**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA HỆ THỐNG THÔNG TIN**



**IMPLEMENTATION OF CLASSIFICATION ALGORITHMS FOR PREDICTING CUSTOMER SUBSCRIPTION TO TERM DEPOSITS ON APACHE SPARK**

ĐỒ ÁN CUỐI KÌ DỮ LIỆU LỚN

Lớp: IS405.O21

Giảng viên hướng dẫn: ThS. Nguyễn Hồ Duy Trí

**Nhóm sinh viên thực hiện:**

|  |  |
| --- | --- |
| Nguyễn Hoàng Việt | 21522791 |
| Nguyễn Hoàng Phúc | 21520400 |
| Trần Lê Tứ | 21522746 |

**TP. HỒ CHÍ MINH, NĂM 2024**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA HỆ THỐNG THÔNG TIN**

**IMPLEMENTATION OF CLASSIFICATION ALGORITHMS FOR PREDICTING CUSTOMER SUBSCRIPTION TO TERM DEPOSITS ON APACHE SPARK**

ĐỒ ÁN CUỐI KÌ DỮ LIỆU LỚN

Lớp: IS405.O21

Giảng viên hướng dẫn: ThS. Nguyễn Hồ Duy Tri

**Nhóm sinh viên thực hiện:**

|  |  |
| --- | --- |
| Nguyễn Hoàng Việt | 21522791 |
| Nguyễn Hoàng Phúc | 21520400 |
| Trần Lê Tứ | 21522746 |

**TP. HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Chúng em xin chân thành cảm ơn trường Đại học Công nghệ Thông tin và Khoa Hệ Thống Thông Tin đã tạo điều kiện cho chúng em hoàn thành tốt đồ án môn học Dữ liệu lớn. Trong quá trình học tập môn này đã giúp cho nhóm chúng em có được rất nhiều kinh nghiệm quý báu. Đặc biệt, nhóm chúng em xin gửi lời biết ơn sâu sắc đến thầy Nguyễn Hồ Duy môn Dữ liệu lớn (IS405.021) đã dành thời gian quý báo và trực tiếp hướng dẫn tận tình, đóng góp ý kiến giúp nhóm hoàn thành tốt báo cáo môn học.

Thông qua quá trình thực hiện đồ án, chúng em đã phần nào củng cố, tích lũy được những kiến thức về môn Dữ liệu lớn. Bên cạnh đó là biết sử dụng thêm một số công cụ để phân tích, rất có ích cho chúng em sau này. Bên cạnh đó, chúng em còn rút ra được nhiều kinh nghiệm trong công việc làm nhóm. Hi vọng mọi thứ sẽ được áp dụng và phát triển trong tương lai.

Trong thời gian thực hiện đề tài, vì thời gian và kiến thức có hạn nên không tránh khỏi nhiều sai sót. Vì vậy, nhóm rất mong nhận được những góp ý bổ sung từ thầy để đề tài ngày một hoàn thiện hơn.

Chúng em xin chân thành cảm ơn.

Nhóm sinh viên thực hiện

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*Thành phố Hồ Chí Minh, ngày……...tháng……năm 2024*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên****)***

# MỤC LỤC

[MỤC LỤC 1](#_Toc168246356)

[DANH MỤC BẢNG 3](#_Toc168246357)

[DANH MỤC HÌNH ẢNH 4](#_Toc168246358)

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 5](#_Toc168246359)

[1.1 Lý do chọn đề tài 5](#_Toc168246360)

[1.2 Mô tả bài toán 5](#_Toc168246361)

[1.3 Mô tả dữ liệu 6](#_Toc168246362)

[1.3.1 Mô tả thành phần dữ liệu 6](#_Toc168246363)

[1.3.2 Phân tích dữ liệu 8](#_Toc168246364)

[1.3.2.1 Giá trị lớn nhất, giá trị nhỏ nhất: 8](#_Toc168246365)

[1.3.2.2 Giá trị phổ biến nhất, hiếm nhất 11](#_Toc168246366)

[1.3.2.3 Giá trị trung bình, trung vị 13](#_Toc168246367)

[1.3.3 Trực quan hóa dữ liệu 16](#_Toc168246368)

[1.3.3.1 Thống kê tỉ lệ khách hàng đăng ký gửi tiền có kỳ hạn theo mức độ học vấn 16](#_Toc168246369)

[1.3.3.2 Thống kê tỉ lệ khách hàng đăng ký gửi tiền có kỳ hạn theo tình trạng hôn nhân 17](#_Toc168246370)

[1.3.3.3 So sánh tỉ lệ đăng ký gửi tiền có kì hạn giữa các nhóm khách hàng có khoản vay nhà và có khoản vay cá nhân 19](#_Toc168246371)

[1.3.3.4 Phân tích tần suất đăng ký gửi tiền có kỳ hạn theo từng tháng trong năm 20](#_Toc168246372)

[1.4 Công cụ sử dụng 21](#_Toc168246373)

[CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU 22](#_Toc168246374)

[2.1 Loại bỏ thuộc tính không cần thiết 22](#_Toc168246375)

[2.2 Nhận diện và xử lý dữ liệu bị nhiễu 22](#_Toc168246376)

[2.3 Nhận diện dữ liệu bị thiếu 25](#_Toc168246377)

[2.4 Xử lý ý nghĩa dữ liệu 26](#_Toc168246378)

[2.4.1 Cột ‘job’ và ‘education’ 26](#_Toc168246379)

[2.4.2 Cột “month” 26](#_Toc168246380)

[2.5 Chuẩn hóa dữ liệu 27](#_Toc168246381)

[2.6 Tách dữ liệu huấn luyện và kiểm thử 29](#_Toc168246382)

[CHƯƠNG 3: ÁP DỤNG CÁC GIẢI THUẬT KHAI THÁC DỮ LIỆU 31](#_Toc168246383)

[3.1 Giải thuật K-Nearest Neighbors 31](#_Toc168246384)

[3.1.1 Khái niệm 31](#_Toc168246385)

[3.1.2 Song song hóa giải thuật với mô hình Map Reduce 32](#_Toc168246386)

[3.1.2.1 Giai đoạn Map 32](#_Toc168246387)

[3.1.2.2 Giai đoạn Reduce 33](#_Toc168246388)

[3.1.3 Áp dụng giải thuật trên Apache Spark 33](#_Toc168246389)

[3.2 Giải thuật Random Forest 40](#_Toc168246390)

[3.2.1. Lý do lựa chọn 40](#_Toc168246391)

[3.2.2. Lập trình với Apache Spark 41](#_Toc168246392)

[3.2.3. Hình vẽ minh họa song song 47](#_Toc168246393)

[3.3 Giải thuật Support Vector Machine 54](#_Toc168246394)

[3.3.1 Khái niệm 54](#_Toc168246395)

[3.3.2 Song song hóa giải thuật dựa trên MapReduce 55](#_Toc168246396)

[CHƯƠNG 4: ĐÁNH GIÁ CÁC THUẬT KHAI THÁC DỮ LIỆU 61](#_Toc168246397)

[4.1 Đánh giá về thời gian chạy thuật toán 61](#_Toc168246398)

[4.2 Đánh giá về độ chính xác tổng quát của thuật toán (Accuracy) 61](#_Toc168246399)

[4.3 Đánh giá về độ chính xác của thuật toán (Precision) 62](#_Toc168246400)

[4.3.1 So sánh độ chính xác của thuật toán trên dữ liệu 0 63](#_Toc168246401)

[4.3.2 So sánh độ chính xác của thuật toán trên dữ liệu 1 64](#_Toc168246402)

[4.4 Đánh giá về độ nhạy của thuật toán (Recall) 64](#_Toc168246403)

[4.4.1 So sánh độ nhạy của thuật toán trên dữ liệu 0 65](#_Toc168246404)

[4.4.2 So sánh độ nhạy của thuật toán trên dữ liệu 1 66](#_Toc168246405)

[4.5 Nhận xét các thuật toán 66](#_Toc168246406)

[4.5.1 K-Nearest Neighbor 66](#_Toc168246407)

[4.5.3 Support Vector Machine 67](#_Toc168246408)

[CHƯƠNG 5: TỔNG KẾT 69](#_Toc168246409)

[5.1 Kết quả đạt được 69](#_Toc168246410)

[5.2 Hướng phát triển 70](#_Toc168246411)

[BẢNG PHÂN CHIA CÔNG VIỆC 71](#_Toc168246412)

[DANH MỤC TÀI LIỆU THAM KHẢO 72](#_Toc168246413)

# DANH MỤC BẢNG

# DANH MỤC HÌNH ẢNH

[Hình 1. 1 Dataset 6](#_Toc168181473)

[Hình 1. 2 Phân tích dữ liệu Min Max 8](#_Toc168181474)

[Hình 1. 3 Phân tích dữ liệu Min Max – Kết quả 8](#_Toc168181475)

[Hình 1. 4 Phân tích dữ liệu Most Least 10](#_Toc168181476)

[Hình 1. 5 Phân tích dữ liệu Most Least – Kết quả 10](#_Toc168181477)

[Hình 1. 6 Phân tích dữ liệu – Mean Median 13](#_Toc168181478)

[Hình 1. 7 Phân tích dữ liệu – Mean Median Kết quả 13](#_Toc168181479)

[Hình 1. 8 Trực quan hóa 1 15](#_Toc168181480)

[Hình 1. 9 Trực quan hóa 1 – Kết quả 16](#_Toc168181481)

[Hình 1. 10 Trực quan hóa 2 17](#_Toc168181482)

[Hình 1. 11 Trực quan hóa 2 – Kết quả 17](#_Toc168181483)

[Hình 1. 12 Trực quan hóa 3 18](#_Toc168181484)

[Hình 1. 13 Trực quan hóa 3 – Kết quả 19](#_Toc168181485)

[Hình 1. 14 Trực quan hóa 4 19](#_Toc168181486)

[Hình 1. 15 Trực quan hóa 4 – Kết quả 20](#_Toc168181487)

[Hình 2. 1 Tiền xử lý: Loại bỏ ‘ID’ 22](#_Toc168181506)

[Hình 2. 2 Code vẽ biểu đồ BoxPlot xác định Outlier 22](#_Toc168181507)

[Hình 2. 3 Kết quả vẽ biểu đồ BoxPlot xác định Outlier 23](#_Toc168181508)

[Hình 2. 4 Tiền xử lý: Thống kê Outlier bằng phương pháp Box plot 24](#_Toc168181509)

[Hình 2. 5 Tiền xử lý: Hàm QuantileOutlierClipper 25](#_Toc168181510)

[Hình 2. 6 Tiền xử lý: Áp dụng hàm QuantileOutlierClipper 25](#_Toc168181511)

[Hình 2. 7 Tiền xử lý: Nhận diện giá trị bị thiếu 26](#_Toc168181512)

[Hình 2. 8 Tiền xử lý ‘job’ và ‘education’ 26](#_Toc168181513)

[Hình 2. 9 Tiền xử lý ‘month’ 27](#_Toc168181514)

[Hình 2. 10 Tiền xử lý: Xử lý ý nghĩa dữ liệu – Cột month 27](#_Toc168181515)

[Hình 2. 11 Tiền xử lý: Thống kê thuộc tính 28](#_Toc168181516)

[Hình 2. 12 Tiền xử lý: Code Chuẩn hóa dữ liệu 28](#_Toc168181517)

[Hình 2. 13 Tiền xử lý: Kết quả chuẩn hóa dữ liệu 29](#_Toc168181518)

[Hình 2. 14 Tiền xử lý: Dữ liệu mới sau xử lý 29](#_Toc168181519)

[Hình 2. 15 Tiền xử lý: Tách dữ liệu thành X, y 30](#_Toc168181520)

[Hình 2. 16 Tiền xử lý: Kích thước Train Test 30](#_Toc168181521)

[Hình 3. 1 Song song hóa giải thuật KNN 31](#_Toc168181489)

[Hình 3. 2 KNN - Bước 1 33](#_Toc168181490)

[Hình 3. 3 KNN - Bước 2 33](#_Toc168181491)

[Hình 3. 4 KNN - Bước 3 33](#_Toc168181492)

[Hình 3. 5 KNN - Bước 4 34](#_Toc168181493)

[Hình 3. 6 KNN - Bước 4 Kết quả 34](#_Toc168181494)

[Hình 3. 7 KNN - Bước 5 34](#_Toc168181495)

[Hình 3. 8 KNN – Bước 6 35](#_Toc168181496)

[Hình 3. 9 KNN – Bước 7 35](#_Toc168181497)

[Hình 3. 10 KNN – Bước 8 36](#_Toc168181498)

[Hình 3. 11 KNN – Bước 9 36](#_Toc168181499)

[Hình 3. 12 KNN – Bước 10 36](#_Toc168181500)

[Hình 3. 13 KNN – Bước 11 37](#_Toc168181501)

[Hình 3. 14 KNN – Bước 12 37](#_Toc168181502)

[Hình 3. 15 KNN – Bước 13 37](#_Toc168181503)

[Hình 3. 16 KNN – Bước 14 38](#_Toc168181504)

[Hình 3. 17 KNN – Bước 15 38](#_Toc168181505)

# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài

* Trong thời đại kinh tế khó khăn hiện nay, việc gửi tiền có kỳ hạn trong ngân hàng có tầm ảnh hưởng rất lớn.
* Đối với khách hàng, giúp tăng giá trị tiền gửi. Đây là hình thức lưu trữ tiền an toàn và ổn định, được bảo vệ bởi ngân hàng. Khách hàng có thể quản lý tài chính hiệu quả hơn, tích lũy tiền cho các mục tiêu dài hạn và lựa chọn đa dạng về kỳ hạn để phù hợp với nhu cầu cá nhân. Ngoài ra, gửi tiền có kỳ hạn tạo nguồn dự trù tài chính cho tương lai và hỗ trợ kiểm soát chi tiêu.
* Đối với ngân hàng, các khoản tiền gửi có kỳ hạn là nguồn vốn quan trọng để duy trì hoạt động kinh doanh, đầu tư và cung cấp dịch vụ tài chính. Việc tăng cường lượng tiền gửi có kỳ hạn từ khách hàng giúp ngân hàng có thể quản lý tài sản hiệu quả hơn, cung cấp các sản phẩm và dịch vụ tốt hơn cho khách hàng.
* Trong bối cảnh cạnh tranh gay gắt trong ngành ngân hàng tại Việt Nam, các tổ chức tài chính cần có chiến lược tiếp thị hiệu quả để thu hút và giữ chân khách hàng. Việc dự đoán khả năng khách hàng đăng ký gửi tiền có kỳ hạn sẽ giúp ngân hàng tối ưu hóa chiến lược tiếp thị, tập trung vào những khách hàng có khả năng cao nhất để tăng doanh số.
* Nhận thấy được điều đó, nhóm đã quyết định chọn đề tài phân tích dữ liệu về các chiến dịch tiếp thị trực tiếp của một tổ chức ngân hàng Việt Nam trên các nền tảng dữ liệu lớn như Apache Spark.

## Mô tả bài toán

* Bài toán của đề tài là dự đoán khả năng khách hàng đăng ký gửi tiền có kỳ hạn từ các chiến dịch tiếp thị của một tổ chức ngân hàng Việt Nam. Cụ thể, mục tiêu là áp dụng các thuật toán phân lớp sử dụng dữ liệu khách hàng, thông tin từ các chiến dịch tiếp thị và các yếu tố liên quan khác để dự đoán khả năng khách hàng sẽ đăng ký gửi tiền có kỳ hạn. Nhằm tối ưu hóa chiến lược tiếp thị, tập trung vào những khách hàng có khả năng cao nhất để tăng doanh số.

Các bước thực hiện trong báo cáo có thể bao gồm:

* **Thu thập dữ liệu**: Sử dụng bộ dữ liệu về chiến dịch tiếp thị trực tiếp của một tổ chức ngân hàng Việt Nam từ Kaggle.
* **Xử lý tiền dữ liệu**:Phân tích dữ liệu để tìm hiểu các yếu tố ảnh hưởng đến quyết định gửi tiền có kỳ hạn của khách hàng. Dọn dẹp và tiền xử lý dữ liệu để loại bỏ thông tin không cần thiết hoặc không chính xác.
* **Áp dụng các giải thuật toán khai thác dữ liệu**: Sử dụng các giải thuật khai thác dữ liệu để dự đoán dựa trên bộ dữ liệu tìm được.
* **Đánh giá mô hình**: Kiểm tra và đánh giá hiệu suất của các giải thuật khai thác dữ liệu bằng cách sử dụng các chỉ số đo lường như độ chính xác, độ nhạy, độ đặc hiệu, v.v.
* Báo cáo sẽ trình bày chi tiết về quy trình trên, kết quả phân tích, và các khuyến nghị dựa trên mô hình dự đoán để giúp ngân hàng tối ưu hóa chiến lược tiếp thị và thu hút khách hàng gửi tiền có kỳ hạn.

## Mô tả dữ liệu

### Mô tả thành phần dữ liệu

- Tên dataset: Bank Customer Data in VietNam

- Nguồn dữ liệu: <https://www.kaggle.com/datasets/tomculihiddleston/bank-customer-data-in-vietnam/data>

- Mô tả dữ liệu: Dữ liệu có liên quan đến các chiến dịch tiếp thị trực tiếp (cuộc gọi điện thoại) của một tổ chức ngân hàng Việt Nam. Mục tiêu là dự đoán xem khách hàng có đăng ký gửi tiền có kỳ hạn hay không.

A screenshot of a computer

Description automatically generated

Hình 1. 1 Dataset

Dữ liệu gồm 42600 dòng và 16 cột dữ liệu, trong đó:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thuộc tính | Kiểu dữ liệu | Ý nghĩa |
| 1 | ID | int | Số thứ tự của khách hàng đã tư vấn của chiến dịch |
| 2 | age | int | Độ tuổi của khách hàng |
| 3 | job | string | Mô tả công việc của khách hàng |
| 4 | marital | int | Mô tả tình trạng hôn nhân của khách hàng. 0(ly hôn hoặc không xác định), 1(độc thân),2(đã kết hôn cũng như có gia đình) |
| 5 | education | string | Mô tả trình độ học vấn. “primary” (học hết tiểu học), “secondary” (học hết cấp hai),”teriary” (từ cấp ba trở lên: cấp ba, đại học, cao học), “unknown”(không xác định) |
| 6 | default | binary | Mô tả khách hàng có sài thẻ tín dụng mặc định hay không. 0(không), 1(có) |
| 7 | housing | binary | Mô tả khách hàng có khoản vay nhà hay không.0(không), 1(có) |
| 8 | loan | binary | Mô tả khách hàng có khoản vay cá nhân hay không.0(không), 1(có) |
| 9 | day | int | Ngày liên hệ cuối cùng với khách hàng ở chiến dịch lần này |
| 10 | month | string | Tháng của ngày liên hệ cuối cùng với khách hàng ở chiến dịch lần này |
| 11 | duration | int | Tổng số thời gian liên lạc với khách hàng ở chiến dịch lần này(tính bằng giây) |
| 12 | campaign | int | Số lần liên hệ với khách hàng ở chiến dịch này |
| 13 | pdays | int | Số ngày trôi qua kể từ lần liên hệ cuối cùng với khách hàng này ở chiến dịch lần trước(-1 tức là khách hàng này chưa được liên hệ trước đó) |
| 14 | previous | int | Số lần liên hệ được thực hiện trước với khách hàng này ở chiến dịch này |
| 15 | term\_deposit | binary | Khách hàng có đăng kí gửi tiền kì hạn không?0(không), 1(có) |

### Phân tích dữ liệu

* + - 1. Giá trị lớn nhất, giá trị nhỏ nhất:

A screenshot of a computer

Description automatically generated

Hình 1. 2 Phân tích dữ liệu Min Max

Kết quả thu được

A screenshot of a computer

Description automatically generated

Hình 1. 3 Phân tích dữ liệu Min Max – Kết quả

Từ kết quả trên, ta có thể rút ra được kết luận:

* age (tuổi)
* Giá trị lớn nhất (max\_age): 95
* Giá trị nhỏ nhất (min\_age): 18
* Kết luận: Độ tuổi của các khách hàng trong tập dữ liệu dao động từ 18 đến 95 tuổi, cho thấy tập dữ liệu bao gồm cả người trẻ tuổi và người cao tuổi.
* balance (số dư tài khoản)
* Giá trị lớn nhất (max\_balance): 102127
* Giá trị nhỏ nhất (min\_balance): -8019
* Kết luận: Số dư tài khoản dao động từ -8019 đến 102127. Điều này cho thấy có khách hàng bị nợ (số dư âm) và có những khách hàng có số dư rất cao.
* duration (thời lượng cuộc gọi)
* Giá trị lớn nhất (max\_duration): 4918
* Giá trị nhỏ nhất (min\_duration): 0
* Kết luận: Thời lượng cuộc gọi dao động từ 0 đến 4918 giây (tương đương hơn 81 phút). Thời lượng bằng 0 có thể cho thấy các cuộc gọi không thành công hoặc bị từ chối ngay lập tức.
* campaign (số lần liên hệ trong chiến dịch)
* Giá trị lớn nhất (max\_campaign): 63
* Giá trị nhỏ nhất (min\_campaign): 1
* Kết luận: Số lần liên hệ trong chiến dịch dao động từ 1 đến 63 lần. Điều này cho thấy có khách hàng đã được liên hệ rất nhiều lần trong một chiến dịch.
* pdays (số ngày kể từ khi khách hàng được liên hệ lần cuối trong chiến dịch trước)
* Giá trị lớn nhất (max\_pdays): 536
* Giá trị nhỏ nhất (min\_pdays): -1
* Kết luận: Giá trị -1 có thể chỉ ra rằng khách hàng chưa từng được liên hệ trước đó. Số ngày kể từ lần liên hệ trước dao động từ 0 đến 536 ngày đối với những khách hàng đã được liên hệ trước đó.
* previous (số lần liên hệ với khách hàng trước chiến dịch hiện tại)
* Giá trị lớn nhất (max\_previous): 275
* Giá trị nhỏ nhất (min\_previous): 0
* Kết luận: Số lần liên hệ trước chiến dịch hiện tại dao động từ 0 đến 275 lần, với nhiều khách hàng chưa từng được liên hệ trước đó.
  + - 1. Giá trị phổ biến nhất, hiếm nhất

A close-up of a computer code

Description automatically generated

Hình 1. 4 Phân tích dữ liệu Most Least

Kết quả thu được:

A screenshot of a computer

Description automatically generated

Hình 1. 5 Phân tích dữ liệu Most Least – Kết quả

Từ kết quả trên, ta có thể rút ra được kết luận:

* age (tuổi)
* Giá trị phổ biến nhất (most\_common\_age): 32 (1999 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_age): 94 (1 lần xuất hiện)
* Nhận xét: Độ tuổi 32 là độ tuổi phổ biến nhất trong tập dữ liệu, trong khi độ tuổi 94 chỉ xuất hiện duy nhất một lần.
* job (nghề nghiệp)
* Giá trị phổ biến nhất (most\_common\_job): blue-collar (9536 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_job): unknown (264 lần xuất hiện)
* Nhận xét: Nghề nghiệp blue-collar là phổ biến nhất, còn nghề nghiệp unknown là hiếm nhất trong dữ liệu, điều này có thể chỉ ra rằng phần lớn khách hàng làm các công việc liên quan đến lao động chân tay.
* marital (tình trạng hôn nhân)
* Giá trị phổ biến nhất (most\_common\_marital): 2 (25868 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_marital): 0 (4965 lần xuất hiện)
* Nhận xét: Tình trạng hôn nhân 2 là phổ biến nhất, trong khi tình trạng hôn nhân 0 ít phổ biến hơn.
* education (trình độ học vấn)
* Giá trị phổ biến nhất (most\_common\_education): secondary (22066 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_education): unknown (1690 lần xuất hiện)
* Nhận xét: Trình độ học vấn secondary là phổ biến nhất trong tập dữ liệu, trong khi unknown là ít phổ biến nhất.
* default (Có thẻ tín dụng)
* Giá trị phổ biến nhất (most\_common\_default): 0 (41828 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_default): 1 (811 lần xuất hiện)
* Nhận xét: Phần lớn khách hàng không có nợ xấu (default = 0), trong khi số lượng khách hàng có nợ xấu (default = 1) là rất ít.
* housing (vay mua nhà)
* Giá trị phổ biến nhất (most\_common\_housing): 1 (24590 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_housing): 0 (18049 lần xuất hiện)
* Nhận xét: Số lượng khách hàng có vay mua nhà (housing = 1) nhiều hơn so với những khách hàng không vay mua nhà (housing = 0).
* loan (vay cá nhân)
* Giá trị phổ biến nhất (most\_common\_loan): 0 (35554 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_loan): 1 (7085 lần xuất hiện)
* Nhận xét: Phần lớn khách hàng không có vay cá nhân (loan = 0), trong khi số lượng khách hàng có vay cá nhân (loan = 1) ít hơn.
* day (ngày liên hệ)
* Giá trị phổ biến nhất (most\_common\_day): 20 (2703 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_day): 1 (235 lần xuất hiện)
* Nhận xét: Ngày 20 là ngày phổ biến nhất để liên hệ khách hàng, trong khi ngày 1 là ít phổ biến nhất.
* month (tháng liên hệ)
* Giá trị phổ biến nhất (most\_common\_month): may (13532 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_month): dec (214 lần xuất hiện)
* Nhận xét: Tháng 5 là tháng phổ biến nhất để liên hệ khách hàng, trong khi tháng 12 là ít phổ biến nhất.
* term\_deposit (quyết định gửi tiết kiệm)
* Giá trị phổ biến nhất (most\_common\_term\_deposit): 0 (38678 lần xuất hiện)
* Giá trị hiếm nhất (least\_common\_term\_deposit): 1 (3961 lần xuất hiện)
* Nhận xét: Phần lớn khách hàng không chọn gửi tiết kiệm (term\_deposit = 0), trong khi số lượng khách hàng chọn gửi tiết kiệm (term\_deposit = 1) là ít hơn nhiều.
  + - 1. Giá trị trung bình, trung vị

A screenshot of a computer code

Description automatically generated

Hình 1. 6 Phân tích dữ liệu – Mean Median

Kết quả thu được:

A screenshot of a computer

Description automatically generated

Hình 1. 7 Phân tích dữ liệu – Mean Median Kết quả

Từ kết quả trên, ta có thể rút ra được kết luận:

* Age (Tuổi)
* Trung bình (avg\_age): 40.79
* Trung vị (median\_age): 39.0
* Nhận xét: Độ tuổi trung bình của khách hàng là khoảng 40.79, trong khi tuổi trung vị là 39. Điều này cho thấy độ tuổi của khách hàng tập trung xung quanh mức 40, với một sự phân bố khá đồng đều.
* Balance (Số dư)
* Trung bình (avg\_balance): 1331.86
* Trung vị (median\_balance): 417.0
* Nhận xét: Số dư trung bình của khách hàng là khoảng 1331.86, trong khi số dư trung vị là 417.0. Sự khác biệt lớn giữa trung bình và trung vị cho thấy dữ liệu có thể bị ảnh hưởng bởi một số giá trị cực đại (outliers) lớn.
* Duration (Thời gian cuộc gọi)
* Trung bình (avg\_duration): 255.96
* Trung vị (median\_duration): 175.0
* Nhận xét: Thời gian cuộc gọi trung bình là khoảng 255.96 giây, trong khi trung vị là 175 giây. Điều này cho thấy phần lớn cuộc gọi có thời gian ngắn hơn so với mức trung bình, có thể có một số cuộc gọi rất dài kéo mức trung bình lên cao.
* Campaign (Số lần liên hệ trong chiến dịch)
* Trung bình (avg\_campaign): 2.82
* Trung vị (median\_campaign): 2.0
* Nhận xét: Số lần liên hệ trung bình trong chiến dịch là khoảng 2.82, trong khi trung vị là 2. Điều này cho thấy đa số khách hàng được liên hệ khoảng 2 lần trong chiến dịch.
* Pdays (Số ngày kể từ lần liên hệ cuối cùng trong chiến dịch trước)
* Trung bình (avg\_pdays): 34.17
* Trung vị (median\_pdays): -1.0
* Nhận xét: Giá trị trung vị là -1 cho thấy phần lớn khách hàng chưa từng được liên hệ trong chiến dịch trước. Giá trị trung bình là 34.17 bị kéo cao bởi một số khách hàng có pdays dương.
* Previous (Số lần liên hệ trước đó)
* Trung bình (avg\_previous): 0.47
* Trung vị (median\_previous): 0.0
* Nhận xét: Số lần liên hệ trước đó trung bình là 0.47, trong khi trung vị là 0. Điều này cho thấy hầu hết khách hàng chưa từng được liên hệ trước đó.

### Trực quan hóa dữ liệu

#### 1.3.3.1 Thống kê tỉ lệ khách hàng đăng ký gửi tiền có kỳ hạn theo mức độ học vấn

* Ta tính toán tỉ lệ bằng cách đếm tổng số lượng khách hàng trong từng ‘education’ gán vào biến ‘education\_counts’. Sau đó tính toán số lượng khách hàng đăng ký gửi tiền kỳ hạn trong từng mức độ học vấn bằng cách trong từng ‘education’ lấy số lượng khách hàng có gửi tiền kỳ hạn (‘term\_deposit’ == 1) gán vào biến ‘term\_deposit\_counts’ .Cuối cùng, tính toán phần trăm ‘education\_ratio’ bằng cách (term\_deposit\_counts / education\_counts) \* 100. Chuyển tỉ lệ thành DataFrame và vẽ biểu đồ bằng pyplot để trực quan hóa.

A screenshot of a computer code

Description automatically generated

Hình 1. 8 Trực quan hóa 1

* Kết quả thu được:

A graph with different colored rectangles

Description automatically generated

Hình 1. 9 Trực quan hóa 1 – Kết quả

* Kết luận: Từ biểu đồ ta dễ dàng nhận thấy nhóm khách hàng có học vấn từ cấp 3 trở lên (teriatry) có tỉ lệ đăng ký gửi tiền kỳ hạn cao nhất (khoảng 12 %). Ngược lại nhóm khách hàng có học vấn tiểu học (primary) có tỷ lệ đăng ký gửi tiền kỳ hạn thấp nhất (khoảng 7%). Điều này dễ hiểu vì hầu hết những người có học vấn cao thì sẽ có thu nhập cao và ổn định hơn những người có mức học vấn thấp. Vì thế chuyện tiền dư dả quyết định gửi tiền kỳ hạn vào ngân hàng để sinh lời là điều hiển nhiên với nhóm người có mức thu nhập cao.

#### 1.3.3.2 Thống kê tỉ lệ khách hàng đăng ký gửi tiền có kỳ hạn theo tình trạng hôn nhân

- Sử dụng hàm barplot trong Seaborn để xây dựng đồ thị dạng cột thống kê tỷ lệ khách hàng đăng ký gửi tiền có kỳ hạn theo tình trạng hôn nhân (term\_deposit = 1).

A screenshot of a computer code

Description automatically generated

Hình 1. 10 Trực quan hóa 2

- Kết quả thu được

A graph with different colored squares

Description automatically generated with medium confidence

Hình 1. 11 Trực quan hóa 2 – Kết quả

- Kết luận: Từ biểu đồ ta dễ dàng nhận thấy nhóm khách hàng đã kết hôn có tỷ lệ gửi tiền ngân hàng cao nhất ( gần 12%). Tương tự, nhóm khách hàng độc thân có tỷ lệ gửi tiền thấp (xấp xỉ 8%). Điều này dễ thấy ở ngoài thực tế, những người có gia đình hay tích góp còn những người độc thân hay có xu hướng sống tự do và chưa có mục tiêu tích lũy.

#### 1.3.3.3 So sánh tỉ lệ đăng ký gửi tiền có kì hạn giữa các nhóm khách hàng có khoản vay nhà và có khoản vay cá nhân

- Phân loại khách hàng thành 4 nhóm:

* Không có khoản vay nhà (0) và không có khoản vay cá nhân (0)
* Có có khoản vay nhà (1) và không có khoản vay cá nhân (0)
* Không có khoản vay nhà (0) và có khoản vay cá nhân (1)
* Có có khoản vay nhà (1) và có khoản vay cá nhân (1)

Sau đó sử bụng Bar Plot để hiển thị tỉ lệ đăng ký gửi tiền có kì hạn cho từng nhóm khách hàng trên.

A screenshot of a computer code

Description automatically generated

Hình 1. 12 Trực quan hóa 3

* Kết quả thu được:

A graph with blue squares

Description automatically generated

Hình 1. 13 Trực quan hóa 3 – Kết quả

* Nhận xét: Từ biểu đồ ta dễ dàng nhận thấy nhóm khách hàng không có khoản vay nhà và không có khoản vay cá nhân có tỉ lệ đăng ký gửi tiền có kì hạn cao nhất (gần 14%). Tương tự, nhóm khách hàng có khoản vay nhà và có khoản vay cá nhân có tỉ lệ đăng ký gửi tiền có kì hạn thấp nhất (khoảng 6%). Điều này cũng dễ hiểu so với thực tế khi khách hàng có các vay thì xác suất người đó đi gửi tiền sẽ giảm đi.

#### 1.3.3.4 Phân tích tần suất đăng ký gửi tiền có kỳ hạn theo từng tháng trong năm

- Chuyển dữ liệu cột month thành dạng Cetegorical để đảm bảo thứ tự các tháng từ tháng 1 đến tháng 12. Sau đó sử dụng Cat Plot của thư viện Seaborn để hiển thị tỉ lệ khách hàng đăng ký gửi tiền có kỳ hạn trong từng tháng trong năm.

A computer code on a white background

Description automatically generated

Hình 1. 14 Trực quan hóa 4

- Kết quả thu được:

A graph with a line and a dotted line

Description automatically generated

Hình 1. 15 Trực quan hóa 4 – Kết quả

- Nhận xét: Từ biểu đồ ta có thể thấy khách hàng có xu hướng gửi tiền có kì hạn cao vào tháng 3 (tháng sau tết), và các tháng về cuối năm (9, 10, 12). Tỉ lệ khách hàng gửi tiền thấp vào các tháng 1,2 (đầu năm mới) và các tháng về giữa năm (5, 6, 7, 8).

## 1.4 Công cụ sử dụng

* Nền tảng xử lý dữ liệu lớn: Apache Spark (Pyspark)
* Jupyter Notebook
* Công cụ khai thác: Jupyter Notebook
* Phiên bản Python: 3.11.5
* Thư viện kèm theo:
* from pyspark.sql import SparkSession
* from pyspark import SparkContext
* from pyspark.sql.functions import col, min as spark\_min, max as spark\_max, avg, mean, stddev, udf, when
* from pyspark.sql.types import DoubleType, IntegerType
* from pyspark.sql import Row
* import matplotlib.pyplot as plt
* import seaborn as sns
* import math

# CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU

## 2.1 Loại bỏ thuộc tính không cần thiết

- Loại bỏ thuộc tính ‘ID’ vì không cần thiết cho quá trình khai thác dữ liệu

A white rectangular object with a black border

Description automatically generated

Hình 2. 1 Tiền xử lý: Loại bỏ ‘ID’

## 2.2 Nhận diện và xử lý dữ liệu bị nhiễu

- Vẽ biểu bồ Box Plot của các thuộc tính định lượng và xem xét các giá trị nhiễu nằm ngoài khoảng chặn trên / chặn dưới.

A computer screen shot of a computer

Description automatically generated

Hình 2. 2 Code vẽ biểu đồ BoxPlot xác định Outlier

- Kết quả thu được:

A group of graphs with numbers

Description automatically generated with medium confidence

Hình 2. 3 Kết quả vẽ biểu đồ BoxPlot xác định Outlier

- Thống kê số lượng và tỉ lệ phần trăm giá trị nhiễu của từng thuộc tính bằng phương pháp tứ phân vị:

A screenshot of a computer code

Description automatically generated

Hình 2. 4 Tiền xử lý: Thống kê Outlier bằng phương pháp Box plot

- Xử lý ngoại lệ bằng Tứ phân vị trong đó, định nghĩa:

* Tứ phân vị là một phương pháp để phân chia tập dữ liệu thành bốn phần bằng nhau, mỗi phần chiếm 25% dữ liệu. Các tứ phân vị chính bao gồm:
* Q1 (Quartile 1): Điểm giữa của 25% đầu tiên (hay phân vị thứ 25).
* Q2 (Quartile 2) hay Median: Điểm giữa của tập dữ liệu (hay phân vị thứ 50).
* Q3 (Quartile 3): Điểm giữa của 25% cuối cùng (hay phân vị thứ 75).
* Công thức:
* IQR = Q3 - Q1
* Khoảng tứ phân vị (IQR) là khoảng cách giữa Q1 và Q3, biểu thị sự trải rộng của phần giữa dữ liệu.
* Lower bound (Giới hạn dưới) = Q1 - 1.5 \* IQR
* Upper bound (Giới hạn trên) = Q3 + 1.5 \* IQR
* Xác định Giá Trị Ngoại Lệ:
* Giá trị ngoại lệ thấp hơn: Các giá trị nhỏ hơn Lower bound.
* Giá trị ngoại lệ cao hơn: Các giá trị lớn hơn Upper bound.

A screenshot of a computer code

Description automatically generated

Hình 2. 5 Tiền xử lý: Hàm QuantileOutlierClipper

- Áp dụng vào dữ liệu của các thuộc tính trước đó, sau đó thực hiện lại thống kê số lượng và tỉ lệ phần trăm giá trị nhiễu của từng thuộc tính bằng phương pháp tứ phân vị trên dữ liệu mới:

A screenshot of a computer code

Description automatically generated

Hình 2. 6 Tiền xử lý: Áp dụng hàm QuantileOutlierClipper

- Nhận xét : Ta thấy rằng, dữ liệu đã không còn giá trị ngoại lệ.

## 2.3 Nhận diện dữ liệu bị thiếu

- Đếm số lượng giá trị null ở từng thuộc tính trong dữ liệu ban đầu

A screenshot of a computer code

Description automatically generated

Hình 2. 7 Tiền xử lý: Nhận diện giá trị bị thiếu

- Nhận xét: Dữ liệu ban đầu không có giá trị null nên ta không cần áp dụng kĩ thuật xử lý dữ liệu bị thiếu trên tập dữ liệu này.

## 2.4 Xử lý ý nghĩa dữ liệu

### 2.4.1 Cột ‘job’ và ‘education’

- Ta chuyển dữ liệu cột ‘job’ và ‘education’ đang có dạng danh mục thể hiện cho từng loại hình công việc và từng mức học vấn ta thực hiện chuyển dữ liệu của cột ‘job’ và ‘education’ thành dạng số bằng phương pháp StringIndexer để phục vụ cho các thuật toán khai thác dữ liệu. Sau đó bỏ đi dữ liệu ở 2 cột dữ liệu cũ.

A computer screen shot of a computer code

Description automatically generated

Hình 2. 8 Tiền xử lý ‘job’ và ‘education’

### 2.4.2 Cột “month”

A screen shot of a computer

Description automatically generated

Hình 2. 9 Tiền xử lý ‘month’

- Ta thấy dữ liệu cột month đang ở dạng danh mục với các giá trị dạng chuỗi đại diện cho các tháng trong năm. Ta thực hiện chuyển kiểm dữ liệu cột “month” thành kiểu số tương tứng từ tháng 1 đến tháng 12.

A screen shot of a computer

Description automatically generated

Hình 2. 10 Tiền xử lý: Xử lý ý nghĩa dữ liệu – Cột month

## 2.5 Chuẩn hóa dữ liệu

- Thống kê 2 thuộc tính định lượng ‘balance’ và ‘duration’ của tập dữ liệu sau các bước tiền xử lý trước đó.

A screenshot of a computer

Description automatically generated

Hình 2. 11 Tiền xử lý: Thống kê thuộc tính

- Ta thấy dữ liệu các cột balance và duration có độ lệch chuẩn lớn và chênh lệch giữa giá trị lớn nhất và giá trị nhỏ nhất là khá cao gây ảnh hướng đến việc khai thác dữ liệu. Ta sử dụng phương pháp Min-Max Nomalization để đưa giá trị cột balance và duration về khoảng [0,1].

A screenshot of a computer code

Description automatically generated

Hình 2. 12 Tiền xử lý: Code Chuẩn hóa dữ liệu

- Sau khi chuẩn hóa, dữ liệu cột balance và duration đã có độ lệch chuẩn bé hơn 1.

A screenshot of a computer program

Description automatically generated

Hình 2. 13 Tiền xử lý: Kết quả chuẩn hóa dữ liệu

## 2.6 Tách dữ liệu huấn luyện và kiểm thử

- Ta thực hiện chuyển thuộc tính dự đoán ‘term\_deposit’ về cuối cùng và xem dữ liệu mới sau các bước tiền xử lý dữ liệu đã thực hiện ở trên.

A screenshot of a computer

Description automatically generated

Hình 2. 14 Tiền xử lý: Dữ liệu mới sau xử lý

* Ta tiến hành tách dữ liệu làm 2 phần theo tỉ lệ 8:2. 80% dữ liệu cho phần huấn luyện (train), 20% dữ liệu cho phần kiểm định (test). Lưu kết quả vào các biến tương ứng X\_train. X\_test, y\_train, y\_test. Trong đó: X chứa các thuộc tính độc lập, y chứa thuộc tính quyết định (term\_deposit).

A white background with colorful text

Description automatically generated

Hình 2. 15 Tiền xử lý: Tách dữ liệu thành X, y

* Kiểm tra kích thước tập dữ liệu Train, Test

A computer screen shot of a computer code

Description automatically generated

Hình 2. 16 Tiền xử lý: Kích thước Train Test

# CHƯƠNG 3: ÁP DỤNG CÁC GIẢI THUẬT KHAI THÁC DỮ LIỆU

# 3.1 Giải thuật K-Nearest Neighbors

## 3.1.1 Khái niệm

- Thuật toán K-Nearest Neighbor (KNN) là một thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi huấn luyện, thuật toán này không học một điều gì từ dữ liệu training (đây 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. KNN có thể được áp dụng cho cả 2 bài toán là Phân lớp (Classification) và Hồi quy (Regression).

- 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 nó nhất trong tập traning set. Nhãn của một dữ liệu test có thể được quyết định bằng cách bầu chọn giữa các điểm gần nhất (số lần xuất hiện), hoặc cũng có thể được suy ra bằng các đánh trọng số khác nhau cho mỗi trong điểm trong các điểm gần nhất đó rồi suy ra nhãn. Trong bài toán Hồi quy, đầu ra của một điểm dữ liệu được tính bằng chính đầu ra của điểm gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của các điểm gần nhất. 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 traning set 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.

- Lý do lựa chọn:

* K-nearest Neighbor là một trong những thuật toán học có giám sát đơn giản nhất được ứng dụng cho các bài toán Phân lớp và Hồi quy. Hoàn toàn đáp ứng được yêu cầu khai thác dữ liệu của đồ án.
* Thời gian huấn luyện của mô hình là bằng 0, KNN được gọi là Lazy Learning vì nó không thực hiện quá trình huấn luyện nào trên dữ liệu huấn luyện. Thay vào đó, nó chỉ lưu trữ dữ liệu huấn luyện trong bộ nhớ và thực hiện tính toán khi có dữ liệu mới.
* Việc dữ đoán kết quả của dữ liệu mới rất đơn giản vì thuật toán này không yêu cầu đào tạo trước khi đưa ra dự đoán. Dữ liệu mới có thể được thêm vào một cách liền mạch mà không làm thay đổi độ chính xác của thuật toán.
* KNN là một mô hình không có tham số (nonparametric). Quá trình tính toán được thực hiện độc lập với mỗi điểm dữ liệu khác nhau. Hơn nữa, thuật toán KNN yêu cầu rất ít tham số, chỉ cần 2 tham số là K – số điểm dữ liệu và một độ đo khoảng cách. Điều này là ít hơn khi so sánh với các thuật toán Machine Learning khác.

## 3.1.2 Song song hóa giải thuật với mô hình Map Reduce

A diagram of a diagram

Description automatically generated

Hình 3. 1 Song song hóa giải thuật KNN

- (Jesus Maillo, Isaac Triguero, Francisco Herrera, 2015) Trong mô hình MapReduce, quá trình tính toán được tổ chức thành 2 giai đoạn chính: giai đoạn map và giai đoạn reduce. Giai đoạn map sẽ tính toán các lớp (class) và khoảng cách đến k lân cận gần nhất của mỗi điểm dữ liệu kiểm tra trong các phần khác nhau của dữ liệu huấn luyện. Giai đoạn Reduce sẽ xử lý khoảng cách của k lân cận gần nhất từ mỗi map và tạo ra danh sách k lân cận gần nhất bằng cách chọn những điểm có khoảng cách nhỏ nhất. Sau đó thực hiện quá trình bỏ phiếu như thường lệ giống như mô hình KNN tuần tự để dự đoán lớp kết quả. Chi tiết từng quá trình được trình bày bên dưới:

### 3.1.2.1 Giai đoạn Map

- Gọi TR và TS lần lượt là tập dữ liệu huấn luyện và tập dữ liệu kiểm tra ban đầu. Giai đoạn map bắt đầu chia tập TR thành một số tập con không giao nhau nhất định. Gọi m là số lượng quy trình map được xác định trước. Mỗi tác vụ map (Map1, Map2,… Mapm) sẽ làm việc với một TRj tương ứng, với 1 ≤ j ≤ m ứng với mỗi tập con của dữ liệu huấn luyện đã được chia. Mapj sẽ tương ứng với phần dữ liệu con j của tập dữ liệu huấn luyện. Kết quả là mỗi map sẽ phân tích một tập con của dữ liệu huấn luyện có số lượng mẫu như nhau.

- Mục tiêu của thuât toán là thu được một triển khai chính xác của thuật toán KNN gốc, tập dữ liệu kiểm tra TS sẽ không bị chia nhỏ vì ta cần truy cập vào từng mẫu của TS trong tất cả các map. Bằng cách này, khi mỗi map đã tạo thành tập TRj tương ứng với nó, ta sẽ tính toán khoảng cách của mỗi mẫu dữ liệu kiểm tra x\_t đối với các mẫu của TR\_j. Nhãn lớp (class) của k lân cận gần nhất cho mỗi mẫu kiểm tra sẽ được lưu lại. Kết quả là, ta sẽ thu được ma trận CD,j của các cặp <class, distance> với kích thước n (kích thước tập TS) x k. Mỗi hàng ứng với thứ tự của mẫu kiểm tra sẽ được sắp xếp theo thứ tự tăng dần về khoảng cách, do đó Dist(neigh1) < Dist(neigh2) < …. < Dist(neighk). Khi mỗi map hoàn thành quy trình tính toán, kết quả được chuyển đến một tác vụ Reduce duy nhất.

### 3.1.2.2 Giai đoạn Reduce

- Giai đoạn Reduce bao gồm việc xác định lại k lân cận gần nhất từ kết quả của các map cho toàn bộ dữ liệu kiểm tra. Giai đoạn Reduce được chia thành 3 quy trình như sau:

* Setup: Quá trình này sẽ tạo một ma trận CD\_reducer gồm các cặp <class, distance> với kích thước cố định n x k trong đó n là số lượng mẫu của tập TS, k là số lân cận gần nhất, hàng i tương ứng với mẫu dữ liệu thứ i. Ma trận được khởi tạo với giá trị None cho các lớp (class) và giá trị dương vô cùng cho các khoảng cách (distance). Thao tác này chỉ được khởi tạo một lần khi quá trình Reduce diễn ra. Trong MapReduce, reducer có thể bắt đầu nhận dữ liệu khi mapper đầu tiên hoàn thành.
* Reduce: Khi giai đoạn map kết thúc. Ma trận class-distance CD\_reducer được cập nhật bằng cách so sánh các giá trị khoảng cách hiện tại của nó với ma trận CDj đến từ mỗi map j. Vì các ma trận đến từ các map được sắp xếp theo khoảng cách, quá trình cập nhật sẽ diễn ra nhanh hơn. Nó bao gồm quá trình hợp nhất 2 danh sách đã sắp xếp đến khi có k giá trị (tương tự như thuật toán Merge Sort). Do đó đối với mỗi mẫu kiểm tra x\_test, ta so sánh từng giá trị khoảng cách của các lân cận một các tuần tự, bắt đầu từ lân cận gần nhất. Nếu khoảng cách nhỏ hơn giá trị hiện tại, lớp và khoảng cách của vị trí này được cập nhật với các giá trị tương ứng, nếu không thì xét với giá trị tiếp theo. Thao tác này được chạy mỗi khi một map hoàn thành. Do đó, nó có thể được hiểu như một quy trình lặp lại kết hợp kết quả do các map cung cấp. Như vậy, thao tác này không cần gửi bất kỳ cặp <key, value> nào. Nó được thực hiện trong quy trình tiếp theo.
* Cleanup: Sau khi tất cả đầu vào của các map đã được xử lý bởi quy trình Reduce trước đó, giai đoạn này sẽ tiến hành xử lý kết quả thu được. Tại thời điểm này, CD\_reducer sẽ chứa danh sách cuối cùng của các lân cận (lớp và khoảng cách) cho tất cả các mẫu của TS. Do đó, quá trình này sẽ thực hiện bỏ phiếu đa số của mô hình k-NN và xác định lớp được dự đoán do TS. Kết quả là các lớp được dự đoán cho tất cả tập TS được cung cấp làm đầu ra cuối cùng của giai đoạn Reduce.

## 3.1.3 Áp dụng giải thuật trên Apache Spark

- Bước 1: Import các thư viện cần thiết và bắt đầu một Session trong Spark.

A screenshot of a computer program

Description automatically generated

Hình 3. 2 KNN - Bước 1

- Bước 2: Đọc dữ liệu đã được tiền xử lý trước đó thành một DataFrame.

A screenshot of a computer

Description automatically generated

Hình 3. 3 KNN - Bước 2

* Bước 3: Đầu tiên ta thêm cột ‘id’ để làm khóa cho mỗi mẫu của tập dữ liệu. Sau đó tách dữ liệu thành tập huấn luyện và kiểm tra theo tỉ lệ 8:2.

A close-up of a computer screen

Description automatically generated

Hình 3. 4 KNN - Bước 3

* Bước 4: Chọn ra các cột features của dữ liệu. Sau đó sử dụng VectorAssemler để tạo column features từ các features đã chọn.

A screen shot of a computer code

Description automatically generated

Hình 3. 5 KNN - Bước 4

Kết quả thu được:

A screenshot of a computer

Description automatically generated

Hình 3. 6 KNN - Bước 4 Kết quả

* Bước 5: Thực hiện tính toán khoảng cách giữa điểm dữ liệu mới và tất cả các điểm dữ liệu huấn luyện. Độ đo khoảng các thường được sử dụng là khoảng cách Euclidean – là đường thẳng nối giữa 2 điểm tính bằng công thức:



Hình 3. 7 KNN - Bước 5

* Bước 6: Tạo hàm tính khoảng cách và tìm k điểm lân cận gần nhất cho mỗi mẫu dữ liệu kiểm thử gồm 3 tham số:

def compute\_distances(test\_point, train\_data\_parttion, k)

Trong đó:

* test\_point: Một mẫu dữ liệu của tập kiểm tra
* train\_data\_parttion: Một partition của tập dữ liệu huấn luyện đã được chia tách hành các phần bằng nhau.
* k: Số lượng điểm lân cận cần xét

A computer screen shot of a computer code

Description automatically generated

Hình 3. 8 KNN – Bước 6

Quy trình tính toán

* Lần lượt lặp qua các mẫu dữ liệu của Parttion
* Thực hiện tính khoảng cách đến test\_point và thêm vào mảng neighbors danh sách 2 giá trị, lớp của mẫu dữ liệu huấn luyện và khoảng cách tính được.
* Thực hiện sắp xếp mảng neighbors theo khoảng cách và sắp xếp tăng dần. Kết quả trả về k lân cận gần nhất.
* Bước 7: Tạo hàm thực hiện giai đoạn Map để tính toán và tìm k lân cận gần nhất cho toàn bộ dữ liệu kiểm thử trên một parttion của dữ liệu huấn luyện gồm 3 tham số.

def map\_phase(partition, test\_data\_broadcast, k)

* partition: Một partition của tập dữ liệu huấn luyện đã được chia tách hành các phần bằng nhau.
* test\_data\_broadcast: Toàn bộ mẫu dữ liệu tập kiểm thử đã được broadcast cho toàn bộ các worker.
* k: Số lượng điểm lân cận cần xét

A screen shot of a computer program

Description automatically generated

Hình 3. 9 KNN – Bước 7

* Bước 8: Tạo hàm khởi tạo giá trị ban đầu cho ma trận CD\_reducer cho toàn bộ mapper (Quá trình Setup) gồm 2 tham số: số lượng mẫu tập kiểm tra và số lân cận.

A computer screen shot of a code

Description automatically generated

Hình 3. 10 KNN – Bước 8

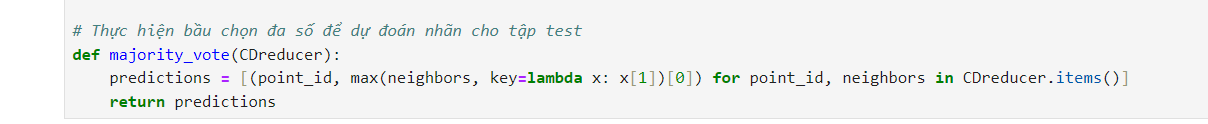
- Bước 9: Tạo hàm cập nhật ma trận ma trận CD\_reducer khi nhận được kết quả là ma trận CDj từ các map (Quá trình Reduce) gồm 2 tham số: Kết quả của các map và ma trận CDreducer.

A computer code with text

Description automatically generated

Hình 3. 11 KNN – Bước 9

* Bước 10: Sau khi có được ma trận CD\_reducer cuối cùng từ quá trình Reduce. Tạo hàm để thực hiện đự đoán lớp cho tập dữ liệu kiểm tra như thuật toán KNN tuần tự (Quá trình Cleanup)



Hình 3. 12 KNN – Bước 10

- Bước 11: Sau khi đã tạo thành công các hàm cho từng giai đoạn Map, Reduce. Thực hiện phân tách tập dữ liệu huấn luyện thành 4 partition. Sau đó Broadcast toàn bộ dữ liệu tập kiểm tra để có thể truy cập từ các worker. Khởi tạo k = 5

A computer code with text

Description automatically generated with medium confidence

Hình 3. 13 KNN – Bước 11

- Bước 12: Lần lượt map qua các parttion của dữ liệu Train với hàm mapPartitions, mỗi partion thực hiện gọi hàm map\_phase. Sau đó khởi tạo giá trị ban đầu cho CD\_reducer. Cập nhật ma trận CD\_reducer với mỗi kết quả trả về từ các map với hàm update\_CDreducer. Sau đó tiến hành dự đoán lớp cho tập kiểm tra với hàm majority\_vote.

A computer screen shot of a computer code

Description automatically generated

Hình 3. 14 KNN – Bước 12

- Bước 13: Thực hiện đánh giá độ chính xác, thời gian chạy và ma trận nhầm lẫn của mô hình Map Reduce KNN.

A screenshot of a computer

Description automatically generated

Hình 3. 15 KNN – Bước 13

Kết quả thu được:

A screenshot of a computer

Description automatically generated

Hình 3. 16 KNN – Bước 14

Từ kết quả thu được ta có thể thấy:

* Độ chính xác của mô hình vào khoảng 86.38%, một giá trị tương đối ổn với một mô hình được vận hành trên dữ liệu lớn.
* Thời gian chạy của mô hình lên tới 3667s (khoảng hơn 1 tiếng)
* Ma trận nhầm lẫn tương ứng

- Bước 15: Trực quan hóa ma trận nhầm lẫn

A screenshot of a computer

Description automatically generated

Hình 3. 17 KNN – Bước 15

# 3.2 Giải thuật Random Forest

## 3.2.1. Lý do lựa chọn

Thuật toán Random Forest được sử dụng chủ yếu cho các bài toán phân loại và hồi quy, thuật toán sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định.

**Nguyên lý hoạt động của Random Forest:**

**A diagram of a diagram

Description automatically generated**

Hình 3 1 Nguyên lý hoạt động của Random Forest

* **Tạo nhiều mẫu dữ liệu huấn luyện (Bootstrapping):** Từ tập dữ liệu ban đầu, nhiều mẫu dữ liệu ngẫu nhiên được tạo ra bằng phương pháp bootstrapping. Điều này có nghĩa là mỗi mẫu dữ liệu con sẽ chứa các bản ghi được chọn ngẫu nhiên từ tập dữ liệu ban đầu, có thể có sự trùng lặp.
* **Xây dựng các cây quyết định:** Với mỗi mẫu dữ liệu con, một cây quyết định sẽ được xây dựng. Mỗi cây quyết định này chỉ lấy tập hơn con k thuộc tính từ n thuộc tính ban đầu (k < n)
* **Dự đoán bằng cách bỏ phiếu (Ensemble Voting)**
  + **Đối với bài toán phân loại**: Kết quả dự đoán của mô hình Random Forest là kết quả của việc bỏ phiếu đa số (majority vote) từ các cây quyết định. Cụ thể, mỗi cây đưa ra một dự đoán và dự đoán nào có nhiều phiếu nhất sẽ là kết quả cuối cùng.
  + **Đối với bài toán hồi quy**: Kết quả dự đoán là giá trị trung bình của tất cả các giá trị dự đoán từ các cây quyết định.

Ưu điểm của thuật toán :

* **Độ chính xác cao:** Random Forest thường cho kết quả chính xác hơn so với việc sử dụng một cây quyết định đơn lẻ.
* **Tránh bị overfitting:** Do việc sử dụng nhiều cây quyết định và lấy trung bình hoặc bỏ phiếu đa số, Random Forest ít bị overfitting so với các phương pháp Decision Tree ban đầu.
* **Đa dạng hóa**: Sự ngẫu nhiên trong việc chọn mẫu dữ liệu và lựa chọn các thuộc tính (features) khi phân chia tại các nút của cây quyết định giúp mô hình tránh bị lệ thuộc vào một phần của dữ liệu.

## 3.2.2. Lập trình với Apache Spark

Lớp Node: Lớp này đại diện cho một nút trong cây quyết định.

* Phương thức \_\_init\_\_: Khởi tạo nút với các thuộc tính feature (chỉ mục của đặc trưng để chia nhánh), threshold (ngưỡng để chia nhánh đặc trưng), left và right (các nút con), và value (nhãn lớp nếu nút là lá).
* Phương thức is\_leaf\_node: Kiểm tra xem nút có phải là nút lá không bằng cách kiểm tra xem value có khác None hay không

A computer screen shot of a computer code

Description automatically generated

Lớp DecisionTreeSpark: Lớp này đại diện cho một cây quyết định sử dụng Apache Spark để xử lý dữ liệu phân tán.

* Phương thức \_\_init\_\_: Khởi tạo cây với min\_samples\_split (số lượng mẫu tối thiểu để chia một nút), max\_depth (độ sâu tối đa của cây), n\_features (số lượng đặc trưng để xem xét cho việc chia nhánh), và root (nút gốc của cây).
* Phương thức fit: Xây dựng cây quyết định bằng cách gọi phương thức \_grow\_tree với dữ liệu đầu vào.

A computer code with text

Description automatically generated

Phương thức \_grow\_tree: Xây dựng cây quyết định một cách đệ quy.

* Trường hợp cơ bản cho đệ quy: Nếu độ sâu hiện tại vượt quá max\_depth, tất cả các mẫu có cùng nhãn, hoặc có ít hơn mẫu hơn min\_samples\_split, tạo một nút lá.
* Tạo một nút lá: Tính toán nhãn phổ biến nhất trong dữ liệu hiện tại và tạo một nút với nhãn đó là giá trị của nút.

Chia nhánh của một nút:

* Chọn một tập hợp con ngẫu nhiên của các đặc trưng (feature\_idxs) để xem xét cho việc chia nhánh.
* Tìm đặc trưng và ngưỡng tốt nhất để chia dữ liệu bằng cách sử dụng \_best\_split.
* Chia dữ liệu thành các tập con trái và phải bằng cách sử dụng \_split.
* Xây dựng đệ quy các cây con bên trái và bên phải.
* Tạo và trả về một nút với đặc trưng và ngưỡng tốt nhất.

A computer screen shot of a code

Description automatically generated

Phương thức \_best\_split: Tìm đặc trưng và ngưỡng tốt nhất để chia dữ liệu dựa trên lợi ích thông tin.

Khởi tạo: Đặt best\_gain thành -1, chỉ ra không có lợi ích nào được tìm thấy.

Lặp qua các đặc trưng: Đối với mỗi đặc trưng trong feature\_idxs, thu thập các giá trị ngưỡng duy nhất.

Đánh giá các chia nhánh: Đối với mỗi ngưỡng, tính toán lợi ích thông tin và cập nhật best\_gain, split\_feature, và split\_threshold nếu lợi ích là tốt nhất

A screenshot of a computer code

Description automatically generated

Phương thức \_information\_gain: Tính toán lợi ích thông tin từ việc chia dữ liệu trên một đặc trưng tại một ngưỡng nhất định.

* Entropy của cha: Tính toán entropy của dữ liệu hiện tại.
* Chia dữ liệu: Chia dữ liệu thành các tập con trái và phải dựa trên đặc trưng và ngưỡng.
* Kiểm tra chia nhánh trống: Nếu bất kỳ tập con nào là trống, trả về lợi ích bằng 0.
* Tính toán entropy của con: Tính toán entropy có trọng số của các tập con trái và phải.
* Trả về lợi ích thông tin: Sự khác biệt giữa entropy của cha và entropy của con.

A screenshot of a computer code

Description automatically generated

Phương thức \_split: Chia dữ liệu thành hai tập con dựa trên một đặc trưng và ngưỡng.

* Tập con trái: Chứa các mẫu trong đó giá trị đặc trưng nhỏ hơn hoặc bằng ngưỡng.
* Tập con phải: Chứa các mẫu trong đó giá trị đặc trưng lớn hơn ngưỡng

A close-up of a computer code

Description automatically generated

Phương thức \_entropy: Tính toán entropy của dữ liệu.

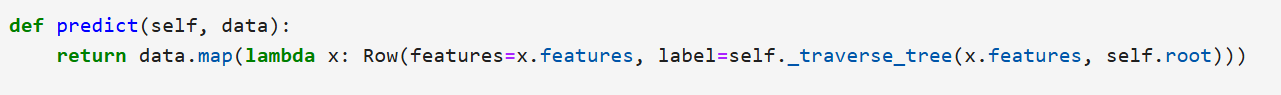
* Tổng số lượng: Đếm số lượng mẫu cho mỗi nhãn.
* Xác suất của nhãn: Tính toán xác suất của mỗi nhãn.
* Tính toán entropy: Sử dụng công thức entropy: -sum(p \* log(p)).

A close-up of a computer code

Description automatically generated

Phương thức predict: Dự đoán nhãn cho dữ liệu đầu vào.

* Ánh xạ dữ liệu: Áp dụng phương thức \_traverse\_tree cho mỗi mẫu để lấy nhãn dự đoán.



Phương thức \_traverse\_tree: Duyệt qua cây để dự đoán nhãn cho một mẫu.

* Nút lá: Trả về giá trị nếu nút là nút lá.
* Nút nội: Đệ quy duyệt qua cây con trái hoặc phải dựa trên giá trị của đặc trưng.

A computer code with text

Description automatically generated

Lớp RandomForestSpark: Lớp này đại diện cho một rừng ngẫu nhiên, là một tập hợp các cây quyết định.

* Phương thức \_\_init\_\_: Khởi tạo rừng với số cây (n\_trees), độ sâu tối đa của mỗi cây (max\_depth), số lượng mẫu tối thiểu để chia một nút (min\_samples\_split), số lượng đặc trưng để xem xét cho việc chia nhánh (n\_features), và một danh sách rỗng để lưu trữ các cây (trees).

A computer code with black and white text

Description automatically generated

Phương thức fit: Huấn luyện rừng ngẫu nhiên.

* Khởi tạo danh sách cây: Làm trống danh sách các cây.
* Huấn luyện mỗi cây: Đối với mỗi cây, tạo một cây quyết định mới với các tham số được chỉ định, lấy mẫu dữ liệu với việc thay thế, và huấn luyện cây với dữ liệu đã lấy mẫu, sau đó thêm cây vào danh sách các cây

A computer code with text

Description automatically generated

Phương thức predict: Dự đoán nhãn cho dữ liệu đầu vào bằng rừng ngẫu nhiên.

* Dự đoán ban đầu: Bắt đầu với dữ liệu đầu vào.
* Tích hợp dự đoán: Đối với mỗi cây, áp dụng phương thức predict cho mỗi mẫu và cập nhật các dự đoán.
* Trả về các dự đoán cuối cùng.

A close-up of a computer screen

Description automatically generated

Tiến hành chạy thuật toán

* Xác định số lượng cột đặc trưng và tên của cột nhãn.
* Đặt feature\_columns là danh sách các cột đặc trưng (cột từ đầu đến cột thứ 14 của DataFrame).
* Đặt label\_column là tên của cột nhãn.
* Định nghĩa hàm row\_to\_labeled\_point để chuyển đổi mỗi hàng của DataFrame thành một LabeledPoint.
* Áp dụng hàm này cho mỗi hàng của DataFrame và thu thập kết quả thành một danh sách data.
* Chuyển đổi danh sách data thành một RDD.
* Sử dụng phương thức randomSplit để chia RDD thành tập huấn luyện (80%) và tập kiểm tra (20%) với một hạt giống xác định
* Khởi tạo một mô hình Rừng Ngẫu nhiên với các tham số:
  + Số cây tối đa là 10
  + Độ sâu tối đa mỗi cây là 10
  + Số lượng mẫu yêu cầu tối thiểu để phân chia là 2
  + Số lượng đặc trưng được chọn ngẫu nhiên để xem xét khi tìm kiếm sự phân chia tốt nhất. None là lấy tất cả

A screenshot of a computer code

Description automatically generated

## 3.2.3. Hình vẽ minh họa song song

Ta so sánh các hàm trong hai cách thực hiện.

* Với Apache Spark: nền trắng
* Với numpy: nền đen
* Class Node

A computer screen shot of a computer code

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

* Class Random Forest

A screen shot of a computer code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

* Class Decision Tree

A screenshot of a computer code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

# 3.3 Giải thuật Support Vector Machine

## 3.3.1 Khái niệm

**Nguyên lý hoạt động của Support Vector Machine:**

A diagram of a mathematical equation

Description automatically generated

* SVM là một thuật toán máy học thường đường sử dụng cho các bài toán phân lớp nhị phân.
* SVM hoạt động bằng cách tìm một siêu phẳng (hyperplane) với biên lớn nhất phân chia dữ liệu thành các lớp , siêu phẳng được tính theo công thức: , trong đó
  + w, b là các trọng số
  + x là các thuộc tính feature của bộ dữ liệu
* SVM sẽ có công thức như sau:

**Phương pháp Stochastic Gradient Descent:**

- Để tính toán được giá trị tối ưu của các trọng số chúng ta sẽ tính bằng cách giảm tối thiểu giá trị của hàm cost:

- Chúng ta có thể gộp w và b lại thành một bằng cách đưa b vào vec tơ W như sau:

,

với,

- Sau đó thì hàm cost sẽ trở thành:

- Dựa vào hàm cost, chúng ta tính đạo hành riêng (gradient) theo W:

- Và để giảm tối thiểu giá trị của hàm cost thì chúng ta sử dụng phương pháp SGD (Stochastic Gradient Descent)

A diagram of a weight loss

Description automatically generated

- SGD hoạt động như sau:

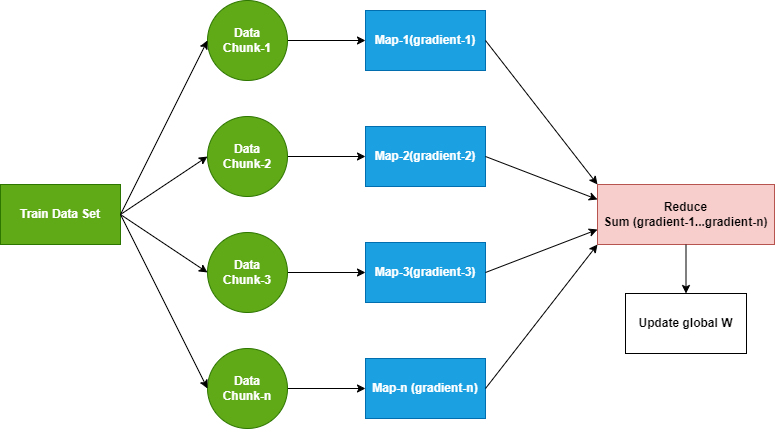
* Bước 1: Tính đạo hàm riêng của hàm cost
* Bước 2: Di chuyển trọng số hướng ngược lại bằng công thức W = W – gradient
* Bước 3: Lặp lại đến khi ta tìm được W sao cho J(W) đạt nhỏ nhất
* Đạo hàm riêng là hướng tăng nhanh nhất của hàm số vì vậy đi ngược lại sẽ giảm tối thiểu hàm J(W).

- Lý do phải giảm tối thiểu hàm cost là vì về cơ bản hàm cost là một thước đo mức độ kém hiệu quả của mô hình trong việc đạt được mục tiêu. Nếu nhìn vào J(w), để tìm giá trị nhỏ nhất của nó, chúng ta phải:

* Giảm tối thiểu nghĩa là tối đa (biên của siêu phẳng)
* Giảm tối hiểu tổng của nghĩa là tối thiểu việc phân lớp sai

## 3.3.2 Song song hóa giải thuật dựa trên MapReduce

**Ý tưởng:**



Việc đầu tiên là ta sẽ khởi tạo một mảng trọng số W (global W) với giá trị toàn bộ là 0, sau đó ta tiến hành training. Trong quá trình training:

* Đầu tiên tập dữ liệu train sẽ được chia nhỏ ra thành từng phần (data chunk).
* Các mapper sẽ tiến hành tính toán gradient của hàm cost dựa trên data chunk nhận được và global W.
* Các giá trị gradient sau đó sẽ được gửi đến reducer và tiến hành quá trình reduce.
* Cuối cùng dựa trên ouput của reducer, ta sẽ cập nhật global W.
* Sau đó ta tiến hành lặp lại quá trình training dựa vào global W đã cập nhật cho đến khi thu được thu được trọng số W mong muốn.

**Cài đặt:**

- Đầu tiên chúng ta sẽ cập nhật lại bộ dữ liệu train test, thêm vào X\_train và X\_test một cột là intercept với toàn bộ giá trị là 1 (dựa theo cách gộp w và b ở phần khái niệm)

A screen shot of a computer code

Description automatically generated

- Khởi tạo các RDD, với y sẽ chỉ chứa 1 và -1:

A screenshot of a computer code

Description automatically generated

- Thiết lập số lượng partition trong train\_rdd, tương tự số lượng data chunk mà dữ liệu sẽ có:



- Xây dựng svm\_calculate\_cost\_gradient để tính toán giá trị gradient của hàm cost:

A screenshot of a computer program

Description automatically generated

* Đầu vào của hàm tính toán là dữ liệu và một mảng trọng số W.
* Bước 1: Ta sẽ chia dữ liệu vào hai mảng là X\_batch và Y\_batch, X\_batch sẽ chưa các thuộc x và Y\_batch sẽ chứa thuộc tính y.
* Bước 2: Tính toán distance bằng X\_batch, Y\_batch và W dựa theo công thức:
* Bước 3: Tạo một mảng 0 là dw.
* Bước 4: Tiến hành tính toán gradient dựa trên công thức:
  + Nếu max(0, distance) == 0 thì t lấy W.
  + Ngược lại thì ta lấy W – (reg\_strength \* Y\_batch[ind] \* X\_batch[ind])
  + Sau đó ta cập nhật mảng dw
* Sau khi kết thúc vòng lập ta chia dw cho len(Y\_batch) tức N.
* Cuối cùng đầu ra của hàm sẽ là một mảng cái giá trị gradient.

- Xây dựng hàm svm\_combine\_cost\_gradient, tính tổng hai mảng gradient:

A close-up of a computer code

Description automatically generated

- Xây dựng hàm svm\_sgd, đây sẽ được xem là hàm train:

A screen shot of a computer

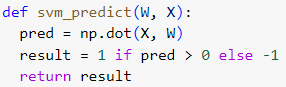
Description automatically generated

* Đầu vào của hàm sẽ là tập dữ liệu các thuộc tính feature(X\_train)
* Bước 1: Ta sẽ khởi tạo một mảng trọng số W (global W) ban đầu sẽ có giá trị toàn bộ là 0.
* Bước 2: Khởi tạo vòng lặp training, số lần lặp sẽ tùy vào người sử dụng, điều kiện dừng sẽ dựa trên sự thay đổi của các giá trị gradient, nếu sự thay đổi bé hơn threshold thì dừng vòng lặp, threshold cũng sẽ tùy vào người sử dụng.
  + Bên trong vòng lặp sẽ tiến hành tính toán các gradient, hàm mapPartions sẽ lấy các partition (data chunk) bên trong X và đưa các partition đến các mapper để tính toán song song, hàm Map ở đây sẽ là svm\_calculate\_cost\_gradient.
  + Sau khi tính toán, các output sẽ được reduce bằng hàm svm\_combine\_cost\_gradient.
  + Sau đó output của reduce sẽ chia cho số lượng partition để cho ra kết quả trung bình các gradient.
  + Cuối cùng global W sẽ được cập nhật dựa trên gradient theo công thức:

W = W - gradient

* + Và vòng lặp sẽ tiếp tục với global W vừa được cập nhật cho đến khi kết thúc.
* Đầu ra của hàm sẽ là một mảng trọng số W tối ưu.

- Xây dựng hàm svm\_predict, đây sẽ là hàm sử dụng để dự đoán:



* Hàm được xây dựng dựa trên công thức:
* Kết quả dự đoán > 0 thì sẽ thuộc lớp 1, ngược lại thì sẽ thuộc lớp -1.

- Tiến hành train và predict:

A screenshot of a computer

Description automatically generated

- Thực hiện đánh giá độ chính xác, thời gian chạy và ma trận nhầm lẫn của mô hình, kết quả thu được:

A screenshot of a computer code

Description automatically generated

Từ kết quả thu được ta có thể thấy:

* Độ chính xác của mô hình vào khoảng 72.9%, một giá trị tương đối ổn với một mô hình được vận hành trên dữ liệu lớn.
* Thời gian chạy của mô hình lên tới 4041s (khoảng hơn 1 tiếng)
* Ma trận nhầm lẫn tương ứng

A blue squares with white text

Description automatically generated

# CHƯƠNG 4: ĐÁNH GIÁ CÁC THUẬT KHAI THÁC DỮ LIỆU

## 4.1 Đánh giá về thời gian chạy thuật toán

A graph showing the number of different types of data

Description automatically generated with medium confidence

Hình 4. 1 Đánh giá thời gian chạy

- Từ kết quả trên ta thu được thời gian chạy của 2 thuật toán tương ứng:

* K-Nearest Neighbor: 3667s
* Support Vector Machine: 4041s

- Nhận xét:

* **K-Nearest Neighbor (KNN):** KNN có thời gian chạy ít hơn SVM, điều này là do KNN chỉ cần tính toán khoảng cách giữa các điểm dữ liệu và không cần phải xây dựng mô hình phức tạp. KNN chủ yếu dựa vào việc tìm kiếm và so sánh khoảng cách, điều này có thể được phân phối dễ dàng trong mô hình MapReduce. Thời gian chạy chủ yếu phụ thuộc vào số lượng điểm dữ liệu và số lượng láng giềng cần tìm.
* **Support Vector Machine (SVM):** SVM thường phức tạp hơn KNN vì nó liên quan đến việc tìm kiếm một siêu phẳng tối ưu để phân tách các lớp, điều này đòi hỏi giải quyết một bài toán tối ưu hóa. Việc này mất nhiều thời gian hơn, đặc biệt là khi xử lý dữ liệu lớn trong môi trường phân tán như Apache Spark.

## 4.2 Đánh giá về độ chính xác tổng quát của thuật toán (Accuracy)

- Khái niệm: Accuracy (độ chính xác) là một thước đo đánh giá hiệu suất của một mô hình phân loại. Accuracy cho biết tỷ lệ phần trăm các dự đoán đúng của mô hình so với tổng số các dự đoán. Nói cách khác, nó là tỷ lệ giữa số lượng dự đoán đúng (bao gồm cả True Positives và True Negatives) trên tổng số các trường hợp được đánh giá.

- Công thức tính:

=

A blue rectangular object with white text

Description automatically generated

Hình 4. 2 Đánh giá Accuracy

- Từ kết quả trên ta thu được độ chính xác tổng quát của 2 thuật toán tương ứng:

* K-Nearest Neighbor: khoảng 86.38%
* Support Vector Machine: khoảng 72.88%

- Nhận xét:

* **K-Nearest Neighbor (KNN):** Đạt được độ chính xác cao hơn trong bài toán này. Điều này có thể cho thấy rằng KNN phù hợp hơn với đặc điểm của tập dữ liệu được sử dụng, hoặc có thể do việc lựa chọn số lượng láng giềng k hợp lý, dẫn đến khả năng phân loại chính xác hơn.
* **Support Vector Machine (SVM):** Độ chính xác thấp hơn KNN trong trường hợp này. Mặc dù SVM thường mạnh mẽ trong việc tìm siêu phẳng tối ưu để phân tách các lớp, có thể có các yếu tố ảnh hưởng như việc lựa chọn tham số không tối ưu, hoặc dữ liệu có thể không phân lớp rõ ràng.

## 4.3 Đánh giá về độ chính xác của thuật toán (Precision)

- Khái niệm: Precision đo lường tỷ lệ các mẫu được dự đoán là dương tính (positive) bởi mô hình mà thực sự là dương tính. Nó cho biết mô hình chính xác đến mức nào khi dự đoán một mẫu là thuộc lớp dương tính.

- Công thức tính:

Precision =

Trong đó:

* TP (True Positives) là số lượng các mẫu mà mô hình dự đoán là dương tính và thực sự là dương tính.
* FP (False Positives) là số lượng các mẫu mà mô hình dự đoán là dương tính nhưng thực sự là âm tính.

### 4.3.1 So sánh độ chính xác của thuật toán trên dữ liệu 0

A blue squares with black numbers

Description automatically generated

Hình 4. 3 Đánh giá Precision trên dữ liệu 0

- Từ kết quả trên ta thu được độ chính xác của 2 thuật toán trên dữ liệu 0 tương ứng:

* K-Nearest Neighbor: khoảng 92.29%
* Support Vector Machine: khoảng 98.74%

- Nhận xét:

* **KNN:**Với độ chính xác cao (86.38%), KNN cho thấy hiệu quả tốt trong việc phân loại tổng thể. Precision cao (92.29%) chỉ ra rằng KNN cũng đáng tin cậy trong các dự đoán positive. KNN là lựa chọn tốt trong các bài toán khi yêu cầu tỷ lệ đúng tổng thể cao và vẫn duy trì được độ chính xác cao trong các dự đoán positive.
* **SVM:** Với độ chính xác thấp hơn (72.88%), SVM không phân loại đúng nhiều trường hợp như KNN.Tuy nhiên, với precision rất cao (98.74%), SVM rất đáng tin cậy trong các dự đoán positive, làm cho nó trở thành lựa chọn tốt trong bài toán khi mà false positives có thể gây ra vấn đề nghiêm trọng.

### 4.3.2 So sánh độ chính xác của thuật toán trên dữ liệu 1

A blue and white rectangular graph

Description automatically generated with medium confidence

Hình 4. 4 Đánh giá Precision trên dữ liệu 1

- Từ kết quả trên ta thu được độ chính xác của 2 thuật toán trên dữ liệu 1 tương ứng:

* K-Nearest Neighbor: khoảng 17.59%
* Support Vector Machine: khoảng 24.58%

- Nhận xét:

* Cả hai thuật toán KNN và SVM đều có precision thấp trên tập dữ liệu 1, với SVM (24.6%) cao hơn KNN (17.5%). Điều này cho thấy cả hai thuật toán đều không thực sự hiệu quả trong việc phân loại chính xác các điểm positive trên tập dữ liệu này.
* SVM có hiệu suất tốt hơn KNN về mặt precision, nhưng sự chênh lệch không đủ lớn để có thể coi là đáng kể. Điều này có thể chỉ ra rằng tập dữ liệu này có đặc điểm khó khăn đối với cả hai thuật toán, hoặc có thể cần phải điều chỉnh thêm các siêu tham số (hyperparameters) hoặc phương pháp tiền xử lý dữ liệu (data preprocessing) để cải thiện hiệu suất.

## 4.4 Đánh giá về độ nhạy của thuật toán (Recall)

- Khái niệm: Recall, còn được gọi là độ nhạy (sensitivity) hoặc tỷ lệ true positive (true positive rate), là một thước đo hiệu suất của một mô hình phân loại, đặc biệt quan trọng trong các bài toán mà việc phát hiện đúng các trường hợp positive là quan trọng. Recall đo lường khả năng của mô hình trong việc nhận diện chính xác tất cả các trường hợp positive. Một recall cao nghĩa là mô hình có khả năng tốt trong việc phát hiện ra các trường hợp positive, giảm thiểu số lượng false negatives (các trường hợp positive bị dự đoán sai là negative).

* Công thức tính:

Recall=

Trong đó:

* **True Positives (TP):** Số lượng các trường hợp positive được dự đoán đúng bởi mô hình.
* **False Negatives (FN):** Số lượng các trường hợp positive bị dự đoán sai là negative bởi mô hình.

### 4.4.1 So sánh độ nhạy của thuật toán trên dữ liệu 0

A blue rectangular object with white text

Description automatically generated

Hình 4. 5 Đánh giá Recall trên dữ liệu 0

* Từ kết quả trên ta thu được độ nhạy của 2 thuật toán trên dữ liệu 0 tương ứng:
* K-Nearest Neighbor: khoảng 92.94%
* Support Vector Machine: khoảng 70.97%
* Nhận xét:

 **K-Nearest Neighbor (KNN):** Thuật toán KNN có recall cao hơn, đạt khoảng 92.9%. Điều này cho thấy KNN có khả năng nhận diện các trường hợp term\_deposit = 0 tốt hơn so với SVM trên dữ liệu này.

 **Support Vector Machine (SVM):** SVM có recall thấp hơn, đạt khoảng 71.0%. Mặc dù độ chính xác của SVM không quá thấp, nhưng nó còn có xu hướng bỏ sót một số trường hợp term\_deposit = 0, khiến cho recall của SVM thấp hơn so với KNN trên dữ liệu này.

### 4.4.2 So sánh độ nhạy của thuật toán trên dữ liệu 1

A graph showing a number of blue squares

Description automatically generated with medium confidence

Hình 4. 6 Đánh giá Recall trên dữ liệu 1

* Từ kết quả trên ta thu được độ nhạy của 2 thuật toán trên dữ liệu 1 tương ứng:
* K-Nearest Neighbor: khoảng 16.25%
* Support Vector Machine: khoảng 91.26%

- Nhận xét:

* **K-Nearest Neighbor (KNN):** Recall trên dữ liệu 1 (16.25%) thấp hơn đáng kể so với recall trên dữ liệu 0 (92.9%). Điều này cho thấy KNN không hoạt động hiệu quả trong việc nhận diện các trường hợp term\_deposit = 1 trên tập dữ liệu này.
* **Support Vector Machine (SVM):** Recall trên dữ liệu 1 (91.25%) cao hơn rất nhiều so với recall trên dữ liệu 0 (71.0%). Điều này cho thấy SVM hoạt động tốt trong việc nhận diện các trường hợp term\_deposit = 1, có sự cải thiện đáng kể so với dữ liệu 0.

## 4.5 Nhận xét các thuật toán

### 4.5.1 K-Nearest Neighbor

- Từ kết quả của các độ đo đánh giá, so sánh các thuật toán ở trên, ta có thể suy ra nhận xét về ưu, nhược điểm của thuật toán KNN trên tập dữ liệu như sau:

- Ưu điểm:

* Đơn giản và dễ hiểu: KNN là một thuật toán không tham số và dễ hiểu. Nó chỉ đơn giản dựa trên khoảng cách đến các điểm lân cận.
* Triển khai trên dữ liệu lớn: Thuật toán KNN tương đối đơn giản nên có thể dễ dàng song song hóa giải thuật bằng mô hình Map Reduce trên Apache Spark.
* Nhược điểm:
* Thời gian chạy: KNN có thời gian chạy quá lâu lên tới hơn 1 giờ. Điều này là do KNN phải tính khoảng cách đến tất cả các điểm dữ liệu trong tập huấn luyện mỗi khi dự đoán với độ phức tạp O(n x d), khiến nó không hiệu quả trên các tập dữ liệu lớn.
* Tiêu tốn tài nguyên:Để tính toán nhanh khoảng cách, mô hình k-NN thường có thể yêu cầu lưu trữ dữ liệu huấn luyện trong bộ nhớ. Khi dữ liệu huấn luyện quá lớn, nó có thể dễ dàng vượt quá bộ nhớ RAM hiện có.
* Độ chính xác chưa quá cao: Chỉ đạt độ chính xác của mô hình trên tập dữ liệu kiểm tra là khoảng 86%, chứng tỏ mô hình chưa quá đáng tin cậy.
* Nhạy cảm với nhiễu: KNN dễ bị ảnh hưởng bởi nhiễu trong dữ liệu vì nó xét các điểm gần kề, bao gồm cả các điểm nhiễu.

### 4.5.3 Support Vector Machine

- Từ kết quả của các độ đo đánh giá, so sánh các thuật toán ở trên, ta có thể suy ra nhận xét về ưu, nhược điểm của thuật toán SVM trên tập dữ liệu như sau:

- Ưu điểm:

* Đơn giản và dễ hiểu: Mô hình SVM tuyến tính sử dụng SGD để tối ưu hóa trọng số là một phương pháp đơn giản và dễ hiểu.
* Khả năng song song hóa: Mô hình này có thể dễ dàng triển khai trên các hệ thống phân tán như Apache Spark, giúp tận dụng tài nguyên của các cụm máy tính để tăng tốc độ huấn luyện.
* Tính tổng quát hóa cao: SVM với biên phân cách tối ưu có khả năng tổng quát hóa tốt trên các tập dữ liệu chưa được nhìn thấy.

- Nhược điểm:

* **Thời gian chạy dài**: Quá trình huấn luyện SVM với phương pháp SGD trên tập dữ liệu lớn có thể mất nhiều thời gian, đặc biệt khi số lượng iteration lớn.
* Tiêu tốn tài nguyên: Mặc dù sử dụng SGD giúp giảm thiểu bộ nhớ cần thiết so với các phương pháp tối ưu hóa khác, nhưng với tập dữ liệu quá lớn, yêu cầu về bộ nhớ và tài nguyên tính toán vẫn cao.
* Độ chính xác chưa cao: Độ chính xác của mô hình SVM trên tập dữ liệu kiểm tra chỉ đạt khoảng 73%, điều này cho thấy mô hình có thể chưa đủ tốt cho những bài toán yêu cầu độ chính xác cao.
* Nhạy cảm với tham số: Hiệu suất của mô hình phụ thuộc vào các tham số như learning rate (alpha) và regularization strength, yêu cầu phải tinh chỉnh cẩn thận để đạt kết quả tốt nhất.
* Cập nhật chậm khi gradient nhỏ: Khi gradient rất nhỏ, quá trình cập nhật trọng số trở nên chậm, điều này có thể dẫn đến thời gian huấn luyện kéo dài và khó đạt được sự hội tụ.

# CHƯƠNG 5: TỔNG KẾT

## 5.1 Kết quả đạt được

- Về cơ bản, trong đồ án lần này, nhóm chúng em đã triển khai được toàn bộ quá trình của một đồ án khai thác dữ liệu. Đi từ những bước cơ bản nhất như mô tả được dữ liệu cũng như mô tả được bài toán mà đồ án cần giải quyết được. Tiếp theo là biết cách sử dụng các công cụ, thư viện của Apache Spark để trực quan hóa dữ liệu trước tiền xử lý, giúp chúng em có được cái nhìn khái quát về tập dữ liệu trước khi đi sâu bên trong.

- Hiểu rõ quy trình tiền xử lý dữ liệu trên Apache Spark không sử dụng thư viện xử lý dữ liệu tập trung (Pandas) bao gồm các bước được triển khai trong chương 2:

* Loại bỏ thuộc tính không cần thiết.
* Nhận diện và xử lý dữ liệu bị nhiễu
* Xử lý ý nghĩa dữ liệu
* Chuẩn hóa dữ liệu
* Tách dữ liệu huấn luyện và kiểm thử.

- Từ mục tiêu của bài toán dự đoán khả năng khách hàng đăng ký gửi tiền có kỳ hạn, chúng em đã đề xuất 3 thuật toán phân lớp từ trong Machine Learning được triển khai trên dữ liệu lớn. Quá trình triển khai các thuật toán được đi từ việc học tập, hiểu các khái niệm, đặc điểm của mỗi thuật toán cũng như công thức tính toán cơ bản của thuật toán trên môi trường tập trung. Đến việc tìm kiểm và song song hóa thuật toán bằng mô hình Map Reduce để vận hành xử lý dữ liệu lớn:

* K-Nearest Neighbor
* Support Vector Machine

- Kết hợp với việc triển khai các thuật toán, nhóm em đã đưa ra các độ đo để đánh giá, so sánh cũng như rút ra nhận xét về ưu, nhược điểm của các thuật toán trên tập dữ liệu:

* Đánh giá về thời gian chạy thuật toán
* Đánh giá về độ chính xác tổng quát của thuật toán (Accuracy)
* Đánh giá về độ chính xác của thuật toán (Precision)
* Đánh giá về độ nhạy của thuật toán (Recall)

## 5.2 Hướng phát triển

- Từ kết quả đạt được, về cơ bản nhóm em đã đưa ra được các thuật toán phân lớp để dự đoán khả năng gửi tiền có kì hạn của khách hàng trên tập dữ liệu đơn giản. Tuy nhiên vì thời gian học tập và nghiên cứu hạn chế nên nhóm em chỉ triển khai các thuật toán tổng quát và tương đối đơn giản để phục vụ đồ án.

- Để tiếp tục phát triển đồ án này, một số hướng nghiên cứu và cải tiến có thể được thực hiện trong tương lai:

* Cải thiện KNN: Áp dụng các kỹ thuật giảm nhiễu và chọn lựa số lượng lân cận (k) phù hợp để giảm thiểu overfitting và cải thiện khả năng tổng quát hóa của KNN.
* Tăng cường dữ liệu (Data Augmentation): Sử dụng các phương pháp tăng cường dữ liệu để cải thiện khả năng học của các mô hình, đặc biệt là các mô hình phức tạp như SVM.
* Kết hợp nhiều mô hình (Ensemble Methods): Sử dụng các phương pháp kết hợp nhiều mô hình như boosting, bagging hoặc stacking để cải thiện hiệu suất và độ chính xác của các mô hình.
* Đánh giá trên nhiều tập dữ liệu: Kiểm tra và đánh giá các mô hình trên nhiều tập dữ liệu khác nhau để đảm bảo tính tổng quát hóa và hiệu suất ổn định của các mô hình.
* Triển khai hệ thống thực tế: Xây dựng một hệ thống dự đoán thực tế dựa trên mô hình đã huấn luyện và kiểm định, đồng thời theo dõi và cập nhật mô hình dựa trên dữ liệu mới thu thập được.

# BẢNG PHÂN CHIA CÔNG VIỆC

|  |  |  |  |
| --- | --- | --- | --- |
| **Nội Dung** | **Nguyễn Hoàng Việt** | **Nguyễn Hoàng Phúc** | **Trần Lê Tứ** |
| Chương 1.1 + 1.2 + 1.4 |  |  | X |
| Chương 1.3.1 + 1.3.2 |  | X |  |
| Chương 1.3.3 | X |  |  |
| Chương 2 |  | X |  |
| Chương 3.1 | X |  |  |
| Chương 3.2 |  | X |  |
| Chương 3.3 |  |  | X |
| Chương 4.1+4.2 | X |  |  |
| Chương 4.3+4.4 |  |  | X |
| Chương 4.5 | X | X | X |
| Chương 5 | X | X | X |
| **Hoàn thành** | **100%** | **100%** | **100%** |

# DANH MỤC TÀI LIỆU THAM KHẢO

Huu, T. V. (2017, 1 8). *Bài 6: K-nearest neighbors.* Được truy lục từ https://machinelearningcoban.com/2017/01/08/knn/

Illuri, L. T. (2023, 9 26). *Understanding Weighted k-Nearest Neighbors (k-NN) Algorithm.* Được truy lục từ https://medium.com/@lakshmiteja.ip/understanding-weighted-k-nearest-neighbors-k-nn-algorithm-3485001611ce

Jesus Maillo, Isaac Triguero, Francisco Herrera. (2015). A MapReduce-based k-Nearest Neighbor Approach for Big Data Classification. *2015 IEEE Trustcom/BigDataSE/ISPA*, 1-6.

Vu, T. (2021). *Xử lý các giá trị ngoại lệ¶.* Đã truy lục 4 28, 2024, từ https://machinelearningcoban.com/tabml\_book/ch\_data\_processing/process\_outliers.html

Anushree Priyadarshini, Dr.Sonali Agarwal. (2015). A Map Reduce based Support Vector Machine for Big Data Classification. International Journal of Database Theory and Application Vol.8, No.5 (2015), pp.77-98.