

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**Đề tài:**

**Dự đoán kết quả round đầu CS:GO**

**Giảng viên:** Thân Quang Khoát

**Nhóm sinh viên thực hiện**

STT	Họ và tên	MSSV
1	Phạm Gia Khánh	20200323
2	Trần Xuân Nam	20204672
3	Đàm Phương Mai	20204667
4	Nguyễn Hữu Việt	20200667

*Hà Nội, tháng 06 năm 2023*

# MỤC LỤC

Mục lục .....	1
Lời nói đầu.....	3
Chương 1. GIỚI THIỆU ĐỀ TÀI.....	4
1.1. Mô tả bài toán thực tế.....	4
1.2. Mô tả hệ thống.....	4
Chương 2. Cơ sở lý thuyết.....	5
2.1. Neuron network .....	5
2.2. naïve bayes .....	5
2.3. Logistic Regression .....	7
2.4. K-Nearest Neighbors .....	8
2.5. Decision Tree.....	10
2.6. random forest.....	11
Chương 3. Quá trình thực nghiệm và kết quả đạt được .....	13
3.1. Trực quan hoá và phân tích dữ liệu .....	13
3.1.1 Thống kê dữ liệu.....	13
3.2. Tiền xử lý dữ liệu .....	15
3.2.1 Xử lý dữ liệu dạng phân loại .....	15
3.3. Phương pháp đánh giá .....	16
3.4. Huấn luyện mô hình học máy.....	17
3.4.1 Neuron Network .....	17
3.4.2 Naïve Bayes.....	18
3.4.3 Logistic Regression .....	18
3.4.4 Decision Tree.....	18
3.4.5 K-Nearest Neighbors .....	19
3.4.6 Random forest .....	19

Chương 4. Kết quả, đánh giá .....	20
4.1. NEURON NETWORK .....	20
4.2. naïve BAYES .....	21
4.3. LOGISTIC REGRESSION .....	22
4.4. decision tree .....	23
4.5. K-nearest neighbor .....	24
4.6. random forest .....	25
4.7. nhận xét chung .....	26
Chương 5. KẾT LUẬN VÀ KINH NGHIỆM RÚT RA .....	27
5.1. Các vấn đề, khó khăn gặp phải và phương hướng giải quyết .....	27
5.2. Các khám phá và kết luận .....	27

## LỜI NÓI ĐẦU

Xã hội ngày càng phát triển, nhu cầu giải trí của con người ngày càng tăng cao, và trong thời đại công nghệ hiện đại, trò chơi điện tử đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày. Với sự tiến bộ của công nghệ, trò chơi điện tử không chỉ đơn thuần là một hình thức giải trí mà còn là một nền công nghiệp khổng lồ, thu hút hàng triệu người chơi trên toàn thế giới.

Trong danh mục các trò chơi điện tử phổ biến, CS:GO (Counter-Strike: Global Offensive) đã nhanh chóng trở thành một hiện tượng và thu hút sự quan tâm của đông đảo người chơi. CS:GO không chỉ đem lại những trải nghiệm hấp dẫn và kịch tính mà còn tạo nên một cộng đồng lớn mạnh xung quanh trò chơi này.

Với sự phát triển và phổ biến của CS:GO, nhu cầu dự đoán kết quả các trận đấu và round đấu trong trò chơi này cũng gia tăng. Người chơi, nhà cái và những người yêu thích eSports đều quan tâm tới việc có thể dự đoán được kết quả chính xác, từ đó đưa ra các quyết định thông minh về đặt cược hoặc theo dõi trận đấu.

Do đó, trong báo cáo này, chúng ta sẽ tập trung vào việc nghiên cứu và phát triển một hệ thống dự đoán kết quả round đấu CS:GO. Điều này không chỉ mang lại giá trị giải trí và tiện ích cho cộng đồng người chơi CS:GO mà còn đóng góp vào sự phát triển của lĩnh vực dự đoán trong eSports và trò chơi trực tuyến.

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1. MÔ TẢ BÀI TOÁN THỰC TẾ

CS:GO là một trò chơi trực tuyến bắn súng đối kháng, mỗi trận đấu có 2 đội (CT và T) đấu với nhau 30 vòng đấu, mỗi vòng đấu dài 1 phút 55s. Mỗi đội có 5 người chơi (tổng là 10 người chơi) và đội nào thắng 16 vòng trước thì sẽ chiến thắng trận đấu đó. Khi bắt đầu trận đấu, một đội sẽ chơi với vai trò T và đội kia là CT, sau 15 vòng thì sẽ đổi lại. Có 7 bản đồ khác nhau trong trò chơi mà người chơi có thể lựa chọn. Bạn sẽ thắng một vòng với vai trò T nếu đội bạn đặt bom thành công hoặc tiêu diệt hết người chơi team CT. Ngược lại, bạn sẽ thắng với vai trò CT nếu đội bạn gỡ bom thành công hoặc tiêu diệt hết người chơi team T.

Từ những dữ liệu đầu vào như thời gian còn lại, bản đồ, tỉ số 2 đội,... chúng ta sẽ tạo mô hình học máy để dự đoán xem đội nào sẽ chiến thắng vòng đấu đó.

## 1.2. MÔ TẢ HỆ THỐNG

**Đầu vào:** Một vector các giá trị thuộc tính mô tả vòng đấu.

**Đầu ra:** Dự đoán xem đội T hay CT sẽ chiến thắng vòng đấu đó.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

### 2.1. NEURON NETWORK

Mạng neuron (neural network) dựa trên mô phỏng cấu trúc và hoạt động của hệ thần kinh sinh học. Mạng neuron là một mô hình tính toán được thiết kế để xử lý thông tin tương tự như cách mà hệ thần kinh của con người hoạt động.

Cơ bản, mạng neuron gồm một số lượng lớn các "nơ-ron" nhân tạo (artificial neurons) hoặc "đơn vị xử lý" (processing units), được kết nối với nhau bằng các liên kết trọng số (weight). Mỗi nơ-ron nhân tạo nhận đầu vào từ các nơ-ron khác thông qua các liên kết trọng số, thực hiện một phép tính toán đơn giản trên đầu vào, và sau đó truyền kết quả qua các liên kết ra đến các nơ-ron khác.

Các liên kết trọng số trong mạng neuron đại diện cho sự tương tác giữa các nơ-ron và có thể được điều chỉnh để mạng neuron "học" từ dữ liệu. Quá trình điều chỉnh các trọng số này được gọi là "huấn luyện" (training) và thường sử dụng các thuật toán học máy (machine learning) như lan truyền ngược (backpropagation) để điều chỉnh các trọng số theo cách tối ưu hóa mục tiêu.

Cấu trúc của mạng neuron có thể được tổ chức thành các lớp (layers) khác nhau, bao gồm lớp đầu vào (input layer), lớp đầu ra (output layer) và một hoặc nhiều lớp ẩn (hidden layers) ở giữa. Mạng neuron có khả năng tự học các đặc trưng và mối quan hệ phức tạp trong dữ liệu thông qua quá trình huấn luyện, và sau đó có thể được sử dụng để dự đoán hoặc phân loại dữ liệu mới.

### 2.2. NAÏVE BAYES

Naive Bayes Classification là một thuật toán dựa trên định lý Bayes về lý thuyết xác suất để đưa ra các phán đoán cũng như phân loại dữ liệu dựa trên các dữ liệu được quan sát và thống kê.

Naive Bayes Classification là một trong những thuật toán được ứng dụng rất nhiều trong các lĩnh vực Machine learning dùng để đưa các dự đoán chính xác nhất dựa trên một tập

dữ liệu đã được thu thập, vì nó khá dễ hiểu và độ chính xác cao. Nó thuộc vào nhóm thuật toán học có giám sát, tức là máy học từ các ví dụ từ các mẫu dữ liệu đã có.

Định lý Bayes cho phép tính xác suất xảy ra của một sự kiện ngẫu nhiên A khi biết sự kiện liên quan B đã xảy ra. Xác suất này được ký hiệu là  $P(A|B)$ , và đọc là “xác suất của A nếu có B”. Đại lượng này được gọi xác suất có điều kiện hay xác suất hậu nghiệm vì nó được rút ra từ giá trị được cho của B hoặc phụ thuộc vào giá trị đó.

Theo định lý Bayes, xác suất xảy ra A khi biết B sẽ phụ thuộc vào 3 yếu tố:

- Xác suất xảy ra A của riêng nó, không quan tâm đến B. Ký hiệu là  $P(A)$  và đọc là xác suất của A. Đây được gọi là xác suất biên duyên hay xác suất tiên nghiệm, nó là “tiên nghiệm” theo nghĩa rằng nó không quan tâm đến bất kỳ thông tin nào về B.
- Xác suất xảy ra B của riêng nó, không quan tâm đến A. Ký hiệu là  $P(B)$  và đọc là “xác suất của B”. Đại lượng này còn gọi là hằng số chuẩn hóa (normalizing constant), vì nó luôn giống nhau, không phụ thuộc vào sự kiện A đang muốn biết.
- Xác suất xảy ra B khi biết A xảy ra. Ký hiệu là  $P(B|A)$  và đọc là “xác suất của B nếu có A”. Đại lượng này gọi là khả năng (likelihood) xảy ra B khi biết A đã xảy ra. Chú ý không nhầm lẫn giữa khả năng xảy ra B khi biết A và xác suất xảy ra A khi biết B.

Tóm lại định lý Bayes sẽ giúp ta tính ra xác suất xảy ra của một giả thuyết bằng cách thu thập các bằng chứng nhất quán hoặc không nhất quán với một giả thuyết nào đó. Khi các bằng chứng tích lũy, mức độ tin tưởng vào một giả thuyết thay đổi. Khi có đủ bằng chứng, mức độ tin tưởng này thường trở nên rất cao hoặc rất thấp, tức là xác suất xảy ra giả thuyết sẽ thay đổi thì các bằng chứng liên quan đến nó thay đổi. Công thức của định luật Bayes được phát biểu như sau:

Trong đó:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{\text{likelihood} * \text{prior}}{\text{normalizing.constant}}$$

- $P(A)$  là xác suất xảy ra của riêng A, không liên quan đến B (prior).

- $P(B)$  là xác suất xảy ra của riêng B.  $P(B)$  còn được gọi là normalizing.constant vì nó là hằng số không bị ảnh hưởng bởi A.
- $P(B|A)$  là xác suất xảy ra B khi biết A đã xảy ra (likelihood), còn được gọi là xác suất của B nếu có A.

Ở trên ta có thể thấy xác suất xảy ra của giả thuyết A phụ thuộc vào xác suất của giả thuyết B, nhưng trong thực tế xác suất A có thể phụ thuộc vào xác suất của nhiều các giả thuyết khác có thể là  $B_1, B_2, B_3 \dots B_n$ . Vậy định luật Bayes có thể được mở rộng bằng công thức sau:

$$P(A|B) = \frac{(P(B_1|A) \times P(B_2|A) \times P(B_3|A) \dots \times P(B_n|A)) \times P(A)}{P(B_1) \times P(B_2) \times P(B_3) \times \dots \times P(B_n)}$$

### 2.3. LOGISTIC REGRESSION

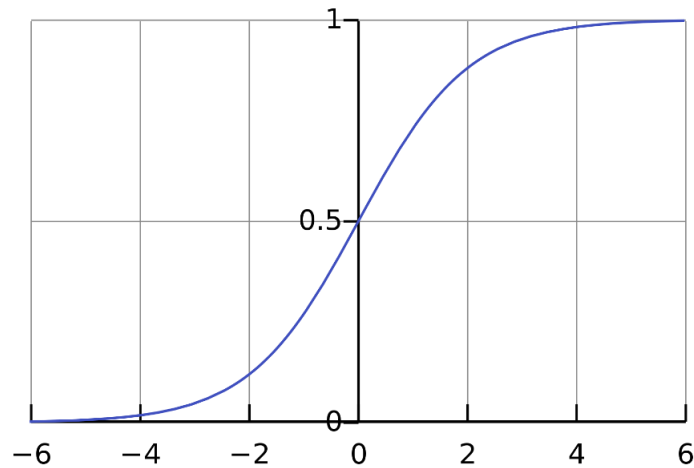
Hồi quy logistic là mô hình hồi quy nhằm dự đoán đầu ra rời rạc  $y$  tương ứng với một vector đầu vào  $x$ . Tương đương với việc phân loại các đầu vào  $x$  vào các nhóm  $y$  tương ứng.

Hồi quy logistic là một mô hình thống kê sử dụng hàm logistic, hay hàm logit trong toán học làm phương trình giữa  $x$  và  $y$ . Hàm logit ánh xạ  $y$  làm hàm sigmoid của  $x$ :

$$f(x) = \frac{1}{1 + e^{-x}}$$

Phương trình hồi quy logistic khi vẽ ra sẽ được đồ thị như sau:





Hình 1 Đồ thị phương trình hồi quy logistic

Từ đồ thị trên ta có thể thấy hàm logit chỉ trả về các giá trị giữa 0 và 1 cho biến phụ thuộc, dù giá trị của biến độc lập là gì. Đây là cách hồi quy logistic ước tính giá trị của biến phụ thuộc. Phương pháp hồi quy logistic cũng lập mô hình phương trình giữa nhiều biến độc lập và một biến phụ thuộc.

Trong nhiều trường hợp, nhiều biến giải thích ảnh hưởng đến giá trị của biến phụ thuộc. Để lập mô hình các tập dữ liệu đầu vào như vậy, công thức hồi quy logistic phải giả định mối quan hệ tuyến tính giữa các biến độc lập khác nhau. Ta sửa đổi hàm sigmoid và tính toán biến đầu ra cuối cùng như sau:

$$y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_n x_n)$$

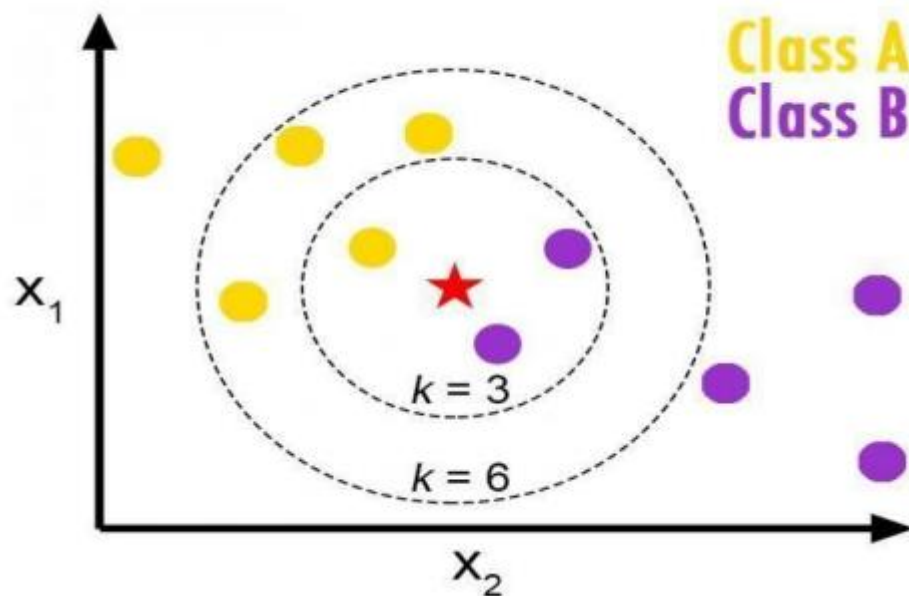
Ký hiệu  $\beta$  đại diện cho hệ số hồi quy. Mô hình logit có thể đảo ngược tính toán các giá trị hệ số này khi ta cho nó một tập dữ liệu thực nghiệm đủ lớn có các giá trị đã xác định của cả hai biến phụ thuộc và biến độc lập.

## 2.4. K-NEAREST NEIGHBORS

Ý tưởng của của phương pháp học K-Nearest Neighbors (KNN) là dựa vào mối quan hệ của một ví dụ cần phân loại/hồi quy với các láng giềng gần nhất để gán giá trị của hàm mục tiêu (một nhãn lớp, hoặc một giá trị thực).

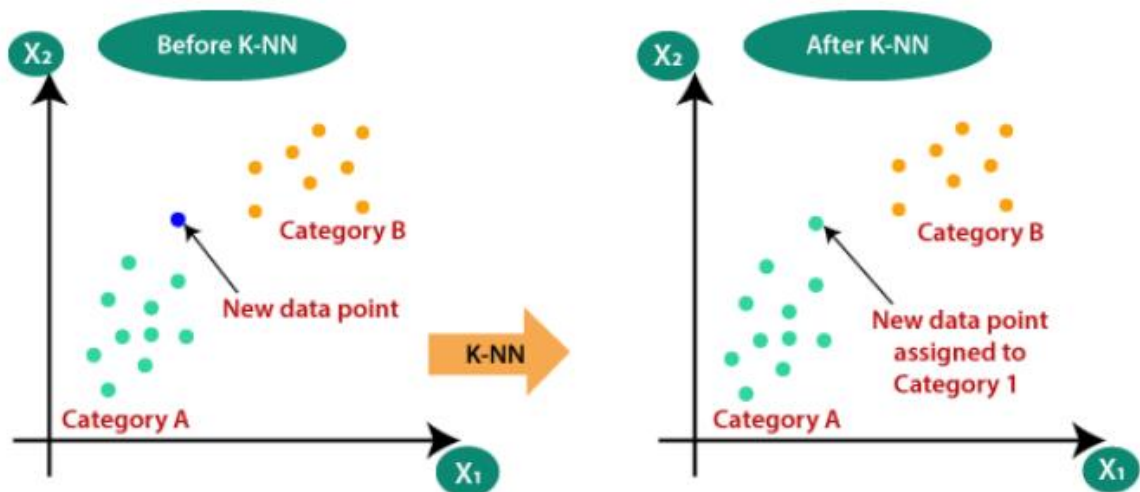
Với KNN, trong bài toán Classification, nhãn của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Nhãn của một test data có thể được quyết định bằng major voting (bầu chọn theo số

phieu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho các điểm gần nhất đó rồi suy ra nhãn.



Các bước hoạt động của K-NN

- Bước 1: Chọn số K của những người hàng xóm
- Bước 2: Tính khoảng cách Euclide của K số hàng xóm
- Bước 3: Lấy K láng giềng gần nhất theo khoảng cách Euclide được tính toán.
- Bước 4: Trong số k lân cận này, đếm số điểm dữ liệu trong mỗi loại.
- Bước 5: Gán các điểm dữ liệu mới cho danh mục đó mà số lượng hàng xóm là tối đa.
- Bước 6: Mô hình đã sẵn sàng.



## 2.5. DECISION TREE

Mô hình cây quyết định là một mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán phân loại và dự báo của học có giám sát. Khác với những thuật toán khác trong học có giám sát, mô hình cây quyết định không tồn tại phương trình dự báo. Mọi việc chúng ta cần thực hiện đó là tìm ra một cây quyết định dự báo tốt trên tập huấn luyện và sử dụng cây quyết định này dự báo trên tập kiểm tra.

Trong thuật toán cây quyết định chúng ta có thể quan tâm tới một số tham số có thể được sử dụng để tuning.

`DecisionTreeClassifier(*, criterion = 'gini', splitter = 'best', max_depth = None, min_samples_split = 2, min_samples_leaf = 1, max_features = None, max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split = None)`

Trong đó:

- **criterion:** Là hàm số để đo lường chất lượng phân chia ở mỗi node. Có hai lựa chọn là gini và entropy.
- **max\_depth:** Độ sâu tối đa cho một cây quyết định. Đối với mô hình bị quá khớp thì chúng ta cần giảm độ sâu và vị khớp thì gia tăng độ sâu.
- **min\_samples\_split:** Kích thước mẫu tối thiểu được yêu cầu để tiếp tục phân chia đối với node quyết định. Được sử dụng để tránh kích thước của node lá quá nhỏ nhằm giảm thiểu hiện tượng quá khớp.

- `max_features`: Số lượng các biến được lựa chọn để tìm kiếm ra biến phân chia tốt nhất ở mỗi lượt phân chia.
- `max_leaf_nodes`: Số lượng các node lá tối đa của cây quyết định. Thường được thiết lập khi muốn kiểm soát hiện tượng quá khớp.
- `min_impurity_decrease`: Chúng ta sẽ tiếp tục phân chia một node nếu như sự suy giảm của độ tinh khiết nếu phân chia lớn hơn ngưỡng này.
- `min_impurity_split`: Ngưỡng dừng sớm để kiểm soát sự gia tăng của cây quyết định. Thường được sử dụng để tránh hiện tượng quá khớp. Chúng ta sẽ tiếp tục chia node nếu độ tinh khiết cao hơn ngưỡng này.

## 2.6. RANDOM FOREST

Cơ sở lý thuyết của Random Forest (Rừng ngẫu nhiên) dựa trên ý tưởng của Ensemble Learning (học tập tập hợp), trong đó nhiều mô hình học máy được kết hợp để tạo thành một mô hình mạnh hơn.

Random Forest là một thuật toán học máy giám sát được sử dụng cho các tác vụ phân loại và dự đoán. Nó sử dụng một tập hợp các cây quyết định (decision trees) để tạo ra dự đoán cuối cùng. Mỗi cây quyết định trong Rừng ngẫu nhiên được xây dựng dựa trên một mẫu dữ liệu ngẫu nhiên và một phương pháp gọi là "bootstrap aggregating" (bagging).

Dưới đây là các bước cơ bản để xây dựng một Random Forest:

**Bước 1: Chuẩn bị tập dữ liệu huấn luyện:** Random Forest yêu cầu một tập dữ liệu huấn luyện có các đặc trưng (features) và nhãn (labels) tương ứng cho các ví dụ huấn luyện.

**Bước 2: Xây dựng cây quyết định:** Mỗi cây quyết định trong Random Forest được xây dựng bằng cách chọn một mẫu ngẫu nhiên từ tập dữ liệu huấn luyện. Cây quyết định được xây dựng bằng cách chia tập dữ liệu thành các nhánh dựa trên các quy tắc tối ưu như chỉ số Gini hoặc độ tương đồng.

**Bước 3: Xây dựng Rừng ngẫu nhiên:** Rừng ngẫu nhiên được tạo bằng cách xây dựng nhiều cây quyết định (số lượng được định trước) thông qua việc lặp lại bước 2. Mỗi cây quyết định trong Rừng ngẫu nhiên có thể được xây dựng độc lập với nhau.

Bước 4: Dự đoán: Khi Rừng ngẫu nhiên đã được xây dựng, nó có thể được sử dụng để dự đoán nhãn cho các dữ liệu mới. Dự đoán được thực hiện bằng cách tính toán trung bình hoặc đa số phiếu bầu của các dự đoán từ các cây quyết định.

Cơ sở lý thuyết của Random Forest nằm ở khả năng kết hợp dự đoán của nhiều cây quyết định để tạo ra dự đoán cuối cùng. Việc kết hợp các cây quyết định giúp giảm thiểu tác động của nhiễu và overfitting (quá khớp) trong mô hình. Đồng thời, việc sử dụng các mẫu dữ liệu ngẫu nhiên và phương pháp bagging giúp tăng tính đa dạng và khả năng tổng quát hóa của Rừng ngẫu nhiên.

Random Forest cũng cung cấp các độ đo như độ quan trọng của đặc trưng (feature importance) để đánh giá tầm quan trọng của các đặc trưng trong quá trình dự đoán. Nó cũng có thể được sử dụng để ước lượng độ chính xác và thực hiện việc phân loại, dự đoán, hoặc xác định mức độ không chắc chắn của dự đoán.

Tổng quan, Random Forest là một thuật toán mạnh mẽ trong lĩnh vực học máy, nó kết hợp các cây quyết định để tạo ra dự đoán đáng tin cậy và ổn định.

## CHƯƠNG 3. QUÁ TRÌNH THỰC NGHIỆM VÀ KẾT QUẢ ĐẠT ĐƯỢC

Toàn bộ các quá trình thực nghiệm được thực hiện thông qua dịch vụ Colaboratory – sản phẩm của Google Research.

### 3.1. TRỰC QUAN HOÁ VÀ PHÂN TÍCH VÀ DỮ LIỆU

#### 3.1.1 Thống kê dữ liệu

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122410 entries, 0 to 122409
Data columns (total 97 columns):
#   Column                Non-Null Count  Dtype
---  -
0   time_left              122410 non-null float64
1   ct_score               122410 non-null float64
2   t_score                122410 non-null float64
3   map                    122410 non-null object
4   bomb_planted           122410 non-null bool
5   ct_health              122410 non-null float64
6   t_health               122410 non-null float64
7   ct_armor               122410 non-null float64
8   t_armor                122410 non-null float64
9   ct_money                122410 non-null float64
10  t_money                 122410 non-null float64
11  ct_helmets              122410 non-null float64
12  t_helmets               122410 non-null float64
13  ct_defuse_kits          122410 non-null float64
14  ct_players_alive        122410 non-null float64
15  t_players_alive         122410 non-null float64
16  ct_weapon_ak47          122410 non-null float64
17  t_weapon_ak47           122410 non-null float64
18  ct_weapon_aug            122410 non-null float64
19  t_weapon_aug             122410 non-null float64
20  ct_weapon_awp            122410 non-null float64
21  t_weapon_awp            122410 non-null float64
22  ct_weapon_bizon          122410 non-null float64
23  t_weapon_bizon           122410 non-null float64
```

Hình 2 Kiểm tra bộ dữ liệu

Bộ dữ liệu bao gồm 122410 hồ sơ trận đấu và 97 thuộc tính biểu diễn cho mỗi một vòng đấu. Nguồn từ [CS:GO Round Winner Classification | Kaggle](#).

Các thuộc tính được phân thành bốn loại chính:

- Các cột liên quan đến trận đấu như bản đồ, tỉ số.

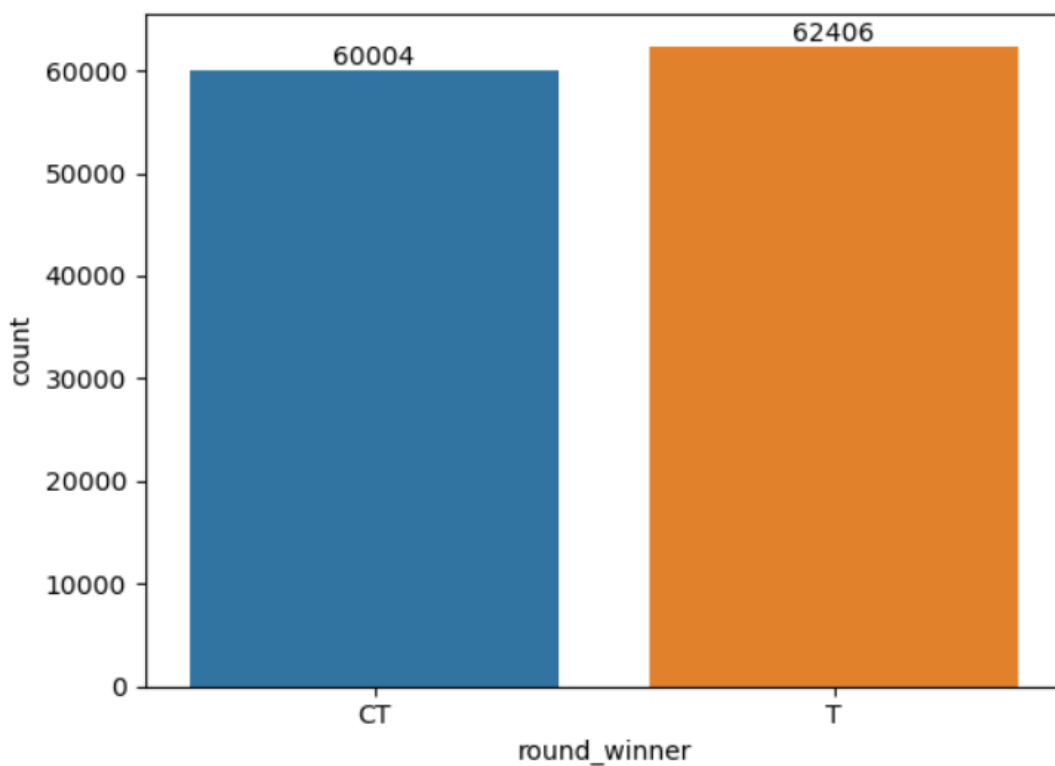
- Các cột liên quan đến vòng đấu đó như thời gian, bom đã đặt hay chưa.
- Các cột liên quan đến người chơi như trang bị, máu, số tiền hiện có.
- Cột “round\_winner” gồm hai object là T và CT hiện thị kết quả bên T hoặc CT thắng vòng đấu đó.

```
data.isnull().sum()

time_left      0
ct_score       0
t_score        0
map            0
bomb_planted   0
..
ct_grenade_molotovgrenade  0
t_grenade_molotovgrenade  0
ct_grenade_decoygrenade   0
t_grenade_decoygrenade   0
round_winner            0
Length: 97, dtype: int64
```

Trong tập dữ liệu không chứa giá trị null.

Sau đó nhóm sử dụng hàm *countplot* của thư viện Seaborn để trực quan hoá dữ liệu:



Hình 3 Phân bố số lượng round thắng của từng bên

Ta có thể thấy rằng số lượng mẫu thử khá là cân bằng.

## 3.2. TIỀN XỬ LÝ DỮ LIỆU

### 3.2.1 Xử lý dữ liệu dạng phân loại

Trong tập dữ liệu chứa một số trường thuộc dạng dữ liệu phân loại (categorical data) như `round_winner`, `map`. Các mô hình học máy thường chỉ nhận các đầu vào là các giá trị dạng số (numerics). Các mô hình này sẽ đánh giá một trọng số cho các thuộc tính này (như với mô hình Logistic Regression) hoặc tính toán khoảng cách giữa các ví dụ (như với mô hình KNN).

Vì vậy để huấn luyện và sử dụng các mô hình này, các trường dữ liệu phân loại buộc phải được đưa về dạng số. Tuy nhiên, điều này có thể không cần thiết với tất cả các mô hình học máy, ví dụ với mô hình cây quyết định, nếu đủ sâu, có thể xử lý được các dạng dữ liệu phân loại.

Đầu tiên ta chuyển cột `round_winner` từ giá trị dạng object T/CT thành dạng nhị phân 0 và 1:

```
df["bomb_planted"] = df["bomb_planted"].astype(np.int16)
df["round_winner"] = df["round_winner"].replace({"T":0, "CT":1})
```

Tiếp đó ta chuyển cột `map` từ các giá trị object là tên của từng map thành các số nguyên từ 0 đến 7 trong một `map_label` và dùng hàm `apply` để áp dụng phép ánh xạ này:

```
map = ['de_dust2', 'de_mirage', 'de_nuke', 'de_inferno', 'de_overpass',
       'de_vertigo', 'de_train', 'de_cache']
map_label = {}

for i in range(len(map)):
    map_label[map[i]] = i
df["map"] = df['map'].apply(lambda x : map_label[x])
```

Sau khi đã chuyển hết dữ liệu về dạng số, ta sử dụng phương pháp scaling data để đưa giá trị của các thuộc tính đầu vào về cùng một phạm vi, giúp cho các mô hình học máy hoạt động một cách hiệu quả hơn. Ở đây ta sẽ sử dụng `RobustScaler`. `RobustScaler` là một phương pháp scaling dựa trên phần vị (percentile), giúp giảm ảnh hưởng của các giá trị ngoại lệ (outliers). Tiếp theo sử dụng `scaler.fit_transform` để thực hiện scaling trên dữ liệu đầu vào `X`. Kết quả trả về là một `DataFrame` mới (được gán lại vào `X`) có



cùng số cột và tên cột như X ban đầu, nhưng các giá trị đã được chuyển đổi theo phép scaling:

```
scaler = RobustScaler()  
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

Cuối cùng, ta chia tập dữ liệu thành 2 phần là train và test với tỉ lệ 8:2 :

```
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True, train_size=0.8, random_state=42)
```

### 3.3. PHƯƠNG PHÁP ĐÁNH GIÁ

Trong quá trình thực nghiệm, nhóm so sánh độ chính xác giữa các mô hình học máy dựa trên bộ các thuộc tính đã được chọn. Độ chính xác của các mô hình được tính toán sử dụng 5-fold cross-validation. Nhóm sử dụng hàm đánh giá từ thư viện scikit-learn là *sklearn.model\_selection.cross\_val\_score*

Độ chính xác (accuracy) là tỉ lệ giữa số điểm được phân loại đúng và tổng số điểm. Độ chính xác sẽ phù hợp hơn với các bài toán mà kích thước các lớp dữ liệu là tương đối như nhau. Khi kích thước các lớp dữ liệu là chênh lệch (imbalanced data hay skew data), precision và recall thường được sử dụng.


### 3.4. HUẤN LUYỆN MÔ HÌNH HỌC MÁY


Nhóm sử dụng các mô hình Neuron Network, Logistic Regression, K-Nearest Neighbors và Decision Tree

#### 3.4.1 Neuron Network

Sử dụng thư viện Tensorflow để cài đặt mô hình:

```
[ ] new_model = Sequential()  
    new_model.add(Dense(128, activation='relu', input_shape=(90,)))  
    new_model.add(Dense(256, activation='relu'))  
    new_model.add(Dense(64, activation='relu'))  
    new_model.add(Dense(16, activation='relu'))  
    new_model.add(Dense(2, activation='sigmoid'))  
  
    new_model.compile(optimizer='Adam', loss=tf.keras.losses.CategoricalCrossentropy(), metrics=['accuracy'])
```

 new\_model.summary()

 Model: "sequential\_2"

Layer (type)	Output Shape	Param #
=====		
dense_10 (Dense)	(None, 128)	11648
dense_11 (Dense)	(None, 256)	33024
dense_12 (Dense)	(None, 64)	16448
dense_13 (Dense)	(None, 16)	1040
dense_14 (Dense)	(None, 2)	34
=====		
Total params: 62,194		
Trainable params: 62,194		
Non-trainable params: 0		

Mô hình được cài đặt với 5 layer, trong đó có 1 input layer, 3 hidden layer và 1 output layer với kiểu lớp, số lượng neuron, hàm kích hoạt được mô tả như hình ở trên.

Ở đây, ta sử dụng hàm mất mát Cross-Entropy và thuật toán Adam (Adaptive Moment Estimation) để tối ưu hóa.

Sau đó, ta biến đổi tập đích y về dạng one-hot encoding để có thể tích hợp với hàm mất mát Crossentropy và giúp cung cấp thông tin về đầu ra rõ ràng giúp tối ưu hóa mô hình:

```
enc = OneHotEncoder()  
y_train_onehot = enc.fit_transform(np.array(y_train).reshape(-1, 1)).toarray()  
y_test_onehot = enc.fit_transform(np.array(y_test).reshape(-1, 1)).toarray()
```

Bây giờ, ta sẽ train mô hình Neuron network với batch\_size là 32 và chạy 20 lần epochs:

```
tf.config.run_functions_eagerly(True)
hist = new_model.fit(X_train, y_train_onehot, batch_size = 32, epochs = 20, validation_data=(X_test, y_test_onehot))
```

### 3.4.2 Naïve Bayes

Sử dụng mô hình GaussianNB có sẵn của sklearn:

```
NB = GaussianNB()
NB.fit(X_train, y_train)
y_predict = NB.predict(X_test)
accuracy1 = accuracy_score(y_test, y_predict)
accuracy1
```

### 3.4.3 Logistic Regression

Sử dụng mô hình có sẵn của sklearn:

```
LR = LogisticRegression(max_iter = 1000)
LR.fit(X_train, y_train)
y_predict = LR.predict(X_test)
accuracy2 = accuracy_score(y_test, y_predict)
accuracy2
```

### 3.4.4 Decision Tree

Sử dụng mô hình có sẵn của sklearn:

```
DTC = DecisionTreeClassifier(random_state=0)
DTC.fit(X_train, y_train)
y_predict = DTC.predict(X_test)
accuracy3 = accuracy_score(y_test, y_predict)
accuracy3
```

### 3.4.5 K-Nearest Neighbors

Sử dụng mô hình có sẵn của sklearn, cài đặt tham số `n_neighbors = 3`, tức là KNN sẽ tìm 3 hàng xóm của mỗi điểm dữ liệu mới và sử dụng đa số nhãn của các hàng xóm đó để dự đoán cho điểm dữ liệu đó.

```
KNN = KNeighborsClassifier(n_neighbors=3)
KNN.fit(X_train, y_train)
y_predict = KNN.predict(X_test)
accuracy4 = accuracy_score(y_test, y_predict)
print(accuracy4)
```

### 3.4.6 Random forest

Sử dụng mô hình có sẵn của sklearn:

```
RF = RandomForestClassifier(random_state=0)
RF.fit(X_train, y_train)
y_predict = RF.predict(X_test)
accuracy5 = accuracy_score(y_test, y_predict)
accuracy5
```

## CHƯƠNG 4. KẾT QUẢ, ĐÁNH GIÁ

Accuracy: tính toán bằng cách lấy số dự đoán đúng chia cho toàn bộ các dự đoán.

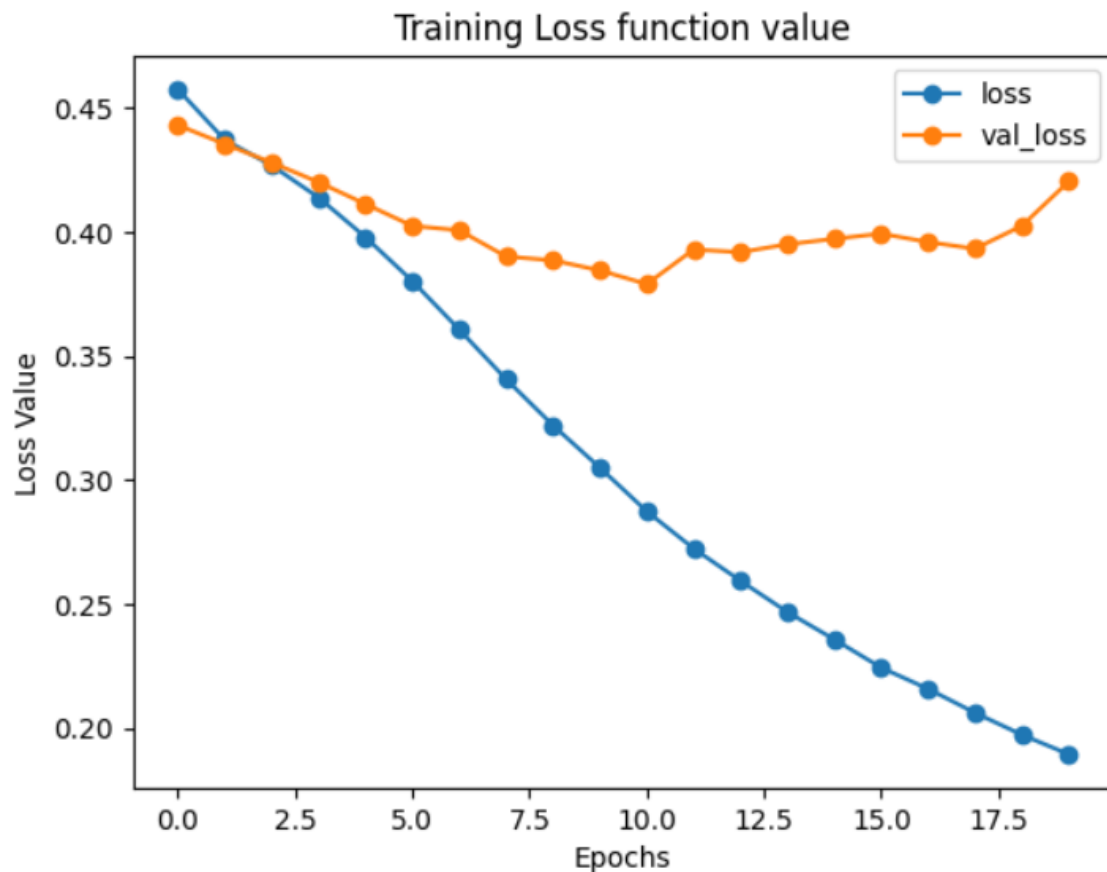
Precision: cho biết thực sự có bao nhiêu dự đoán Positive là thật sự True.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Recall: đo lường tỷ lệ dự báo chính xác các trường hợp Positive trên toàn bộ các mẫu thuộc nhóm Positive.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

### 4.1. NEURON NETWORK

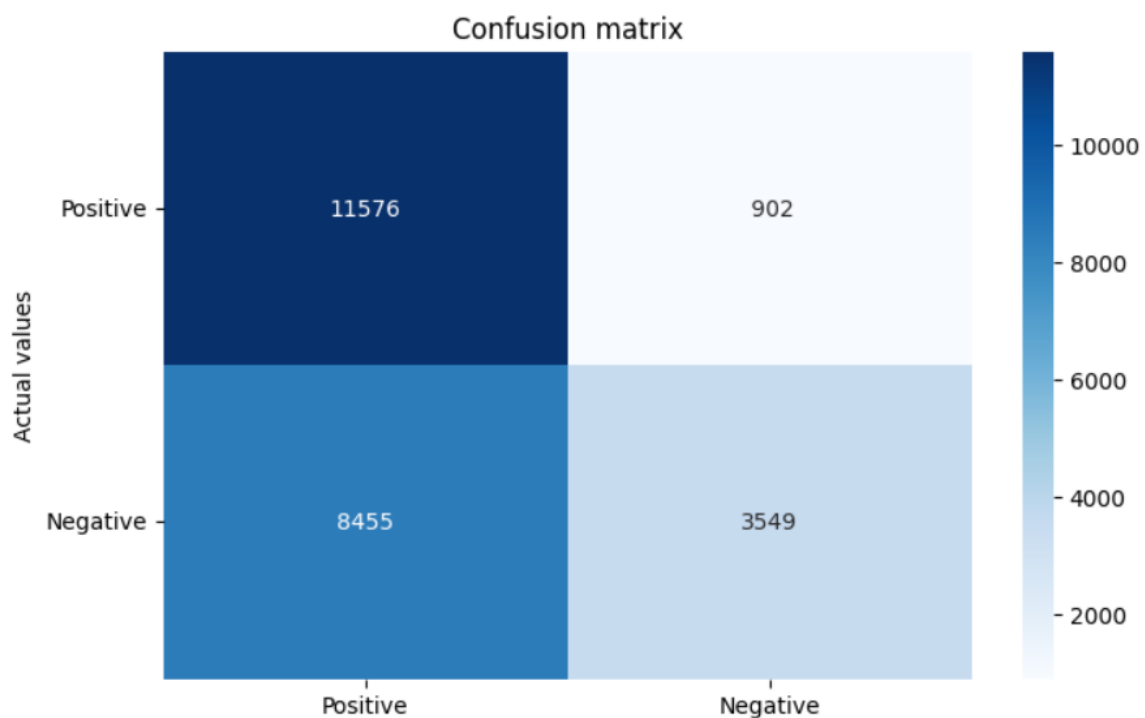


### Nhận xét:

- Mô hình Neuron Network hoạt động khá tốt, nhưng đòi hỏi thời gian train lâu, nhất là với số lượng epochs lớn.
- Dễ bị overfitting trong quá trình huấn luyện.
- Cần lựa chọn số lượng layer, số neuron của mỗi layer và hàm kích hoạt của mỗi layer sao cho phù hợp.

## 4.2. NAÏVE BAYES

	precision	recall	f1-score	support
0	0.58	0.93	0.71	12478
1	0.80	0.30	0.43	12004
accuracy			0.62	24482
macro avg	0.69	0.61	0.57	24482
weighted avg	0.69	0.62	0.57	24482

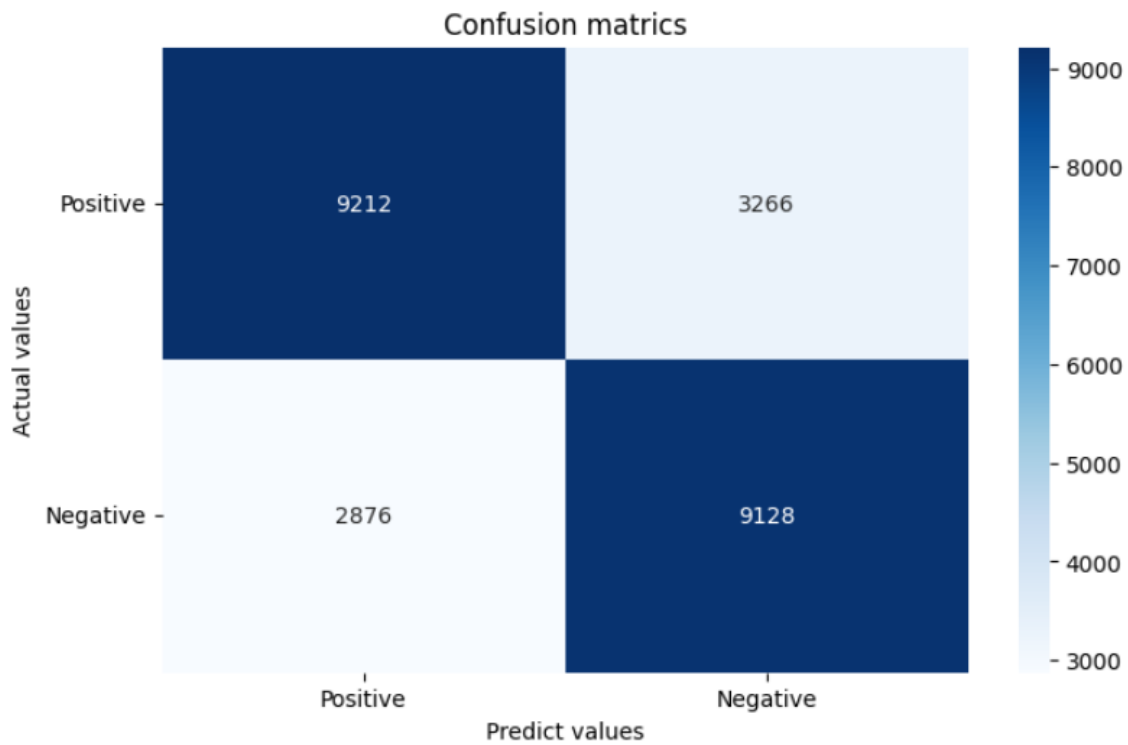


### Nhận xét:

- Ở đây ta có thể thấy rằng mô hình Naïve Bayes khá tệ (đa phần đều dự đoán kết quả là T thắng). Điều này có thể giải thích là do dữ liệu huấn luyện là dữ liệu không có dạng phân phối chuẩn và có nhiều đặc trưng khiến cho việc tính toán trở nên khó khăn với mô hình như Naïve Bayes.

### 4.3. LOGISTIC REGRESSION

	precision	recall	f1-score	support
0	0.76	0.74	0.75	12478
1	0.74	0.76	0.75	12004
accuracy			0.75	24482
macro avg	0.75	0.75	0.75	24482
weighted avg	0.75	0.75	0.75	24482

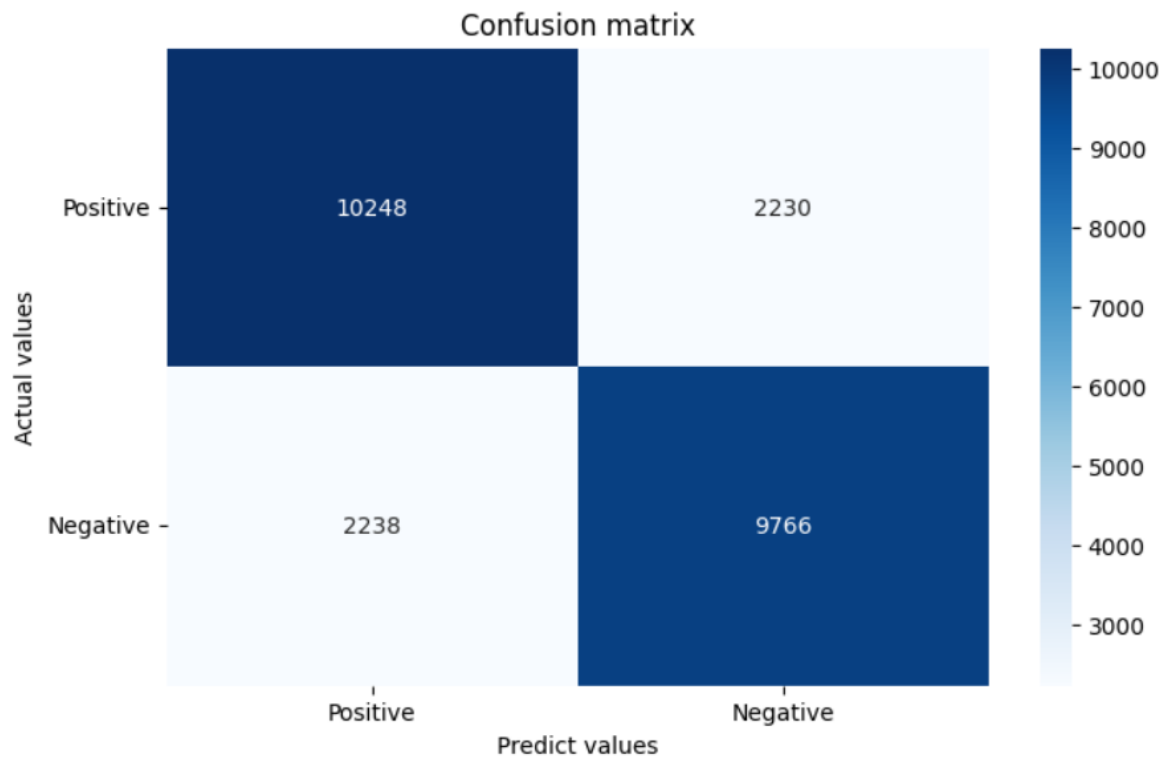


#### Nhận xét:

- Cũng như Naïve Bayes, logistic regression là mô hình đơn giản. Việc có quá nhiều đặc trưng khiến cho việc tối ưu hóa loss function trở nên khó khăn và xảy ra mất mát cao, nên độ chính xác vào ở mức vừa phải. Nếu có thể giảm số chiều của dữ liệu đầu vào thì sẽ tăng độ chính xác, nhưng việc này khá khó khăn.

#### 4.4. DECISION TREE

	precision	recall	f1-score	support
0	0.82	0.82	0.82	12478
1	0.81	0.81	0.81	12004
accuracy			0.82	24482
macro avg	0.82	0.82	0.82	24482
weighted avg	0.82	0.82	0.82	24482



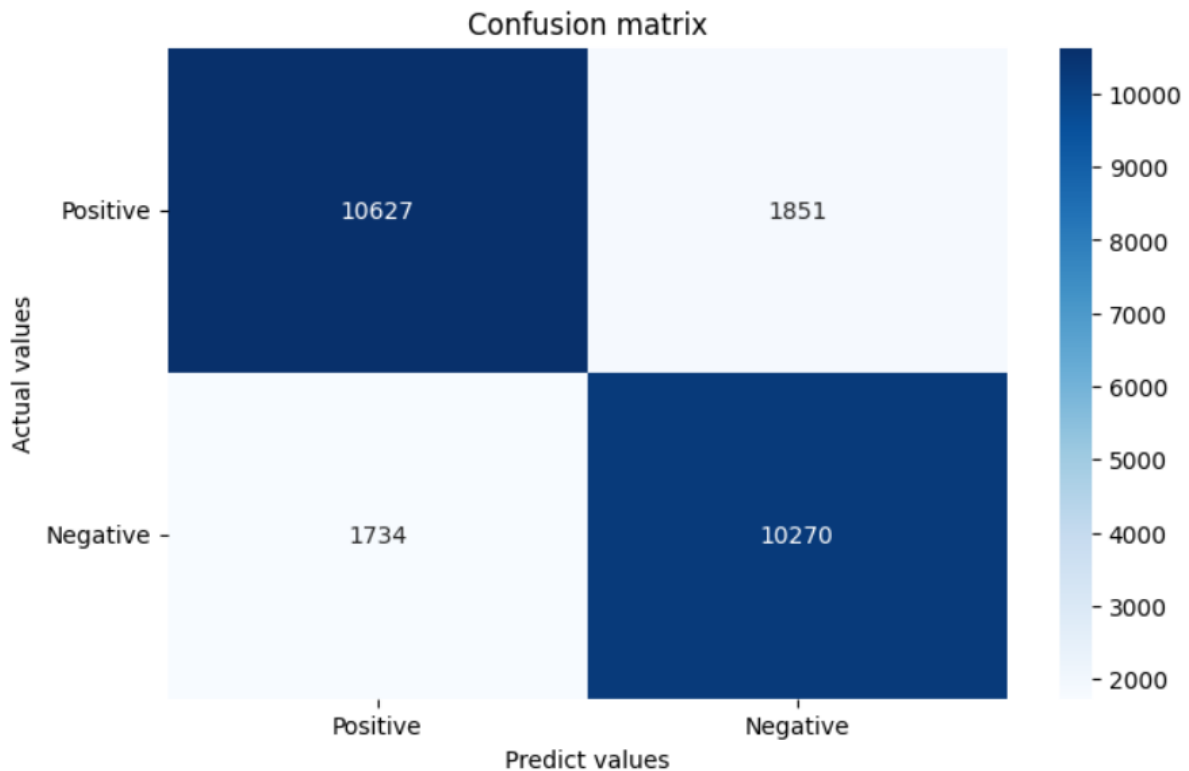
#### Nhận xét:

- Độ chính xác của cây quyết định khá tốt, có thể là vì kết quả dựa trên khá nhiều yếu tố nên mô hình tìm được ngưỡng để phân loại tốt.
- Tuy nhiên nếu cây quá sâu hoặc mô hình quá phức tạp có thể dẫn đến overfitting.



## 4.5. K-NEAREST NEIGHBOR

	precision	recall	f1-score	support
0	0.86	0.85	0.86	12478
1	0.85	0.86	0.85	12004
accuracy			0.85	24482
macro avg	0.85	0.85	0.85	24482
weighted avg	0.85	0.85	0.85	24482

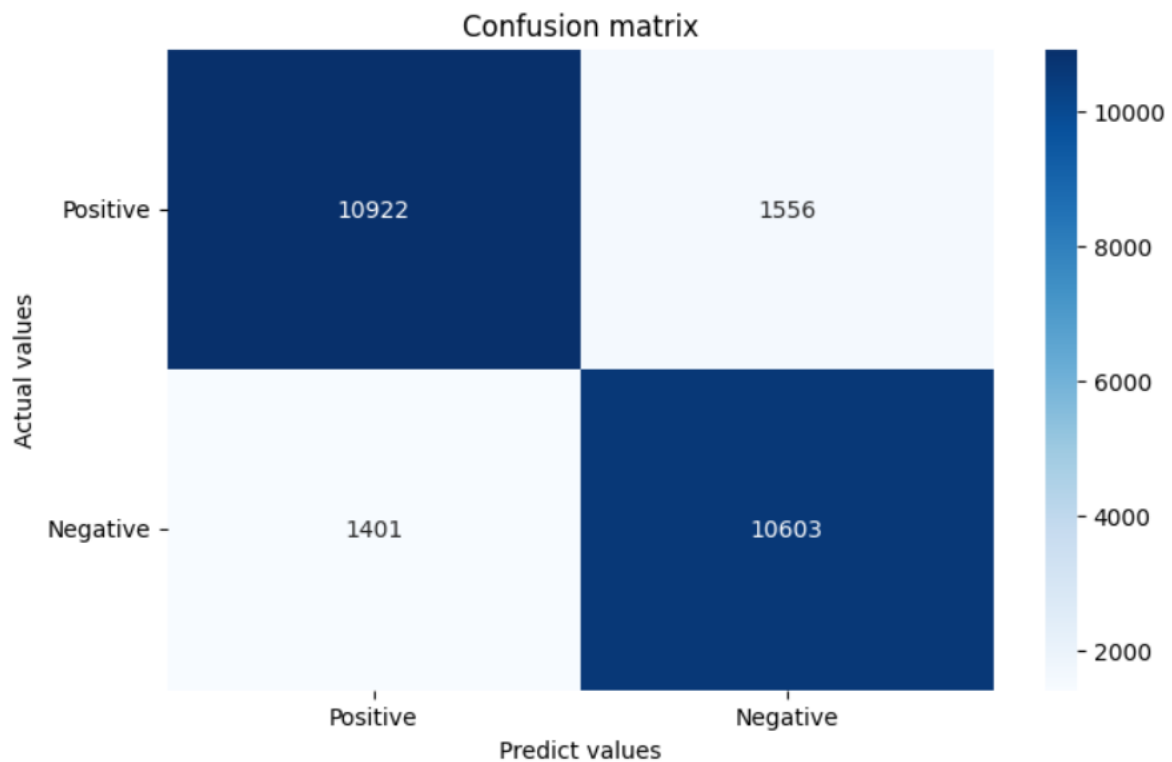


### Nhận xét:

- Mô hình hoạt động khá tốt, có thể là do dữ liệu lớn và có nhiều đặc trưng. Việc có nhiều đặc trưng cung cấp cho mô hình nhiều thông tin hơn về mối quan hệ giữa các điểm dữ liệu và tăng khả năng tìm ra các điểm lân cận gần nhất. Tuy rằng có nhiều đặc trưng sẽ dễ dẫn đến “sự thừa thớt”, nhưng do số lượng dữ liệu lớn nên đã tránh được việc này.

## 4.6. RANDOM FOREST

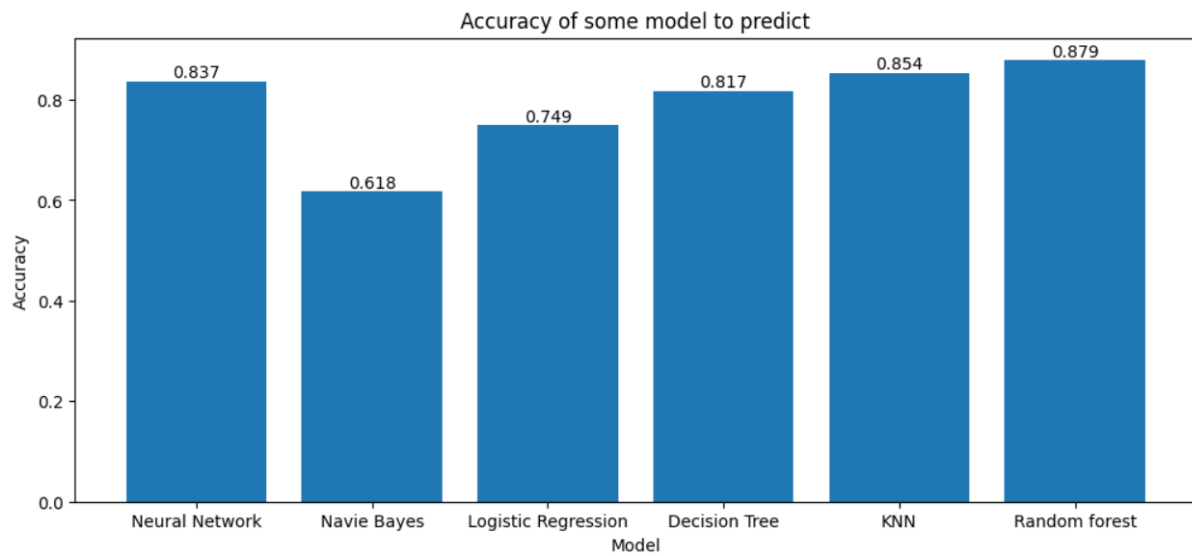
	precision	recall	f1-score	support
0	0.89	0.88	0.88	12478
1	0.87	0.88	0.88	12004
accuracy			0.88	24482
macro avg	0.88	0.88	0.88	24482
weighted avg	0.88	0.88	0.88	24482



### Nhận xét:

- Mô hình hoạt động tốt nhất. Có thể là do random forest kết hợp nhiều cây quyết định để đưa ra dự đoán cuối cùng, và vì decision tree hoạt động tốt nên random forest cũng sẽ tốt theo. Ngoài ra random forest dựa trên nhiều cây nên có thể giảm thiểu tình trạng overfitting và đưa ra dự đoán chính xác hơn decision tree.

## 4.7. NHẬN XÉT CHUNG



Tổng kết, mô hình Neuron network, KNN và random forest cho thấy độ chính xác tốt trên tập dữ liệu ta đưa ra. Các mô hình còn lại có độ chính xác thấp hơn có thể là do mô hình không phù hợp hoặc khả năng mô hình hóa các dữ liệu phức tạp của các mô hình này không tốt trong trường hợp này.

## **CHƯƠNG 5. KẾT LUẬN VÀ KINH NGHIỆM RÚT RA**

### **5.1. CÁC VẤN ĐỀ, KHÓ KHĂN GẶP PHẢI VÀ PHƯƠNG HƯỚNG GIẢI QUYẾT**

Trong giai đoạn đầu khi giải quyết bài toán này, nhóm chúng em có ý tưởng là sẽ chỉ sử dụng một phương pháp học máy, tuy nhiên sau khi xây dựng xong chương trình giải quyết bài toán sử dụng phương pháp đó, chúng em nhận ra một vấn đề là không biết liệu đây đã là phương pháp tốt nhất cho bài toán này hay chưa, có phương pháp nào đem lại hiệu quả tốt hơn không. Do đó, chúng xem quyết định chuyển hướng từ sử dụng một phương pháp để giải quyết bài toán của mình thành việc sẽ sử dụng nhiều phương pháp khác nhau, từ đó chúng em có thể đánh giá được độ hiệu quả của từng thuật toán khác nhau.

### **5.2. CÁC KHÁM PHÁ VÀ KẾT LUẬN**

Việc xây dựng chương trình giải quyết bài toán này đã giúp chúng em có nhiều khám phá và kết luận mới đối với bản thân:

- Việc xử lý dữ liệu tốt sẽ góp phần to lớn tăng hiệu năng của chương trình.
- Mỗi một bài toán cụ thể sẽ có nhiều phương pháp khác nhau để giải quyết, tuy nhiên mỗi một phương pháp lại có điểm mạnh yếu khác nhau nên cần có một cái nhìn khách quan để có thể chọn được thuật toán phù hợp.
- Việc quan sát kĩ data trước khi đưa vào mô hình sẽ giúp chúng ta có thể lựa chọn các tham số cho mô hình một cách phù hợp hơn.
- Ngoài ra việc lựa chọn phương pháp đánh giá phù hợp cũng rất quan trọng