**PHẦN 1: TỔNG QUAN**

**I. Giới thiệu vấn đề**

*1. Tổng quan về Big Data*

Dữ liệu lớn (Big Data) là thuật ngữ dùng để miêu tả tập hợp các tập dữ liệu cực kỳ lớn và phức tạp mà các công cụ truyền thống xử lý và quản lý dữ liệu không thể xử lý một cách hiệu quả.

Có rất nhiều định nghĩa khác nhau về dữ liệu lớn, từ định nghĩa truyền thống Big Data = 3Vs (Volume, Variety, Velocity). Theo thời gian đặc tính của Big Data cũng được bổ sung nhiều chữ V hơn mà hiện nay được hiểu chủ yếu theo 6Vs với đặc điểm như sau:

- Volume (khối lượng): khối lượng dữ liệu được tạo ra, lưu trữ và xử lý cực lớn

(hàng trăm ngàn Terabyte, thậm chí hiện nay đã phải dùng đến zettabyte (ZB) để

đo lường). Và sau này có thể tăng lên còn hơn thế nữa, thậm chí không thể tính

bằng đơn vị. Điều này đặt ra thách thức về việc lưu trữ và quản lý dữ liệu một cách

hiệu quả.

**-** Velocity (Tốc độ sinh): Dữ liệu lớn thường được tạo ra và lưu lại với tốc độ

nhanh chóng. Điển hình hình đó chính là sự phát triển của Internet of Things

(IoT) dẫn đến việc tốc độ truyền dữ liệu càng nhanh hơn.

**-** Variety (Độ đa dạng): Dữ liệu lớn không chỉ bao gồm dữ liệu có cấu trúc

(structure) dữ liệu trong cơ sở dữ liệu truyền thống, mà còn bao gồm dữ liệu

phi cấu trúc (unstructured) như văn bản (text), ảnh (pictures), video, audio, …

hay thậm chí cả dữ liệu bán cấu trúc (semi-structure) như file json hay file

xml.

**-** Veracity (Độ tin cậy): Dữ liệu các nguồn khác nhau có độ tin cậy khác nhau.

**-** Value (Giá trị): Mục tiêu chính của việc xử lý dữ liệu lớn là tạo ra giá trị từ

dữ liệu vì vậy cần xác định giá trị mà dữ liệu mang lại đến như thế nào từ đó

mới xác định được có nên triển khai dữ liệu hay không

**-** Variability (Tính biến đổi): Đề cập đến sự không nhất quán có thể được hiển

thị bởi dữ liệu, do đó có thể cản trở quá trình có thể xử lý và quản lý dữ liệu

một cách hiệu quả.

*2. Phân tích cảm xúc*

Phân tích cảm xúc (Sentiment Analysis) là lĩnh vực đã và đang thu hút được sự quan tâm của cộng đồng các nhà nghiên cứu cũng như các nhà phát triển các ứng dụng trong công nghiệp. Trong những năm gần đây, do sự phát bùng nổ lượng dữ liệu đánh giá của người dùng trên các trang mạng xã hội, các diễn đàn, các trang đánh giá sản phẩm, việc phát triển các phương pháp mới và công cụ nhằm phân tích và rút trích ra quan điểm giúp có thể hiểu được xu thế mọi người đang bình luận hay đánh giá về một thực thể mục tiêu. Kết quả của những nghiên cứu này là hữu ích cho các cá nhân và doanh nghiệp khi họ cần tham khảo các thông tin đánh giá về thực thể mục tiêu mà họ quan tâm.

Đã có nhiều tiếp cận nghiên cứu khác nhau được đề xuất cho các nhiệm vụ trong phân tích quan điểm. Các tiếp cận này thường dựa trên việc trích chọn các đặc trưng thể hiện quan điểm, nhận xét, đánh giá, tình cảm hay cảm xúc người dùng về thực thể được đánh giá cho các bài toán mục tiêu.

*3. Phân loại dữ liệu văn bản*

Phân loại văn bản (Text classification) là quá trình xác định và gán một hoặc

nhiều nhãn (labels) cho các mẫu văn bản dựa trên nội dung của chúng. Mục tiêu của

text classification là tự động phân loại các đoạn văn bản vào các danh mục cụ thể,

giúp tổ chức và hiểu được thông tin từ các nguồn văn bản lớn. Text classification có

ý nghĩa quan trọng trong nhiều lĩnh vực và ứng dụng khác nhau. Dưới đây là một số

ý nghĩa chính của text classification:

- Tổ chức thông tin.

- Phân tích ý kiến và đánh giá.

- Phân loại nội dung.

- Phát hiện tin tức giả và thư rác.

- Hỗ trợ ra quyết định.

**II. Mục tiêu nghiên cứu**

Mục tiêu nghiên cứu của đề tài là ứng dụng Sentiment Analysis và các phương

pháp máy học và xử lý ngôn ngữ vào bộ dữ liệu để tìm hiểu và phân tích ý kiến, cảm

xúc của du khách đối với công viên. Cuối cùng dựa vào kết quả có được để sử dụng

mô hình để phân lớp dữ liệu và đề ra hướng phát triển cho đề tài.

**III. Phương pháp nghiên cứu**

Trước hết, nhóm tập trung áp dụng các phương pháp nghiên cứu để khám phá và

hiểu sâu hơn về bộ dữ liệu “Banking review”. Sau đó tiền xử lý sao cho phù hợp

và thuận lợi cho việc sử dụng các mô hình phân lớp và học máy

Tiếp theo nhóm tiến hành Sentiment Analysis cho bộ dữ liệu bằng cách sử dụng

những phương pháp như Vader, NLTK và Wordcloud để phân tích những

thông tin chi tiết về những bình luận từ bộ dữ liệu.

Thực hiện xây dựng các mô hình phân lớp văn bản, mô hình học máy dựa trên

so sánh giữa các mô hình để tìm ra mô hình tốt nhất cho việc

áp dụng vào bộ dữ liệu mà nhóm tự đào để phân tích dự đoán cảm xúc

Cuối cùng, nhóm sẽ tóm tắt lại những kết quả nghiên cứu để đánh giá và kết luận.

Từ đó, chỉ ra những hạn chế mà nghiên cứu gặp phải và đề ra hướng phát triển để cải

thiện đề tài

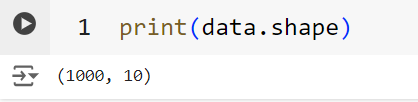
IV. Dữ liệu đề tài

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả thuộc tính |
| 1 | author | Người dùng đã viết đánh giá, cung cấp thông tin giá trị về danh tính và quan điểm của người đánh giá. |
| 2 | date | Ngày mà đánh giá được gửi, |
| 3 | address | Vị trí từ nơi đánh giá được viết |
| 4 | bank | Tên của ngân hàng được đánh giá |
| 5 | rating | Đánh giá bằng số của người dùng về dịch vụ của ngân hàng, cho thấy sự hài lòng của người dùng |
| 6 | review\_title\_ by\_user | Tiêu đề do người dùng chỉ định cho đánh giá của họ |
| 7 | review | Nội dung chi tiết của đánh giá của người dùng về ngân hàng |
| 8 | bank\_image | URL dẫn đến logo hoặc hình ảnh của ngân hàng liên quan đến đánh giá |
| 9 | rating\_title\_by\_user | Tiêu đề do người dùng chỉ định cho đánh giá của họ |
| 10 | useful\_count | Số lượng người dùng cho rằng đánh giá hữu ích |

**PHẦN 2: XỬ LÝ VÀ PHÂN TÍCH DỮ LIỆU**

1. **Tiền xử lý dữ liệu**
2. **Tổng quan về dữ liệu**
3. **Số chiều dủa dữ liệu**

Ta dùng phương thức **shape** để in ra số chiều của dữ liệu



Hình 1 : Kết quả in ra số chiều của dữ liệu

Dữ liệu gồm 1000 mẫu thu thập từ người dùng và 10 cột đặc tính.

1. In ra các dòng đầu tiên của dữ liệu

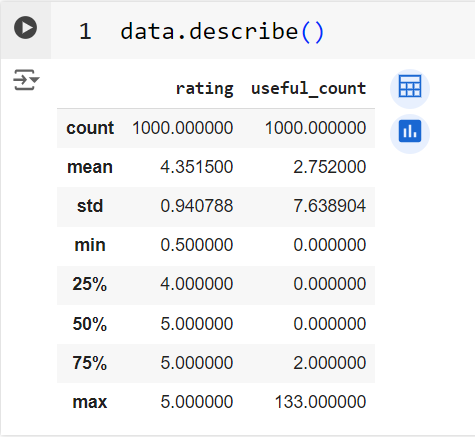
Để in ra 5 dòng đầu tiên về dữ liệu, ta sử dụng phương thức **head()**



Hình 2: Kết quả in ra 5 dòng đầu tiên về dữ liệu

1. Các chỉ số thống kê của dữ liệu

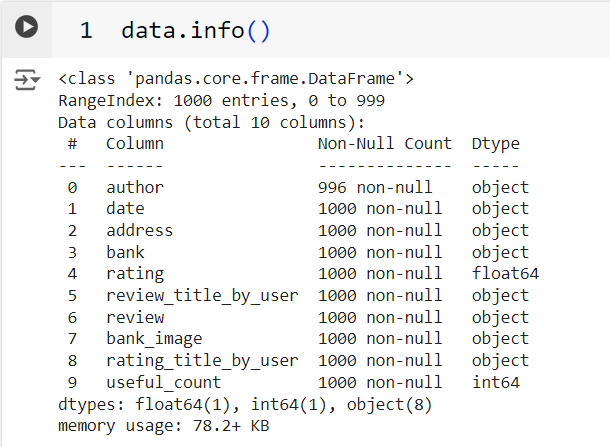
Để in ra các chỉ số thống kê như trung bình, trung vị,… ta dùng phương thức describe()



Hình 3 : Kết quả in ra các chỉ số thống kê của dữ liệu

1. Kiểm tra thuộc tính và số lượng dữ liệu bị thiếu

Để kiểm tra thuộc tính và số lượng dữ liệu bị thiếu ta dùng phương thức info()



Hình 4 : Kết quả in ra thuộc tính và số lượng mất mát của dữ liệu

1. Phân loại biến

Các biến liên tục bao gồm:

A close up of a computer screen

Description automatically generated

Hình 5 : Kết quả in ra các biến liên tục của dữ liệu

Các biến phân loại gồm:

A screenshot of a computer code

Description automatically generated

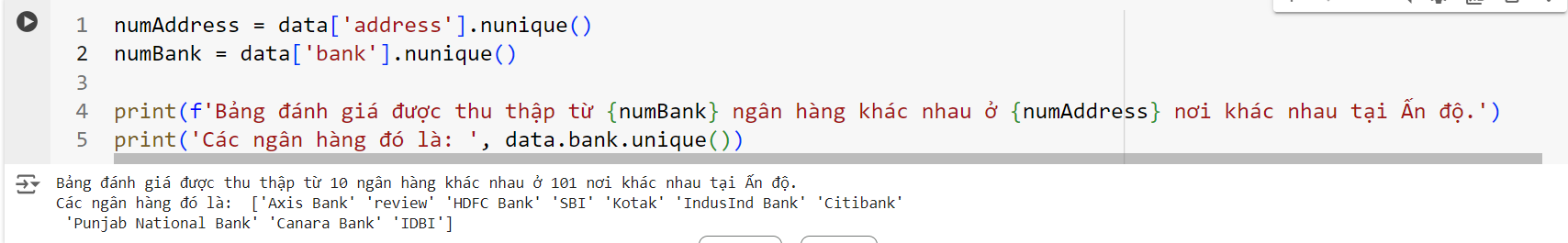
Hình 6 : Kết quả in ra các biến phân loại của dữ liệu

1. Tiền xử lí
2. Loại bỏ các cột không đóng góp vào quá trình phân tích

Ta có thể bỏ các cột *[, 'image']* vì chúng không có ý nghĩa để phân tích

1. Kiểm tra các dữ liệu duy nhất (unique)

Kiểm tra dữ liệu thu thập ở ngân hàng nào và ở đâu



Hình 7 : Kết quả in ra các dữ liệu khác nhau của dữ liệu

1. Chỉnh sửa các dữ liệu không hợp lí

Một số dữ liệu trong cột “bank” có dạng “review” không mang ý nghĩa xác định ngân hàng nào nên ta chuyển đổi những dữ liệu này thành “Unknow”



Hình 8 : Đoạn mã dùng để chỉnh sửa cột “bank”

1. Gán nhãn số cho biến “Rating”

Để thực hiện quá trình gán nhãn dữ liệu trước khi đưa vào huấn luyện, nghiên

cứu áp dụng phương pháp phân loại cảm xúc theo điểm số đánh giá (Rating) của người dùng, để phân chia tập dữ liệu đã thu thập được thành 3 bộ dữ liệu được gán nhãn theo quy tắc sau:

* Rating <=3 : bình luận nào đánh giá dưới 3 sao sẽ được dán nhãn là tiêu cực (Bad).
* Rating >=4: bình luận nào đánh giá trên 4 sao sẽ được dán nhãn là tích cực (Good).
* Còn lại: Neural

A computer code with text

Description automatically generated with medium confidence

Hình 9: Đoạn mã dùng để gán nhãn biến “Rating”

1. Dùng biểu thức chính quy để loại bỏ các kí tự đặc biệt



Hình 10: Đoạn mã dùng để gán nhãn biến “Rating

* **[^a-zA-Z0-9\s]** là một biểu thức chính quy để tìm tất cả các ký tự không phải là chữ cái (a-z, A-Z), số (0-9), hoặc khoảng trắng (space).
* **str.replace()** dùng thay thế tất cả các ký tự tìm được bằng chuỗi rỗng (''), tức là loại bỏ những ký tự này khỏi chuỗi.

1. Khám phá dữ liệu
2. Những ngân hàng thu thập nhiều dữ liệu nhất

Tiến hành in ra 10 ngân hàng bằng biểu đồ grid và pie

A graph with red and yellow bars

Description automatically generated

Hình 10: Biểu đồ 10 ngân hàng thu thập nhiều dữ liệu nhất

* Bangalore có số lượng đánh giá cao nhất với 245 đánh giá.
* Chennai xếp thứ hai với 172 đánh giá.
* Hyderabad đứng thứ ba với 127 đánh giá.
* Jaipur và Kochi có số lượng đánh giá thấp nhất, mỗi nơi chỉ có 13 đánh giá.

Các phần của biểu đồ tròn được phân chia theo tỷ lệ:

* Bangalore chiếm tỷ trọng lớn nhất với **32.11%** trong tổng số đánh giá.
* Chennai chiếm **22.54%**.
* Hyderabad chiếm **16.64%**.
* Các thành phố còn lại như Kolkata, Pune, Mumbai, New Delhi, Coimbatore, Jaipur và Kochi đóng góp phần trăm nhỏ hơn, với Jaipur và Kochi đóng góp ít nhất (1.7%).

1. Ngân hàng có số lượt đánh giá cao nhất

A graph with different colored bars

Description automatically generated

Hình 11: Biểu đồ 10 ngân hàng có số lượt đánh giá cao nhất

* **Unknown (không xác định)** có số lượng đánh giá cao nhất với 285 đánh giá.
* **SBI (State Bank of India)** đứng thứ hai với 219 đánh giá.
* **HDFC Bank** đứng thứ ba với 188 đánh giá.
* Các ngân hàng có số lượng đánh giá ít nhất là **Punjab National Bank**, **Citibank**, và **IDBI**, mỗi ngân hàng chỉ có 14 đánh giá.

1. Thống kê về phân phối của biến Rating

A graph with blue squares

Description automatically generated

Hình 11: Biểu đồ về phân phối của biến Rating

* Mức điểm 5.0 có số lượng đánh giá cao nhất với hơn 500 đánh giá.
* Mức điểm 4.0 cũng có lượng lớn với khoảng 300 đánh giá.
* Mức điểm từ 2.0 đến 3.5 có ít đánh giá hơn, nhưng vẫn có một số lượng nhất định, dao động trong khoảng từ 20 đến 100 đánh giá.
* Các mức điểm rất thấp như 0.5, 1.0, 1.5 gần như không có đánh giá nào hoặc rất ít, dưới 20 đánh giá cho mỗi mức.

Ý nghĩa:

Hầu hết các đánh giá tập trung ở mức điểm 5.0 và 4.0, cho thấy rằng phần lớn người dùng có trải nghiệm tích cực với các dịch vụ của ngân hàng.

1. Phân phối của biến **Userful Count**

A graph of a number of objects

Description automatically generated

Hình 12: Biểu đồ về phân phối của biến Userful Count

1. **Sentiment Analysis**

Sentiment Analysis về cơ bản đây là quá trình ứng dụng của trí tuệ nhân tạo, sử

dụng các thuật toán phức tạp để xử lý ngôn ngữ tự nhiên của con người (NLP). Qua

đó xác định được cảm xúc của du khách mà trong bài này nhóm sử dụng 3 loại là

Positive, Negative và Neutral.

Để có thể có nhận định sâu hơn về reviews của khách hàng, nhóm đã dùng

phương pháp Word Cloud,Valder

1. Word Cloud

Nhóm đã dùng thư viện WordCloud để làm rõ những từ ngữ xuất hiện nhiều lần

nhất trong bộ dữ liệu. Những từ có kích thước lớn hơn sẽ xuất hiện nhiều lần hơn và

ngược lại.

A close up of words

Description automatically generated

Hình 13: Wordcloud của biến Reviews

Ngoài ra , khi điểm score < 0 , ta phân loại vào nhóm 'To be Improved'

Đoạn mã là dưới đây tạo hình ảnh Word Cloud cho các đánh giá được phân loại trong danh mục **"** 'To be Improved' từ dữ liệu ban đầu

A screen shot of a computer program

Description automatically generated

Hình 14: Đoạn mã vẽ hình wordcloud\_tobeImproved

A close up of words

Description automatically generated

Hình 14: Wordcloud của UserReviewTitle

1. VADER

Valence Aware Dictionary và Sentiment Reasoner (VADER) là mô hình dựa trên

quy tắc từ vựng để phân tích tình cảm. Kết quả do VADER tạo ra gồm một từ điển gồm 4 khóa: neg, neu, pos và compound. Trong đó, neg, neu và pos lần lượt có nghĩa là tiêu cực, trung tính và tích cực.

Tổng của chúng phải bằng 1 hoặc gần bằng 1.

Compound tương ứng với tổng điểm của mỗi từ vựng và nằm trong đoạn [-1;1] với 1 là tích cực nhất và -1 là tiêu cực nhất.

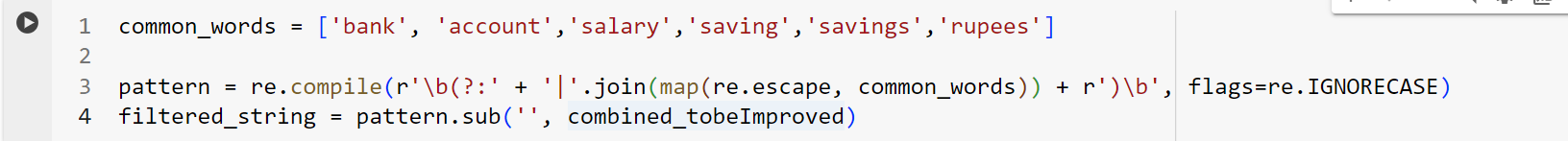
Chúng ta cùng phân tích sentiment tiêu cực

A close up of words

Description automatically generated

Hình 15: Wordcloud của UserReviewTitle với Sentiment tiêu cực

Lọc bỏ các từ thông dụng (common words) khỏi chuỗi văn bản



Hình 16: Đoạn mã dùng loại bỏ các từ thông dụng

Giải thích:

**r'\b(?:' + '|'.join(map(re.escape, common\_words)) + r')\b'** gồm:

* **\b**: Đây là ký tự đặc biệt trong biểu thức chính quy để chỉ **ranh giới từ** (word boundary). Nó đảm bảo rằng chỉ những từ nguyên vẹn mới được tìm thấy và thay thế. Ví dụ, nếu ta tìm từ "bank", nó sẽ không khớp với từ "banking" hoặc "banker".
* **(?: ... )**: Đây là một nhóm không bắt (non-capturing group), nghĩa là nó sẽ nhóm các từ bên trong nhưng không giữ lại chúng cho bất kỳ mục đích nào ngoài việc so khớp.
* **'|'.join(map(re.escape, common\_words))**: Dùng map() để áp dụng re.escape() lên từng từ trong danh sách common\_words. re.escape() giúp đảm bảo các từ trong danh sách (ví dụ như "rupees") được xử lý chính xác ngay cả khi chúng chứa các ký tự đặc biệt.
  + **'|'.join(...)**: Sau đó, các từ được nối với nhau bằng ký tự **|**, tạo thành một biểu thức chính quy. Ví dụ, sau khi nối, nó sẽ trông như: 'bank|account|salary|saving|savings|rupees'.
* **flags=re.IGNORECASE**: Sử dụng cờ IGNORECASE để làm cho việc tìm kiếm **không phân biệt chữ hoa và chữ thường**. Điều này có nghĩa là từ "Bank", "bank", và "BANK" đều như nhau.



Hình 17: Wordcloud của UserReviewTitle với Sentiment tiêu cực sau khi loại bỏ từ thông dụng

**PHẦN III**. XÂY DỰNG MÔ HÌNH

1. **Tạo tập dữ liệu huấn luyện và kiểm tra**

Trước khi chạy các mô hình phân lớp, nhóm sẽ tiến hành phân bộ dữ liệu thành

train và test set độc lập với nhau .

Training set sẽ dùng để huấn luyện mô hình, và testing set sẽ dùng để kiểm thử, đánh giá độ chính xác của mô hình phân lớp đã được huấn luyện.

Ta dùng biến đầu vào là các biến

| **author** | **date** | **address** | **bank** | **review\_title\_by\_user** | **review** | **rating\_title\_by\_user** | **useful\_count** | **polarity** | **compound** | **Sentiment\_Category** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Bảng 1: Tổng hợp các biến đầu vào

Và biến mục tiêu là biến **rating** nhằm phân lớp xem rating sẽ nằm ở lớp nào từ đó đưa ra nhận xét độ hài lòng của khách hàng là bao nhiêu

A screenshot of a computer

Description automatically generated

Hình 18: Đoạn mã chia dữ liệu ra 2 tập train và test và in ra 5 dòng đầu của tập train

Ta chia tỉ lệ giữa train và test là 75% cho train và 25% cho test, đây là tỉ lệ phù hợp để tranh tình trạng overfitting hay underfitting của các mô hình học máy

1. Chuyển đổi các giá trị của cột phân loại sang dạng số
2. Label Encoder ( Mã hóa nhãn)

Việc mã hóa dữ liệu chủ yếu được sử dụng trước khi dữ liệu có thể được đưa vào các thuật toán học máy. Đây là bước tiền xử lý quan trọng trong một dự án học máy.\

Có nhiều kỹ thuật mã hóa khác nhau và được sử dụng rộng rãi như:

* OneHot Encoder
* Label Encoder
* get\_dummies.

Trong khuôn khổ bài này, nhóm em sử dụng Label Encoder cho việc mã hóa dữ liệu

Ta xem xét ví dụ dưới đây để hiểu cách áp dụng Label Encoder

Giả sử chúng ta có một cột màu (color) trong một số tập dữ liệu có các phần tử là Đỏ (Red), Green( Xanh lá cây) và Blue (Xanh da trời). Để chuyển đổi cột danh mục này thành một cột số, chúng ta sẽ áp dụng mã hóa nhãn cho cột này. Sau khi áp dụng mã hóa nhãn, cột Chiều cao được chuyển đổi thành một cột số có các phần tử 0, 1 và 2 trong đó 0 là nhãn cho Red, 1 là nhãn cho Green và 2 là nhãn cho chiều cao Blue.

A yellow arrow pointing to a yellow rectangular object

Description automatically generated

Hình 19: Quy trình mã hóa biến dùng Label Encoder

Ta áp dụng lên 2 tập dữ liệu là Train và Test ở các cột dữ liệu phân loại

Đối với tập Train sau khi mã hóa các cột dữ liệu phân loại

A screenshot of a computer

Description automatically generated

Hình 20: Đoạn mã dùng mã hóa các biến trong tập Train

Đối với tập Test sau khi mã hóa các cột dữ liệu phân loại

A screenshot of a computer

Description automatically generated

Hình 20: Đoạn mã dùng mã hóa các biến trong tập Test

1. Chuẩn hóa dữ liệu
   1. Vì sao cần chuẩn hóa dữ liệu

Các trọng số của mô hình được khởi tạo từ các giá trị ngẫu nhiên nhỏ và được cập nhật bằng thuật toán tối ưu trong quá trình backward, việc cập nhật dựa trên lỗi dự đoán (loss) trong quá trình huấn luyện.

Vì các trọng số nhỏ của mô hình nhỏ và được cập nhật dựa vào lỗi dự đoán nên việc scale giá trị của đầu vào X và đầu ra Y của tập dữ liệu huấn luyện là một yếu tố quan trọng. Nếu đầu vào không được scaling có thể dẫn đến quá trình huấn luyện không ổn định. Ngoài ra nếu đầu ra Y không được scale trong các bài toán regression có thể dẫn đến exploding gradient khiến thuật toán không chạy được.

Scaling có thể tạo ra sự khác biệt giữa một mô hình kém và một mô hình tốt.

Bước tiền xử lý dữ liệu liên quan đến kỹ thuật normalization và standardization để rescale lại input và output trước khi huấn luyện mô hình.

1. Scale các biến đầu vào

Các biến đầu vào là các biến đưa vào mạng neuron để dự đoán.

Một nguyên tắc chung là các biến đầu vào phải có giá trị nhỏ, có thể nằm trong khoảng [0:1] hoặc được chuẩn hóa với giá trị trung bình bằng 0 và độ lệch chuẩn (standard deviation) bằng 1. Các biến đầu vào có cần phải scaling hay không phụ thuộc vào từng bài toán cụ thể và từng biến cụ thể.

Nếu phân bố các giá trị của biến là phân bố chuẩn thì biến nên được standardization, nếu không dữ liệu nên được normalization. Điều này áp dụng khi phạm vi giá trị lớn (10, 100…) hoặc nhỏ (0.01, 0.0001).

Nếu giá trị của biến nhỏ (gần trong khoảng [0:1]) và phân phối bị giới hạn (ví dụ độ lệch chuẩn gần với 1) thì chúng ta không cần phải scale dữ liệu.

Các bài toán có thể phức tạp hoặc không rõ ràng nên ta không xác định được việc sử dụng kỹ thuật nào để scale dữ liệu là tốt nhất. Vì thế nên thường thì mình hay thử nghiệm scale dữ liệu và không scale có khác biệt nhau thế nào bằng việc cho mô hình chạy rồi tiến hành đánh giá.

1. Scale các biến đầu ra

Biến đầu ra Y là biến được dự đoán bởi mô hình.

Chúng ta cần đảm bảo là giá trị của Y phải khớp với phạm vi biểu diễn của hàm kích hoạt (activation function) trong lớp output của mô hình mạng nơ-ron.

Nếu đầu ra của activation function thuộc vào miền [0, 1] thì giá trị biến đầu ra Y cũng phải nằm trong miền giá trị này. Tuy nhiên chúng ta nên chọn hàm kích hoạt phù hợp với phân bố của đầu ra Y hơn là đưa Y về miền giá trị của hàm kích hoạt.

Ví dụ nếu bài toán của bạn là regression thì đầu ra sẽ là một giá trị số thực. Mô hình tốt nhất cho bài toán này đó là lựa chọn hàm kích hoạt tuyến tính (linear activation). Nếu đầu ra có phân bố chuẩn thì chúng ta có thể standardize biến đầu ra. Nếu không thì đầu ra Y có thể được normalize.

Có nhiều cách để chuẩn hóa dữ liệu, như Normalization,Min-max Scaler,StandardScaler, nhóm em chọn StandardScaler để chuẩn hóa dữ liệu cho bài toán này.

* 1. StandardScaler

StandardScaler là việc scale dữ liệu về một phân bố trong đó giá trị trung bình của các quan sát bằng 0 và độ lệch chuẩn = 1. Kỹ thuật này còn được gọi là “whitening.”. Nhờ việc chuẩn hóa, các thuật toán như linear regression, logistic regression được cải thiện.

Công thức chuẩn hóa như sau**:**

**A mathematical equation with black lines

Description automatically generated with medium confidence**

Hình 21: Công thức chuẩn hóa

trong đó μ là giá trị trung bình và σ là độ lệch chuẩn so với giá trị trung bình

StandardScaler làm cho một phân bố có độ lệch chuẩn bằng 1 và phương sai cũng bằng 1, vì phương sai = bình phương độ lệch chuẩn. Nên 1 bình phương = 1.

StandardScaler làm sai lệch khoảng cách tương đối giữa các đặc trưng ( feature)

A diagram of a function

Description automatically generated

Hình 22: Phân phối Gaussian hay Phân phối chuẩn

Quay lại bài toán , ta đưa X\_train vào để chuẩn hóa

A screenshot of a computer code

Description automatically generated

Hình 23: Đoạn mã mã hóa dữ liệu X\_train

Đầu ra của X\_train bây giờ là 1 mảng với các phần tử của mảng tập trung xung quanh giá trị 0

Tiếp theo ta cần giảm chiều dữ liệu xuống để tránh tính toán phức tạp

1. Giảm chiều dữ liệu với PCA

PCA là viết tắt của cụm từ *principal component analysis*. Thuật ngữ Tiếng Việt còn gọi là *phân tích thành phần chính*. Đây là một phương pháp giảm chiều dữ liệu (*dimensionality reduction*) tương đối hiệu quả dựa trên phép phân tích suy biến (*singular decomposition*) mà ở đó chúng ta sẽ chiếu các điểm dữ liệu trong không gian **cao chiều** xuống một số ít những véc tơ thành phần chính trong không gian **thấp chiều** mà đồng thời vẫn bảo toàn tối đa **độ biến động** của dữ liệu sau biến đổi. Ưu điểm của PCA đó là **sử dụng tất cả** các biến đầu vào nên phương pháp này không bỏ sót những biến quan trọng.

**Mục đích của giảm chiều dữ liệu:**

Những bộ dữ liệu lớn thường tiêu tốn nhiều bộ nhớ lưu trữ và thời gian huấn luyện. Do đó khi đối mặt với những bộ dữ liệu kích thước lớn chúng ta thường tìm cách giảm chiều dữ liệu từ không gian cao chiều (*high dimensionality*) xuống không gian thấp chiều (*low dimensionality*) mà vẫn giữ được những đặc trưng chính của dữ liệu nhưng tiết kiệm được chi phí huấn luyện và dự báo.

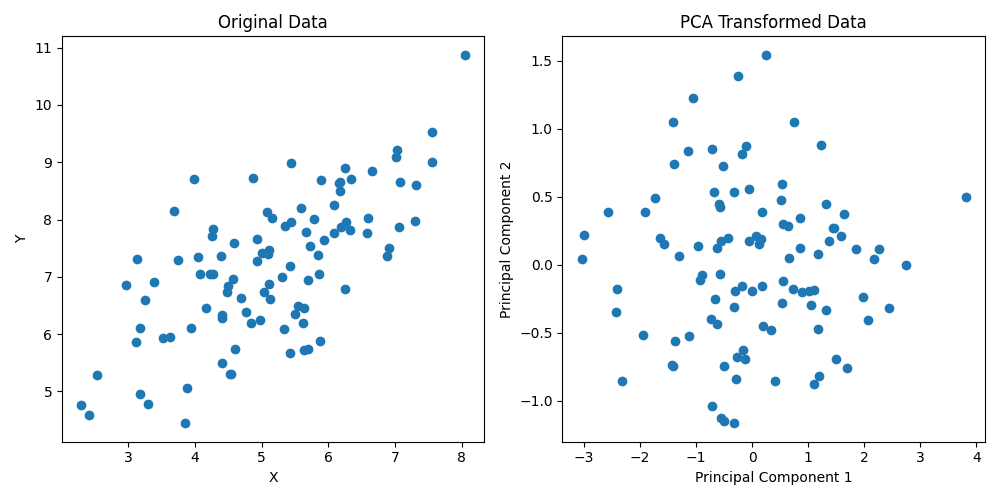
Lấy một ví dụ, giả sử bạn đang cần phân loại tác vụ ảnh với 1000 nhãn mục tiêu và kích thước ảnh đầu vào là 1000x1000x3. Như vậy nếu véc tơ hóa ma trận ảnh ta thu được một véc tơ với kích thước 3 triệu chiều. Để xây dựng một mạng *thần kinh nơ ron nông* (*shallow neural network*)với một layer kết nối toàn bộ 3 triệu điểm ảnh này tới 1000 nhãn mục tiêu sẽ cần số lượng tham số là 3 tỷ. Đây là một mạng nơ ron có kích thước quá lớn và thường vượt quá khả năng tính toán của các máy tính thông thường. Nếu huấn luyện được một mạng nơ ron khổng lồ như vậy thì khả năng mô hình gặp hiện tượng *overfitting* cũng rất cao. Khi đối mặt với tình huống này chúng ta có thể sử dụng các phương pháp giảm chiều dữ liệu để đạt được hiệu quả tính toán và tránh *overfitting*.

Những bộ dữ liệu cao chiều cũng thường xuất hiện trong dữ liệu dạng bảng (*tabular data*). Thông thường chúng ta sẽ không sử dụng toàn bộ các biến đầu vào mà thực hiện xếp hạng mức độ quan trọng của chúng nhằm lọc ra một phần nhỏ các biến được coi là quan trọng nhất và loại bỏ những biến nhiễu.. Một cách khác cũng thường được thực hiện đó là giảm chiều dữ liệu. Phương pháp này không yêu cầu phải loại bỏ bất kì một biến đầu vào nào mà tất cả chúng sẽ được tận dụng nhằm tạo ra những biến được tổ hợp tuyến tính từ chúng.

Trong bài toán phân cụm, các phương pháp giảm chiều dữ liệu có thể biến đổi dữ liệu về không gian hai chiều hoặc ba chiều nhằm biểu diễn dữ liệu một cách trực quan. Thông qua đó phát hiện được những bất thường dữ liệu (*anomaly detection*) và nhận biết phân bố cụm trong những bài toán học không giám sát.

Ví dụ :

Trong ví dụ này, trước tiên nhóm em tạo một số dữ liệu tổng hợp với 100 mẫu và có mối tương quan tích cực với dữ liệu này. Sau đó, nhóm em sử dụng PCA để giảm kích thước xuống còn 2 thành phần. Thuộc tính giải thích\_variance\_ratio\_ cho chúng ta biết tỷ lệ tổng phương sai được giải thích bởi từng thành phần chính được chọn. Cuối cùng, nhóm em trực quan hóa dữ liệu đã rút gọn trong biểu đồ phân tán.



Hình 24: Dữ liệu trước khi dùng PCA ( hình trái )

Dữ liệu sau khi dùng PCA ( hình phải )

Áp dụng vào bài toán của nhóm em bằng cách chỉ giữ lại 8 thành phần chính

A screenshot of a computer code

Description automatically generated

Hình 25: Đoạn mã dùng PCA trên tập X\_train và in ra Variance ratio

Mỗi số trong mảng trên biểu thị tỷ lệ phương sai của dữ liệu gốc được giải thích bởi **thành phần chính (principal component)**

* Thành phần chính đầu tiên giải thích **25.23%** tổng phương sai của dữ liệu.
* Thành phần chính thứ hai giải thích **12.29%** tổng phương sai.
* Thành phần chính thứ ba giải thích **10.33%** phương sai
* …

PCA tìm cách giảm số chiều của dữ liệu bằng cách biến đổi các biến ban đầu thành một tập hợp các thành phần chính không tương quan (orthogonal), sắp xếp chúng sao cho các thành phần đầu tiên giải thích phần lớn phương sai của dữ liệu

Ta vẽ biểu đồ để xác định giữ lại bao nhiêu thành phần chính mà vẫn giữ được 90% tổng phương sai

A blue line with numbers and lines

Description automatically generated with medium confidence*Community Verified icon*

Hình 26: Biểu đồ thể hiện độ giải thích của phương sai so với số chiều của dữ liệu

1. Mô hình hồi quy Logictics
2. Giới thiệu mô hình hồi quy Logictics

Hồi quy logistic là một thuật toán được sử dụng để phân loại các quan sát theo các danh mục rời rạc. Thay vì có đầu ra là các giá trị liên tục như thuật toán hồi quy tuyến tính, hồi quy logistic đưa đầu ra qua hàm logistic sigmoid để trả về một giá trị biểu thị xác suất mà quan sát đầu vào thể hiện tính chất của hai hay nhiều hơn các danh mục rời rạc.

Các loại hồi quy Logictics

* Nhị phân (Đỗ/Trượt)
* Đa lớp (Chó, Mèo, Thỏ, ...)
* Thứ tự (Cao, Trung bình, Thấp)

Trong bài này nhóm em sử dụng hồi quy Logictics đa lớp vì biến đầu ra có một trong 3 lớp : [Bad,Neural,Good]

1. Áp dụng mô hình hồi quy Logictics

A diagram of different types of geometric shapes

Description automatically generated

Hình 27: Phân loại nhị phân ( hình trái ) và phân loại nhiều lớp ( hình phải)

Nhóm em sử dụng thư viện Skicitlearn để hỗ trợ các thư viện học máy bao gồm Logictics Regression để huấn luyện trên tập tập train gồm (X\_train,y\_train) và đánh giá dựa trên X\_test

A close-up of a computer screen

Description automatically generated

Hình 28: Đoạn mã xây dựng mô hình LogicticRegression

1. Đánh giá kết quả

Kết quả đánh giá dựa trên tập X\_test

A screenshot of a computer screen

Description automatically generated

Bảng 2: Kết quả thu được từ mô hình LogicticRegression

Với độ chính xác trung bình là 0.92, ta thấy đây là mô hình tốt để phân loại cho bài toán này. Tuy nhiên ta cần tối ưu hóa siêu tham số để tìm ra bộ siêu tham số tốt nhất

1. Tối ưu siêu tham số dùng grid\_search

A computer screen shot of a computer code

Description automatically generated

Hình 30: Đoạn mã dùng tối ưu hóa tham số mô hình Logictics Regression

Giải thích các siêu tham số:

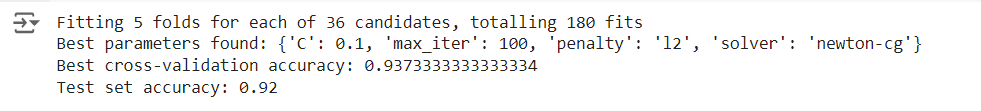
- **C**: Đây là tham số điều chuẩn (regularization parameter). Giá trị nhỏ của C tương ứng với điều chuẩn mạnh hơn, giúp ngăn chặn mô hình quá khớp (overfitting). Trong lưới này, ta thử các giá trị: [0.1, 1, 10, 100].

-**solver**: Đây là thuật toán được sử dụng để tối ưu hóa hàm mất mát. Các lựa chọn bao gồm:

* 'newton-cg': Dựa trên phương pháp Newton.
* 'lbfgs': Một thuật toán gradient giảm dần, phù hợp cho dữ liệu lớn.
* 'liblinear': Một solver phù hợp cho các mô hình nhỏ và xử lý nhanh.

- **penalty**: Hình phạt điều chuẩn, ta lựa chọn 'l2' (điều chuẩn Ridge). Một số solver như newton-cg và lbfgs chỉ hỗ trợ 'l2'.

-**max\_iter**: Số lần lặp tối đa để solver hội tụ, với các giá trị lần lượt là [100, 200, 300]. Giúp kiểm soát việc bao nhiêu vòng lặp cần thiết để tìm ra nghiệm của bài toán.



Hình 31: Kết quả sau khi dùng tối ưu hóa tham số mô hình Logictics Regression

Quá trình tìm kiếm đã chạy 180 lần thử nghiệm, với 5 lần kiểm tra chéo cho mỗi 36 tổ hợp tham số. Các tham số tốt nhất được tìm thấy là

|  |  |  |  |
| --- | --- | --- | --- |
| C | max\_iter | penalty | solver |
| 0.1 | 100 | L2 | newton-cg |

In ra bảng ma trận nhầm lần trước và sau khi tối ưu hóa siêu tham số với mô hình Logictics Regression

A blue square with white text

Description automatically generated

Hình 31: *Confusion matrix của mô hình Logistic Regression*

Ma trận confusion này hiển thị sự so sánh giữa nhãn thực tế (true labels) và nhãn dự đoán (predicted labels) của mô hình Logistic Regression. Mỗi ô trong ma trận biểu thị số lượng mẫu mà mô hình dự đoán đúng hoặc sai cho từng nhãn.

* **Hàng 0 (True Label = 0)**:
  + Dự đoán đúng: 13 mẫu được dự đoán là 0 và đúng là 0 (ô [0,0]).
  + Dự đoán sai: 14 mẫu được dự đoán là 1 nhưng thực tế là 0 (ô [0,1]).
* **Hàng 1 (True Label = 1)**:
  + Dự đoán đúng: 218 mẫu được dự đoán là 1 và đúng là 1 (ô [1,1]).
  + Không có dự đoán sai nhãn cho lớp này.
* **Hàng 2 (True Label = 2)**:
  + Dự đoán sai: 5 mẫu được dự đoán là 1 nhưng thực tế là 2 (ô [2,1]).

Kết luận :

* Mô hình dự đoán tốt cho lớp 1 với 218 dự đoán đúng.
* Mô hình gặp khó khăn khi phân biệt giữa lớp 0 và lớp 1 (14 mẫu lớp 0 bị nhầm thành lớp 1).
* Mô hình không dự đoán được đúng bất kỳ mẫu nào của lớp 2, khi 5 mẫu của lớp 2 bị nhầm lẫn thành lớp 1.

Nhìn chung, mô hình hoạt động tốt với lớp 1 nhưng không tốt lắm với lớp 0 và lớp 2.

Ta tiếp tục xem có tối ưu hơn khi dùng tối ưu siêu tham số hay không

A blue square with white text

Description automatically generated

Hình 32: *Confusion matrix của mô hình Logistic Regression sau khi dùng grid\_search*

Trong ma trận confusion sau khi áp dụng **Grid Search** cho mô hình **Logistic Regression**, ta có các kết quả như sau:

1. **Hàng 0 (True Label = 0)**:
   * Có **12** mẫu thực sự thuộc lớp 0 đã được mô hình dự đoán chính xác là 0.
   * Có **15** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán sai thành lớp 1.
   * Không có mẫu nào của lớp 0 bị nhầm lẫn thành lớp 2.
2. **Hàng 1 (True Label = 1)**:
   * Có **218** mẫu thuộc lớp 1 được mô hình dự đoán chính xác.
   * Không có mẫu nào thuộc lớp 1 bị nhầm lẫn thành lớp 0 hoặc lớp 2.
3. **Hàng 2 (True Label = 2)**:
   * Có **5** mẫu thực sự thuộc lớp 2 nhưng bị dự đoán sai thành lớp 1.
   * Không có mẫu nào thuộc lớp 2 được mô hình dự đoán chính xác.
   * Không có mẫu nào của lớp 2 bị nhầm lẫn thành lớp 0.

Kết luận:

Mô hình hoạt động tốt cho lớp 1, nhưng chưa đủ hiệu quả khi phân loại lớp 0 và lớp 2. Điều này do sự mất cân bằng trong dữ liệu hoặc việc điều chỉnh siêu tham số chưa hoàn toàn tối ưu cho hai lớp này.

1. Mô hình Random Forest
2. Giới thiệu mô hình Random Forest

Random forest (RF) là một phương pháp học máy (machine learning statistic) dùng để phục vụ các mục đích phân loại, tính hồi quy và các nhiệm vụ khác bằng cách xây dựng nhiều cây quyết định (Decision tree).

Mỗi Node của cây sẽ là các thuộc tính, và các nhánh là giá trị lựa chọn của thuộc tính đó.

A diagram of a tree

Description automatically generated

Hình 29. Sơ đồ biểu diễn các cây quyết định trong phương pháp random forest

Từ hình 1 chúng ta thấy rằng Random Forest được cấu thành bởi một số cây quyết định. Thuật toán lấy mẫu cho phương pháp random forest ứng dụng cho các phương pháp sử dụng thuật toán mô tả thống kê để ước lượng số lượng từ một mẫu dữ liệu (bagging).

Một tập mẫu X = x\_1, ..., x\_n với các câu trả lời Y = y\_1, ..., y\_n, lấy giá trị trung bình (B lần), chọn một mẫu ngẫu nhiên từ bộ mẫu phù hợp với cây quyết định: Lặp b = 1,…, B; n mẫu từ giá trị tọa độ (X, Y); gọi là (X\_b, Y\_b); lớp dữ liệu hay kết quả hồi quy f\_b của biến X\_b, Y\_b:

Sau khi lấy mẫu, các phép tính toán cho các mẫu là ẩn số x’ có thể được thực hiện bằng cách lấy trung bình các giá trị nội suy từ tất cả các cây hồi quy riêng lẻ của biến x' hoặc lấy giá trị từ đa số của các mẫu trong cây quyết định:

A black and white math symbol

Description automatically generated

Random forest có thể sắp xếp sự quan trọng của các biến trong các bài toán phân loại hay hồi quy, được mô tả trong các nghiên cứu của Breiman, xác định các biến quan trọng trong 1 tập dữ liệu là làm phù hợp phương pháp random forest với tập dữ liệu:



Để xác định được tính quan trọng của đối tượng thứ i sau khi lấy mẫu, các giá trị của mẫu i được hoán vị trong tập mẫu và các lỗi dự báo được tính toán lại trong tập dữ liệu. Độ quan trọng của đối tượng được tính bằng điểm, các điểm được tính toán bằng cách lấy trung bình của độ chênh lệch giữa các lỗi dự báo trước và sau khi hoán vị. Các đối tượng có giá trị lớn được xếp quan trọng hơn các điểm có giá trị nhỏ.

1. Áp dụng mô hình Random Forest

Nhóm em huấn luyện và đánh giá một mô hình **RandomForestClassifier** trên tập dữ liệu huấn luyện và kiểm tra. Sau đó đưa ra báo cáo hiệu suất mô hình được tạo ra bằng cách sử dụng **classification\_report**

A screen shot of a computer code

Description automatically generated

Hình 30: Đoạn mã xây dựng mô hình Random Forest

1. Kết quả

A screenshot of a computer

Description automatically generated

Bảng 3: Kết quả thu được từ mô hình Random Forest

Độ chính xác trung bình lên tới 91%, cho thấy Random Forest là thuật toán mạnh cho các bài toán phân loại.

1. Tối ưu siêu tham số

A computer screen shot of a computer code

Description automatically generated

Hình 31: Đoạn mã xây dựng mô hình Random Forest khi dùng grid\_search

param\_grid là một từ điển định nghĩa các siêu tham số cho mô hình **Random Forest** để thực hiện tìm kiếm lưới (grid search). Cụ thể:

* **n\_estimators**: Danh sách số lượng cây trong rừng, với các giá trị [100, 200, 300], ảnh hưởng đến độ chính xác và thời gian huấn luyện.
* **max\_depth**: Chiều sâu tối đa của mỗi cây, với các giá trị [10, 20, 30, None], kiểm soát độ phức tạp của mô hình để tránh overfitting.
* **min\_samples\_split**: Số lượng mẫu tối thiểu cần thiết để tách một node, với các giá trị [2, 5, 10], giúp xác định độ sâu của cây.
* **min\_samples\_leaf**: Số lượng mẫu tối thiểu trong một node lá, với các giá trị [1, 2, 4], để điều chỉnh mức độ chia nhỏ của dữ liệu.
* **max\_features**: Số lượng đặc trưng tối đa để xem xét khi tách, với các giá trị ['auto', 'sqrt'], ảnh hưởng đến tính ngẫu nhiên và khả năng tổng quát của mô hình.
* **bootstrap**: Quyết định có lấy mẫu lại dữ liệu khi xây dựng cây hay không, với các giá trị [True, False], ảnh hưởng đến tính đa dạng của các cây trong rừng.

Mục tiêu của việc điều chỉnh các siêu tham số này là tối ưu hóa hiệu suất của mô hình.

A white background with black text

Description automatically generated

Bảng : Kết quả sau khi điều chỉnh siêu tham số mô hình Random Forest

Quá trình đã thực hiện 2160 lần huấn luyện, với 5 lần kiểm tra chéo cho mỗi 432 tổ hợp tham số khác nhau.

Các tham số tốt nhất được tìm thấy cho mô hình là:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| bootstrap | max\_depth | max\_features | min\_samples\_leaf | min\_samples\_split | n\_estimators |
| False | 10 | 'sqrt' | 1 | 2 | 200 |

Bảng : Các siêu tham số được chọn

- **bootstrap: False**: Không sử dụng kỹ thuật lấy mẫu lại dữ liệu.

- **max\_depth: 10**: Chiều sâu tối đa của mỗi cây là 10, giúp kiểm soát độ phức tạp và tránh overfitting.

- **max\_features: 'sqrt'**: Số lượng đặc trưng tối đa để xem xét khi tách là căn bậc hai của tổng số đặc trưng, giúp tăng tính ngẫu nhiên và khả năng tổng quát.

- **min\_samples\_leaf: 1**: Mỗi node lá phải có ít nhất 1 mẫu, cho phép chia nhỏ dữ liệu một cách linh hoạt.

- **min\_samples\_split: 2**: Mỗi node cần có ít nhất 2 mẫu để tách, giúp giữ cho cây không bị tách quá sớm.

- **n\_estimators: 200**: Số lượng cây trong rừng là 200, cung cấp một mô hình mạnh mẽ mà vẫn giữ được hiệu suất.

Ngoài ra Độ chính xác cao nhất qua kiểm tra chéo đạt khoảng 97.47%, cho thấy mô hình có khả năng phân loại tốt và có thể tổng quát tốt trên tập dữ liệu không thấy trước (unseen data).

1. Ma trận nhầm lẫn

Ta tiến hình in ra và phân tích ma trận nhầm lẫn của mô hình Random Forest trước và sau khi tối ưu siêu tham số

A blue square with white text

Description automatically generated

Hình : Ma trận nhầm lẫn của mô hình Random Forest

* **Hàng 0 (True Label = 0)**:
* Có **9** mẫu thực sự thuộc lớp 0 đã được mô hình dự đoán chính xác là 0.
* Có **4** mẫu thuộc lớp 0 nhưng bị dự đoán sai thành lớp 1.
* Có **14** mẫu thuộc lớp 0 nhưng bị dự đoán sai thành lớp 2.
* **Hàng 1 (True Label = 1)**:
* Có **218** mẫu thực sự thuộc lớp 1 được mô hình dự đoán chính xác là 1.
* Không có mẫu nào thuộc lớp 1 bị nhầm lẫn thành lớp 0 hoặc lớp 2.
* **Hàng 2 (True Label = 2)**:
* Có **5** mẫu thực sự thuộc lớp 2 nhưng bị dự đoán sai thành lớp 1.
* Không có mẫu nào thuộc lớp 2 được dự đoán chính xác và không có mẫu nào bị nhầm lẫn thành lớp 0.

Đánh giá :

* **Lớp 1**: Mô hình hoạt động rất tốt với 218 dự đoán đúng và không có sai sót.
* **Lớp 0**: Mô hình gặp khó khăn trong việc phân loại, với tổng cộng **18** mẫu (4 + 14) bị nhầm lẫn với các lớp khác.
* **Lớp 2**: Mô hình có hiệu suất kém, chỉ có **5** mẫu bị nhầm lẫn và không có dự đoán đúng cho lớp này

Kết luận:

Mô hình hoạt động hiệu quả nhất cho lớp 1 nhưng chưa đủ khả năng phân loại cho lớp 0 và lớp 2, đặc biệt là lớp 2, khi không có dự đoán chính xác nào.

A blue square with white text

Description automatically generated

Hình : Ma trận nhầm lẫn của mô hình Random Forest sau khi dùng Grid search

* **Hàng 0 (True Label = 0)**:
* Có **13** mẫu thực sự thuộc lớp 0 được dự đoán chính xác là 0.
* Có **4** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán nhầm thành lớp 1.
* Có **10** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán nhầm thành lớp 2.
* **Hàng 1 (True Label = 1)**:
* Có **217** mẫu thực sự thuộc lớp 1 được dự đoán chính xác là 1.
* Có **1** mẫu thực sự thuộc lớp 1 bị dự đoán sai thành lớp 0.
* Không có mẫu nào thuộc lớp 1 bị nhầm lẫn thành lớp 2.
* **Hàng 2 (True Label = 2)**:
* Không có mẫu nào thuộc lớp 2 được dự đoán chính xác.
* Có **5** mẫu thực sự thuộc lớp 2 nhưng bị dự đoán sai thành lớp 1.
* Không có mẫu nào thuộc lớp 2 bị nhầm lẫn thành lớp 0.

Đánh giá :

* **Lớp 1**: Mô hình hoạt động rất tốt với 217 dự đoán đúng và chỉ 1 mẫu bị nhầm lẫn thành lớp 0.
* **Lớp 0**: Mô hình có hiệu suất khá tốt với 13 dự đoán đúng, nhưng có **14** mẫu (4 + 10) bị nhầm lẫn với các lớp khác.
* **Lớp 2**: Mô hình không có dự đoán chính xác nào cho lớp này, với tất cả **5** mẫu bị nhầm lẫn thành lớp 1.

Kết luận:

Mô hình hoạt động hiệu quả nhất cho lớp 1, nhưng gặp khó khăn trong việc phân loại lớp 0 và không có khả năng phân loại cho lớp 2

1. Mô hình Catboost
2. Giới thiệu mô hình Catboost

CatBoost, viết tắt của **Categorical Boosting**, là một thuật toán machine learning mạnh mẽ do Yandex phát triển, dựa trên kỹ thuật boosting cho cây quyết định. Đặc điểm nổi bật của CatBoost là khả năng xử lý hiệu quả các biến phân loại mà không cần mã hóa phức tạp. Điều này giúp đơn giản hóa quá trình tiền xử lý dữ liệu và cải thiện hiệu suất tổng thể **của mô hình**. CatBoost **hoạt động** theo nguyên lý tăng cường độ dốc, trong đó các cây quyết định được xây dựng một cách tuần tự. Mỗi cây mới được xây dựng dựa trên lỗi của các cây trước đó, nhằm tối ưu hóa hàm mất mát và cải thiện **độ chính xác** của mô hình.

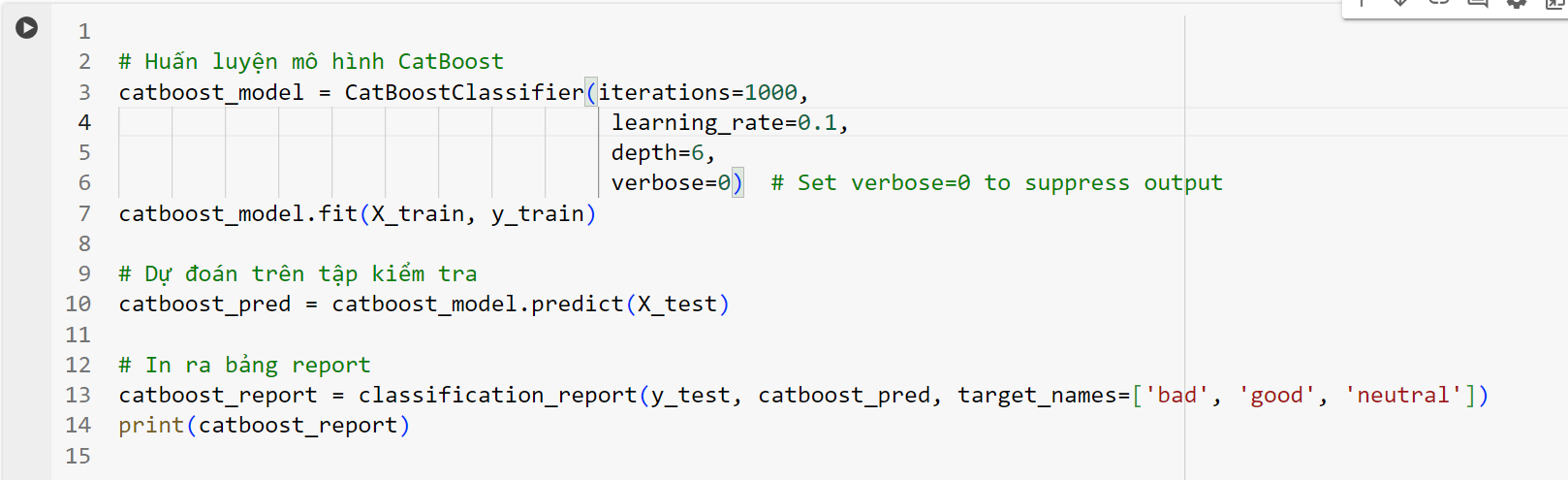
Một điểm đặc biệt của CatBoost là kỹ thuật "tăng cường theo thứ tự", giúp xử lý các biến phân loại bằng cách tạo ra các mã số thứ tự dựa trên giá trị mục tiêu. Kỹ thuật này không chỉ giúp giảm thiểu lỗi dự đoán mà còn ngăn chặn hiện tượng quá khớp, một vấn đề phổ biến trong các mô hình machine learning. CatBoost cũng sử dụng các phương pháp chính quy hóa (regularization) để ngăn chặn **overfitting**, chẳng hạn như **"L2 regularization"** và kỹ thuật **"leaf-wise"**. Những phương pháp này giúp duy trì sự cân bằng giữa độ chính xác và khả năng tổng quát hóa của mô hình.

CatBoost dễ dàng tích hợp với các thư viện machine learning phổ biến như **scikit-learn**, giúp nó trở nên linh hoạt và tiện lợi cho nhiều loại bài toán khác nhau, bao gồm phân loại, hồi quy và nhiều bài toán khác. Tuy nhiên, thời gian huấn luyện của CatBoost có thể dài, đặc biệt là với các tập dữ liệu lớn, và nó yêu cầu nhiều tài nguyên tính toán. Việc điều chỉnh các siêu tham số của CatBoost cũng phức tạp và đòi hỏi kiến thức chuyên sâu về thuật toán để đạt được hiệu quả tối ưu.

1. Áp dụng mô hình Catboots

Em áp dụng mô hình Catboots với các siêu tham số

* **iterations=1000**: Số lần lặp (iterations) hay còn gọi là số cây quyết định (trees) sẽ được xây dựng trong quá trình huấn luyện. Mô hình sẽ huấn luyện qua 1000 cây.
* **learning\_rate=0.1**: Tốc độ học (learning rate) là một tham số quyết định mức độ điều chỉnh mô hình trong mỗi lần lặp. Giá trị 0.1 có nghĩa là các thay đổi nhỏ hơn sẽ được thực hiện trong quá trình tối ưu hóa mô hình, tránh sự thay đổi quá lớn và giảm khả năng overfitting.
* **depth=6**: Đây là độ sâu của cây quyết định (decision tree). Giá trị này quyết định số lượng node trong mỗi cây và ảnh hưởng đến khả năng học các mẫu phức tạp của mô hình. Độ sâu càng lớn thì cây càng có khả năng học các mối quan hệ phức tạp hơn, nhưng cũng làm tăng nguy cơ overfitting.
* **verbose=0**: Tùy chọn này kiểm soát việc in ra thông tin trong quá trình huấn luyện. Khi verbose=0, không có thông tin nào được in ra màn hình, giúp giảm lượng thông báo trong quá trình huấn luyện.



Hình 31: Đoạn mã xây dựng mô hình Catboots

1. Kết quả

A screenshot of a computer screen

Description automatically generated

Bảng 4: Kết quả mô hình Catboots

1. Tối ưu hoá siêu tham số mô hình Catboots

Tối ưu hoá siêu tham số là một quá trình quan trọng trong việc xây dựng mô hình ML, giúp tìm ra các giá trị siêu tham số tốt nhất để cải thiện hiệu suất dự đoán của mô hình trên một tập dữ liệu. Mô hình ML với các siêu tham số tối ưu có thể có độ chính xác và độ ổn định cao hơn so với các mô hình không được tối ưu.

A screenshot of a computer code

Description automatically generated

Hình 32: Đoạn mã xây dựng mô hình Catboots với tối ưu siêu tham số

Tập siêu tham số của mô hình

* **iterations**: Số lần lặp của mô hình (từ 100 đến 1000), tức là số cây quyết định trong quá trình boosting.
* **learning\_rate**: Tốc độ học (từ 0.01 đến 0.3). Giá trị này điều chỉnh mức độ thay đổi khi mô hình cập nhật.
* **depth**: Độ sâu của các cây quyết định (từ 4 đến 10). Độ sâu lớn hơn giúp mô hình học được nhiều mẫu phức tạp hơn, nhưng cũng có thể dẫn đến **overfitting**.
* **l2\_leaf\_reg**: Hệ số regularization L2 trên các lá của cây (từ 1 đến 10), giúp ngăn chặn **overfitting**.
* **random\_state**: Đảm bảo tính ngẫu nhiên có thể tái lập được trong huấn luyện.
* **verbose**: Bằng 0 để tắt đầu ra chi tiết khi huấn luyện.

1. Kết quả sau khi tối ưu hóa siêu tham số

A screenshot of a number

Description automatically generated

Bảng 5 : Kết quả mô hình Catboots sau khi tối ưu hóa siêu tham số

1. Ma trận nhầm lẫn

A diagram with blue squares and numbers

Description automatically generated

Hình : Ma trận nhầm lẫn của mô hình Catboost

* **Hàng 0 (True Label = 0)**:
* Có **4** mẫu thực sự thuộc lớp 0 được dự đoán chính xác là 0.
* Có **5** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán nhầm thành lớp 1.
* Có **18** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán nhầm thành lớp 2.
* **Hàng 1 (True Label = 1)**:
* Có **145** mẫu thực sự thuộc lớp 1 được dự đoán chính xác là 1.
* Có **73** mẫu thực sự thuộc lớp 1 bị dự đoán nhầm thành lớp 0.
* Không có mẫu nào thuộc lớp 1 bị nhầm lẫn thành lớp 2.
* **Hàng 2 (True Label = 2)**:
* Không có mẫu nào thuộc lớp 2 được dự đoán chính xác.
* Có **5** mẫu thực sự thuộc lớp 2 nhưng bị dự đoán sai thành lớp 1.
* Không có mẫu nào thuộc lớp 2 bị nhầm lẫn thành lớp 0.

Đánh giá:

* **Lớp 1**: Mô hình hoạt động khá tốt với 145 dự đoán đúng, tuy nhiên có **73** mẫu bị nhầm lẫn thành lớp 0, cho thấy một số khó khăn trong việc phân loại chính xác lớp này.
* **Lớp 0**: Mô hình có **4** dự đoán đúng nhưng có **23** mẫu (5 + 18) bị nhầm lẫn với các lớp khác, cho thấy rằng mô hình không đủ mạnh để phân loại chính xác lớp này.
* **Lớp 2**: Mô hình không có dự đoán chính xác nào cho lớp này, với tất cả **5** mẫu bị nhầm lẫn thành lớp 1.

Kết luận :

Mô hình hoạt động tốt nhất với lớp 1, nhưng gặp khó khăn trong việc phân loại chính xác cho lớp 0 và không có khả năng phân loại cho lớp 2. Điều này có thể là do sự mất cân bằng trong dữ liệu hoặc cần điều chỉnh thêm cho mô hình để cải thiện khả năng phân loại cho các lớp còn lại.

Ta tiếp tục xem qua sau khi tinh chỉnh siêu tham số, mô hình có tốt hơn không

A blue squares with white text

Description automatically generated

Hình : Ma trận nhầm lẫn của mô hình Catboost sau khi tinh chỉnh siêu tham số

* **Hàng 0 (True Label = 0)**:
* Có **5** mẫu thực sự thuộc lớp 0 được dự đoán chính xác là 0.
* Có **5** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán nhầm thành lớp 1.
* Có **17** mẫu thực sự thuộc lớp 0 nhưng bị dự đoán nhầm thành lớp 2.
* **Hàng 1 (True Label = 1)**:
* Có **154** mẫu thực sự thuộc lớp 1 được dự đoán chính xác là 1.
* Có **64** mẫu thực sự thuộc lớp 1 bị dự đoán nhầm thành lớp 0.
* Không có mẫu nào thuộc lớp 1 bị nhầm lẫn thành lớp 2.
* **Hàng 2 (True Label = 2)**:
* Không có mẫu nào thuộc lớp 2 được dự đoán chính xác.
* Có **5** mẫu thực sự thuộc lớp 2 nhưng bị dự đoán sai thành lớp 1.
* Không có mẫu nào thuộc lớp 2 bị nhầm lẫn thành lớp 0.

Đánh giá:

* **Lớp 1**: Mô hình hoạt động tốt với **154** dự đoán đúng, nhưng có **64** mẫu bị nhầm lẫn thành lớp 0, cho thấy rằng một số khó khăn trong việc phân loại chính xác lớp này.
* **Lớp 0**: Mô hình có **5** dự đoán đúng nhưng **22** mẫu (5 + 17) bị nhầm lẫn với các lớp khác, cho thấy rằng mô hình không đủ mạnh để phân loại chính xác lớp này.
* **Lớp 2**: Mô hình không có dự đoán chính xác nào cho lớp này, với tất cả **5** mẫu bị nhầm lẫn thành lớp 1.

Kết luận :

Mô hình hoạt động tốt nhất với lớp 1, nhưng gặp khó khăn trong việc phân loại chính xác cho lớp 0 và không có khả năng phân loại cho lớp 2

**PHẦN 5: ĐÁNH GIÁ KẾT QUẢ VÀ KẾT LUẬN**

* 1. **So sánh kết quả của các mô hình**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuật toán**  **phân lớp** | ***Logistic***  ***Regression*** | **Random**  **Forest** | **Catboost** |
| ***Precision*** | 0.64 | 0.65 | 0.33 |
| ***Recall*** | 0.49 | 0.44 | 0.27 |
| ***F1\_score*** | 0.54 | 0.49 | 0.28 |
| ***Accuracy*** | 0.92 | 0.91 | 0.60 |

Bảng : So sánh kết quả các mô hình trên tập test của bộ dữ liệu banking\_reviews3

Bảng so sánh này đánh giá hiệu suất của ba thuật toán phân loại: **Logistic Regression**, **Random Forest**, và **CatBoost** dựa trên các chỉ số **precision**, **recall**, **f1-score**, và **accuracy**.

**1. Precision:**

* **Logistic Regression**: 0.64
* **Random Forest**: 0.65
* **CatBoost**: 0.33

Precision của Logistic Regression và Random Forest tương đối cao, cho thấy khả năng dự đoán đúng các mẫu dương tính tốt hơn. CatBoost có precision thấp hơn đáng kể, chỉ đạt 0.33.

**2. Recall:**

* **Logistic Regression**: 0.49
* **Random Forest**: 0.44
* **CatBoost**: 0.27

Logistic Regression có recall cao nhất (0.49), cho thấy mô hình này có khả năng phát hiện đúng nhiều mẫu dương tính hơn so với Random Forest và CatBoost, vốn có recall lần lượt là 0.44 và 0.27.

**3. F1-Score:**

* **Logistic Regression**: 0.54
* **Random Forest**: 0.49
* **CatBoost**: 0.28

F1-score của Logistic Regression (0.54) cũng cao nhất, thể hiện sự cân bằng tốt nhất giữa precision và recall. Random Forest đứng thứ hai với 0.49, còn CatBoost có F1 thấp nhất, chỉ 0.28.

**4. Accuracy:**

* **Logistic Regression**: 0.92
* **Random Forest**: 0.91
* **CatBoost**: 0.60

Về độ chính xác tổng thể, Logistic Regression đạt **92%**, Random Forest **91%**, và CatBoost thấp hơn nhiều với **60%**.

Kết luận :

* **Logistic Regression** và **Random Forest** hoạt động tốt hơn, đặc biệt trong việc duy trì cân bằng giữa các chỉ số. Logistic Regression có kết quả tốt hơn một chút về recall và F1-score.
* **CatBoost** có hiệu suất kém hơn rõ rệt, với precision, recall, và F1-score thấp, cùng với độ chính xác chỉ 60%.
  1. **Tóm tắt kết quả và nhận xét**

Thứ nhất, em đã thực hiện phần phân tích cảm xúc (Sentiment Analysis) cho

bộ dữ liệu banking\_reviews3 ở phần 2. Từ đó, em đã phân tích được cảm xúc của khách

hàng cũng như một số sự khác biệt giữa các chi nhánh ngân hàng ở Ấn Độ. Qua quá trình

thực hiện phân tích, em cũng đã tìm hiểu và biết thêm nhiều kiến thức và phương

pháp phân tích dữ liệu trong phần này.

Thứ hai, em thực hiện 2 mô hình học máy có giám sát: Logistic Regression, Random Forest từ scikit- learn và mô hình Ensemble là Catboots và thêm vào đó là dùng mô hình PCA để giảm chiều dữ liệu. 4 mô hình này được sử dụng để phân loại sắc thái cảm xúc cho bài toán phân tích đánh giá và bình luận của khách khi dùng dịch vụ tại các ngân hàng ở Ấn Độ theo 3 mức độ cảm xúc: Bad, Neutral và Good. Dữ liệu thực nghiệm là các review về mức độ hài lòng về chất lượng dịch vụ tại các ngân hàng của Ấn Độ.

* 1. **Những hạn chế và hướng phát triển**

*1. Hạn chế*

- Kết quả mô hình sau mỗi lần chạy sẽ khác nhau, em chỉ thực hiện bài đánh

giá trên một lần chạy.

- Em chỉ chia dữ liệu thành 3 mức độ cảm xúc, trên thực tế thì nhiều hơn.

-Dữ liệu thu thập còn ít chỉ có 1000 mẫu và có sự mất cân bằng về dữ liệu của biến đầu ra.

-Vì không có quá nhiều thời gian nghiên cứu nên bộ dữ liệu đầu vào mà em sử dụng là bộ dữ liệu có sẵn trên Kaggle mà chưa thể khai thác và thu thập dữ liệu đầu vào theo thời gian thực.

-Tối ưu siêu tham số chạy mất khá nhiều thời gian và có thể chưa mang lại tính chính

xác hoàn toàn vì mỗi lần chạy mô hình cho ra mỗi kết quả khác nhau.

*2. Hướng phát triển*

Từ việc phân tích phản hồi của khách hàng theo biến phân loại Bad, Neutral

và Good, nhóm có thể phát triển xây dựng mô hình học máy phân loại đa lớp (với

biến phân loại Rating 1-5 sao hoặc theo nhiều loại cảm xúc khác nhau: giận dữ, hạnh

phúc, buồn bã, bất ngờ,...). Ngoài ra có thể bổ sung thêm thông tin về khách hàng để

xây dựng mô hình phân lớp đánh giá phản hồi của khách hàng theo từng nhóm khách

hàng, hoặc theo địa điểm khác nhau,...

Vấn đề gia tăng độ chính xác có thể giải quyết bằng việc kết hợp thuật toán phân

lớp để gia tăng độ chính xác đồng thời không ảnh hưởng quá nhiều tốc độ xử lý thông

tin

Việc xử lý ngôn ngữ tự nhiên hiện tại nhóm chỉ phân tích trên ngôn ngữ là tiếng

Anh. Trong tương lai có thể phân tích nhận diện cảm xúc của khách hàng bằng tiếng

Việt (tham khảo và áp dụng các mô hình trong các dự án của nhóm Vietnamese NLP

Research Group - Under The Sea,vncoreNLP,…).

Trong quá trình thu thập phản hồi khách hàng có thể lọc ra những phản hồi không phù hợp (toxic comment classification) để dự đoán liệu phản hồi của khách hàng có thật sự phù hợp hay không

**PHẦN 6: GIAO DIỆN DEMO**

* 1. **Thư viện Gradio và Interface**

Gradio là một thư viện Python mạnh mẽ được sử dụng để xây dựng giao diện

người dùng (UI) cho các mô hình máy học và các ứng dụng liên quan đến dữ liệu. Nó

cho phép tạo ra các giao diện người dùng tương tác cho các mô hình máy học mà

không cần kiến thức vững về phát triển giao diện người dùng.

Thư viện Gradio có thể sử dụng linh hoạt trên nhiều dạng tập lệnh Python,

notebook Jupyter hoặc Colaboratory.

A close up of a text

Description automatically generated

Hình : *install và import Gradio trên colaboratory*

Interface: một API cấp cao cho phép bạn tạo một bản demo học máy đầy đủ chỉ

đơn giản bằng cách cung cấp danh sách các đầu vào và đầu ra.

**II. Xây dựng giao diện**

Em sử dụng Gradio để tạo giao diện đơn giản để dự đoán cảm xúc của văn bản

đầu vào. Khi chạy giao diện này sẽ có một ô văn bản để nhập văn bản cần phân loại

cảm xúc. Kết quả sẽ là cảm xúc dự đoán của văn bản đó.

Giao diện được xây dựng dựa theo mô hình tốt nhất sau khi thực hiện xây dựng

và đánh giá các mô hình phân lớp văn bản ở các phần trên đã thực hiện ở phần trên.

Đó là mô hình Logistic Regression và thiết lập các chỉ số như trước.

Dữ liệu huấn luyện được sử dụng từ kết quả sau khi tiền xử lý dữ liệu của bộ dữ

liệu bank\_reviews3.csv được thực hiện ở phần trên.

A computer screen shot of a program code

Description automatically generated

*Hình : Code xây dựng giao diện demo*

A screenshot of a computer

Description automatically generated

*Hình 42: Giao diện demo*

**PHỤ LỤC**

1. **Source code và file liên quan:’**

https://drive.google.com/drive/folders/1Yzdei18lyiT21f63qWRdmm4xR8eVHE6z?usp=sharing

1. **Bảng phân công nhiệm vụ**
2. **Danh mục hình ảnh**

Tham khảo :

[1]<https://medium.com/@chexki_/using-label-encoder-on-unbalanced-categorical-data-in-machine-learning-using-python-435f521323b1>

[2] <https://spotintelligence.com/2023/08/27/dimensionality-reduction/>

[3]<https://medium.com/analytics-vidhya/logistic-regression-from-scratch-multi-classification-with-onevsall-d5c2acf0c37c>

[4] <https://jstt.vn/index.php/vn/article/view/244/208>