Report-1 - report ml football prediction

Nhập môn học máy và khai phá dữ liệu (Trường Đại học Bách khoa Hà Nội)

ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO ĐỒ ÁN MÔN HỌC

IT3190 – Nhập môn Học máy và Khai phá dữ liệu

Đề tài: Dự đoán kết quả các trận đấu Premier League (K-NN, XGBoost, ANN)

Giảng viên: **Nguyễn Nhật Quang**

Nhóm sinh viên thực hiện:

STT	Họ và tên	MSSV
1	Vũ Tuấn Kiệt	20200308
2	Lê Đình Hiếu	20194280
3	Nguyễn Quốc Nhật Minh	20200408
4	Trần Văn Long	20200372

Hà Nội, tháng 01 năm 2023

MỤC LỤC

I. MỞ ĐẦU

Học máy là môn khoa học nhằm phát triển những thuật toán và mô hình thống kê mà các hệ thống máy tính sử dụng để thực hiện các tác vụ dựa vào khuôn mẫu và suy luận mà không cần hướng dẫn cụ thể. Các hệ thống máy tính sử dụng thuật toán máy học để xử lý khối lượng lớn dữ liệu trong quá khứ và xác định các khuôn mẫu dữ liệu. Việc này cho phép chúng dự đoán kết quả chính xác hơn từ cùng một tập dữ liệu đầu vào cho trước. Ví dụ: các nhà khoa học dữ liệu có thể đào tạo một ứng dụng y tế chẩn đoán ung thư từ ảnh chụp X-quang bằng cách lưu trữ hàng triệu ảnh quét và chẩn đoán tương ứng.

Học máy giúp các doanh nghiệp thúc đẩy phát triển, tạo ra các dòng thu nhập mới và giải quyết những vấn đề mang tính thách thức. Dữ liệu là động lực thúc đẩy tối quan trọng đằng sau các quyết định của doanh nghiệp nhưng theo truyền thống, các công ty sử dụng dữ liệu từ nhiều nguồn như phản hồi của khách hàng, nhân viên và bộ phận tài chính. Nghiên cứu của máy học giúp tự động hóa và tối ưu hóa quá trình này. Bằng cách sử dụng phần mềm phân tích khối lượng lớn dữ liệu ở tốc độ cao, các doanh nghiệp có thể đạt được kết quả nhanh hơn.

Một trong những bài toán phổ biến nhất của học máy là bài toán phân lớp. Với mục đích tìm hiểu, so sánh các mô hình và phân loại các lớp khác nhau, nhóm đã chọn đề tài “Dự đoán kết quả các trận đấu của Premier League” dựa trên bộ dữ liệu dạng bảng về thông số trận đấu của các đội bóng sân nhà và sân khách trong khuôn khổ giải đấu Premier League với 3 mô hình học máy KNN, XGBoost và ANN. Các mô hình trên sẽ được áp dụng để kiểm tra bài toán này và kiểm tra độ hiệu quả thông qua các thực nghiệm.

Quá trình gồm các giai đoạn:

- Xử lý dữ liệu: Đây được coi là giai đoạn quan trọng nhất . Dữ liệu là đầu vào của các mô hình học máy để giải quyết bài toán. Tuy nhiên, những dữ liệu ban đầu khi mới thu thập chưa thể đưa vào mô hình do những thuộc tính dư thừa hay định dạng của một số thuộc tính không phù hợp với mô hình. Do đó, dữ liệu cần phải được chọn ra những đặc trưng tốt, xử lý những thông tin bị thiếu hoặc thay đổi định dạng của dữ liệu. Trong quá trình này, chúng ta cũng cần phải thực hiện phân chia dữ liệu thành các tập khác nhau bao gồm:

tập huấn luyện (training set), tập xác thực (validation set) và tập kiểm thử (test set) với mục đích đo đạc, chọn ra mô hình với thông số phù hợp và kiểm tra độ chính xác của mô hình trong thực tế.

- Xây dựng mô hình và điều chỉnh tham số: Mục đích của bước này là xây dựng các mô hình cho việc huấn luyện. Các tham số của các mô hình sẽ được điều chỉnh phù hợp dựa trên kết quả học của tập huấn luyện và tập xác thực. Sau đó, các mô hình tốt nhất đại diện cho mỗi thuật toán sẽ được đưa vào đánh giá hiệu quả trên tập kiểm thử.

Các quá trình trên sẽ được trình bày rõ hơn trong các phần sau.

II. CHUẨN BỊ DỮ LIỆU

2.1. DỮ LIỆU

Dữ liệu các mùa giải Premier League riêng lẻ được lấy từ trang https://www.football-data.co.uk/data.php, với tỉ số các trận đấu và các thông số phụ, nhóm đã lấy dữ liệu các mùa riêng lẻ từ mùa 2004-2005 đến mùa 2021-2022 để tổng hợp lại thành 1 dữ liệu lớn bao gồm 18 mùa giải liên tục, với các thuộc tính chúng ta cần xét vì chúng ảnh hưởng tới kết quả trận đấu, bao gồm:

- FTHG : Số bàn thắng đội nhà ghi được trong trận
- FTAG : Số bàn thắng đội khách ghi được trong trận
- HS : Số cú sút của đội nhà
- AS : Số cú sút của đội khách
- HST : Số cú sút trúng đích của đội nhà
- AST : Số cú sút trúng đích của đội khách
- HF : Số lần phạm lỗi của đội nhà
- AF : Số lần phạm lỗi của đội khách
- HC : Số thẻ của đội nhà
- AC : Số thẻ của đội khách
- FTR : Số thẻ của đội khách

2.2. TIỀN XỬ LÍ DỮ LIỆU

Chúng ta sẽ chỉ quan tâm đến các thuộc tính đã được liệt kê trong phần trước và bỏ đi các thuộc tính dư thừa.

2.2.1 Tạo thêm các thuộc tính mới

Chúng ta sẽ tính toán và tạo ra những thuộc tính mới từ những thuộc tính kể trên như :

- HomeFullPoint : Số điểm mà đội nhà đã có trong mùa giải đấy trước khi trận đấu diễn ra

- AwayFullPoint : Số điểm mà đội khách đã có trong mùa giải đấy trước khi trận đấu diễn ra
- HomeGoal : Số bàn thắng mà đội nhà đã có trong mùa giải đấy trước khi trận đấu diễn ra
- AwayGoal : Số bàn thắng mà đội khách đã có trong mùa giải đấy trước khi trận đấu diễn ra
- HomeConceded : Số bàn thua mà đội nhà đã có trong mùa giải đấy trước khi trận đấu diễn ra
- AwayGoal : Số bàn thua mà đội khách đã có trong mùa giải đấy trước khi trận đấu diễn ra
- HomeAvgPoint : Số điểm trung bình 1 trận mà đội nhà đã có từ mùa giải 2004/05 trước khi trận đấu diễn ra
- AwayAvgPoint : Số điểm trung bình 1 trận mà đội khách đã có từ mùa giải 2004/05 trước khi trận đấu diễn ra
- pastHP : Số điểm mà đội nhà có trong 2 trận gần nhất
- pastAP : Số điểm mà đội khách có trong 2 trận gần nhất
- pastHG : Số bàn thắng mà đội nhà có trong 2 trận gần nhất
- pastAG : Số bàn thắng mà đội khách có trong 2 trận gần nhất
- pastHGC : Số bàn thua mà đội nhà có trong 2 trận gần nhất
- pastAGC : Số bàn thua mà đội khách có trong 2 trận gần nhất
- pastHS : Số cú sút mà đội nhà có trong 2 trận gần nhất
- pastAS : Số cú sút mà đội khách có trong 2 trận gần nhất
- pastHSC : Số cú sút mà đội nhà phải nhận trong 2 trận gần nhất
- pastASC : Số cú sút mà đội khách phải nhận trong 2 trận gần nhất
- pastHST : Số cú sút trúng đích mà đội nhà có trong 2 trận gần nhất
- pastAST : Số cú sút trúng đích mà đội khách có trong 2 trận gần nhất
- pastHSTC : Số cú sút trúng đích mà đội nhà phải nhận trong 2 trận gần nhất

- pastASTC : Số cú sút trúng đích mà đội khách phải nhận trong 2 trận gần nhất
- pastHP-AP = pastHP – pastAP
- pastHG-AG = pastHG – pastAG
- pastAGC-HGC = pastAGC – pastHGC
- pastHS-AS = pastHS – pastAS
- pastASC-HSC = pastASC – pastHSC
- pastHST-AST = pastHST – pastAST
- pastASTC-HSTC = pastASTC – pastHSTC

Khi huấn luyện, ta sẽ bỏ đi những trận đấu trong 3 vòng đầu để loại ra những dữ liệu ngoại lai, giúp mô hình ổn định khi huấn luyện.

2.2.2 Lựa chọn và biểu diễn dữ liệu

2.2.2.1 Mô tả dữ liệu:

	pastHP-AP	pastHG-AG	pastAGC-HGC	pastHS-AS	pastHST-AST
count	6300.000000	6300.000000	6300.000000	6300.000000	6300.000000
mean	-0.241270	-0.169524	-0.171270	-1.026349	-0.449206
std	2.703513	2.551920	2.494515	11.041117	6.206771
min	-6.000000	-11.000000	-11.000000	-43.000000	-28.000000
25%	-2.000000	-2.000000	-2.000000	-8.000000	-4.000000
50%	0.000000	0.000000	0.000000	-1.000000	0.000000
75%	2.000000	1.000000	1.000000	6.000000	4.000000
max	6.000000	11.000000	10.000000	39.000000	26.000000

	pastASC-HSC	pastASTC-HSTC	HomeFullPoint	AwayFullPoint	HomeGoal
count	6300.000000	6300.000000	6300.000000	6300.000000	6300.000000
mean	-1.043333	-0.494444	1.364822	1.384733	1.333932
std	10.879773	6.065615	0.533781	0.534752	0.514849
min	-42.000000	-33.000000	0.000000	0.000000	0.000000
25%	-8.000000	-4.000000	1.000000	1.000000	1.000000
50%	-1.000000	0.000000	1.278889	1.304348	1.222222
75%	6.000000	3.000000	1.722222	1.750000	1.615385
max	34.000000	26.000000	3.000000	3.000000	4.500000

	AwayGoal	HomeConceded	AwayConceded	HomeAvgPoint	AwayAvgPoint
count	6300.000000	6300.000000	6300.000000	6300.000000	6300.000000
mean	1.349257	1.346752	1.332348	1.378555	1.381959
std	0.518144	0.433097	0.432475	0.410521	0.408791
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	1.064516	1.041667	1.100167	1.100000
50%	1.241379	1.333333	1.333333	1.246377	1.249650
75%	1.625000	1.633333	1.611111	1.621266	1.624167
max	4.666667	3.500000	4.000000	3.000000	3.000000

- Count : Số dữ liệu

- mean: Trung bình cộng của tập dữ liệu. Đây là giá trị trung tâm của tập dữ liệu.

- std: Độ lệch chuẩn của tập dữ liệu. Đây là sự sai khác của các giá trị trong tập dữ liệu so với trung bình.

- 25%: Phần vị 25%. Đây là giá trị dưới đáy của khoảng phân vị tương đương với 25% các giá trị trong tập dữ liệu.

- 50%: Phần vị 50%. Đây là giá trị dưới đáy của khoảng phân vị tương đương với 50% các giá trị trong tập dữ liệu. Nó còn được gọi là trung vị.

- 75%: Phần vị 75%. Đây là giá trị dưới đáy của khoảng phân vị tương đương với 75% các giá trị trong tập dữ liệu.

Ta thấy các thuộc tính pastHG-AG, pastHP-AP, pastHS-AS, pastHST-AST, pastASC-HSC, pastASTC-HSTC có độ lệch chuẩn cao, có nghĩa là các giá trị trong tập dữ liệu có khác biệt lớn với nhau và phân phối của dữ liệu không phải là phân phối chuẩn. Điều này có thể làm cho mô hình quá nhạy cảm với các giá trị dữ liệu này

2.2.2.2 Chuẩn hóa dữ liệu

Tính giá trị kỳ vọng (mean) và độ lệch chuẩn (std) của các giá trị mỗi đặc tính trong bộ huấn luyện, sau đó chuẩn hóa cả bộ dữ liệu huấn luyện, xác thực và kiểm tra theo giá trị kỳ vọng và độ lệch chuẩn đó theo công thức:

Với x là giá trị cũ của dữ liệu khi chưa được chuẩn hóa và X là giá trị của dữ liệu sau khi được chuẩn hóa

Chuẩn hóa dữ liệu trong quá trình phân loại có thể giúp giảm thiểu độ nhạy cảm của mô hình với các giá trị đặc biệt trong dữ liệu đầu vào và tăng khả năng dự đoán chính xác của mô hình.

	pastHP-AP	pastHG-AG	pastAGC-HGC	pastHS-AS	pastHST-AST
count	6.300000e+03	6.300000e+03	6.300000e+03	6.300000e+03	6.300000e+03
mean	2.731149e-16	1.966328e-16	-1.392184e-17	-3.334194e-17	4.934325e-18
std	1.000079e+00	1.000079e+00	1.000079e+00	1.000079e+00	1.000079e+00
min	-2.130261e+00	-4.244387e+00	-4.341360e+00	-3.801878e+00	-4.439181e+00
25%	-6.505868e-01	-7.173507e-01	-7.331586e-01	-6.316575e-01	-5.721293e-01
50%	8.925018e-02	6.643518e-02	6.866401e-02	2.386651e-03	7.237934e-02
75%	8.290872e-01	4.583281e-01	4.695753e-01	6.364308e-01	7.168879e-01
max	2.308761e+00	4.377257e+00	4.077777e+00	3.625496e+00	4.261685e+00

	pastASC-HSC	pastASTC-HSTC	HomeFullPoint	AwayFullPoint
count	6.300000e+03	6.300000e+03	6.300000e+03	6.300000e+03
mean	-3.513944e-17	-5.780209e-17	-6.097592e-15	4.604588e-15
std	1.000079e+00	1.000079e+00	1.000079e+00	1.000079e+00
min	-3.764777e+00	-5.359413e+00	-2.557100e+00	-2.589694e+00
25%	-6.394636e-01	-5.779849e-01	-6.835222e-01	-7.195186e-01
50%	3.983243e-03	8.152244e-02	-1.610022e-01	-1.503347e-01
75%	6.474301e-01	5.761529e-01	6.696173e-01	6.831132e-01
max	3.221218e+00	4.368320e+00	3.063633e+00	3.020833e+00

	HomeGoal	AwayGoal	HomeConceded	AwayConceded	HomeAvgPoint
count	6.300000e+03	6.300000e+03	6.300000e+03	6.300000e+03	6.300000e+03
mean	-5.626011e-15	4.659218e-15	1.893336e-14	-2.711270e-15	-6.716990e-15
std	1.000079e+00	1.000079e+00	1.000079e+00	1.000079e+00	1.000079e+00
min	-2.591124e+00	-2.604228e+00	-3.109829e+00	-3.080998e+00	-3.358330e+00
25%	-6.486536e-01	-6.741081e-01	-6.517192e-01	-6.721883e-01	-6.781870e-01
50%	-2.169934e-01	-2.082172e-01	-3.098445e-02	2.278294e-03	-3.220020e-01
75%	5.467131e-01	5.322166e-01	6.617556e-01	6.446274e-01	5.912741e-01
max	6.149994e+00	6.402997e+00	4.972138e+00	6.168830e+00	3.950040e+00

	AwayAvgPoint
count	6.300000e+03
mean	-1.157046e-15
std	1.000079e+00
min	-3.380870e+00
25%	-6.897945e-01
50%	-3.236862e-01
75%	5.925442e-01
max	3.958427e+00

- Một vài lý do tại sao chuẩn hóa dữ liệu là một bước rất quan trọng trong quá trình phân loại:

+ Giá trị các thuộc tính có thể có độ lớn khác nhau nên có thể làm cho mô hình quá nhạy cảm với các đặc trưng có độ lớn lớn hơn. Chuẩn hóa dữ liệu sẽ giúp cho tất cả các thuộc tính có độ lớn giống nhau và giúp cho mô hình không quá nhạy cảm với bất kỳ đặc trưng nào.

+ Độ lệch chuẩn của dữ liệu mới sẽ thấp hơn, điều này làm giảm đi sự nhạy cảm với những môi trường khác nhau.

+ Ngoài ra, nhiều thuật toán học máy cần phải tính toán đạo hàm hoặc ma trận hessian để tìm ra các tham số tốt nhất cho mô hình. Nếu các giá trị các đặc trưng có độ lớn khác nhau, thì các đạo hàm hoặc ma trận hessian có thể rất lớn và khó khăn trong việc tính toán. Chuẩn hóa dữ liệu sẽ giúp giảm độ lớn của các đạo hàm hoặc làm giảm độ phức tạp tính toán.

III. ÁP DỤNG CÁC MÔ HÌNH HỌC MÁY ĐỀ XUẤT

3.1. ARTIFICIAL NEURAL NETWORK

3.1.1 Cơ sở lý thuyết

Mạng thần kinh nhân tạo (ANN) là một mô hình xử lý thông tin được lấy cảm hứng từ bộ não con người. ANN có khả năng học (learn), nhớ lại (recall), và khái quát hóa (generalize) từ các dữ liệu học – bằng cách gán và điều chỉnh (thích nghi) các giá trị trọng số (mức độ quan trọng) của các liên kết giữa các nơ-ron.

Chức năng (hàm mục tiêu) của một ANN được xác định bởi:

- Kiến trúc (topology) của mạng nơ-ron

- Đặc tính vào/ra của mỗi nơ-ron

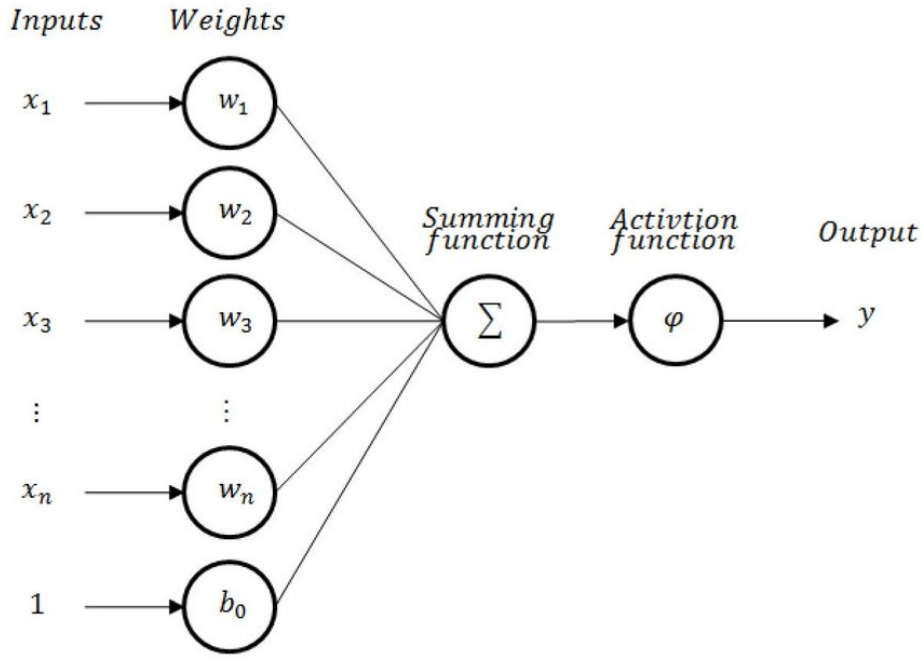
- Chiến lược học (huấn luyện)

- Dữ liệu học

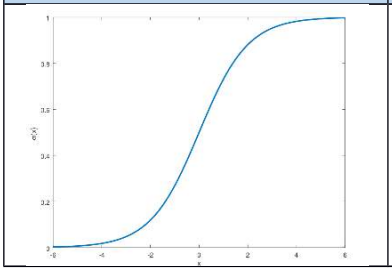
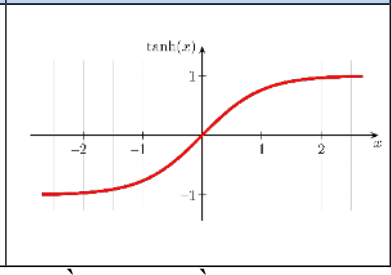
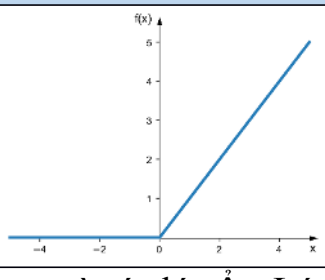
Mỗi nút trong mạng có các tín hiệu đầu vào được gắn với một trọng số với .

Các tín hiệu và trọng số trên được đưa vào một hàm tích hợp các tín hiệu đầu vào gọi là Net input: với là tín hiệu dịch chuyển (bias).

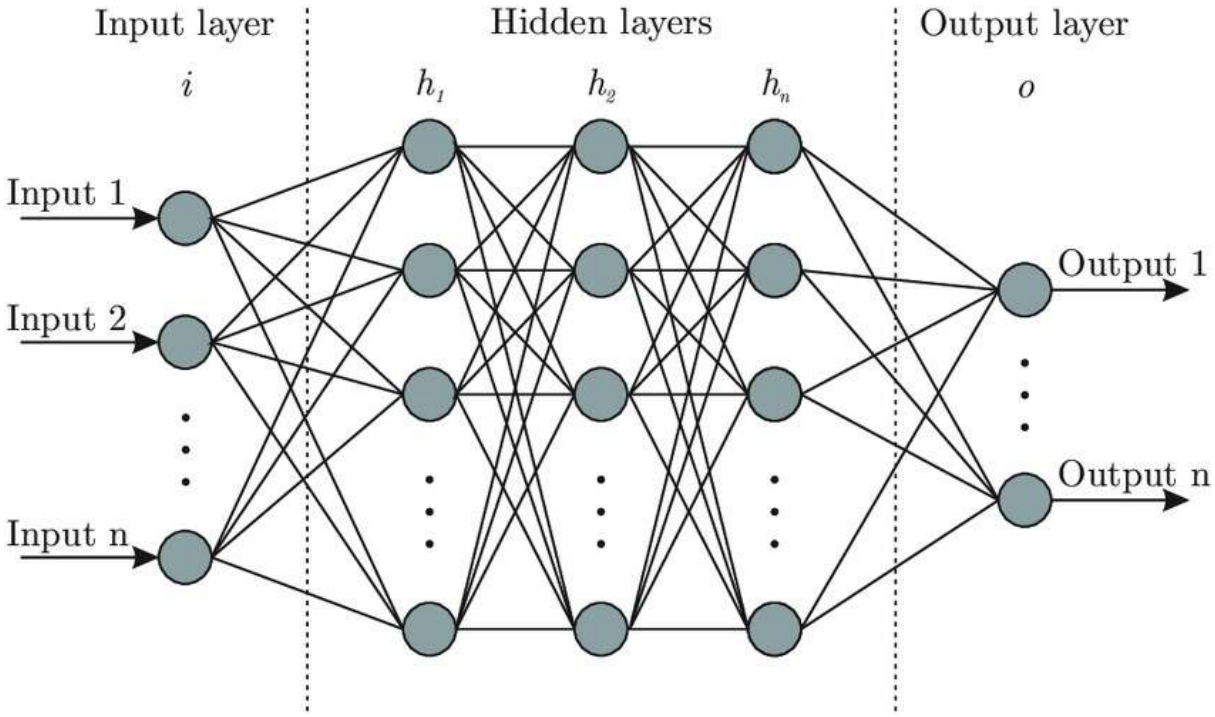
Để tạo giá trị đầu ra, Net input có thể cần đưa qua một hàm kích hoạt (activation funtion) để tạo thành giá trị đầu ra:



Các hàm activation phổ biến:

Sigmoid	Tanh	Relu
		

Một mạng ANN thường bao gồm lớp đầu vào, lớp đầu ra và các lớp ẩn. Lớp đầu vào và lớp đầu ra luôn cần phải có, các lớp này dữ liệu đầu vào và đưa ra dữ liệu đầu ra tương ứng. Các lớp ẩn nằm giữa lớp đầu vào đầu ra, có thể có số lượng nút và số lớp tuỳ ý. Một ANN được gọi là liên kết đầy đủ (fully connected) nếu mọi đầu ra từ một tầng liên kết với mọi nơ-ron của tầng kế tiếp.



Mô hình trong báo cáo sẽ được huấn luyện theo hướng điều chỉnh các trọng số trong mạng, dựa trên việc tối thiểu hàm lỗi:

Trong đó, D là tập dữ liệu huấn luyện, x là dữ liệu đầu vào với nhãn cho ví dụ học tương ứng là y_x. là giá trị dự đoán tương ứng với x. Số lần lặp qua toàn bộ tập D gọi là epoch. Với mỗi epoch tập D sẽ được chia ra thành các tập dữ liệu nhỏ gọi là batch, mỗi batch sẽ được sử dụng để cập nhập trọng số 1 lần.

Một số hàm loss thường gặp:

+ Mean absolute error:

+ Mean square error:

+ Cross-entropy-loss:

Một số hàm tối ưu thường sử dụng:

+ SGD: Giả sử hàm loss có tham số và có đạo hàm và learning rate thì công thức cập nhập trọng số là:

+ SGD with Momentum:

Dưới góc nhìn vật lí, SGD với momentum giúp cho việc hội tụ có gia tốc, làm nhanh quá trình hội tụ trên các đường cong có độ dốc lớn, nhưng cũng đồng thời làm giảm sự dao dộng khi gần hội tụ.

Công thức có thêm bước cập nhập vận tốc bằng cách tính tích vận tốc của lần cập nhập trước đó với hệ số thường sấp xỉ 0.9:

+ Adagrad

Không giống như các cách thức trước, learning rate hầu như giống nhau cho quá trình learning, adagrad coi learning rate cũng là một tham số

Nó update tạo các update lớn với các dữ liệu khác biệt nhiều và các update nhỏ cho các dữ liệu ít khác biệt

Adagrad chia learning rate với tổng bình phương của lịch sử biến thiên (đạo hàm)

Trong đó:

- là hệ số để tránh lỗi chia cho 0, default 1e-8
- là tổng bình phương của đạo hàm vector tham số tại lần lặp t

Một lợi ích dễ thấy của Adagrad là tránh việc điều chỉnh learning rate bằng tay, thường sẽ để default là 0.01 và thuật toán sau đó sẽ tự động điều chỉnh.

Một điểm yếu của Adagrad là tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm learning rate cực kì nhỏ, làm việc traning trở nên đóng băng.

+ RMSprop

RMSprop giải quyết vấn đề tỷ lệ học giảm dần của Adagrad bằng cách chia tỷ lệ học cho trung bình của bình phương gradient.

Ưu điểm rõ nhất của RMSprop là giải quyết được vấn đề tốc độ học giảm dần của Adagrad (vấn đề tốc độ học giảm dần theo thời gian sẽ khiến việc training chậm dần, có thể dẫn tới bị đóng băng)

Thuật toán RMSprop có thể cho kết quả nghiệm chỉ là local minimum chứ không đạt được global minimum như Momentum. Vì vậy người ta sẽ kết hợp cả 2 thuật toán Momentum với RMSprop cho ra 1 thuật toán tối ưu Adam.

+ Adam

Adam là sự kết hợp của Momentum và RMSprop. Nếu giải thích theo hiện tượng vật lí thì Momentum giống như 1 quả cầu lao xuống dốc, còn Adam như 1 quả cầu rất nặng có ma sát, vì vậy nó dễ dàng vượt qua local minimum tới global minimum và khi tới global minimum nó không mất nhiều thời gian dao động qua lại quanh đích vì nó có ma sát nên dễ dừng lại hơn.

3.1.2 Vì sao chọn ANN cho bài toán

Từ tập dữ liệu ta thấy các trận đấu có 3 kết quả thắng, hoà, thua, từ đó ta có thể đưa bài toán dự đoán kết quả trận đấu về bài toán phân loại các nhãn lớp khác nhau. Với ANN, mô hình có thể nhận một lượng lớn dữ liệu từ lớp đầu vào, tính toán, trích xuất các đặc trưng riêng và gửi dữ liệu đến các nút ở lớp tiếp theo. Các lớp ẩn của mạng cũng thực hiện các công việc tương tự. Cuối cùng, các thông số được được đưa qua lớp đầu ra để tính toán tỷ lệ nhãn lớp tương ứng với dữ liệu đầu vào, kết quả chính là nút có tỷ lệ cao nhất.

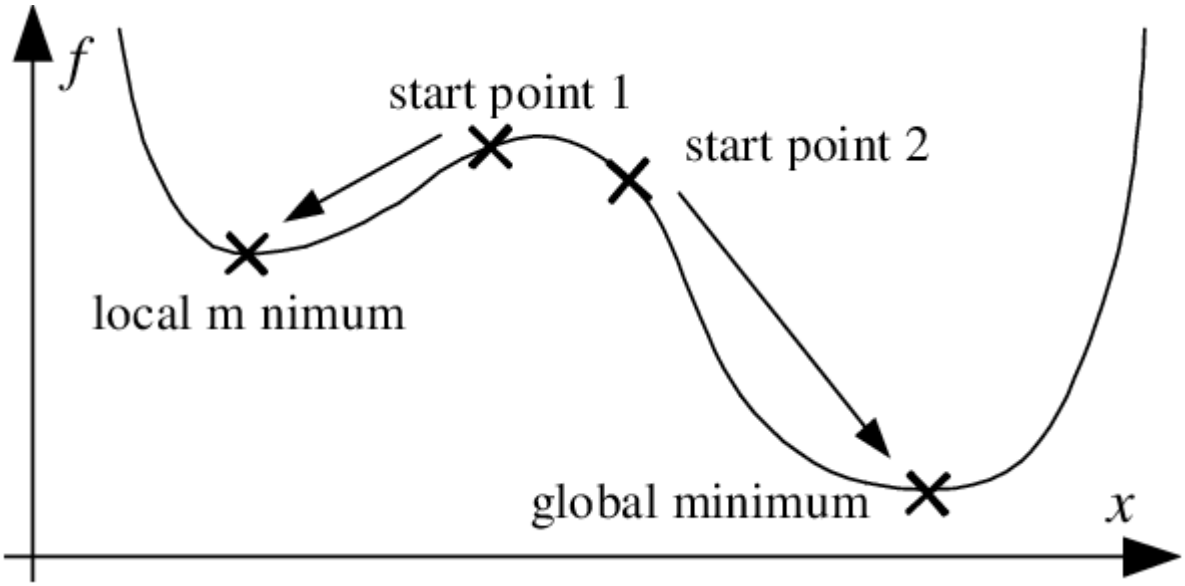
3.1.3 Thực nghiệm

Mô hình mạng nơ ron nhân tạo áp dụng cho bài toán sẽ được kiểm tra độ hiệu quả đạt được khi thay đổi các tham số: learning rate (tốc độ học), activation (hàm kích hoạt). Mô hình chức năng (functional) được xây dựng với 2 lớp trung gian kết nối đầy đủ, mỗi lớp có 128 nút. Lớp đầu vào có 9 nút tương ứng mới số thuộc tính đầu vào. Lớp đầu ra có 3 nút với hàm kích hoạt softmax tương ứng với tỷ lệ lớn nhất của kết quả trận đấu. Thuật toán tối ưu (optimizer) được sử dụng sẽ là Adam. Hàm lỗi là Sparse Categorical Crossentropy. Kích thước của batch, hay số mẫu cho một lần huấn luyện là 512. Số lần lặp qua toàn bộ training set (epoch) là 1000.

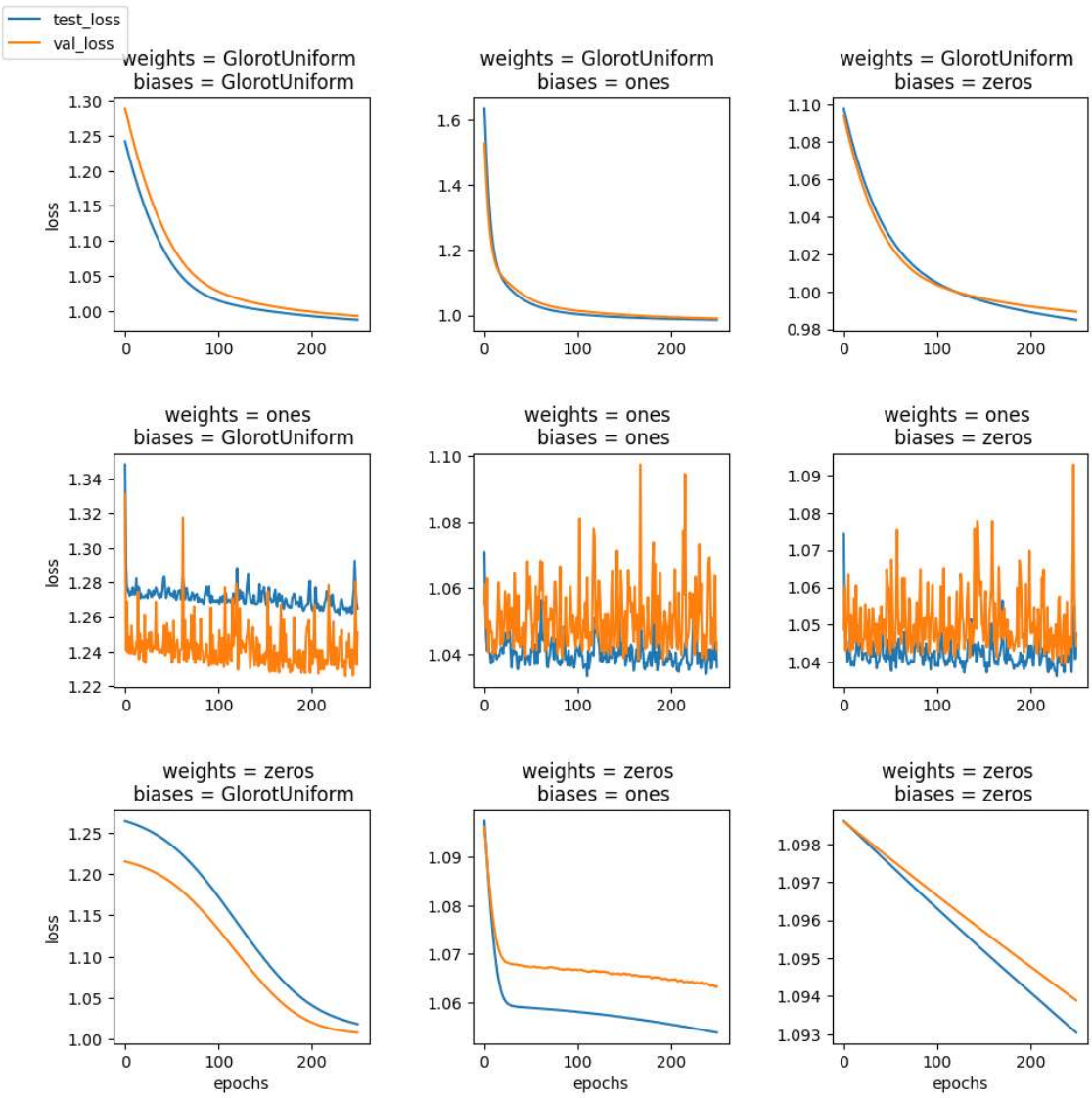
Để đánh giá hiệu năng của mô hình ta sẽ cắt ¼ tập dữ liệu training để làm tập validation

3.1.3.1 Lựa chọn tham số khởi tạo

Mô hình mạng nơ ron tối ưu trọng số (weight) bằng cách sử dụng các thuật toán cơ bản và nâng cao của gradient descent. Các phương pháp này điều chỉnh trọng số của mô hình để cực tiểu hoá hàm loss, với mục đích giúp cho mô hình nâng cao khả năng dự đoán chính xác. Tuy nhiên, hàm loss của bài toán không phải lúc nào cũng là hàm bao lồi, điều này dẫn đến việc các trọng số có thể thay đổi tới nghiệm tối ưu cục bộ thay vì toàn cục. Vì vậy, việc khảo sát để lựa chọn các trọng số khởi tạo chính xác là cần thiết.



Ta sẽ khảo sát sự thay đổi của hàm loss trên tập training và trên tập validation với các cách khởi tạo khác nhau. Đặt learning rate = , hàm activation relu. Chúng ta sẽ khởi tạo cho weights và biases bằng các phương pháp: sử dụng phân phối đồng nhất glorot, khởi tạo toàn bộ trọng số bằng 1, khởi tạo toàn bộ trọng số bằng 0.



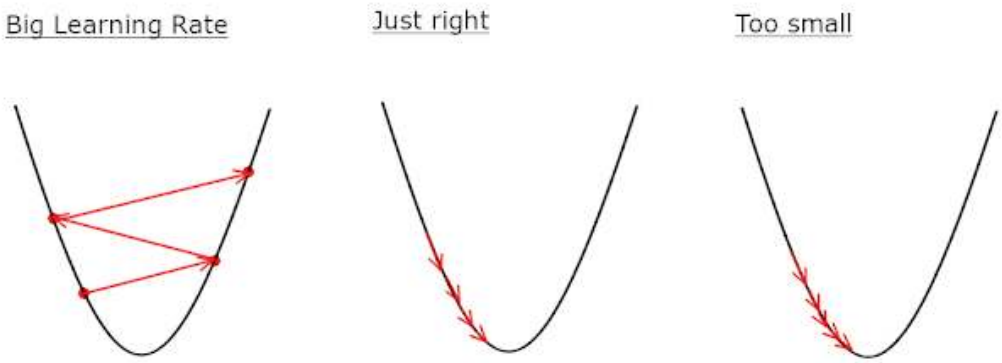
Xét khởi tạo weight = 1, ta thấy hàm loss của tập train và tập validation đều không ổn định. Điều này là do trọng số khởi tạo quá lớn dẫn đến việc bùng nổ gradient của các nút (exploding gradients), dẫn đến việc cập nhập nhập trọng số vượt qua giá trị nghiệm tối ưu.

Xét khởi tạo weights = 0. Với khởi tạo biases = 0, gradient của hàm số sấp xỉ 0 dẫn tới việc trọng số gần như không được cập nhập (vanishing gradient) và hàm loss gần như không thay đổi. Với 2 khởi tạo biases còn lại, ta thấy hàm loss giảm khá nhanh, tuy nhiên tốc độ giảm hàm loss của tập train và tập validation là khác nhau dẫn tới việc mô hình dễ bị overfitting.

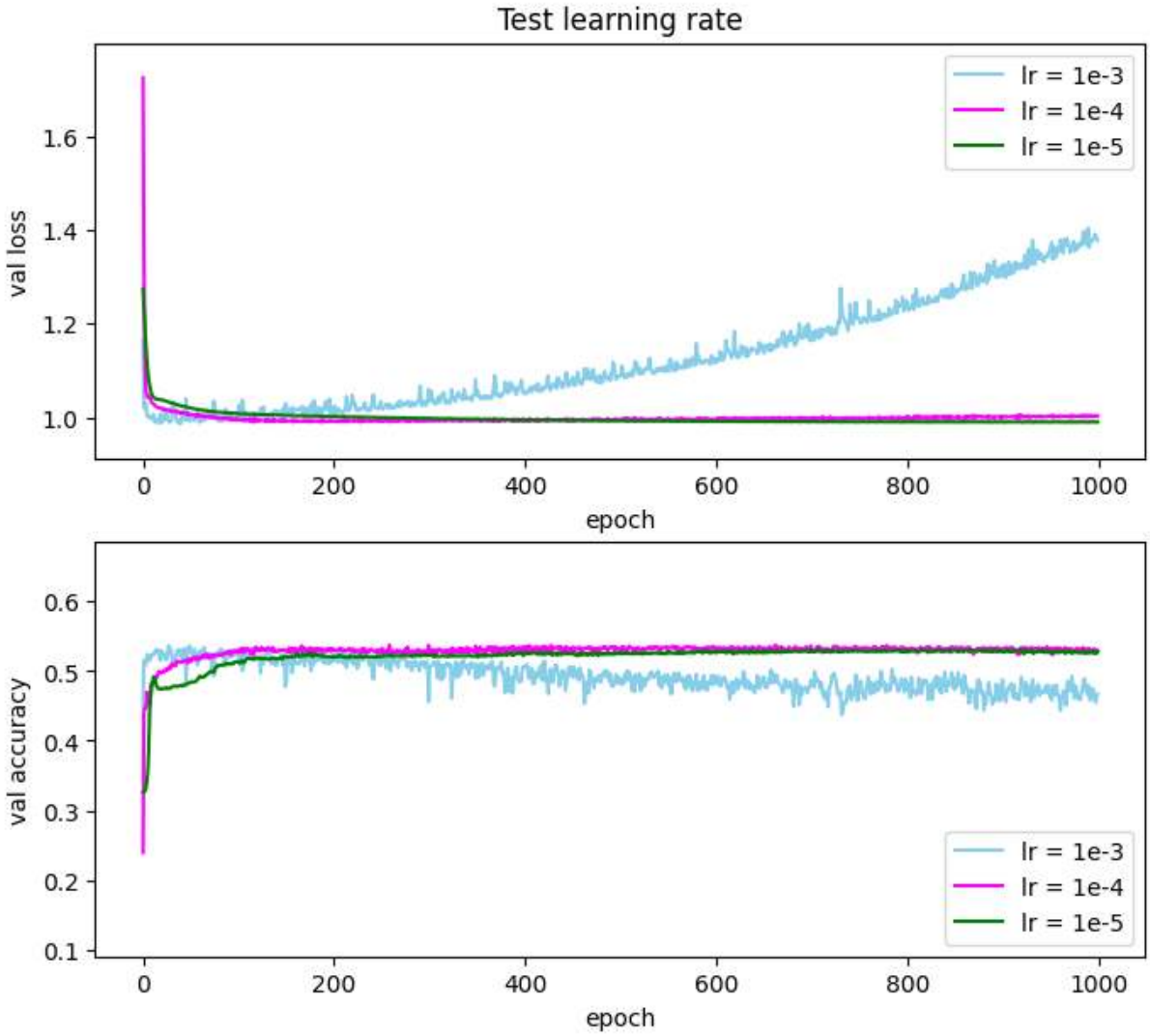
Xét khởi tạo weights = phân phối đồng nhất glorot thì trong cả 3 đồ thị ta thấy hàm loss của tập train và tập validation khá sấp xỉ nhau qua từng epoch. Tuy nhiên, ta sẽ chọn khởi tạo tham số biases = 1 cho các lần điều chỉnh tham số tiếp theo vì

giá trị hàm loss của tập train và tập validation với khởi tạo này gần nhau nhất qua các lần cập nhập trọng số.

3.1.3.2 Điều chỉnh tốc độ học



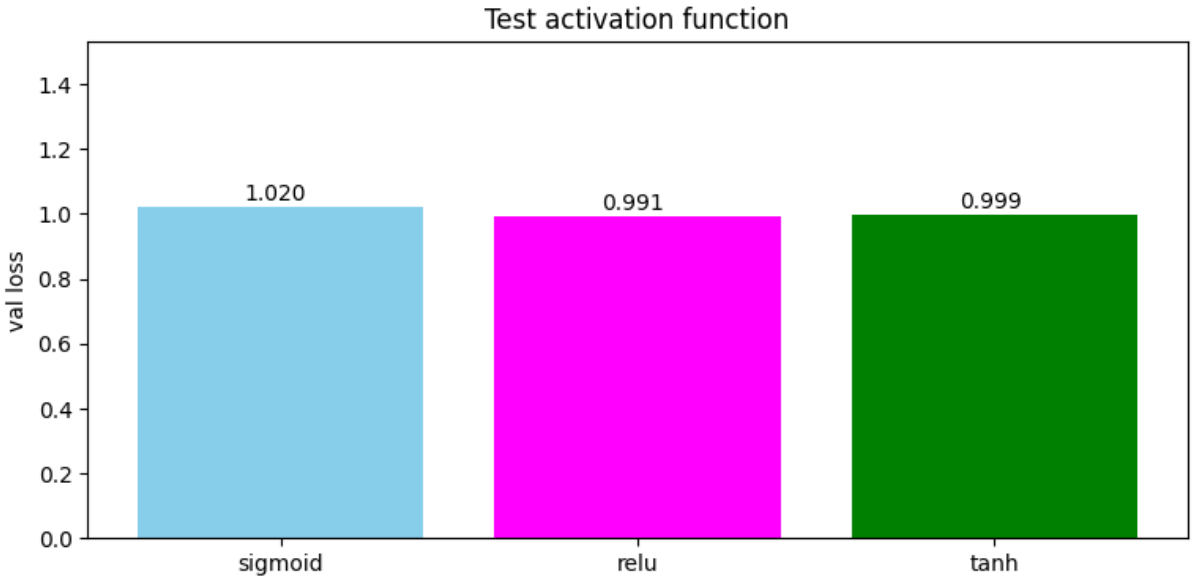
Ta sẽ khảo sát hàm lỗi và độ chính xác trên tập validation của mô hình với các tốc độ học khác nhau (0.001, 0.0001 và 0.00001). 2 lớp trung gian có activation lần lượt là tanh và relu. Ta có thể thấy khi thì giá trị hàm loss ngày càng tăng (xấp xỉ 1.4 ở epoch 1000) và accuracy ngày càng giảm (xấp xỉ 0.46 ở epoch 1000), điều này là do mỗi lần cập nhập, trọng số w bị thay đổi quá lớn dẫn đến trọng số w không những không thể đạt tới điểm tối ưu của hàm loss mà ngược lại, còn khiến cho giá trị loss tăng lên. Với learning rate bằng và , có thể thấy giá trị của hàm loss và accuracy tương ứng với 2 giá trị learning rate này bắt xấp xỉ nhau từ khoảng epoch 400 (giá trị loss duy trì xấp xỉ 1.0 và accuracy xấp xỉ 0.52) và các giá trị này có xu hướng không đổi sau đó.

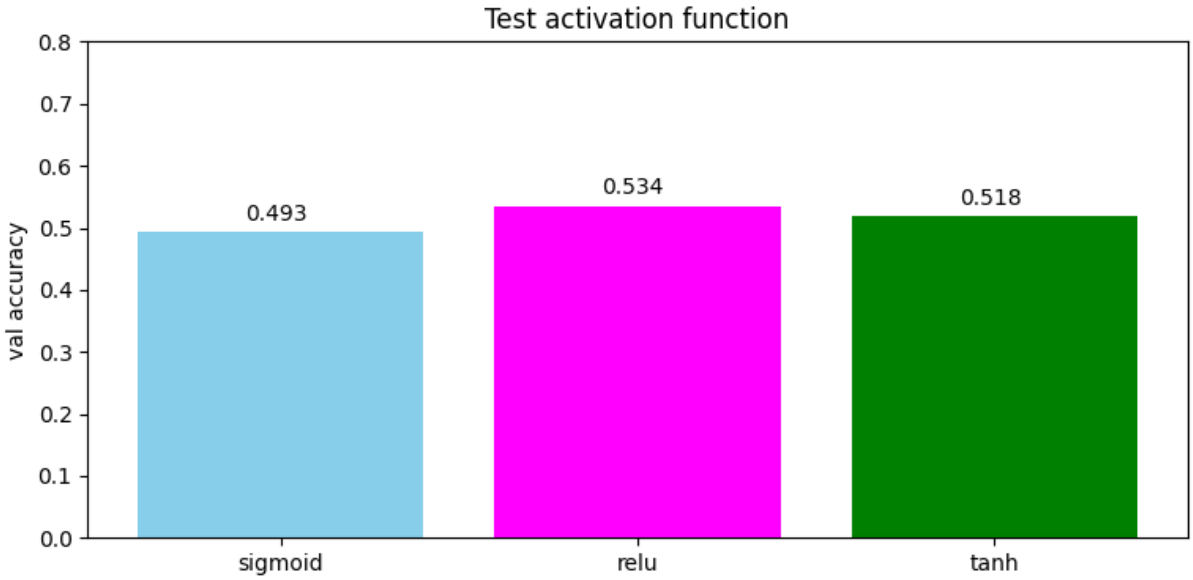


3.1.2.3 Khảo sát độ hiệu quả với các hàm activation khác nhau

Ta sẽ khảo sát hàm loss và accuracy trên tập validation của mô hình với và các hàm activation khác nhau cho 2 lớp trung gian (sigmoid, relu, tanh). Ta có thể thấy (hình dưới) với hàm sigmoid, mô hình có độ chính xác thấp nhất. Ngược lại, khi sử dụng hàm relu, mô hình đạt được hiệu quả cao hơn so với hàm sigmoid và tanh. Hàm Sigmoid có điểm yếu là dễ bị bão hòa và bị ảnh hưởng bởi sự suy giảm gradient. Khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với các nốt đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*) dẫn đến tốc độ học vô cùng chậm. Ngoài ra hàm sigmoid có trung tâm khác 0 cũng làm chậm tốc độ học trong một vài trường hợp cụ thể. Hàm Tanh cũng dễ bị bão hòa về hai đầu, nhưng hàm này có giá trị gradient nói chung lớn hơn so với Sigmoid và nếu giá trị của các thuộc tính đầu vào đối xứng xung quanh điểm

không, tốc độ hội tụ sẽ vô cùng nhanh. Hàm relu có tốc độ học nhanh hơn hẳn. Điều này có được là do hàm relu không bị bão hoà về 2 đầu như tanh và sigmoid. Hơn nữa công thức hàm relu khá đơn giản giúp tiết kiệm chi phí khi tính toán.





3.2 XGBOOST

3.2.1 Cơ sở lý thuyết

Trước khi giới thiệu cụ thể về XGBoost, chúng ta cần biết rằng XGBoost là 1 thuật toán dạng ensemble learning. Vậy thì trước hết chúng ta hãy tìm hiểu Ensemble learning là gì

3.2.1.1 Ensemble learning

Giả sử chúng ta có 3 nhóm giải quyết một bài toán phân loại sản phẩm. Mỗi nhóm sử dụng một thuật toán khác nhau để train model trên cùng 1 tập train và đánh giá kết quả trên cùng 1 tập validation, và ra 3 kết quả khác nhau, giả sử là 60%, 66%, 72%. Điều này hoàn toàn có thể lí giải được bởi mỗi thuật toán có một cách tiếp cận, cách giải quyết khác nhau và tất nhiên cũng cho ra kết quả khác nhau.

Trong machine learning tồn tại định lý "không có bữa trưa miễn phí" (No free lunch theorem), tức là không tồn tại một thuật toán mà luôn tốt cho mọi ứng dụng và mọi tập dữ liệu, vì các thuật toán machiner learning thường dựa trên một tập các tham số (hyperparameters) hoặc một giả thiết nhất định nào đó về phân bố dữ liệu. Điều này giống như việc ta có một đội quân binh hùng tướng mạnh, nhưng mà mỗi một hoàn cảnh, một môi trường, lại có những tướng có những điểm yếu riêng, vậy thì phương pháp tối ưu nhất đó chính là "đoàn kết là sức mạnh", chúng ta sẽ kết hợp những thuật toán này lại, đấy chính là ensemble learning. Các mô hình khác nhau có khả năng khác nhau, có thể thực hiện tốt nhất các loại công việc khác nhau (subtasks), khi kết hợp các mô hình này với nhau một cách hợp lý thì sẽ tạo thành một mô hình kết hợp (combined model) mạnh có khả năng cải thiện hiệu suất tổng thể (overall performance) so với việc chỉ dùng các mô hình một cách đơn lẻ.

- Các phương pháp Ensemble Learning được chia thành 2 dạng:

+ Bagging xây dựng một lượng lớn các models (thường là cùng loại) trên những subsamples khác nhau từ tập training dataset một cách song song nhằm đưa ra dự đoán tốt hơn, với các thuật toán nổi tiếng như là Random Forest,

+ Boosting xây dựng một lượng lớn các models (thường là cùng loại). Tuy nhiên quá trình huấn luyện trong phương pháp này diễn ra tuần tự theo chuỗi (sequence). Trong chuỗi này mỗi model sau sẽ học cách sửa những errors của model trước (hay nói cách khác là dữ liệu mà model trước dự đoán sai). Có thế lấy ví dụ một vài thuật toán như LightGBM, Gradient Boosting, XGBoost, ….

=> Như vậy XGBoost là 1 thuật toán ensemble learning dạng Boosting

3.2.1.2 Cách hoạt động của XGBoost

Ý tưởng của XGB thông qua ví dụ sau:

Cho bảng dữ liệu bên dưới:

Đội bóng	Mùa giải	Bàn thắng	Bàn thua	Điểm số
Man City	2016/2017	90	39	88
Chelsea	2016/2017	85	33	93
Liverpool	2016/2017	83	42	81
Arsenal	2016/2017	77	44	75

Bài toán đạt ra là cần dự đoán điểm số dựa trên các input features: Bàn thắng, bàn thua.

Bước 1: Huấn luyện mô hình decision tree thứ nhất trên tập dữ liệu bên trên (Decision Tree sẽ được đề cập ở phần 3.2.1.3)

Bước 2: Tính toán lỗi dựa theo sai số giữa giá trị thực tế và giá trị dự đoán

Đội bóng	Mùa giải	Bàn thắng	Bàn thua	Điểm số (Output1)	Dự đoán 1	Error 1
Man City	2016/2017	90	39	88	89	-1
Chelsea	2016/2017	85	33	93	91	2
Liverpool	2016/2017	83	42	81	83	-2
Arsenal	2016/2017	77	44	75	73	2

Bước 3: Một mô hình decision tree thứ 2 được tạo, sử dụng cùng input features với model trước đó, nhưng target là Error 1.

Bước 4: Giá trị dự đoán của mô hình thứ 2 được cộng với giá trị dự đoán của mô hình thứ nhất.

Đội bóng	Mùa giải	Bàn thắng	Bàn thua	Điểm số (Output1)	Dự đoán 1	Error 1 (New target)	Dự đoán 2	Combine prediction
Man City	2016/2017	90	39	88	89	-1	-1	88
Chelsea	2016/2017	85	33	93	91	2	1	92
Liverpool	2016/2017	83	42	81	83	-2	-3	80
Arsenal	2016/2017	77	44	75	73	2	1	74

Bước 5: Giá trị kết hợp ở bước 3 coi như là giá trị dự đoán mới. Ta tính lỗi (Error 2) dựa trên sai số giữa giá trị này và giá trị thực tests.

Đội bóng	Mùa giải	Bàn	Bàn	Điểm số	Dự	Error 1	Dự	Combine	Error 2

		thắng	thua	(Output1)	đoán 1	(New target)	đoán 2	prediction	
Man City	2016/2017	90	39	88	89	-1	-1	88	0
Chelsea	2016/2017	85	33	93	91	2	1	92	1
Liverpool	2016/2017	83	42	81	83	-2	-3	80	1
Arsenal	2016/2017	77	44	75	73	2	1	74	1

Bước 6: Lặp lại bước 2-5 cho đến khi số lượng weak learner đạt được hoặc giá trị lỗi không đổi.

3.2.1.3 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 Classification và Regression 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 train và sử dụng cây quyết định này dự báo trên tập kiểm tra.

Mỗi nhánh của cây giống như nhánh if, else nên có thể hiểu cây quyết định là tập hợp các luật nếu thì

Bài toán Classification trong decision tree

- Biểu diễn dữ liệu:

+ Mỗi đối tượng quan sát được thể hiện bởi một vector n chiều, mỗi chiều đại diện cho một thuộc tính đối tượng.

+ Mỗi đỉnh bên trong cây biểu diễn một thuộc tính để kiểm tra dữ liệu đến về sau

+ Mỗi nhánh xuất phát từ một đỉnh tương ứng với mỗi một giá trị trong miền giá trị của thuộc tính đó

+ Mỗi lá lưu nhãn lớp

+ Cho dữ liệu cần phán đoán duyệt qua cây đấy, dùng các thuộc tính của nó duyệt từ gốc đến lá, lấy nhãn ở lá để phán đoán cho dữ liệu mới

Một số thông số quan trọng của Decision Tree:

- Entropy:

$$Entropy(S) = -\sum_{i=1}^{c} p_i log_2 p_i$$

- Information Gain:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- Thuật toán ID3 là thuật toán tham lam được đề xuất bởi Ross Quinlan năm 1986 và được dùng để học cây quyết định. Đây là thuật toán được thiết kế dưới dạng top-down. Thuật toán hoạt động như sau:

+ Tại mỗi đỉnh N, chọn thuộc tính kiểm tra A để giúp chúng ta phân loại tốt nhất dữ liệu tại N.

+ Làm tương tự với các đỉnh còn lại đến khi nào cây phán đoán chính xác tất cả các dữ liệu huấn luyện hoặc các thuộc tính đã dùng hết rồi.

+ Mỗi thuộc tính chỉ được sử dụng nhiều nhất 1 lần trên đường đi từ gốc tới lá.

+ ID3 sử dụng Information Gain và Entropy để chọn thuộc tính kiểm tra A

3.2.1.4 Một số thông số quan trọng của XGBoost

- objective : Lựa chọn hàm mất mát trong XGBoost

- n_jobs: Số lượng cores của hệ thống được sử dụng để chạy model. Giá trị mặc định là -1, XGBoost sẽ tự động phát hiện và sử dụng tất cả các cores.

Ở đây, nhóm chúng ta sẽ đặt thông số n_jobs là -1

- n_estimators : Số decision tree được dùng

- Subsample : chọn ngẫu nhiên các samples từ tập train set, ta sẽ để mặc định là 0.7

- max_depth: Độ sâu tối đa của decision tree, ta sẽ điều chỉnh độ sâu này

- learning_rate: Tốc độ học của mô hình

- n_estimators : Số decision tree được dùng

Vì learning_rate và n_estimators có mối quan hệ tương quan đối nhau, nên ta sẽ để cố định learning rate là 0.1 và thay đổi số cây

- Alpha: Lasso regularization, là một loại regularization được sử dụng để giúp huấn luyện mô hình có độ chính xác cao hơn bằng cách tạo ra mô hình có nhiều feature hơn và giảm đi các feature không cần thiết. Lasso regularization sẽ làm cho các trọng số của các features trở nên âm hoặc bằng 0, điều này giúp cho mô hình trở nên "sparse", tức là chỉ có ít feature được sử dụng trong quá trình dự đoán

- Lambda: Ridge regularization, là một loại regularization khác được sử dụng để giúp huấn luyện mô hình có độ chính xác cao hơn bằng cách giữ nguyên tất cả các feature và giảm đi độ lớn của các trọng số của chúng. Ridge regularization sẽ làm cho các trọng số của các features trở nên nhỏ hơn, nhưng không bằng 0 như trong trường hợp của Lasso regularization.

Các mô hình dùng Ridge thường có phương sai thấp hơn và ít nhạy cảm hơn so với Lasso với lựa chọn dữ liệu huấn luyện cụ thể. Chính vì vậy ta sẽ đặt alpha = 0 và lambda = 1

- Grow policy: Kiểm soát cách các nút mới được thêm vào cây

+ Depthwise: phân chia tại các nút gần gốc nhất.

+ Lossguide: phân chia tại các nút có thay đổi tổn thất cao nhất

Trong bài toán này, ta chọn lossguide

3.2.1.5 Hàm mất mát của XGBoost

Vì đây là dạng bài toán prediction/classfication, vậy nên chúng ta sẽ sử dụng hàm softmax, có công thức là:

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

3.2.1.6 Hàm tối ưu của XGBoost

- Gbtree: sử dụng cây quyết định để tối ưu hóa hàm mất mát.

- Gblinear: sử dụng mô hình tuyến tính để tối ưu hóa hàm mất mát.

- Dart: sử dụng mô hình tuyến tính và cây quyết định để tối ưu hóa hàm mất mát.

Ở đây, chúng ta sẽ sử dụng gbtree, với cách thức hoạt động từ các cây quyết định như đã trình bày ở trên.

3.2.2 Vì sao chọn XGBoost cho bài toán

Khả năng mở rộng: XGBoost được thiết kế để mở rộng tốt cho các bộ dữ liệu rất lớn và có thể được sử dụng trong cài đặt phân tán.

Tính linh hoạt: XGBoost cho phép bạn xác định các hàm mất mát tùy chỉnh, chỉ số đánh giá và xử lý các giá trị bị thiếu, giúp nó rất linh hoạt cho các loại tác vụ phân loại khác nhau.

Độ bền: XGBoost đã được chứng minh là mạnh mẽ đối với nhiễu và ngoại lệ trong dữ liệu, điều này có thể quan trọng đối với các tác vụ phân loại trong đó dữ liệu có thể bị nhiễu hoặc có lỗi.

3.2.3 Thực nghiệm

3.2.3.1 Đánh giá hiệu năng mô hình thông qua kỹ thuật K-fold cross-validation

K-fold cross-validation là một phương pháp để đánh giá hiệu quả của một mô hình học máy trên tập dữ liệu. Nó được thực hiện bằng cách chia tập dữ liệu thành K phần động và sau đó chạy K lần quá trình huấn luyện và đánh giá, mỗi lần sử dụng một phần khác nhau làm tập dữ liệu kiểm tra.

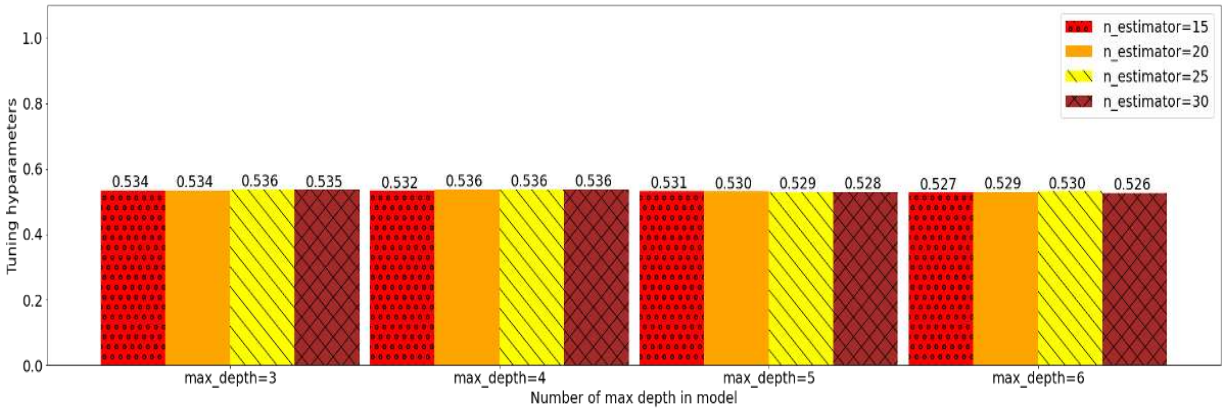
Trong trường hợp đơn giản nhất, đó là "Straight KFold", tập dữ liệu được chia thành K phần động có kích thước tương đương và sau đó mô hình được huấn luyện và đánh giá trên mỗi phần một lần, sử dụng các phần còn lại làm tập dữ liệu kiểm tra.

Ví dụ, nếu chúng ta có một tập dữ liệu gồm 10 mẫu và sử dụng K=5, thì Kfold cross-validation sẽ chia tập dữ liệu thành 5 phần động có kích thước 2 mẫu mỗi phần. Sau đó, mô hình sẽ được huấn luyện và đánh giá 5 lần, mỗi lần sử dụng một phần khác nhau làm tập dữ liệu kiểm tra và các phần còn lại làm tập dữ liệu huấn luyện.

Để đánh giá hiệu năng mô hình mà chúng ta đang xét, ta sẽ thực hiện kỹ thuật này trên tập huấn luyện với K=17 (vì tập huấn luyện của chúng ta bao gồm 17 mùa giải).

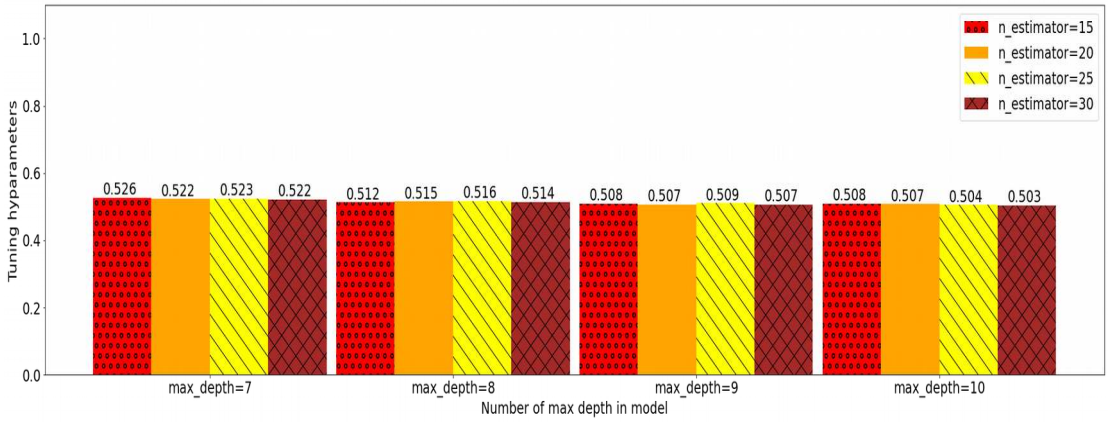
3.2.3.2 Điều chỉnh các tham số

Có 2 tham số rất quan trọng, đó là n_estimator và max_depth, vì vậy ta sẽ khảo sát chúng trên kỹ thuật Straight Kfold. Nếu như điểm của chúng càng cao, tức là mô hình đang hoạt động ổn định, ngược lại, mô hình của bạn sẽ tăng khả năng overfitting nếu điểm Kfold thấp.



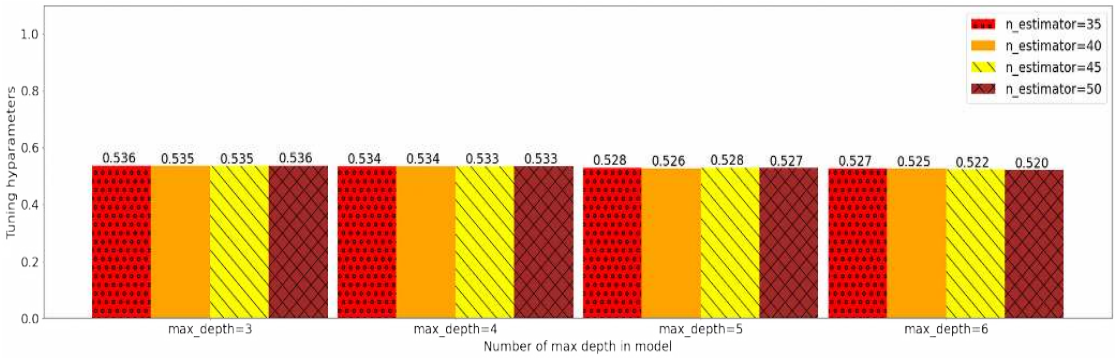
Như ta thấy, với max_depth trong khoảng từ 3 đến 6 và số cây quyết định từ 15đến 30, các mô hình hoạt động rất ổn định và không có quá nhiều sự chênh lệch.

Tiếp theo, ta khảo sát với chỉ số max_depth lớn hơn.



Với chỉ số max_depth từ 7 đến 10, các mô hình bắt đầu giảm dần số điểm. Vì vậy, ta thấy thuật toán của hoạt động hiệu quả với max_depth từ 3 đến 6.

Với max_depth từ 3 đến 6, ta lại thực hiện tăng số cây quyết định lên.

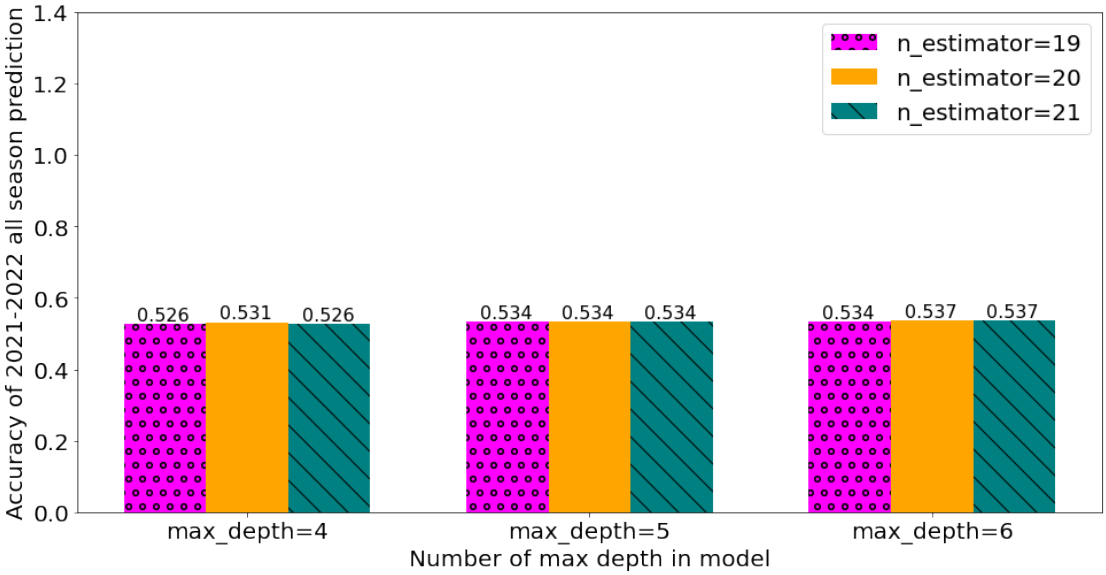


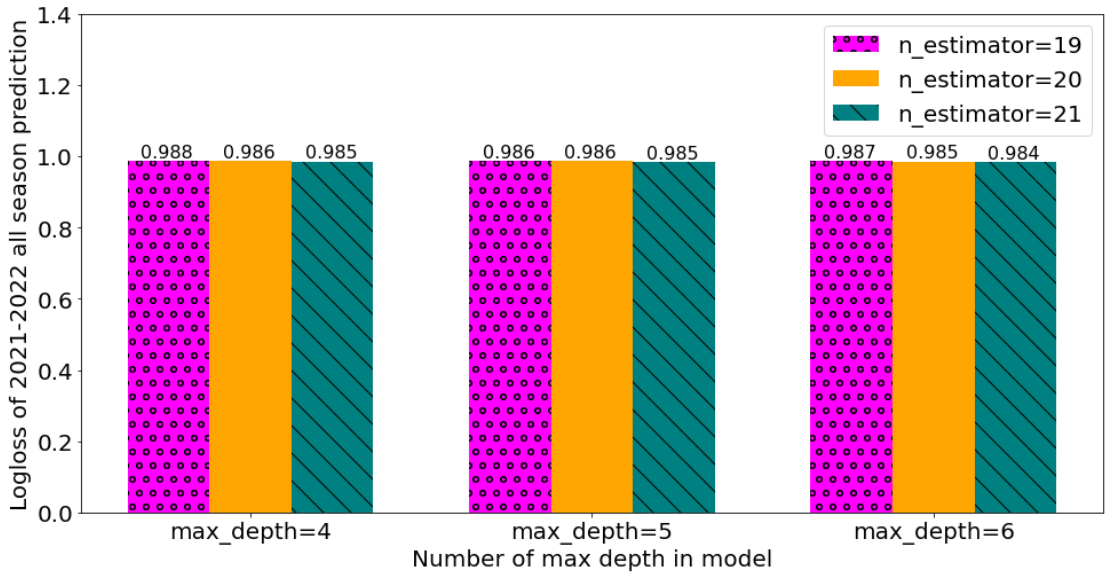
Ta nhận thấy rằng, với số cây lớn hơn, thì thuật toán với max_depth bằng 3 và 4 không có quá nhiều sự thay đổi (thậm chí là còn nhỉnh hơn), tuy nhiên với max_depth bằng 5 và 6, thuật toán lại có độ chính xác thấp hơn hẳn.

Như vậy, với learning rate = 0.1, thì max_depth trong khoảng từ 4 và 6 và số cây trong khoảng từ 15 đến 30, nhất là dao động trong khoảng 20 là hiệu quả nhất.

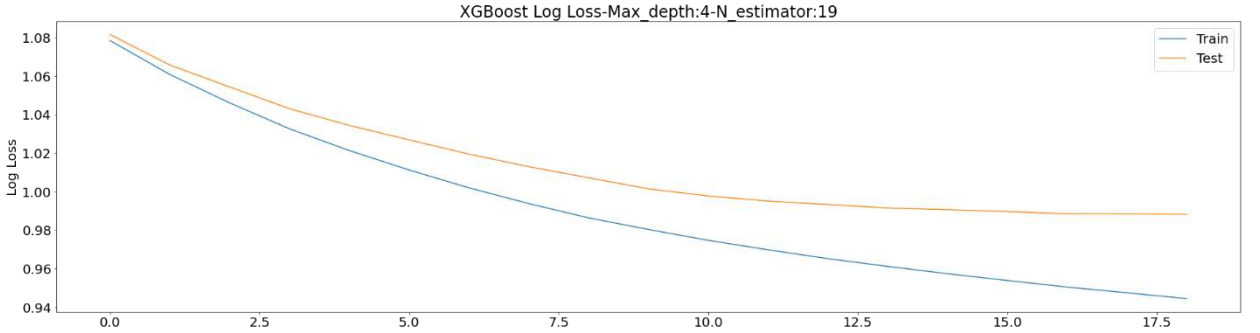
3.2.3.3 Thực hiện kiểm tra trên tập test

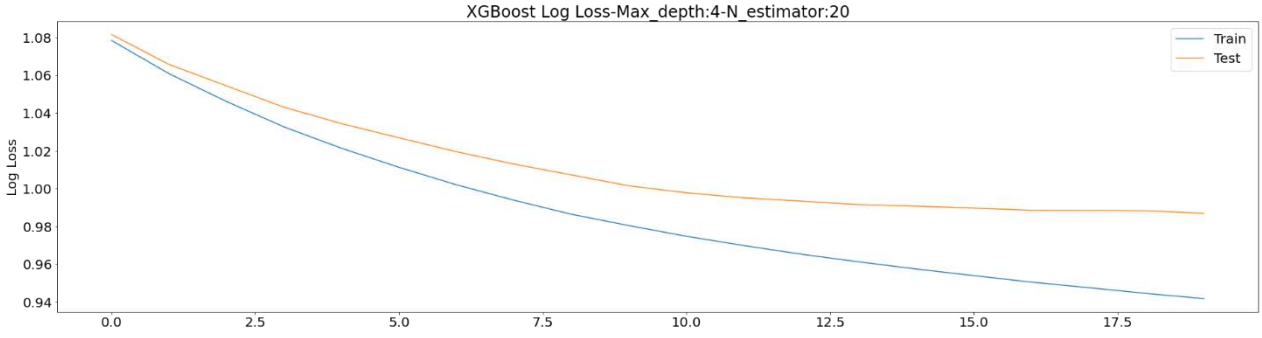
Dựa theo việc điều chỉnh tham số ở mục 2.2.2, ta sẽ kiểm tra mô hình trên tập test, tức là các trận đấu từ vòng 4 đến vòng 38 của Premier League 2021/22 với các tham số max_depth là 4, 5, 6 và n_estimator là 14, 15, 16, và ta thu được những kết quả rất ổn định:

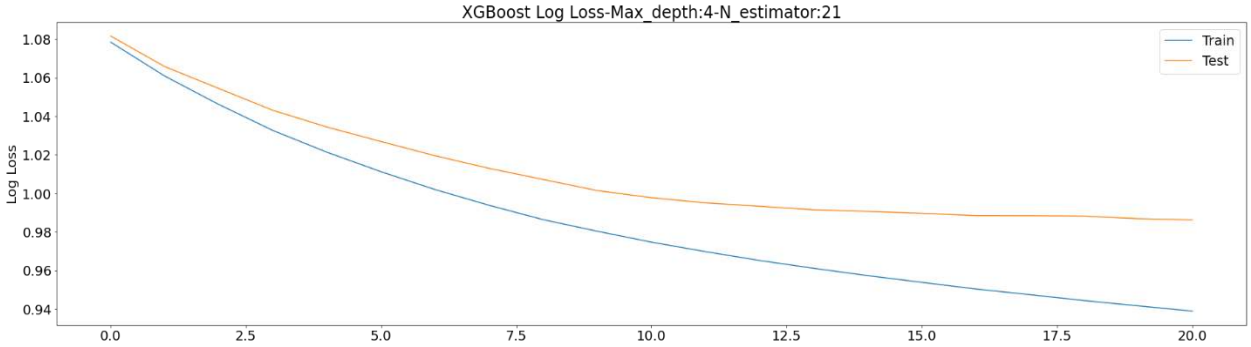




Ta thấy rằng kết quả cross-val score và kết quả khi thực hiện trong tập kiểm tra không có quá nhiều sự chênh lệch (trong khoảng từ 0.526 đến 0.539), và sự chênh lệch giữa các mô hình với nhau trên tập test (trong khoảng từ 0.526 đến 0.534) điều này chứng tỏ các mô hình của chúng ta có 1 sự hội tụ tốt. Ngoài ra, ta có thể thấy, với số cây từ 14 tăng lên 16 thì độ chính xác cũng tăng với từng max_depth, điều này chứng tỏ số cây đang dần hội tụ về số cây tối ưu nhất. Tuy nhiên, không nên tăng số cây quá cao, vì rất dễ overfitting. Tương tự với max_depth, ta có thể chứng minh qua các kết quả sau:

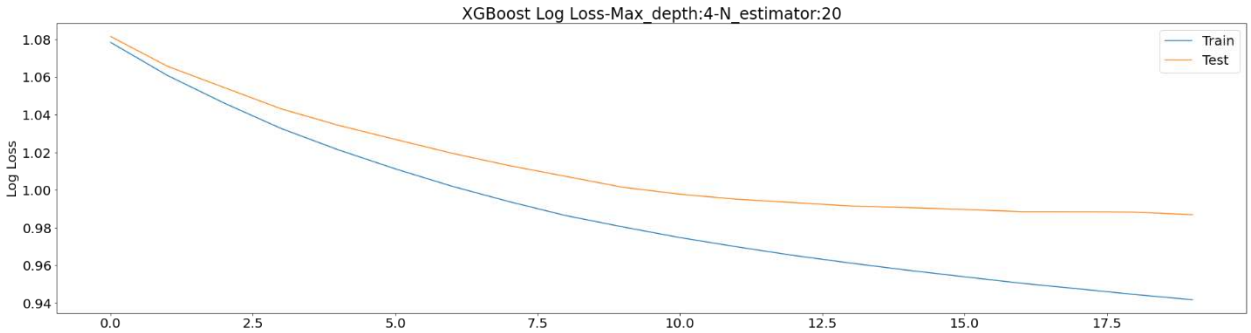


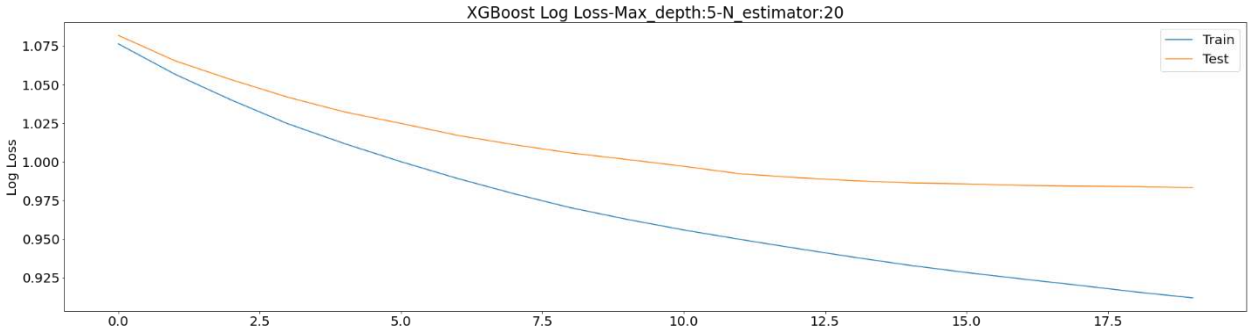


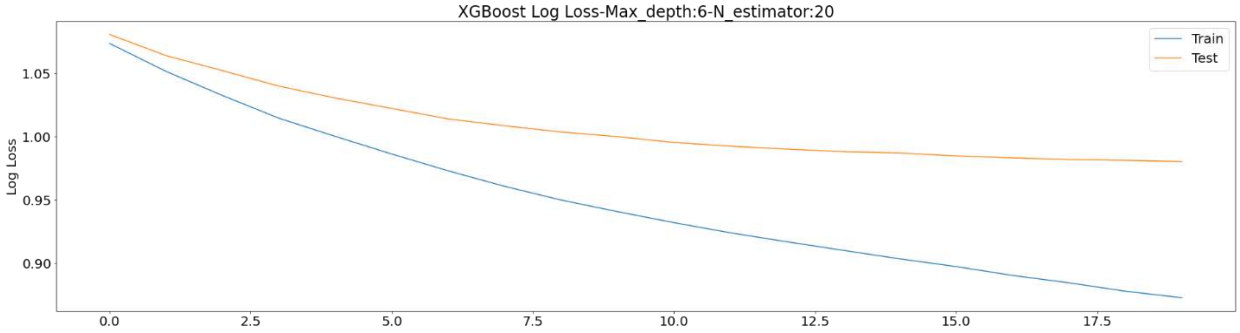


Ta thấy, với n=21, chênh lệch giá trị hàm loss của tập train và test cao hơn so với n=19 và n=20.

Tương tự với max_depth là 5 và 6:







Ta có thể thấy với max_depth tăng từ 4 lên 6, khoảng cách giữa train loss và test loss còn có dấu hiệu tăng rõ rệt hơn, và nếu như ta tăng max_depth > 6, rõ ràng overfitting sẽ xảy ra.

3.3 K-NEAREST NEIGHBORS

3.3.1 Cơ sở lý thuyết

K-nearest neighbors là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi huấn luyện, thuật toán này không học một điều gì từ dữ liệu huấn luyện (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.

Với KNN, trong bài toán Classification, nhãn (label) của một điểm dữ liệu mới được suy ra trực tiếp từ k điểm dữ liệu gần nhất trong tập huấn luyện. Nhãn của một dữ liệu kiểm thử có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra nhãn

Có hai thành phần chính trong KNN là: lựa chọn số hàng xóm và cách tính khoảng cách giữa hai điểm dữ liệu. Trong thực tế, nên sử dụng số hàng xóm hơn để dự đoán ($k > 1$), nhưng không nên chọn quá nhiều. Nếu chọn quá nhiều hàng xóm, thì sẽ bị phá vỡ cấu trúc vốn có của dữ liệu, và do đó, dự đoán có thể không tốt

Trong KNN, việc chuẩn hoá các thuộc tính đôi khi rất quan trọng để có được khả năng dự đoán tốt. Nếu dự liệu không được chuẩn hoá, thì rất có thể độ lớn của một thuộc tính nào đó có thể đóng vai trò quan trọng và lấn át một cách giả tạo các thuộc tính khác.

3.3.1.1 Một số thuộc tính quan trọng trong KNN

- N_neighbors: Số điểm lân cận được xét trong KNN

- Weights: Cách đánh trọng số của các điểm dữ liệu, có 2 cách phổ biến :

- Uniform: Tất cả các điểm trong mỗi vùng lân cận đều có trọng số như nhau

- Distance: Trọng số bằng nghịch đảo của khoảng cách của chúng. trong trường hợp này, các hàng xóm gần điểm truy vấn hơn sẽ có ảnh hưởng lớn hơn các hàng xóm ở xa hơn. Ngoài ra chúng ta có thể tự tạo 1 hàm tính trọng số của riêng mình

- Metrics : Cách tính khoảng cách

+ Minkowski:

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

+ Manhattan:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

+ Euclidian: Là trường hợp đặc biệt của Minkowski khi p=2

+ Chebyshev:

Table 1: Distance Metrics for kNN

Distance	Equation
Euclidean	$\mathbf{d} = \sqrt{\sum_{j=1}^{n}(x_{sj} - x_{tj})^2}$
City Block	$\mathbf{d} = \sum_{j=1}^{n} \lvert x_{sj} - x_{tj} \rvert$
Chebyshev	$\mathbf{d} = \max_j \{ \lvert x_{sj} - x_{tj} \rvert \}$
Cosine	$\mathbf{d} = 1 - \frac{\sum_{j=1}^{n} x_{sj}x_{tj}}{\sqrt{\sum_{j=1}^{n} x_{sj}x_{sj}}\sqrt{\sum_{j=1}^{n} x_{tj}x_{tj}}}$
Correlation	$\mathbf{d} = 1 - \frac{(x_s - \tilde{x}_s)(x_t - \tilde{x}_t)'}{\sqrt{(x_s - \tilde{x}_s)(x_s - \tilde{x}_s)'}\sqrt{(x_t - \tilde{x}_t)(x_t - \tilde{x}_t)'}}$

Có 1 lưu ý rất quan trọng đó là KNN không có hàm mất mát và hàm tối ưu, vậy nên mô hình hoạt động ra sao phụ thuộc tất cả vào các tham số ở trên.

3.3.1.2 KNN cho bài toán Classfication

- Đầu vào: Tập dữ liệu các trận đấu và thông số của các mùa giải trước, bao gồm các features được lựa chọn.

- Biểu diễn dữ liệu: Mỗi đối tượng quan sát được thể hiện bởi một vector n chiều, mỗi chiều đại diện cho một thuộc tính đối tượng. Có tập C là tập các nhãn đã được gán trước.

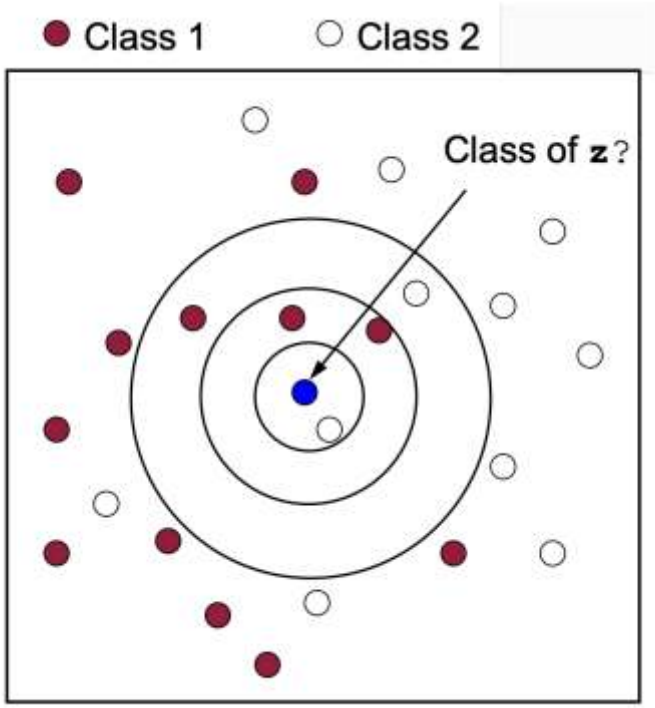
- Giai đoạn học: Lưu tập dữ liệu huấn luyện D và các nhãn của nó

- Dự đoán đầu vào mới z :

+ Mỗi dữ liệu x trong D, tính khoảng cách tương đồng của nó với z

+ Tìm tập NB(z) là tập các hàng xóm gần với z nhất

+ Sử dụng các nhãn của các dữ liệu trong NB(z) để dự đoán z



3.3.2 Vì sao chọn KNN cho bài toán

Thứ nhất, với một bài toán Classfication, KNN hoạt động rất đơn giản vì nó chỉ là việc tìm kiếm các điểm dữ liệu gần nhất trong tập dữ liệu huấn luyện và sử dụng chúng để phân loại điểm dữ liệu mới, ngoài ra không yêu cầu độ dốc tuyến tính và không cần tính toán đạo hàm.

Thứ hai, đây là một thuật toán có khả năng định vị dữ liệu rất tốt thông qua các vector và tọa độ trên một không gian n-chiều, chính vì vậy chúng ta có thể tìm ra được dữ liệu của những mẫu nào có sự tương quan với nhau nhiều và sự tương quan ít hơn. Vì vậy, KNN có thể là một lựa chọn hiệu quả trong các bài toán phân loại cơ bản với tập dữ liệu nhỏ và đơn giản.

3.3.3 Thực nghiệm

3.3.3.1 Đánh giá hiệu năng mô hình thông qua kỹ thuật Kfold cross-validation

K-fold cross-validation là một phương pháp để đánh giá hiệu quả của một mô hình học máy trên tập dữ liệu. Nó được thực hiện bằng cách chia tập dữ liệu thành K phần động và sau đó chạy K lần quá trình huấn luyện và đánh giá, mỗi lần sử dụng một phần khác nhau làm tập dữ liệu kiểm tra.

Trong trường hợp đơn giản nhất, đó là "Straight KFold", tập dữ liệu được chia thành K phần động có kích thước tương đương và sau đó mô hình được huấn luyện và đánh giá trên mỗi phần một lần, sử dụng các phần còn lại làm tập dữ liệu kiểm tra.

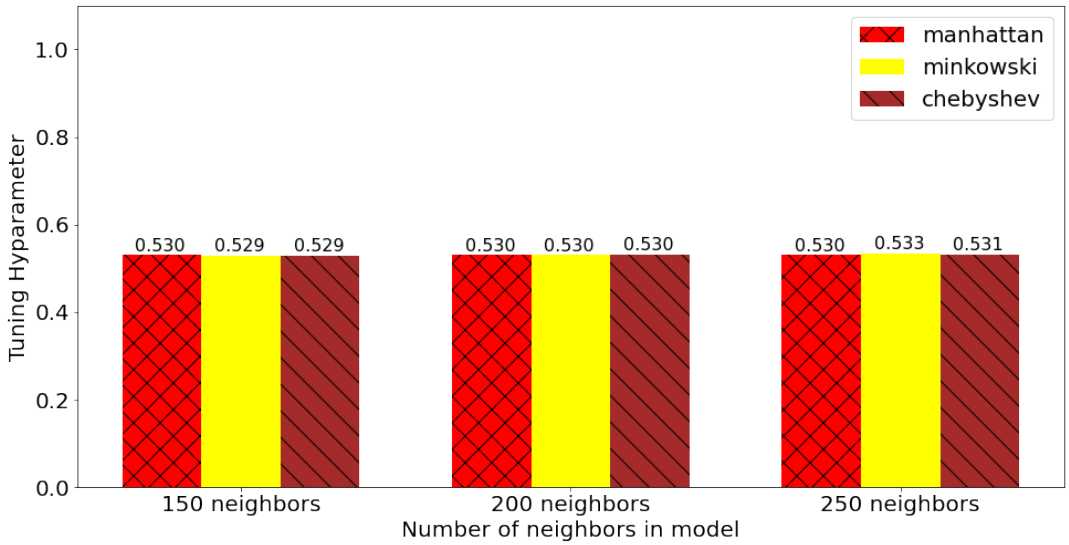
Ví dụ, nếu chúng ta có một tập dữ liệu gồm 10 mẫu và sử dụng K=5, thì K-fold cross-validation sẽ chia tập dữ liệu thành 5 phần động có kích thước 2 mẫu mỗi phần. Sau đó, mô hình sẽ được huấn luyện và đánh giá 5 lần, mỗi lần sử dụng một phần khác nhau làm tập dữ liệu kiểm tra và các phần còn lại làm tập dữ liệu huấn luyện

Để đánh giá hiệu năng mô hình mà chúng ta đang xét, ta sẽ thực hiện kỹ thuật này trên tập huấn luyện với K=17 (vì tập huấn luyện của chúng ta bao gồm 17 mùa giải)

3.3.3.2 Điều chỉnh các tham số

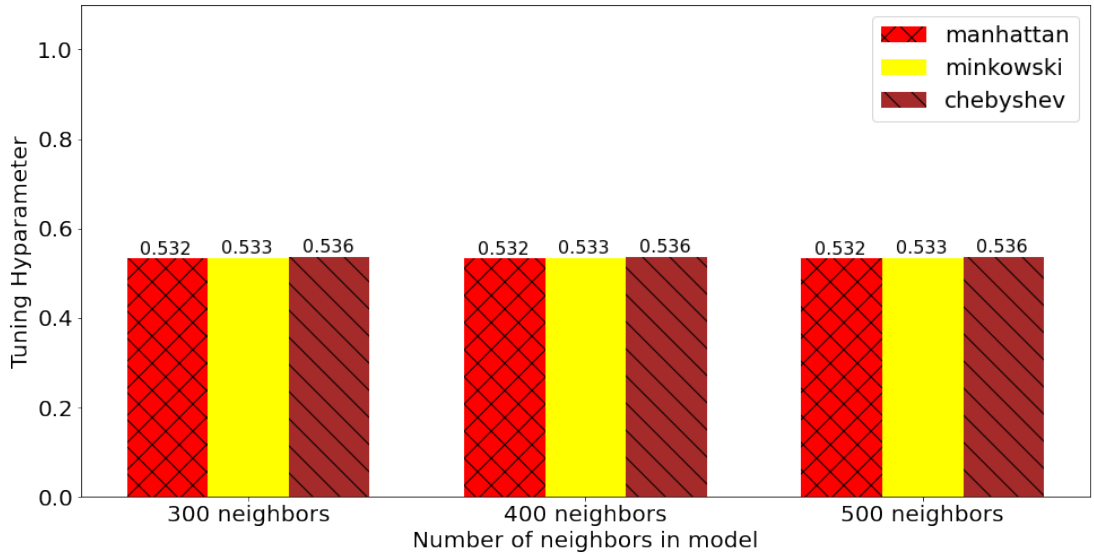
Có 2 tham số chúng ta cần phải điều, đó là chọn tham số k và công thức đo khoảng cách. Nếu như chọn một tham số k quá nhỏ, mô hình dễ bị rơi vào tình trạng overfitting. Ngược lại, nếu như cho tham số k quá lớn, thì mô hình dễ bị underfitting. Chính vì vậy, phải chọn ra 1 số k vừa đủ để có thể giúp mô hình của đạt goodfitting. Xét các công thức đo khoảng cách, mỗi công thức đo như Manhattan, Minkowski, Chebyshev hay Cosine Similarity đều gặp tiện lợi và bất lợi ở những mô hình khác nhau.

Chúng ta sẽ khảo sát 2 thông tin trên kỹ thuật Straight Kfold. Nếu như điểm của chúng càng cao, tức là mô hình đang hoạt động ổn định, ngược lại, mô hình sẽ tăng khả năng overfitting nếu điểm Kfold thấp.

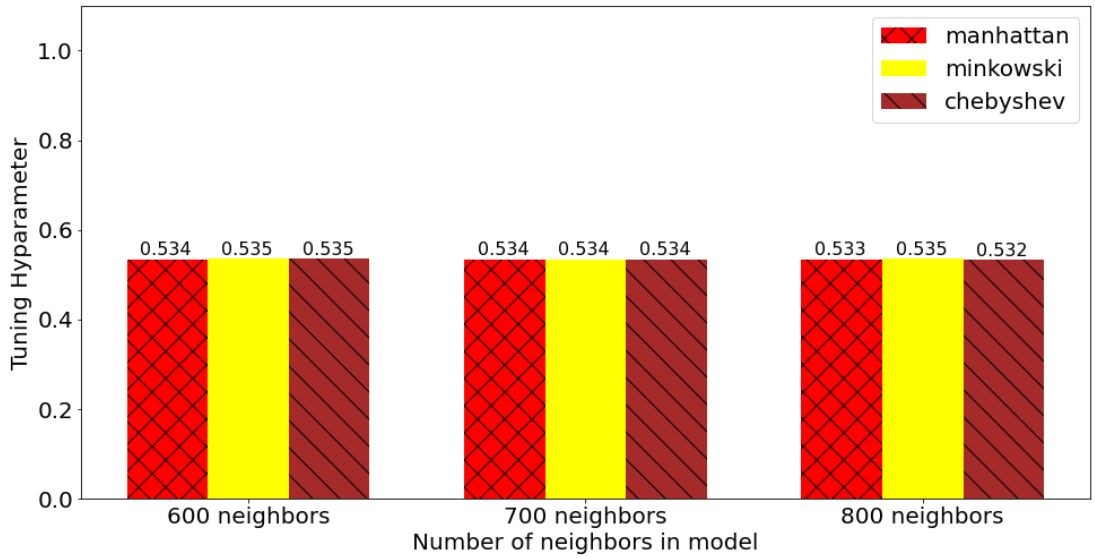


Ta thấy rằng, cả 3 cách đo khoảng cách với số các điểm hàng xóm khác nhau đều cho các kết quả rất gần nhau, điều này chứng tỏ các cách đo khoảng cách của chúng ta không ảnh hưởng quá nhiều tới tập dữ liệu. Ngoài ra, mỗi cách đo khoảng cách trên các số hàng xóm khác nhau là 150, 200, 250 đều cho những kết quả gần tương tự nhau và có chiều hướng tăng dần. Ta sẽ khảo sát tiếp trên các số hàng xóm lớn hơn:

Nhận thấy rằng, khi k tăng lên khoảng 300 đến 500 thì điểm cross-validation score của các mô hình lại càng tăng.

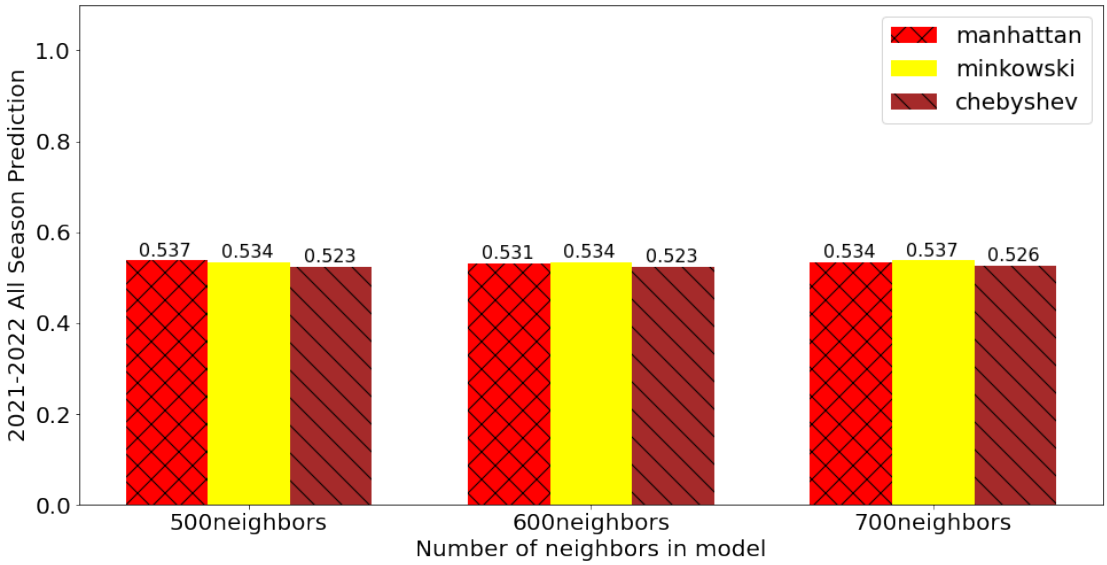


Tiếp tục kiểm tra mô hình với số k cao hơn:



Như ta đã thấy, với k từ 600 đến 700, số điểm vẫn có dấu hiệu tăng, tuy nhiên với k = 800 số điểm lại có dấu hiệu đứng yên hoặc giảm xuống. Như vậy, ta có thể tạm kết luận với k từ 500 đến 700, các mô hình sẽ hoạt động hiệu quả nhất.

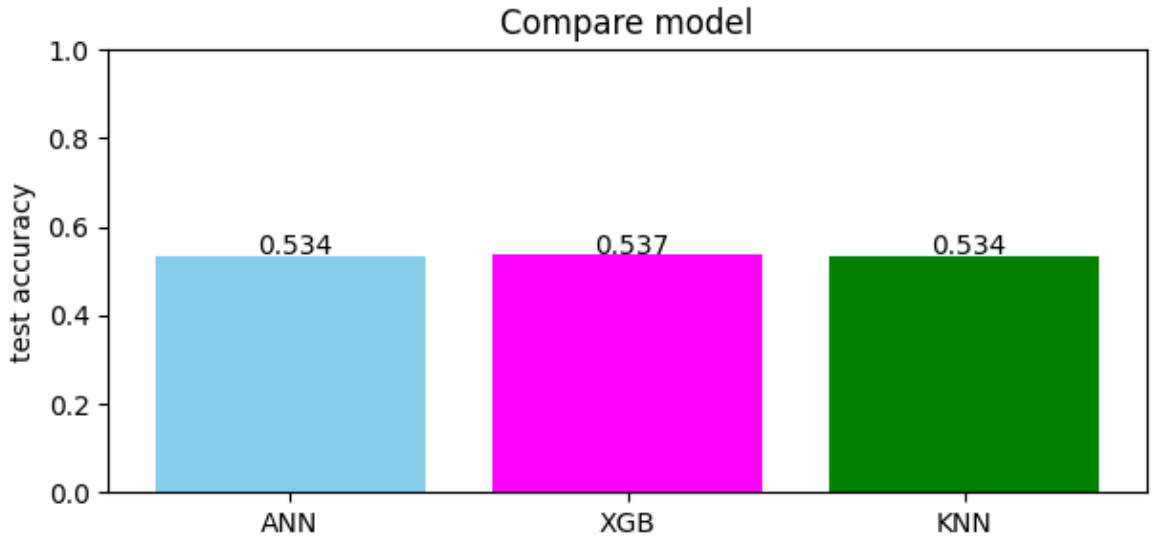
Ta sẽ tiếp tục thử 9 mô hình với k=500, k=600, k=700:



Ta có thể thấy rằng, ở mô hình đo khoảng cách Chebyshev trong giai đoạn xác thực, kết quả là lớn nhất, tuy nhiên khi kiểm tra trên tập xác thực, kết quả lại thấp nhất. Điều này chứng tỏ rằng công thức đo khoảng cách bằng Chebyshev rất nhạy cảm với môi trường. 2 công thức đo còn lại cho ta kết quả không quá chênh lệch, chứng tỏ rằng 2 mô hình đo khoảng cách bằng Manhattan và Minkowski có độ hội tụ tốt hơn, bởi lẽ 2 mô hình này sai lệch giữa tập kiểm chứng và tập kiểm tra là không đáng kể.

III. SO SÁNH KẾT QUẢ CÁC MÔ HÌNH HỌC MÁY TRÊN BỘ DỮ LIỆU KIỂM TRA

Sau khi lựa chọn được các tham số đem lại kết quả tốt nhất ở những bước thực nghiệm trước, các mô hình sẽ được đem dự đoán kết quả trận đấu trên bộ dữ liệu kiểm tra. Độ chính xác được biểu diễn trong hình:



Trước hết, ta phải khẳng định rằng, kết quả đo được của các mô hình đều không quá cao, thậm chí không vượt quá 55%. Có nhiều lí do cho kết quả này có thể kể đến như:

- Có những thuộc tính mà nhóm chưa (hoặc chưa đủ khả năng) có thể thêm vào, ví dụ như phong độ của các cầu thủ, đội hình xuất phát, chiến thuật sử dụng,… đấy đều là những thuộc tính thuần chuyên môn rất là quan trọng ảnh hưởng đến kết quả trận đấu. Trong tương lai, nhóm có thể mở rộng thêm những thuộc tính này.

- Ngoài ra còn những yếu tố trừu tượng và khách quan cũng có thể ảnh hưởng tới kết quả của một trận đấu như: Vận may, trọng tài, mặt cỏ sân, khán giả, tác động từ thị trường chuyển nhượng, thay đổi thượng tầng.

- Các thuộc tính đôi khi cũng có thể gây nhiễu ở một mức độ nhất định. Ví dụ:

+ HomeAvgPoint, AwayAvgPoint có thể định hình rất tốt các đội bóng lớn như Chelsea, MU, Liverpool, tuy nhiên có thể gây nhiễu với những trường hợp một đội bóng mạnh lên trông thấy như Man City, Newcastle.

+ HomePoint, AwayPoint, HomeGoal, AwayGoal, HomeConceded, AwayConceded là những thuộc tính rất mạnh để đo sức mạnh đội bóng trong một mùa giải, tuy nhiên chỉ ở mức tương đối, và chúng ta cũng chứng kiến không thiếu những đội bóng bay cao ở giữa mùa giải song lại sa sút vào cuối mùa, và ngược lại.

+ Các thuộc tính past (lấy phong độ các trận gần nhất) cũng chưa nói lên quá nhiều điều. Ví dụ, đội MU được 6 điểm, ghi 6 bàn và thủng lưới chỉ 1 bàn, tuy nhiên 2 trận đấy MU gặp các đội được cho là yếu thì đó lại là 1 câu chuyện khác so với MU gặp Chelsea và Liverpool trong 2 trận đó.

IV. KẾT LUẬN

Trong quá trình làm bài tập lớn, nhóm đã thực hiện được các công việc sau:

- Tìm hiểu về bài toán dự đoán và phân loại đa lớp.

- Tìm hiểu cách ứng dụng ba mô hình học máy giải quyết bài toán đặt ra.

- Tiến hành các thực nghiệm để đo độ hiệu quả của các cách mô hình đề xuất, và cách thay đổi các tham số của các mô hình

- Phân tích, bàn luận về các kết quả đạt được.

- Tìm hiểu vì sao kết quả thu được, độ chính xác lại không cao.

Các khó khăn gặp phải chủ yếu liên quan đến tiền xử lý dữ liệu (cụ thể là trong việc xây dựng các features), tìm hiểu cách áp dụng các mô hình trong framework scikit-learn, Tensorflow, cũng như là cách tinh chỉnh các tham số của từng thuật toán.

Trong tương lai, nhóm muốn thực nghiệm các mô hình đã sử dụng trên các bài toán phân loại đa lớp khác phức tạp hơn, ngoài ra nhóm sẽ cải tiến những thuộc tính mới cho bài toán “Dự đoán kết quả các trận đấu của Premier League”.

Để hoàn thành được báo cáo này, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến Tiến sĩ Nguyễn Nhật Quang vì những tiết dạy tuyệt vời và hữu ích của thầy ở môn "Nhập môn Học máy và Khai phá dữ liệu".

TÀI LIỆU THAM KHẢO

[1] https://viblo.asia/p/thuat-toan-toi-uu-adam-aWj53k8Q56m

[2] https://nttuan8.com/bai-3-neural-network/

[3] https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu gdsgdadam-Qbq5QQ9E5D8

[6] https://vietanh.dev/blog/2019-09-23-cac-ham-kich-hoat-activation-function-trong-neural-networks

[7] https://machinelearningmastery.com/xgboost-loss-functions/

[8] XGBoost - Bài 2: Toàn cảnh về Ensemble Learning - Phần 2 – VTITechBlog!

[9] XGBoost Parameters — xgboost 1.7.2 documentation

[10] https://www.football-data.co.uk/data.php

[11]https://github.com/Tahuubinh/ML_HUST_PROJECT?fbclid=IwAR0Kh5I0-Pi46DOakxSLVTR58XZAKRJfXCaLPwira5bSvqHXXo5cstoFOHU