thi trên Kaggle.

7 Tài liệu tham khảo

[1] Aurélien Géron, "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow", O'reilly, 2019.
[2] Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong, "Mathematics for Machine Learning", Cambridge University Press, 2020.
[3] "K-Means Clustering in Python: A Practical Guide" - https://realpython.com/k-means-clustering-python/
[4] "Clustering" - https://scikit-learn.org/stable/module/clustering.html#k-means
[5] "Blog: Machine Learning cơ bản" - https://machinelearningcoban.com
[6] Nagesh Singh Chauhan, "Decision Tree Algorithm, Explained" - https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
[7] "Decision tree learning" - https://en.wikipedia.org/wiki/Decision_tree_learning
[8] "ID3 algorithm" - https://en.wikipedia.org/wiki/ID3_algorithm
[9] "Decision Trees" - https://scikit-learn.org/stable/modules/tree.html
[10] "Support Vector Machines" - https://en.wikipedia.org/wiki/Support_vector_machine
[11] "Support Vector Machines" - https://scikit-learn.org/stable/modules/svm.html
[12] Dataset - https://www.kaggle.com/competitions/amex-default-prediction/data
[13] MapReduce - https://en.wikipedia.org/wiki/MapReduce