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

o0o

MÔ HÌNH HỌC MÁY DỰ ĐOÁN KẾT QUẢ CỦA TRÒ CHƠI ĐẤU TRƯỜNG CHÂN LÝ

Học phần: Học Máy

Sinh viên thực hiện: Nguyễn Thị Hà Phương - 21000245
Nguyễn Thành Trung - 21000708
Hoàng Tuấn Tú - 21000709
Lớp: K66A2 Toán tin

Hà Nội - 2024

Mục lục

Mở đầu

Chương 1

Giới thiệu

1.1 Tổng quan về trò chơi Teamfight Tactics

Teamfight Tactics (TFT) là một trò chơi chiến thuật dựa trên lượt chơi được phát triển bởi Riot Games. Trong TFT, người chơi cạnh tranh với nhau thông qua việc xây dựng và tối ưu hóa đội hình của mình để chiến đấu trong một loạt các trận đấu tự động. Mục tiêu chính là để là người cuối cùng sống sót, điều này đòi hỏi khả năng chiến lược và phán đoán về cách xếp đội hình và lựa chọn nhân vật.

Hình 1.1: Giai đoạn chuẩn bị cơ bản trong trò chơi TFT

1.1.1 Hệ thống tài nguyên cần biết của trò chơi

Tướng (Units)

Có 57 vị Tướng trong một mùa giải. Mỗi Tướng đều có các chỉ số và Kỹ Năng riêng. Khi có được, Tướng khởi điểm với 1 Sao. Tướng có thể nâng cấp lên Cấp độ Sao tối đa là 3. Ba Tướng 1 Sao giống nhau có thể được kết hợp để tạo ra Tướng 2 Sao mạnh hơn. Ba Tướng 2 Sao giống nhau có thể được kết hợp để tạo ra Tướng 3 Sao mạnh hơn nữa. Các Tướng tự động kết hợp để lên Cấp khi người chơi có ba Tướng giống nhau, cùng Cấp Sao và tất cả các Tướng đều đang không tham gia chiến đấu.

Trang bị (Items)

Có 9 trang bị thành phần duy nhất. Kết hợp hai trang thành phần sẽ tạo ra một trang bị mới, với tổng cộng 54 vật phẩm mới duy nhất có thể. Người chơi có thể tăng sức mạnh cho Tướng bằng trang bị, tối đa 3 trang bị trên một Tướng.

Tộc/Hệ (Traits)

Mỗi Tộc/Hệ mang các hiệu ứng khác nhau. Mỗi Tướng thường sẽ có 2 hoặc 3 Tộc/Hệ đi kèm. Tộc/Hệ chỉ được kích hoạt nếu như đạt đủ mốc kích hoạt, hay trên sàn đấu phải có đủ số lượng Tướng/đơn vị được triệu hồi khác nhau mang Tộc/Hệ đó (ví dụ: toàn đội nhận được khả năng kháng phép hoặc các tướng cùng Tộc/Hệ nhận thêm 200 máu).

Lõi năng cấp (Augments)

Các Lõi Nâng Cấp giúp thay đổi trận đấu của người chơi thông qua việc cung cấp nhiều loại bùa lợi khác nhau (chẳng hạn như sát thương tấn công bổ sung hoặc các chỉ số khác), nguồn cung cấp tài nguyên (chẳng hạn như Vàng, vật phẩm) và các phần bổ sung quy tắc (chẳng hạn như đạt vượt quá lãi suất, cấp độ tối đa hay tăng, giảm Máu của người chơi). Các Lõi Nâng Cấp xuất hiện 3 lần trong trò chơi, lần lượt ở vòng 2-1, 3-2 và 4-2. Các Lõi Nâng Cấp có độ hiếm từ Bạc, Vàng và Kim Cương, mỗi cấp độ hiếm sẽ cung cấp và mở khóa những lợi ích tiềm năng tốt hơn. Mỗi giai đoạn chọn Lõi Nâng Cấp, người chơi được lựa chọn một trong ba lõi để lựa chọn và người chơi có thể làm mới mỗi tùy chọn một lần.

1.1.2 Luật chơi

8 người chơi sử dụng các Tướng được mua trong Cửa Hàng, sắp xếp đội hình một cách hợp lý trên Chiến Trường để chiến đấu với các người chơi còn lại.

Nhân vật người chơi trong trò chơi là các Linh Thú, khởi điểm với 100 Máu và sẽ bị tiêu diệt khi Máu giảm về 0.

Linh Thú người chơi nhận được một ít Kinh Nghiệm sau mỗi Vòng Đấu. Khi đủ Kinh Nghiệm, Linh Thú sẽ tăng cấp. Linh Thú khởi đầu với cấp độ 1 và cấp độ tối đa là 10.

Cấp độ của Linh Thú càng cao sẽ tăng số lượng Tướng có thể đặt trên Chiến Trường, đồng thời tăng tỉ lệ xuất hiện các Tướng đắt tiền trong Cửa Hàng.

Trò chơi kết thúc khi Linh Thú người chơi bị hạ gục (đối với riêng người chơi đó) hoặc tìm ra được người chiến thắng (đối với toàn bộ người chơi). Người chiến thắng là: i) Linh Thú duy

nhất còn sống sót, hoặc ii) Linh Thú có số Máu còn lại là lớn nhất (số âm) nếu tất cả Linh Thú đều bị đánh bại.

1.2 Mục tiêu và thách thức

1.2.1 Mục tiêu và câu hỏi của dự án

Mục tiêu của dự án này là để khám phá và phân tích liệu các kỹ thuật học máy có thể dự đoán kết quả của các trận đấu trong Teamfight Tactics một cách chính xác hay không. Cụ thể, báo cáo này tập trung vào việc sử dụng và so sánh hiệu quả của nhiều mô hình học máy khác nhau, từ mô hình học máy có giám sát để triển khai dự đoán biến liên tục (thứ hạng trận đấu) hoặc triển khai các mô hình phân loại để dự đoán một biến rời rạc (thắng/thua), để đánh giá xem mô hình nào cung cấp độ chính xác cao nhất trong việc dự đoán. Câu hỏi nghiên cứu chính được đề cập như sau:

- Các kỹ thuật học máy hiện có có hiệu quả trong việc dự đoán kết quả của các trận đấu Teamfight Tactics không?
- Liệu có thể phát triển một mô hình tạo ra một đội hình luôn thắng cuộc không?

1.2.2 Thách thức

- **Định dạng tập dữ liệu:** Tập dữ liệu không chỉ đa chiều (Tướng, trang bị, lõi nâng cấp, vàng,...) mà còn khá phức tạp bởi các biến liên kết với nhau (ví dụ, một tướng có thể sở hữu đến 3 trang bị và 3 Tộc/Hệ). Điều này dẫn đến khó khăn trong việc tiền xử lý để tổ chức dữ liệu vào định dạng dữ liệu chuẩn là file CSV.
- **Trực quan hóa kết quả:** Bởi vì tập dữ liệu có tính chất đa chiều nên thách thức nằm ở việc biểu diễn kết quả, tạo ra biểu đồ dễ hiểu và dễ quan sát.
- **Xảy ra overfitting:** Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa. Khi đó, mô hình cần được điều chỉnh để tổng quát hóa chứ không chỉ đơn thuần ghi nhớ.

Chương 2

Cơ sở lý thuyết

2.1 Các mô hình học máy

2.1.1 Thuật toán Gaussian Naive Bayes

Thuật toán Naive Bayes (NB) được sử dụng cho việc phân loại và hoạt động trên các đặc trưng phân loại. Ưu điểm của thuật toán này là có thể phân loại dữ liệu một cách chính xác sau khi được huấn luyện trên một tập dữ liệu huấn luyện nhỏ. Mô hình này giả định rằng các các thuộc tính trong các đối tượng là độc lập, mặc dù thực tế các thuộc tính thường phụ thuộc lẫn nhau. Tuy nhiên, tập dữ liệu được sử dụng trong dự án này chứa các đặc trưng phân loại và số, đã được mã hóa và tỉ lệ hoá thành các biểu diễn số. Do đó, GNB sẽ được sử dụng vì nó hoạt động tốt trên các tập dữ liệu với các đặc trưng có giá trị thực.

Với mỗi chiều dữ liệu i và một lớp c, x_i tuân theo một phân phối chuẩn có kỳ vọng μ_{ci} và phương sai σ_{ci}^2:

$$p(x_i|c) = p(x_i|\mu_{ci}, \sigma_{ci}^2) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{(x_i - \mu_{ci})^2}{2\sigma_{ci}^2}\right) \tag{2.1}$$

Trong đó, bộ tham số $\theta = \{\mu_{ci}, \sigma_{ci}^2\}$ được xác định bằng Maximum Likelihood:

$$(\mu_{ci}, \sigma_{ci}^2) = \arg\max_{\mu_{ci}, \sigma_{ci}^2} \prod_{n=1}^{N} p(x_i^{(n)}|\mu_{ci}, \sigma_{ci}^2) \tag{2.2}$$

Đây là cách tính của thư viện sklearn. Chúng ta cũng có thể đánh giá các tham số bằng MAP nếu biết trước priors của μ_{ci} và σ_{ci}^2.

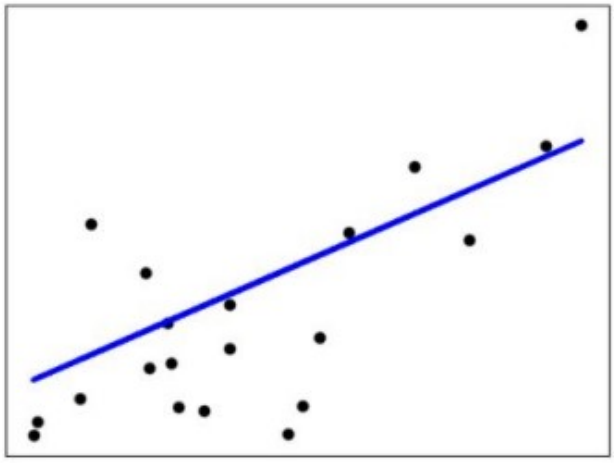
2.1.2 Thuật toán Multiple Linear Regression

Hồi quy tuyến tính bội là một trong những thuật toán cơ bản nhất trong học có giám sát, được sử dụng để dự đoán một biến phụ thuộc liên tục từ hai hoặc nhiều biến độc lập. Công thức của nó có thể được viết như sau:

$$y = b_0 + b_1 x + b_2 x_2 + ... + b_n x_n + \epsilon \tag{2.3}$$

trong đó

- y là biến phụ thuộc (biến dự đoán)
- $b_1, b_2, \ldots, b_n$ là các hằng số
- b_0 còn được gọi là bias
- $x_1, x_2, \ldots, x_n$ là các biến độc lập
- ϵ là sai số ngẫu nhiên



Hình 2.1: Đường thẳng dự đoán tốt nhất trong mô hình Linear Regression

Tuy nhiên có điểm hạn chế của thuật toán này là nó rất nhạy cảm với nhiễu. Nếu dữ liệu có sự biến động lớn không theo quy luật, kết quả mô hình sẽ kém chính xác. Vì vậy, trước khi thực hiện Linear Regression, các nhiễu (outlier) cần phải được loại bỏ.

2.1.3 Thuật toán Logistic Regression

Hồi quy logistic là một mô hình thống kê được sử dụng để mô hình hóa xác suất của một biến phụ thuộc nhị phân. Nó là sự lựa chọn phổ biến cho các bài toán phân loại nhị phân vì nó cho phép tính toán xác suất mà một quan sát cụ thể thuộc về một lớp nhất định. Hồi quy logistic làm việc trên nguyên tắc của hàm logistic, một hàm Sigmoid, cho phép nó đưa ra các dự đoán giữa 0 và 1, rất hữu ích cho việc ước lượng xác suất.

Mô hình dự báo $h_\theta(x)$ trường hợp này được chọn như sau:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2.4}$$

ở đây $g(z) = \frac{1}{1+e^{-z}}$ được gọi là hàm sigmoid, hay gọi chung là hàm logistic

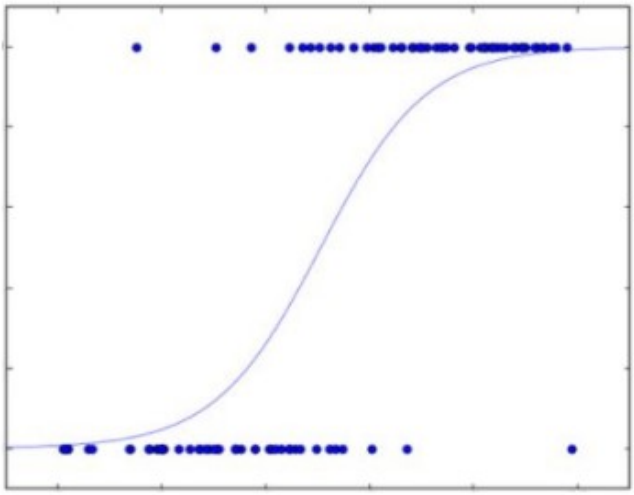
Do y chỉ nhận giá trị 0 hoặc 1 (xung khắc), ta có mô hình hồi quy logistic:

$$P(y = 1|x; \theta) = h_\theta(x) \tag{2.5}$$

$$P(y = 0|x; \theta) = 1 - h_\theta(x) \tag{2.6}$$

với $\theta \in \mathbb{R}^{d+1}$ là tham số của mô hình.

Giả sử đã biết vector tham số θ, ta sử dụng mô hình để phân loại như sau:



Hình 2.2: Đường cong sigmoid trong mô hình Logistic Regression

- Xếp đối tượng x vào lớp 1 nếu

$$P(y = 1|x; \hat{\theta}) > P(y = 0|x; \hat{\theta}) \Leftrightarrow h_{\hat{\theta}}(x) > \frac{1}{2} \Leftrightarrow \hat{\theta}^T x > 0.$$

- Ngược lại xếp x vào lớp 0.

2.1.4 Thuật toán Support Vector Machine

Support Vector Machine (SVMs) là các thuật toán học máy được sử dụng cho cả phân loại và hồi quy, rất hiệu quả khi xử lý các tập dữ liệu có số chiều lớn.

Với bài toán binary classification mà 2 classes là linearly separable, có vô số các siêu mặt phẳng giúp phân biệt hai classes, tức mặt phân cách. Với mỗi mặt phân cách, ta có một classifier. Khoảng cách gần nhất từ 1 điểm dữ liệu tới mặt phân cách ấy được gọi là margin của classifier đó.

Support Vector Machine là bài toán đi tìm mặt phân cách sao cho margin tìm được là lớn nhất, đồng nghĩa với việc các điểm dữ liệu an toàn nhất so với mặt phân cách.

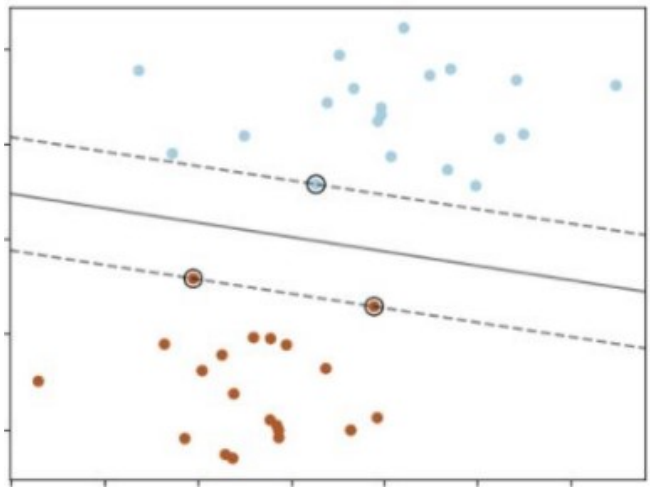
Bài toán tối ưu trong SVM là một bài toán lồi với hàm mục tiêu là stricly convex, nghiệm của bài toán này là duy nhất. Chúng ta có thể viết bài toán tối ưu trong SVM dưới dạng:

$$(w, w_0) = \max_{w,w_0} \frac{1}{||w||^2} \tag{2.7}$$

$$\text{subkect to: } y_i(w^T x_i + b) \geq 1, \forall i \tag{2.8}$$

trong đó:

- w là vector trọng số của siêu phẳng.
- b là hệ số điều chỉnh (bias).
- x_i và y_i là các điểm dữ liệu và nhãn tương ứng của chúng.



Hình 2.3: Minh họa nghiệm tìm được bởi SVM

2.2 Các chỉ số đánh giá

2.2.1 MSE (Mean Squared Error - Sai Số Bình Phương Trung Bình)

MSE là trung bình của bình phương các sai số giữa giá trị dự đoán và giá trị thực tế. MSE càng thấp cho thấy mô hình càng phù hợp với dữ liệu. MSE được tính theo công thức:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

trong đó y_i là giá trị thực tế và $\hat{y}_i$ là giá trị dự đoán. MSE thường được sử dụng để đánh giá các mô hình hồi quy.

2.2.2 R^2 Score (Hệ Số Xác Định)

R^2 là tỷ lệ phần trăm biến đổi của biến phụ thuộc được giải thích bởi mô hình hồi quy. R^2 càng cao cho thấy mô hình càng phù hợp với dữ liệu. R^2 được tính theo công thức:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{2.10}$$

trong đó $\overline{y}$ là giá trị trung bình của y_i.

2.2.3 Accuracy (Độ Chính Xác)

Độ chính xác là tỷ lệ số mẫu được phân loại chính xác trên tổng số mẫu. Độ chính xác được tính theo công thức:

$$Accuracy = \frac{\text{số mẫu được phân loại đúng}}{\text{tổng số mẫu}} \tag{2.11}$$

Độ chính xác thường được sử dụng để đánh giá các mô hình phân loại.

2.2.4 F1 Score

Điểm F1 là trung bình điều hòa của Precision và Recall, cung cấp một cái nhìn toàn diện về hiệu suất của mô hình, đặc biệt khi các lớp có sự mất cân bằng. F1 được tính theo công thức:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.12}$$

trong đó Precision là tỷ lệ số mẫu dương tính thực sự trên tổng số mẫu được phân loại là dương tính, và Recall là tỷ lệ số mẫu dương tính thực sự được phát hiện trên tổng số mẫu dương tính thực sự.

2.2.5 Test Error và Train Error (Sai Số Kiểm Tra và Sai Số Huấn Luyện)

Đây là các chỉ số đánh giá mức độ chênh lệch giữa hiệu suất mô hình trên tập huấn luyện và tập kiểm tra, giúp nhận biết hiện tượng overfitting hoặc underfitting.

- **Train error**: Thường là hàm mất mát áp dụng lên training data. Hàm mất mát này cần có một thừa số $\frac{1}{N_{\text{train}}}$ để tính giá trị trung bình, tức mất mát trung bình trên mỗi điểm dữ liệu. Với Regression, đại lượng này thường được định nghĩa:

  $$\text{train error} = \frac{1}{N_{\text{train}}} \sum_{\text{training set}} \|y - \hat{y}\|_p^2$$

  với p thường bằng 1 hoặc 2.

  Với Classification, trung bình cộng của cross entropy có thể được sử dụng.

- **Test error**: Tương tự như trên nhưng áp dụng mô hình tìm được vào test data. Chú ý rằng, khi xây dựng mô hình, ta không được sử dụng thông tin trong tập dữ liệu test. Dữ liệu test chỉ được dùng để đánh giá mô hình. Với Regression, đại lượng này thường được định nghĩa:

  $$\text{test error} = \frac{1}{N_{\text{test}}} \sum_{\text{test set}} \|y - \hat{y}\|_p^2$$

  với p giống như p trong cách tính train error phía trên.

Một mô hình được coi là tốt (fit) nếu cả train error và test error đều thấp. Nếu train error thấp nhưng test error cao, ta nói mô hình bị overfitting. Nếu train error cao và test error cao, ta nói mô hình bị underfitting. Nếu train error cao nhưng test error thấp, tôi không biết tên của mô hình này, vì cực kỳ may mắn thì hiện tượng này mới xảy ra, hoặc có chỉ khi tập dữ liệu test quá nhỏ.

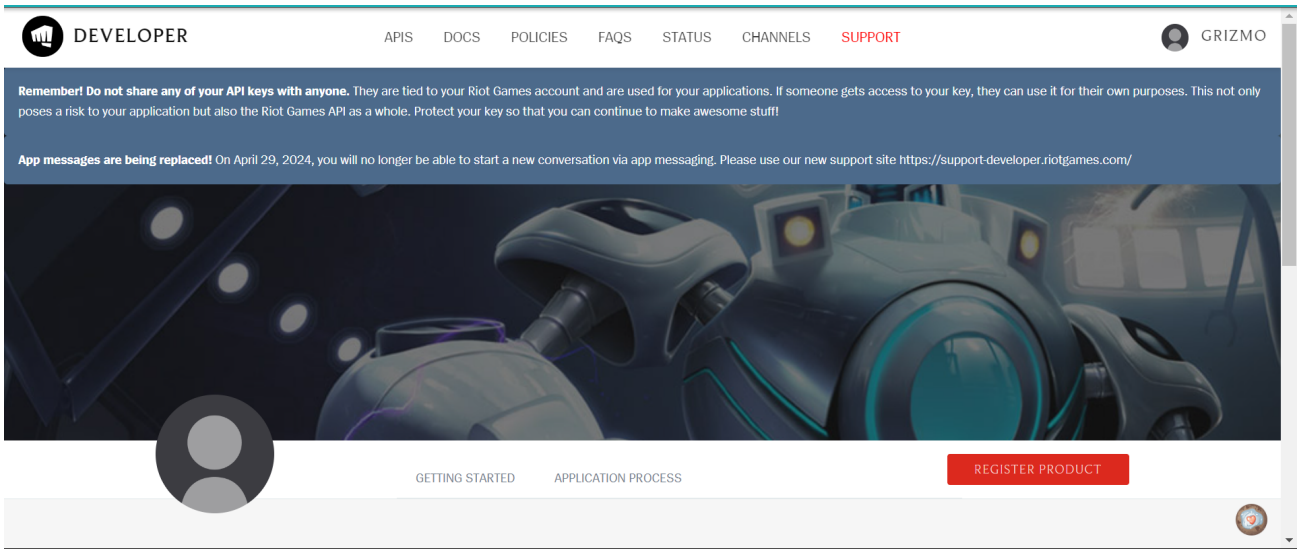
Chương 3

Bộ dữ liệu mô hình

3.1 Quá trình thu thập dữ liệu

Riot Games, nhà phát triển và công ty chủ quản của trò chơi Teamfight Tactics, đã cung cấp một API bán công khai để thu thập dữ liệu cho TFT. API cho phép truy cập vào mọi tài khoản được đăng ký trong TFT và tất cả các trận đấu được chơi bởi tài khoản đó kể từ ngày 26 tháng 6 năm 2019.

Thông qua API, tiến trình thu thập dữ liệu theo các bước sau: Đầu tiên tạo một tài khoản Riot Games tại trang Riot for developer.



Hình 3.1: Giao diện sau khi đăng nhập thành công

Tiếp theo lựa chọn API phù hợp. Ở trong dự án này API được chọn là `TFT-LEAGUE-V1` để lấy thông tin của các người chơi đạt rank thách đấu (Mức rank cao nhất của trò chơi) và `TFT-MATCH-V1` để lấy thông tin về các trận đấu của người chơi.

Cài đặt thư viện `request` trong python

```
pip install request
```

Import và sử dụng `request` trong python

```
import request
response = request.get(url)
```

Trong đó `url` là đường dẫn tới link request như ở trong API docs đã nêu.

Tiếp đến chúng ta sẽ phải đi hơi vòng 1 chút. Sau khi gọi đến API `TFT-LEAGUE-V1` và thông qua tiền xử lí. Ta chỉ thu được thông tin về `Summoner Id` - Summoner Id đã được mã hóa của người chơi. Nhưng định danh này chỉ trên máy chủ của khu vực thôi. Muốn lấy được dữ liệu trận đấu ta cần thông qua PUUID - Player Universal Unique Identifier hay Định danh độc nhất toàn cầu của Người chơi. Chiều dài chính xác là 78 ký tự. Ta sử dụng url /tft/league/v1/entries/by-summoner/summonerId để lấy được puuid của tất cả người chơi.

Bây giờ ta đã có danh sách puuid của các người chơi rank thách đấu, ta sẽ lấy danh sách theo trận. Ta sẽ lấy 100 trận đấu gần nhất của mỗi người chơi để lấy làm dữ liệu. Sử dụng url /tft/match/v1/matches/by-puuid/puuid/ids để lấy được danh sách các `matchId` dựa vào đó chúng ta thông qua url /tft/match/v1/matches/matchId để lấy được dữ liệu của trận đấu. Thực hiện lưu trữ và ghi dữ liệu vào file json tương ứng với matchId để định danh.

Lưu ý

Giới hạn lượt yêu cầu tối đa là 20 lần/ giây và 200 lần / 2 phút.

Đến đây có thể nói đã hoàn tất bước lấy dữ liệu tuy nhiên dữ liệu ở dạng json và chưa thể dùng để xử lý được ngay. Cần thêm một bước là chuyển dữ liệu về dạng csv. Vì tất cả các file đều có cấu trúc cùng với đó là hình thái dữ liệu giống nhau. Chính vì vậy ta áp dụng cùng một cách đọc file json với thư viện `json` trong python. Sau khi đọc xong ta xác định các khóa tương ứng với là các cột quan trọng (Chứa các tài nguyên của người chơi) làm tên cột và ứng với đố dữ liệu của một người chơi sẽ tương ứng là một hàng. Và vì 1 file json sẽ tương ứng là 1 trận nên tổng cộng từ 1 file chúng ta sẽ có 8 dòng dữ liệu.

3.2 Tiền xử lý dữ liệu

3.2.1 Làm sạch dữ liệu

Sau khi đã thu thập và lưu trữ dữ liệu dưới dạng csv từ các file JSON, bước tiếp theo là làm sạch dữ liệu. Quá trình làm sạch dữ liệu sẽ bao gồm các bước sau:

Bước 1: Loại bỏ dữ liệu không cần thiết

Loại bỏ các trường thông tin không liên quan hoặc trùng lặp, chẳng hạn như những trường chỉ chứa giá trị rỗng hoặc các ID không cần thiết cho phân tích.

Bước 2: Định dạng lại các giá trị

Dữ liệu thu thập từ API thường chứa các định dạng không đồng nhất; ví dụ, tên augments có thể chứa dấu gạch dưới hoặc các ký tự đặc biệt. Sử dụng biểu thức chính quy và các hàm chuỗi của pandas để chuẩn hóa và đồng nhất hóa dữ liệu này.

Bước 3: Chọn lọc các cột cần thiết

Trong bước này, chúng ta sẽ tập trung vào việc lọc và giữ lại những cột dữ liệu quan trọng phục vụ cho mục đích phân tích hoặc mô hình học máy, đồng thời loại bỏ những cột không cần thiết. Việc này giúp giảm kích thước dữ liệu, từ đó tăng tốc độ xử lý và cải thiện hiệu quả quản lý dữ liệu.

	augments_1	augments_2	augments_3
0	TFT11_Augment_StoryweaverCrest	TFT9_Augment_StationarySupport3	TFT6_Augment_OneTwoFive
1	TFT11_Augment_Trickshot	TFT11_Augment_TrashToTreasure	TFT9_Augment_Commander_RollingForDays
2	TFT9_Augment_Idealism	TFT6_Augment_GachaAddict	TFT9_Augment_RedBuff
3	TFT9_Augment_YouHaveMyBow	TFT11_Augment_Accomplice	TFT9_Augment_Harmacist1
4	TFT9_Augment_Sleightofhand	TFT9_Augment_StationarySupport3	TFT9_Augment_CyberneticBulk1

Hình 3.2: Hình ảnh dữ liệu lõi nâng cấp trước khi định dạng

	augments_1	augments_2	augments_3
0	Storyweaver Crest	Stationary Support3	One Two Five
1	Trickshot	Trash To Treasure	Rolling For Days
2	Idealism	Gacha Addict	Red Buff
3	You Have My Bow	Accomplice	Harmacist1
4	Sleightofhand	Stationary Support3	Cybernetic Bulk1

Hình 3.3: Hình ảnh dữ liệu lõi nâng cấp sau khi định dạng

Cụ thể, ta sẽ giữ lại các cột liên quan đến nâng cấp lõi, thông tin Tộc/Hệ, Tướng, trang bị, cấp độ Linh Thú và xếp hạng trận đấu (placement). Các thông tin này được xem là cốt lõi và cần thiết để thực hiện các phân tích sâu hơn hoặc để huấn luyện các mô hình.

Bước 4: Lưu các tệp dữ liệu cần xử lý tiếp

Các dữ liệu phức tạp cần được xử lý tiếp sẽ được lưu vào các tệp riêng biệt tương ứng: 'augments.csv' cho dữ liệu lõi nâng cấp, 'units.csv' cho thông tin Tướng và trang bị, và 'traits.csv' cho dữ liệu Tộc/Hệ. Việc phân loại dữ liệu như vậy giúp quá trình phân tích trở nên gọn gàng và hiệu quả hơn, đồng thời tạo điều kiện thuận lợi cho việc trích xuất và xử lý thông tin theo từng phân khúc cụ thể.

3.2.2 Mã hóa dữ liệu

Lúc này dữ liệu đã ở định dạng chuẩn tuy nhiên vẫn còn đó là sự hỗn hợp giữa chữ và số dẫn tới chưa thể tính toán ngay được. Chúng ta cần thực hiện bước chuyển đổi dữ liệu sang dạng số.

Mã hóa thuộc tính lõi nâng cấp

Đối với các thuộc tính như `augments_1`, `augments_2`, và `augments_3`, chúng ta tiến hành chuẩn hóa tên và mã hóa nhị phân chúng. Mỗi lõi nâng cấp (augment) không chỉ được mã hóa dựa trên sự hiện diện mà còn cả cấp độ của lõi đó. Phương pháp này gọi là mã hóa "One-hot Encoding" nhưng được mở rộng để tính đến cấp độ của lõi.

Bước 1: Đọc dữ liệu và tạo ánh xạ

- **Đọc dữ liệu:** Hai file CSV, `Id_augment.csv` và `Type.csv`, được đọc vào. File `Id_augment.csv` chứa ánh xạ từ ID đến tên chuẩn của các lõi nâng cấp, và `Type.csv` chứa thông tin các

lõi và mã hóa tương ứng của nó, xếp hạng nâng cấp của lõi (lõi bạc là 1, lõi vàng là 2, lõi kim cương là 3).

- **Tạo ánh xạ từ ID sang tên:** Sử dụng DataFrame `id_name`, một dictionary được tạo để ánh xạ từ ID sang tên chuẩn.
- **Tạo ánh xạ từ tên lõi sang loại:** Tương tự, một dictionary khác được tạo từ DataFrame `name_type` để ánh xạ từ tên lõi sang loại tương ứng của nó.

	id	name
0	A Cut Above	A Cut Above
1	TFT11_Augment_Accomplice	Accomplice
2	TFT11_Augment_Arcanist	Arcanist
3	Ascension	Ascension
4	TFT11_Augment_AtWhatCost	AtWhatCost
...	...	...
235	TFT11_Augment_Warden	Warden
236	What Doesn't Kill You	What Doesn't Kill You
237	You Have My Bow	You Have My Bow
238	You Have My Sword	You Have My Sword
239	Young and Wild and Free	Young and Wild and Free

240 rows × 2 columns

Hình 3.4: Dữ liệu trong tệp `Id_augment.csv`

	Augment	Type
0	A Cut Above	2
1	Accomplice	3
2	Ascension	2
3	Ba-BOOM	3
4	Balanced Budget	2
...	...	...
236	Wandering Trainer III	3
237	What Doesn't Kill You	2
238	You Have My Bow	2
239	You Have My Sword	2
240	Young and Wild and Free	1

241 rows × 2 columns

Hình 3.5: Dữ liệu trong tệp `Type.csv`

Bước 2: Chuẩn hóa và cập nhật tên lõi

- **Đọc và chuẩn hóa tên từ JSON:** Tên lõi từ một file JSON được đọc và chuẩn hóa để loại bỏ các hậu tố như 'III', 'II', '++', và '+'. Hàm `replace_suffix` được dùng để loại bỏ các hậu tố này và chuẩn hóa tên.
- **Cập nhật giá trị dựa trên tên chuẩn:** Các tên lõi trong DataFrame `augments` được cập nhật sử dụng ánh xạ tên đã tạo. Nếu tên lõi khớp với một trong các tên trong dictionary `names` hoặc `mapping_name`, nó sẽ được thay thế bằng tên chuẩn tương ứng.

Bước 3: Duyệt dữ liệu

- **Tạo DataFrame mới cho tất cả lõi:** Một DataFrame mới `augments_df` được tạo với các cột là tất cả tên lõi đã được chuẩn hóa và không trùng lặp.
- **Điền giá trị vào DataFrame:** Duyệt qua mỗi lõi trong DataFrame `augments`, giá trị được cập nhật dựa trên loại lõi. Nếu lõi có hậu tố 'I', giá trị tương ứng được cộng dồn tùy thuộc vào số lượng 'I'. Nếu lõi khớp với một loại trong `mapping_type`, giá trị tương ứng với loại đó được cộng vào.

Mã hóa thuộc tính Tộc/Hệ

Các cột từ `traits_1_name` đến `traits_16_name` trong dữ liệu của chúng ta đại diện cho các Tộc/Hệ mà nhân vật sở hữu trong trò chơi. Mỗi cột này được chuẩn hóa và sau đó mã hóa dựa trên sự hiện diện của từng Tộc/Hệ. Việc này được thực hiện thông qua kỹ thuật "One-hot Encoding", trong đó mỗi Tộc/Hệ duy nhất sẽ có một cột riêng trong DataFrame mới. Nếu một

nhân vật thuộc về một Tộc/Hệ nào đó, cột tương ứng sẽ được đánh dấu bằng 1, và ngược lại là 0.

Mã hóa thuộc tính Tướng

Các cột từ từ `units_1_name` đến `units_10_name` trong dữ liệu đại diện cho từng Tướng được sử dụng trong ván đấu. Mỗi Tướng được mã hóa theo sự hiện diện và cấp độ (từ 1-3) của họ trong trận đấu. Cách thức mã hóa tương tự như đối với Tộc/Hệ, sử dụng phương pháp "One-hot Encoding".

Mã hóa thuộc tính trang bị

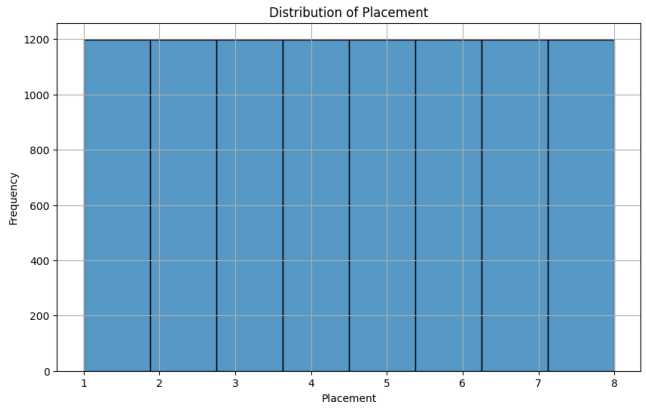
Các trang bị trong cột từ `units_1_item_1` đến `units_10_item_3` cho mỗi nhân vật được mã hóa để phản ánh trang bị nào đang được sử dụng cũng như số lượng của mỗi trang bị đó. Mỗi loại trang bị duy nhất được mã hóa thành một cột trong DataFrame, và được mã hóa dựa trên sự hiện diện, được đánh dấu bằng 1, và ngược lại là 0.

3.3 Phân tích dữ liệu

3.3.1 Kiểm tra tính phân phối của dữ liệu

Một vấn đề chính về dữ liệu được khai thác là liệu phân phối của dữ liệu đó có công bằng không. Điều này có nghĩa là các thuộc tính được đại diện một cách công bằng và dữ liệu không thiên vị về một kết quả nào đó như thắng/thua hoặc vị trí xếp hạng cuối cùng.

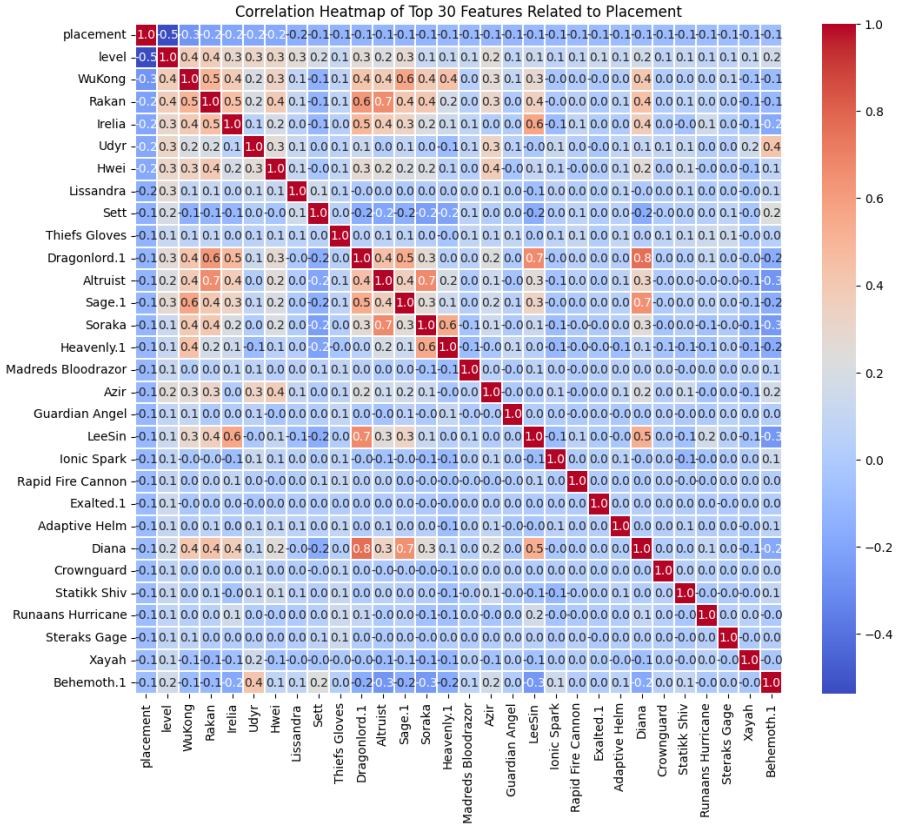
Để xác định những bất thường như vậy, chúng ta sử dụng biểu đồ histogram để trực quan hóa cho thuộc tính `placement`.



Nhìn từ biểu đồ histogram của cột 'placement' cho thấy rằng có một phân phối gần như hoàn hảo của dữ liệu trên tất cả các vị trí có thể (từ 1 đến 8) do đó dữ liệu trong mặt này là công bằng.

3.3.2 Ma trận tương quan

Chúng ta chọn ra 30 thuộc tính có giá trị tương quan cao nhất để vẽ bản đồ nhiệt. Kết quả thu được như sau:



- **Level (-0.537):** Có mối tương quan âm mạnh, có nghĩa là càng ở level cao, khả năng đạt được vị trí thứ hạng tốt hơn (placement thấp) càng cao. Đây là một chỉ báo quan trọng rằng việc tăng cấp độ có thể là chiến lược chính để cải thiện thành tích trong các trận đấu.

- **Mối quan hệ giữa các Tướng và Placement:**

  - **WuKong (-0.262), Rakan (-0.214), Irelia (-0.198), Udyr (-0.187), Hwei (-0.177):** Những Tướng này, khi được sử dụng, có xu hướng liên quan đến việc đạt được xếp hạng cao hơn trong các trận đấu. Điều này cho thấy sự hiệu quả của họ trong trò chơi, và sử dụng các nhân vật này có thể là một lợi thế.
  - **Lissandra (-0.152) và Sett (-0.142):** Cũng cho thấy tương quan âm với placement, tuy không mạnh bằng các tướng phía trên những cũng là lựa chọn tốt để sử dụng.

- **Mối quan hệ giữa Tộc/Hệ, Trang bị và Placement:**

  - **Dragonlord (-0.127):** Tương tự, đây là Tộc/Hệ có chỉ số mạnh, có sự hiệu quả cao trong trận đấu.
  - **Thiefs Gloves (-0.132):** Trang bị này có mối tương quan âm khá cao, tác động tích cực tới xếp hạng của người chơi.

- **Mối quan hệ đáng chú ý giữa các thuộc tính:**

- **Mối tương quan cao giữa Tộc Dragonlord với Diana (0.762), LeeSin (0.675), Rakan (0.617), Sage.1 (0.543) và Irelia (0.478):** cho thấy các tướng Diana, Lee Sin, Rakan, Irelia và hệ Sage có tương tác tốt với Tộc Dragonlord. Từ đó, chúng ta có thể tận dụng tối đa các thành phần này khi chơi Tộc Dragonlord.
- **Mối tương quan cao giữa Hệ Altruist với Soraka (0.675), Rakan (0.658):** Tương tự, sự tương quan cao ở đây cho thấy đây là sự kết hợp tốt khi sử dụng trong trận đấu.

Chương 4

Triển khai mô hình

Trong chương này, chúng ta sẽ trình bày chi tiết các bước triển khai mô hình sau khi tiến hành thu thập, tiền xử lý và phân tích dữ liệu ở Chương 3.

4.1 Các bước thông thường để triển khai mô hình học máy

Nhập các thư viện cần thiết

- Import các thư viện cần thiết `pandas`, `seaborn`, `numpy`, `matplotlib.pyplot`
- Import các class và hàm từ `scikit-learn` để thực hiện phân tích thành phần chính (PCA), phân chia dữ liệu, và xây dựng các mô hình học máy như `Logistic Regression`, `LinearRegression`, `GaussianNB`, `SVC`, `SVR`, và các hàm đánh giá mô hình như `accuracy_score`, `mean_squared_error`, `r2_score`.

Tải bộ dữ liệu

Chúng ta tải dữ liệu từ file `'final_data.csv'`. Chuyển đổi biến mục tiêu 'placement' từ một biến đa cấp thành một biến nhị phân dùng cho phân loại nhị phân. Các biến độc lập x sẽ được tách khỏi biến phụ thuộc y.

Mã hóa nhị phân các biến mục tiêu

Thực hiện việc chuyển đổi biến mục tiêu 'placement' thành một biến phân loại nhị phân. Trong đó, các giá trị từ 1 đến 4 (bao gồm) được gán nhãn là 1, đại diện cho trạng thái 'win' (chiến thắng), trong khi các giá trị từ 5 đến 8 (bao gồm) được gán nhãn là 0, đại diện cho trạng thái 'loss' (thất bại), bằng cách sử dụng hàm apply để áp dụng một hàm lambda lên mỗi phần tử trong cột 'placement'.

Chuẩn hóa các đặc trưng

Để tính toán các giá trị trung bình và độ lệch chuẩn của từng đặc trưng và sau đó chuẩn hóa chúng, ta sử dụng phương thức `fit_transform()` của đối tượng `scaler` để áp dụng lên

biến X. Kết quả là biến X_{scaled} chứa các đặc trưng đã được chuẩn hóa, trong đó mỗi giá trị đã được điều chỉnh sao cho có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

Phân chia dữ liệu thành dữ liệu huấn luyện và dữ liệu kiểm tra

Để phân chia các biến độc lập và phụ thuộc thành các tập huấn luyện và kiểm tra, ta sử dụng hàm `train_test_split`. Trong đoạn mã dưới đây, tỷ lệ 0.60 cho biết rằng 60% dữ liệu sẽ được sử dụng cho tập kiểm tra và 40% còn lại sẽ được sử dụng cho tập huấn luyện.

Huấn luyện mô hình

Bước này thay đổi tùy theo mô hình, vì nó bao gồm việc khởi tạo các lớp khác nhau cho các mô hình ML khác nhau. Việc huấn luyện được thực hiện bằng phương thức fit, phương thức này là chung cho nhiều mô hình dự đoán trên scikit-learn. Nó nhận hai tham số, các biến độc lập x và các giá trị tương ứng y.

Dự đoán kết quả

Để dự đoán kết quả, phương thức `predict` được gọi. Đây là một phương thức tiêu chuẩn cho cả mô hình hồi quy và phân loại. Phương thức đơn giản nhận một vector 2D với tất cả các biến độc lập và sẽ trả về một mảng 2D với các giá trị 1 và 0 cho mỗi điểm dữ liệu trong trường hợp phân loại hoặc với các giá trị liên tục cho hồi quy.

Đánh giá mô hình

Để đánh giá hiệu suất của mô hình, chúng ta sử dụng các chỉ số đánh giá phù hợp.

Đối với các mô hình hồi quy, overfitting được đánh giá thông qua điểm số Sai số bình phương trung bình (Mean Square Error - MSE). Phương thức lỗi bình phương trung bình (Mean Square Error - MSE) nhận biến phụ thuộc y thực sự làm tham số đầu tiên, sau đó là biến phụ thuộc y được dự đoán và cuối cùng, nếu tham số 'squared' được thiết lập thành 'False', nó sẽ trả về điểm số RMSE, ngược lại sẽ trả về điểm số MSE. Ngoài ra, độ chính xác được tính thông qua hệ số xác định được gọi là R^2 (coefficient of determination), sử dụng phương thức nhận hai tham số: biến phụ thuộc y thực sự và biến phụ thuộc y được dự đoán.

Đối với các mô hình phân loại, độ chính xác (Accuracy) và $F1-score$ lần lượt là tỷ lệ phần trăm của các điểm được dự đoán đúng trên tổng số điểm dữ liệu và chỉ số đánh giá về hiệu suất phân loại, qua đó, cung cấp cái nhìn tổng thể và chi tiết về hiệu suất của mô hình.

Đồng thời, kết quả dự đoán không thể được chấp nhận là cuối cùng cho đến khi mô hình được chứng minh có khả năng tổng quát tốt. Để tổng quát tốt, một mô hình không nên overfitting hoặc underfitting. Đối với phân loại, overfitting được đánh giá từ điểm số xác thực chéo (CVS).

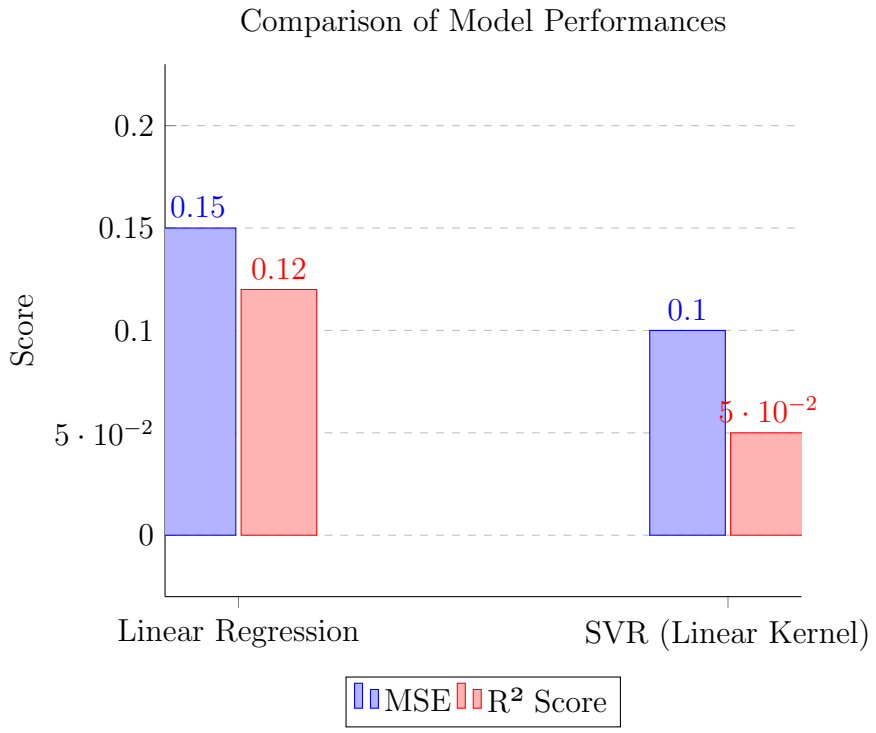
Các thông tin về các chỉ số đánh giá trên đã được trình bày chi tiết ở phụ chương 2.2.

4.2 Kết quả

4.2.1 Các mô hình hồi quy

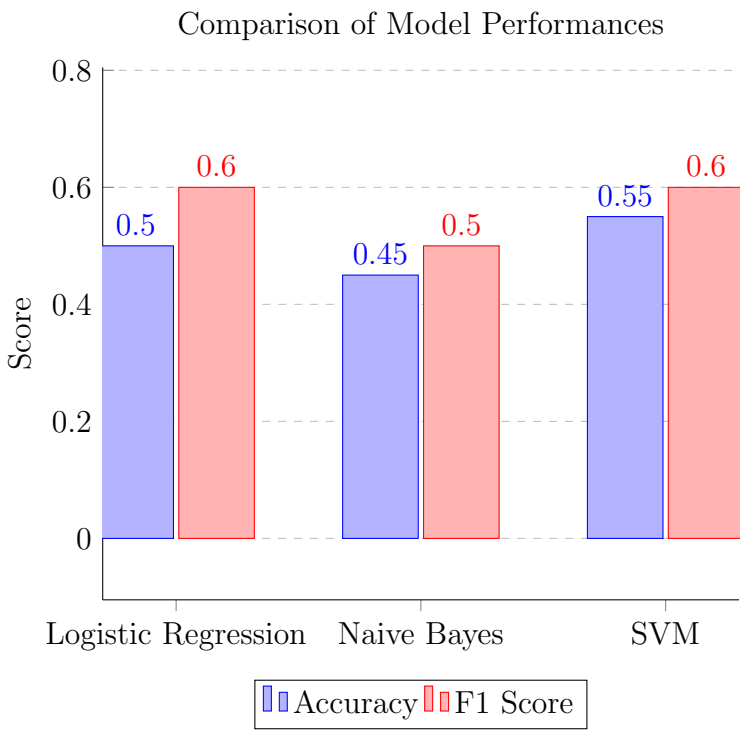
- Linear Regression cho thấy MSE khá thấp so với SVR, nhưng R^2 Score của nó cũng thấp hơn một chút. Điều này cho thấy mô hình Linear Regression có thể là một lựa chọn tốt hơn vì nó cân bằng giữa hiệu quả và khả năng giải thích.

- SVR có R^2 Score cao hơn, nhưng MSE cũng cao hơn đáng kể so với Linear Regression, cho thấy mô hình này có thể không hiệu quả bằng Linear Regression trong việc dự đoán các giá trị mục tiêu.



4.2.2 Các mô hình phân loại

- Logistic Regression có cả Accuracy và F1 Score ở mức vừa phải, cho thấy nó là một mô hình phân loại khá cân bằng.
- Naive Bayes có Accuracy và F1 Score thấp nhất trong số ba mô hình, cho thấy nó có thể không phù hợp với phân phối hoặc đặc trưng của dữ liệu.
- SVM có Accuracy cao hơn so với Naive Bayes nhưng F1 Score của nó lại thấp, điều này cho thấy mặc dù SVM có thể phân loại chính xác tổng thể nhưng nó có thể gặp khó khăn trong việc phân loại cân bằng giữa các lớp hoặc có vấn đề về độ chính xác trong các lớp ít hơn.



4.2.3 Đánh giá tính tổng quan của các mô hình

Các mô hình học máy đã được đánh giá dựa trên sai số huấn luyện và sai số kiểm tra để xem xét khả năng tổng quát hóa của chúng đối với dữ liệu mới. Kết quả thu được như sau:

Model	Train Error	Test Error
Logistic Regression	0.1462	0.2147
Naive Bayes	0.2888	0.3122
SVM	0.0688	0.2366
Linear Regression	0.1299	0.1733
SVR	0.1396	0.1876

Bảng 4.1: Kết quả Train và Test Errors cho các mô hình học máy

Nhận xét: Trong khi Linear Regression và SVR cho thấy khả năng tổng quát hóa tốt nhất với sai số thấp và sự cân bằng giữa huấn luyện và kiểm tra, thì SVM có vẻ như bị overfitting với sai số kiểm tra cao hơn nhiều so với huấn luyện. Logistic Regression và Naive Bayes có hiệu suất ở mức trung bình, với Naive Bayes cho thấy khả năng tổng quát hóa thấp nhất. Việc lựa chọn mô hình phù hợp sẽ phụ thuộc vào mục tiêu cụ thể của bài toán và tính chất của dữ liệu được sử dụng.

Chương 5

Tổng kết

5.1 Những câu hỏi mà dự án đã đưa ra

5.1.1 Các kỹ thuật học máy liệu có hiệu quả trong việc dự đoán kết quả của các trận đấu Teamfight Tactics không?

Qua việc chạy chương trình và xem xét kết quả dự đoán của các mô hình học máy được trình bày trên, ta thấy với bộ dữ liệu đầu vào là sự kết hợp giữa các biến số rời rạc (như Tướng và trang bị) và liên tục (như cấp độ Tướng), mô hình hoàn toàn có thể dự đoán được kết quả của trận đấu dựa trên những yếu tố người chơi đã chọn trong trận đấu đó với mức độ chính xác tương đối.

Tuy nhiên các giải thuật được xét trong bài báo cáo này có vẻ chưa phải là tốt nhất do các chỉ số phương sai và độ chính xác vẫn chưa đạt được đến kì vọng, do vậy vẫn cần phải thử nghiệm thêm các mô hình khác trước khi nghĩ đến việc thực sự áp dụng nó.

5.1.2 Liệu có thể phát triển một mô hình tạo ra một đội hình luôn thắng cuộc không?

Việc này khả thi nhưng cực kì phức tạp, bởi vì vốn dĩ trò chơi này không hề đơn giản.

Vấn đề đầu tiên cho câu hỏi này là mối quan hệ tuyến tính giữa các yếu tố trong trò chơi thực sự quá nhiều. Trong dữ liệu ở báo cáo lần này , mới chỉ xét sự tương quan giữa các yếu tố chứ chưa xem xét đến sự kết hợp của 2 hay nhiều yếu tố nào đó sẽ làm thay đổi sức mạnh đội hình hay kết quả cuối cùng.Ví dụ như vị tướng nào đó khi được kết hợp với lõi phù hợp thì đội hình sẽ mạnh lên rất nhiều lần.

Vấn đề thứ 2 là sự thay đổi liên tục của trò chơi theo thời gian. Các lõi, các tướng, các tộc hệ biến đổi chóng mặt theo từng phiên bản nên việc xây dựng ra một mô hình cố định theo từng phiên bản thật sự là điều không tưởng.

5.2 Hạn chế của dự án

- **Sự đa dạng và độ đầy đủ của dữ liệu:** Dữ liệu được sử dụng trong dự án này chủ yếu tập trung vào các yếu tố như Tướng, trang bị, và lõi nâng cấp. Tuy nhiên, có nhiều yếu tố khác trong trò chơi Đấu Trường Chân Lý có thể ảnh hưởng đến kết quả của trận đấu mà chưa được xét đến, như vị trí đặt Tướng và chiến thuật cụ thể của từng người chơi.
- **Giới hạn của mô hình hiện tại:** Các mô hình được sử dụng trong dự án này, dù đã cho thấy một số kết quả khả quan, vẫn còn thiếu khả năng hiểu và mô phỏng đầy đủ các tương tác phức tạp giữa các yếu tố trong trò chơi. Điều này dẫn đến khả năng tổng quát hóa và dự đoán kết quả còn hạn chế.
- **Thiếu các phân tích sâu hơn:** Dự án này chưa thể thực hiện các phân tích sâu hơn về tác động của từng yếu tố cụ thể đối với thành công của một đội hình, cũng như chưa khai thác hết tiềm năng của dữ liệu về các chiến thuật và cách chơi của người chơi.
- **Chưa khắc phục được các vấn đề còn tồn đọng:** Các vấn đề về trực quan hóa kết quả, vấn đề overfitting hay việc chọn các trường dữ liệu hợp lý để cải thiện mô hình chưa thành công.

5.3 Gợi ý cải thiện trong tương lai

Để khắc phục các hạn chế và nâng cao chất lượng của dự án, một số gợi ý cải thiện có thể được áp dụng trong tương lai:

- **Mở rộng bộ dữ liệu:** Thêm dữ liệu về vị trí đặt Tướng, các tộc hệ kết hợp, và các chiến thuật sử dụng trong từng trận đấu để có thể phân tích và dự đoán kết quả một cách chính xác hơn.
- **Sử dụng mô hình học sâu:** Áp dụng các mô hình học sâu như mạng nơ-ron tích chập (CNN) hoặc mạng nơ-ron hồi quy (RNN) để mô phỏng tốt hơn các tương tác phức tạp và phi tuyến tính giữa các yếu tố trong trò chơi.
- **Tối ưu hóa mô hình:** Thực hiện các kỹ thuật tối ưu hóa mô hình như điều chỉnh tham số, cross-validation, và feature engineering để cải thiện khả năng tổng quát hóa và hiệu suất của mô hình.
- **Theo dõi và cập nhật:** Do trò chơi Đấu Trường Chân Lý liên tục cập nhật và thay đổi, việc theo dõi sát sao các cập nhật và điều chỉnh mô hình cho phù hợp là rất quan trọng để duy trì độ chính xác.