ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÀI TẬP KỊCH BẢN THỰC NGHIỆM

GVHD: ThS. Nguyễn Thị Anh Thư

Nhóm thực hiện: Nhóm 8

Trương Hoàng Khiêm	23520730
Nguyễn Nghĩa Trung Kiên	23520801
Trần Anh Kiệt	23520820
Nguyễn Duy	24520384
Huỳnh Lê Minh Thành	22521346
Võ Phan Kiều My	24521095

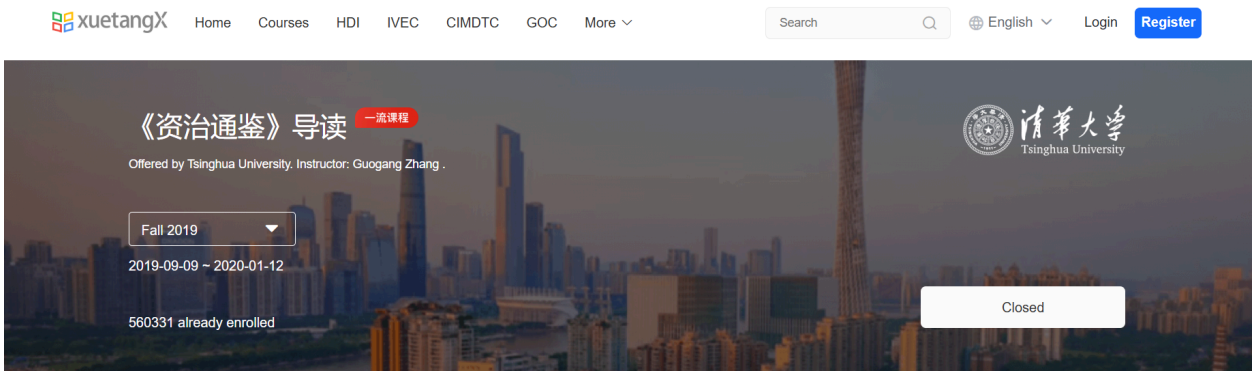
Thành phố Hồ Chí Minh, 11/2025

XÂY DỰNG BỘ DỮ LIỆU TIME SERIES

1. Tổng quan

Để phục vụ mục tiêu dự đoán sớm mức độ hài lòng của học viên, nhóm tiến hành xây dựng bộ dữ liệu dạng chuỗi thời gian (time series) phản ánh quá trình học tập và tương tác của từng học viên theo thời gian. Ý tưởng cốt lõi là chia tiến trình tham gia khóa học của mỗi học viên thành 4 giai đoạn (phase) tương ứng với các khoảng thời gian khác nhau giữa ngày đăng ký khóa học (enroll_time) và ngày kết thúc khóa học (end_date).

Thời điểm bắt đầu và kết thúc của khóa học được lấy từ trang web xuetangx.com, nhóm chỉ lấy ra những khóa học có thời gian bắt đầu và kết thúc cụ thể, không sử dụng các khóa có thời gian là "self-paced" để tránh việc các học viên đăng kí khóa học không có thời hạn kết thúc thì sẽ không chủ động việc học dẫn kết việc trích xuất đặc trưng trở nên khó khăn hơn.



Cụ thể, mỗi khóa học được chia thành 4 phase theo ngày kể từ lúc đăng ký dựa trên thời gian còn lại của người học từ khi đăng kí (remain_day) và chia thành 4 phần bằng nhau.

- **Phase 1:** 0-25% thời gian remain_day (từ enroll_date)

- **Phase 2:** 0-50% thời gian remain_day
- **Phase 3:** 0-75% thời gian remain_day
- **Phase 4:** 0-100% thời gian remain_day (đến end_date của khóa học)

Các giai đoạn có độ dài thời gian linh hoạt dựa trên khoảng thời gian đăng kí của từng học viên và thời gian kết thúc của khóa học, nhằm mô phỏng quá trình học tập theo từng giai đoạn của khóa học. Mỗi phase sẽ tổng hợp các đặc trưng hành vi học tập của học viên dựa trên dữ liệu làm bài tập, xem video và các tương tác bình luận của học viên trên khóa học đó.

Dữ liệu được lọc tiếp tục chỉ giữ ra các cặp khóa (user_id, course_id) có ít nhất một hoạt động trong thời gian học (làm bài tập, xem video hoặc bình luận). Dữ liệu sau cùng có 1350468 dòng (337617 cặp user_course) với 30 features.

	user_id	course_id	phase	exercises_touched	problems_done	total_attempts	correct_submissions	total_earned_score	avg_earned_score	avg_earned_ratio	...	fast_forward_
0	U_1000902	C_697821	1	0	0	0	0	0.0	NaN	0.0000	...	
1	U_1000902	C_697821	2	0	0	0	0	0.0	NaN	0.0000	...	
2	U_1000902	C_697821	3	0	0	0	0	0.0	NaN	0.0000	...	
3	U_1000902	C_697821	4	0	0	0	0	0.0	NaN	0.0000	...	
4	U_1000982	C_947149	1	9	42	42	7	5.6	0.133333	0.1875	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1350463	U_998604	C_934535	4	0	0	0	0	0.0	NaN	0.0000	...	
1350464	U_999821	C_881485	1	0	0	0	0	0.0	NaN	0.0000	...	
1350465	U_999821	C_881485	2	0	0	0	0	0.0	NaN	0.0000	...	
1350466	U_999821	C_881485	3	0	0	0	0	0.0	NaN	0.0000	...	
1350467	U_999821	C_881485	4	0	0	0	0	0.0	NaN	0.0000	...	

1350468 rows × 30 columns

Sau khi hoàn thành bước làm sạch và lọc để giữ lại 337,617 cặp khóa học-học viên có tương tác thực tế (tổng cộng 1,350,468$ dòng), dữ liệu hiện tại đang ở định dạng dài (Long Format), trong đó mỗi cặp (user_id, course_id) được biểu diễn bằng 4 dòng, tương ứng với 4 giai đoạn học tập (phase 1 đến phase 4).

Để chuẩn bị dữ liệu này cho các mô hình học máy truyền thống (như LGBM, XGBoost, SVR…), cần thực hiện bước xoay trục (pivot) dữ liệu thành định dạng rộng (Wide Format). Mục tiêu của việc xoay trục là biến đổi dữ liệu từ 1,350,468 dòng thành

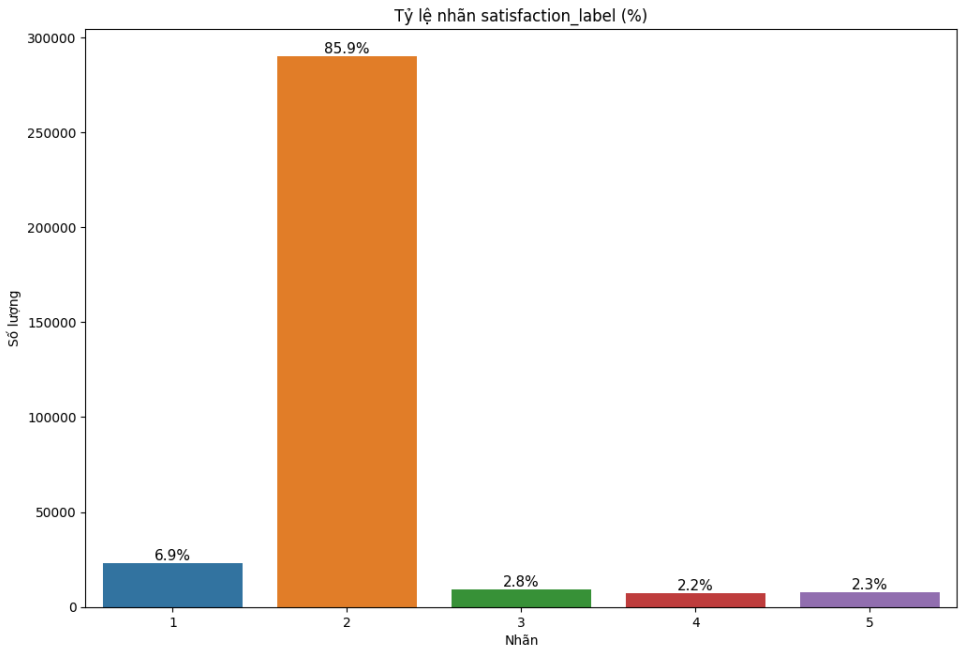
337,617 dòng duy nhất và 110 cột (features), nơi mỗi dòng là một mẫu độc lập đại diện cho một cặp khóa học của một học viên, sẵn sàng để huấn luyện.

	user_id	course_id	gender	total_courses_enrolled	total_students_enrolled	total_videos	total_exercises	num_fields	is_prerequisites	accuracy_p1	...	total_comments_p4	total_earned_score_p1	total_earned_score_p2	total_earned_
0	U_1000902	C_697821	1.0	20	44600	138	69	1	0	0.000000	...	1	0.0	0.0	
1	U_1000982	C_947149	0.0	1	14716	20	14	0	1	0.166667	...	0	5.6	5.6	
2	U_1002814	C_808526	0.0	2	18110	21	18	0	0	0.000000	...	1	0.0	0.0	
3	U_100294	C_682442	0.0	19	10909	113	75	1	0	0.000000	...	1	0.0	0.0	
4	U_10030628	C_936971	1.0	1	231674	38	36	0	0	0.000000	...	12	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
337612	U_99753	C_1428968	1.0	1	23478	49	5	0	1	0.820000	...	7	0.0	0.0	
337613	U_99772	C_1903985	2.0	7	19649	66	7	0	0	0.838384	...	3	122.0	129.0	
337614	U_9980550	C_936971	1.0	2	231674	38	36	0	0	0.000000	...	12	0.0	0.0	
337615	U_998604	C_934535	0.0	1	11390	35	34	0	1	0.000000	...	5	0.0	0.0	
337616	U_999821	C_881485	0.0	13	16165	30	15	0	1	0.000000	...	1	0.0	0.0	

337617 rows × 110 columns

Bảng thống kê số lượng nhãn:

Giai đoạn	Số dòng	Số feature (không tính các thông tin tĩnh)	Phân phối nhãn
Phase_{i}	337617	25	1: 6.9% (23123)
			2: 85.9% (290041)
			3: 2.8% (9411)
			4: 2.2% (7359)
			5: 2.3% (7674)



Tổng hợp các feature của bộ dữ liệu:

Nhóm	Tên cột	Ý nghĩa
Thông tin tĩnh user và tài nguyên khóa học	user_id	Mã học viên
	course_id	Mã khóa học
	gender	Giới tính học viên (0,1,2)
	total_courses_enrolled	Tổng số lượng khóa học mà học viên đã đăng kí
	is_preresiquites	Có môn học tiên quyết hay không (0/1)
	num_fields	Số lượng lĩnh vực liên quan
	course_total_exercises	Số lượng exercise khóa học cung cấp
	course_total_videos	Số lượng video khóa học cung cấp
	total_students_enrolled	Số lượng học viên đã tham gia khóa học
Thành phần cảm xúc (comment)	total_comments_p{i}	Tổng số lượng bình luận của học viên ở khóa học trong giai đoạn {i}
	sent_1_count_p{i}	Tổng số lượng bình luận mang tính tiêu cực trong giai đoạn {i}

	sent_2_count_p{i}	Tổng số lượng bình luận mang tính trung tính trong giai đoạn {i}
	sent_3_count_p{i}	Tổng số lượng bình luận mang tính tích cực trong giai đoạn {i}
Thành phần học tập (problems)	exerercises_touched_p{i}	Số lượng exercise mà học viên đã làm trong giai đoạn {i}
	problems_done_p{i}	Số lượng problem mà học viên đã làm trong giai đoạn {i}
	total_attempts_p{i}	Tổng số lần nộp bài trong giai đoạn {i}
	correct_submissions_p{i}	Số lần nộp bài đúng trong giai đoạn {i}
	total_earned_score_p{i}	Tổng điểm học viên nhận được trong giai đoạn {i}
	avg_earned_score_p{i}	Điểm trung bình đạt được trên mỗi bài tập trong giai đoạn {i}
	avg_earned_ratio_p{i}	Tỉ lệ điểm trung bình đạt được trong giai đoạn {i}
	avg_problem_score_p{i}	Điểm trung bình tối đa của các bài tập mà học viên tham gia trong giai đoạn {i}
	accuracy_p{i}	Tỷ lệ chính xác khi làm bài, được tính bằng correct_submissions / total_attempts trong giai đoạn {i}

	problems_per_day_p{i}	Số lượng bài tập trung bình mà học viên làm mỗi ngày trong giai đoạn {i}
	earned_per_attempt_p{i}	Điểm trung bình đạt được trên mỗi lần làm bài trong giai đoạn {i}
Thành phần xem video (video)	Avg_speed_in_course_p{i}	Tốc độ tua trung bình mà học viên đó xem trong giai đoạn {i}
	coverage_ratio_p{i}	Tỉ lệ phần trăm số video học viên đã học trong một khóa trong giai đoạn {i}
	course_watch_ratio_p{i}	Tỉ lệ phần trăm học viên xem được dựa trên thời gian xem thực tế trong giai đoạn {i}
	weighted_watch_ratio_p{i}	Tỉ lệ phần trăm học viên xem được dựa trên thời gian xem có điều chỉnh về tốc độ tua trong giai đoạn {i}
	interactions_per_video_p{i}	Số tương tác trung bình trên mỗi video đã xem trong giai đoạn {i}
	sessions_per_video_p{i}	Số phiên xem lại trên mỗi video đã xem (nếu sessions_per_video > 1 nghĩa là học viên đã xem đi xem lại video nhiều lần) trong giai đoạn {i}
	speed_changes_per_session_p{i}	Số lần thay đổi tốc độ tua trên mỗi phiên xem trong giai đoạn {i}

	fast_forward_per_watch_p{i}	Tỷ lệ tua nhanh trên thời gian xem trong giai đoạn {i} - ~0 → ít tua nhanh - Cao (~3.39) → học viên tua nhanh nhiều (gấp 3.39 lần thời gian xem) - Cực cao (~13.01) → Học viên chỉ lướt (chủ yếu tua video)
	rewind_per_watch_p{i}	Tỷ lệ tua lại trên thời gian xem: học viên xem lại các đoạn nội dung (thường là các khái niệm khó) trong giai đoạn {i}
	max_watch_point_ratio_p{i}	Tỷ lệ mốc xem xa nhất trong giai đoạn {i}: phân biệt người học lướt và học sâu Một học viên có thể có course_watch_ratio (tổng thời gian xem) thấp, nhưng max_watch_point_ratio lại cao bất thường. Điều này xảy ra khi học viên tua nhanh (fast-forward) đến cuối video mà không xem toàn bộ nội dung
Label	satisfaction_label	Nhãn hài lòng (1-5)

2. Chia dữ liệu train/test/dev

Trước tiên, tập dữ liệu tổng thể được chia ngẫu nhiên thành ba tập độc lập:

- **Tập train - 80%:** Dùng để huấn luyện mô hình.

- **Tập dev - 10%**: Dùng để tinh chỉnh siêu tham số (Hyperparameters) và lựa chọn mô hình tốt nhất.
- **Tập test - 10%**: Dùng để đánh giá hiệu suất cuối cùng của mô hình đã chọn và đặc biệt là cho việc dự báo sớm. Tập này đóng vai trò là dữ liệu "tương lai" chưa từng được mô hình nhìn thấy.

Để mô phỏng dự báo sớm, tập test sẽ được tái cấu trúc thành các phiên bản cắt giảm đặc trưng, tương ứng với các tỷ lệ thời lượng đã hoàn thành của khóa học:

Giai đoạn	Tỉ lệ thời lượng	Đặc trưng sử dụng	Mục đích đánh giá
P1	0-25%	Các đặc trưng trong phase 1 (hậu tố _p1)	Đánh giá khả năng dự đoán rất sớm dựa trên tương tác ban đầu
P2	0-50%	Các đặc trưng trong phase 1 và 2 (hậu tố _p1 và _p2)	Đánh giá khả năng dự đoán giữa kì
P3	0-75%	Các đặc trưng trong phase 1, 2, 3 (hậu tố _p1, _p2, _p3)	Đánh giá khả năng dự đoán gần cuối
P4	0-90%	Các đặc trưng trong phase 1, 2, 3, 4 (hậu tố _p1,_p2, _p3 và _p4)	Đánh giá hiệu suất khi gần như toàn bộ dữ liệu lịch sử đã có

CÁC THUẬT TOÁN DỰ KIẾN THỰC NGHIỆM

Sau khi hoàn thiện bộ dữ liệu chuỗi thời gian (time series) và gán nhãn mức độ hài lòng của học viên, bước tiếp theo trong nghiên cứu là lựa chọn và triển khai các thuật toán học máy phù hợp nhằm dự đoán sớm mức độ hài lòng. Việc lựa chọn mô hình được định hướng theo hai tiêu chí chính:

- Khả năng xử lý dữ liệu chuỗi thời gian theo tiến trình học tập
- Khả năng phát hiện sớm xu hướng thay đổi trong hành vi học viên trước khi khóa học kết thúc.

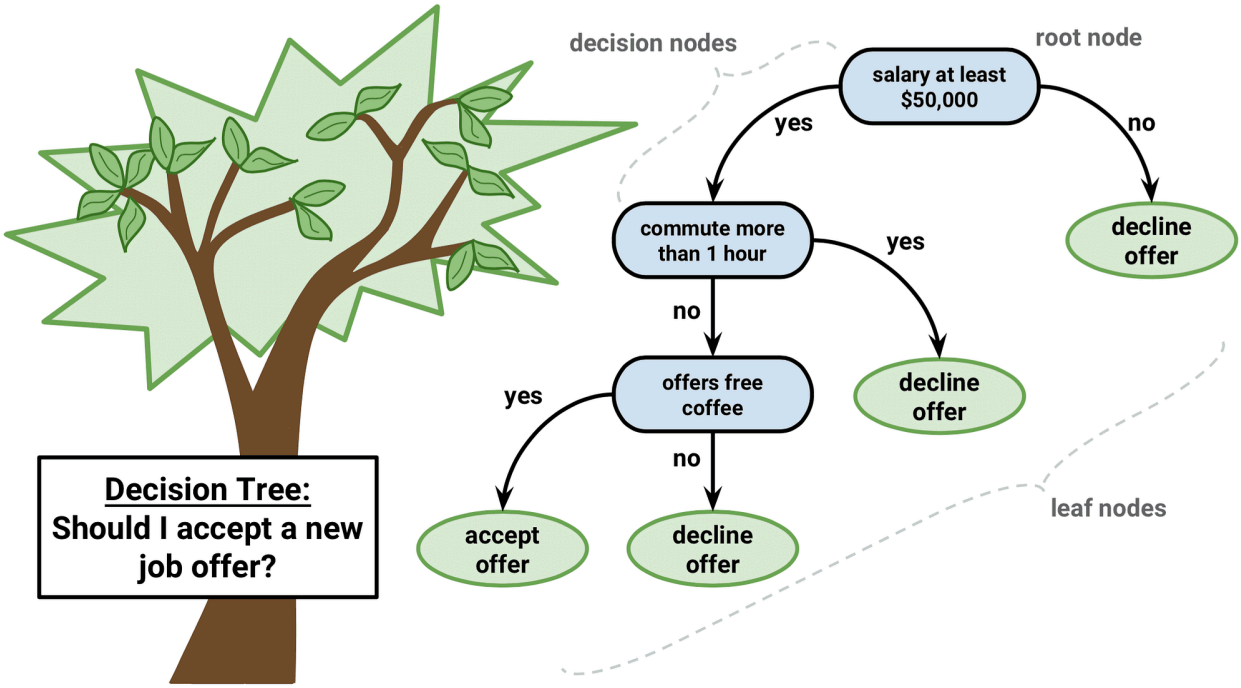
Cụ thể, nhóm dự kiến áp dụng kết hợp giữa các mô hình truyền thống và mô hình học sâu để đánh giá hiệu quả dự đoán.

- **Các mô hình học máy truyền thống:** như Decision tree, Random forest, SVM, Logistic regression, XGBoost,... sẽ được sử dụng để xây dựng baseline và đánh giá khả năng phân loại mức độ hài lòng dựa trên các đặc trưng tổng hợp theo từng phase.
- **Các mô hình học sâu:** như Long Short-Term Memory (LSTM) hoặc Temporal Convolutional Network (TCN) sẽ được áp dụng nhằm khai thác quan hệ phụ thuộc theo thời gian giữa các giai đoạn học tập.
- Ngoài ra, nghiên cứu cũng xem xét **mô hình lai (hybrid)** kết hợp giữa đặc trưng thống kê và đặc trưng học sâu (feature fusion), nhằm tăng độ chính xác và khả năng khái quát hóa của mô hình.

1. Về các mô hình học máy

1.1. Decision Tree

Decision Tree là một đồ thị có hướng được sử dụng cho việc ra quyết định. Mô hình cây quyết định là một mô hình thuật toán học có giám sát, có thể giải quyết cả bài toán hồi quy và phân loại. Mỗi nút trong cây đại diện một đặc trưng (feature), mỗi nhánh đại diện cho một giá trị của đặc trưng đó, và mỗi lá (leaf) đại diện cho một nhãn (label) hoặc giá trị đầu ra.



a. Cách hoạt động

- **Chọn thuộc tính gốc:** Thuật toán chọn thuộc tính "tốt nhất" để làm nút gốc của cây. Lựa chọn các tiêu chí như Information Gain, Gini Impurity hoặc Variance Reduction nhằm phân chia dữ liệu một cách hiệu quả nhất
- **Phân chia dữ liệu:** Dựa trên giá trị của thuộc tính gốc, dữ liệu được chia thành các tập con tương ứng với các nhánh.
- **Xây dựng cây đệ quy:** Quá trình chọn thuộc tính và phân chia dữ liệu được lặp lại đệ quy cho mỗi tập con, tạo thành các nút và nhánh tiếp theo của câu. Quá trình này tiếp

tục cho đến khi tất cả các nút lá đều chứa dữ liệu thuộc cùng một lớp hoặc đạt đến một điều kiện dừng nào đó (vd: độ sâu tối đa của cây)

- **Phân loại/Dự đoán:** Khi một điểm dữ liệu mới, thuật toán sẽ đi theo các nhánh của cây tương ứng với các giá trị thuộc tính của điểm dữ liệu đó cho đến khi đạt đến một nút là. Nhãn lớp (hoặc giá trị dự đoán) tại nút lá đó sẽ được sử dụng để phân loại (hoặc dự đoán) điểm dữ liệu mới
- **Đối với bài toán dự báo:** Chúng ta có thể áp dụng cây quyết định cho bài toán dự báo cũng tương tự như bài toán phân loại. Điểm khác biệt đó là chúng ta không sử dụng hàm tin thu mà thay vào đó là độ suy giảm của phương sai (reduction in variance)

b. Ưu điểm

- **Dễ hiểu và dễ giải thích:** dễ trực quan mối quan hệ giữa các đặc trưng học tập (như số bài làm, điểm trung bình, số lần truy cập, mức độ hoàn thành ở từng phase) và mức độ hài lòng của học viên, hữu ích cho việc phân tích hành vi học tập và diễn giải nguyên nhân dẫn đến sự không hài lòng.
- **Phù hợp với dữ liệu phi tuyến:** có thể nắm bắt được các mẫu phức tạp và phi tuyến trong hành vi học tập (vd: học viên ít xem video nhưng làm nhiều bài tập vẫn hài lòng)
- **Xử lý dữ liệu hỗn hợp tốt:** Mô hình có thể xử lý đồng thời các đặc trưng định lượng (số bài làm, điểm số) và định tính (giới tính, loại khóa học, giai đoạn học) mà không cần chuẩn hóa phức tạp.

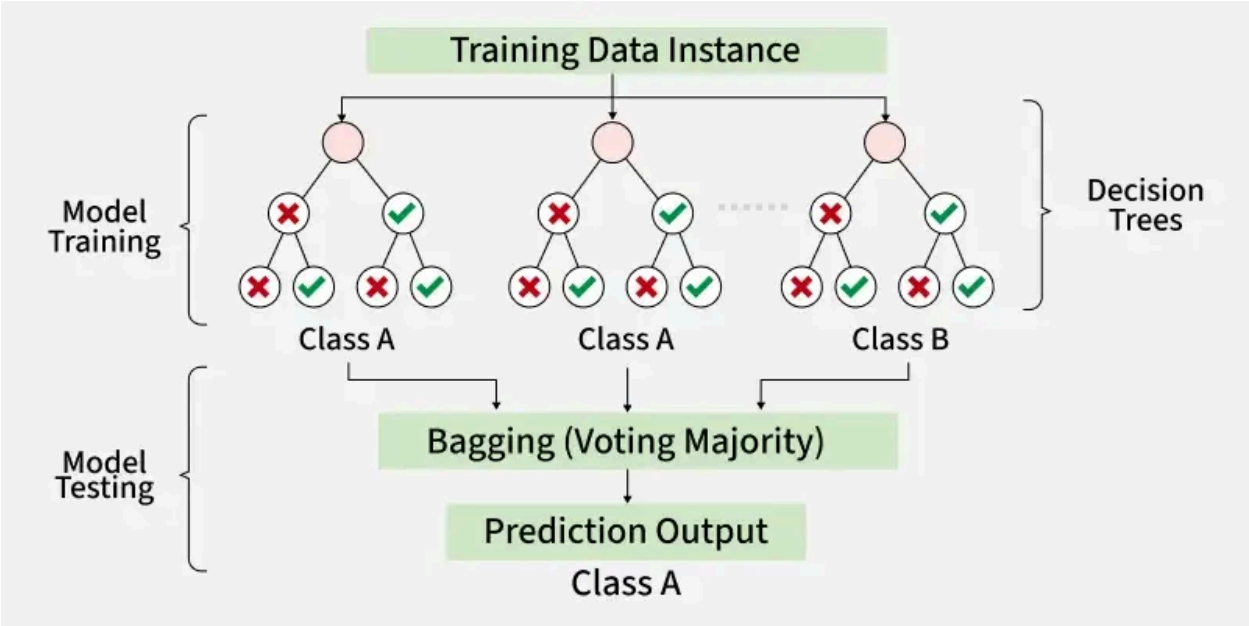
c. Nhược điểm

- **Dễ bị overfitting**: dữ liệu có nhiều đặc trưng hoặc nhiễu có nhiều hành vi học tập không đồng nhất), cây quyết định dễ học thuộc các mẫu cục bộ thay vì khái quát hóa.
- **Hiệu quả giảm khi dữ liệu lớn và phức tạp**: với quy mô dữ liệu lớn thì một cây đơn lẻ có thể trở nên rất sâu và tốn bộ nhớ, làm tăng thời gian huấn luyện và khó tối ưu.

- **Khó mô hình hóa quan hệ thời gian**: vì Decision Tree hoạt động theo từng mẫu độc lập, nó khó nắm bắt được mối quan hệ chuỗi thời gian giữa các phase của học viên (phase 1 → phase 2 → phase 3 → phase 4), nên chỉ phù hợp làm baseline, không tối ưu cho dữ liệu time series.

1.2. Random Forest

Thuật toán Random Forest là một thuật toán học máy có giám sát (supervíed learning) được sử dụng cho cả bài toán phân loại lẫn hồi quy. Dựa trên thuật toán Decision Tree, nó hoạt động bằng cách xây dựng các cây quyết định và kết hợp dự đoán của chúng để đưa ra dự đoán cuối cùng.



a. Cách hoạt động

- **Bagging**: RF tạo ra nhiều tập con dữ liệu huấn luyện bằng việc lấy mẫu ngẫu nhiên có hoàn lại từ tập gốc.
- **Random Subspace:** Đối với mỗi nút của mỗi cây quyết định, thuật toán chỉ xét một tập con ngẫu nhiên để tìm điểm chia tốt nhất
- **Tổng hợp dữ đoán:** RF kết hợp dự đoán của toàn bộ cây quyết định để ra kết quả cuối cùng

b. Ưu điểm

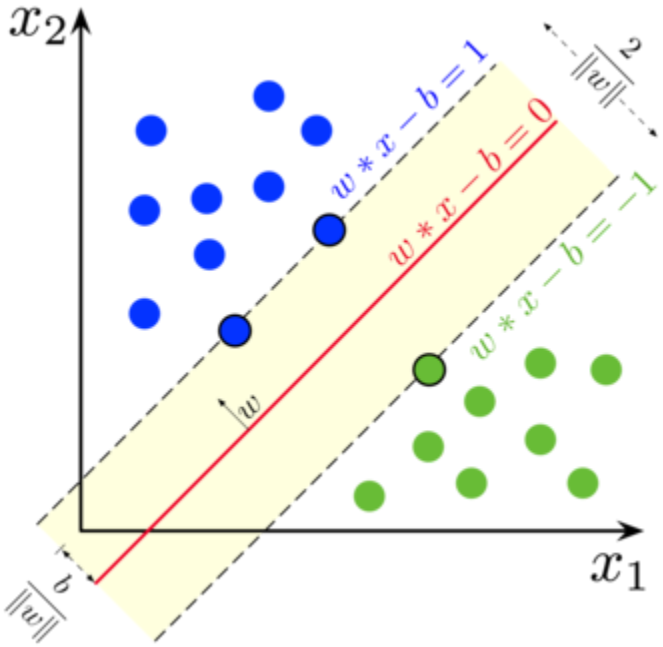
- **Giảm thiểu overfitting so với Decision Tree:** RF kết hợp nhiều cây giúp trung hòa sai số và giảm đáng kể overfitting khi dữ liệu lớn và đa dạng.
- **Xử lý tốt dữ liệu phi tuyến và đặc trưng hỗn hợp:** RF đồng thời có thể xử lý cả đặc trưng định lượng lẫn định tính.
- **Feature Importance:** Mô hình cung cấp chỉ số feature importance, giúp nhận biết yếu tố nào ảnh hưởng mạnh đến mức độ hài lòng (vd: số bài tập hoàn thành có thể quan trọng hơn tổng thời gian học).
- **Ổn định hơn và ít nhạy cảm với nhiễu dữ liệu:** Do kết quả là trung bình của nhiều cây con, RF ít bị ảnh hưởng bởi nhiễu hoặc dữ liệu ngoại lai, giúp tăng độ tin cậy trong bối cảnh dữ liệu hành vi học tập có thể không đồng đều giữa học viên.

c. Nhược điểm

- **Thời gian huấn luyện và dự đoán dài:** với dữ liệu lớn khiến mô hình tốn thời gian và bộ nhớ đáng kể, cần tinh chỉnh số lượng cây (n_estimators) và độ sâu (max_depth) trở nên quan trọng hơn.
- **Giảm khả năng diễn giải mô hình:** so với Decision Tree đơn lẻ, Random Forest là mô hình tập hợp nên khó trực quan hóa hoặc giải thích toàn bộ cấu trúc ra quyết định.
- **Không nắm bắt được quan hệ thời gian giữa các phase:** RF không mô hình hóa mối quan hệ tuần tự giữa các giai đoạn học (phase 1 → phase 2 → phase 3 → phase 4) mà chỉ thể hiện mức độ "snapshot" (tại từng giai đoạn)
- **Cần tinh chỉnh tham số để đạt hiệu năng tối ưu:** cần hyperparams để tối ưu mô hình.

1.3. SVM

Thuật toán Support Vector Machine (SVM) là một thuật toán học có giám sát được sử dụng trong bài toán phân loại lẫn hồi quy. Nó hoạt động bằng cách tìm một siêu mặt phẳng (hyperplane) tối ưu phân tách các điểm dữ liệu thuộc các lớp khác nhau với khoảng cách lớn nhất (margin) giữa siêu phẳng và các điểm dữ liệu gần nhất



a. Cách hoạt động

- **Ánh xạ dữ liệu lên không gian chiều cao hơn:** Nếu dữ liệu không thể phân tách tuyến tính trong không gian ban đầu, SVM sử dụng hàm kernel để ánh xạ dữ liệu lên một không gian chiều cao hơn, nơi dữ liệu có thể được phân tách tuyến tính
- **Tìm siêu phẳng tối ưu:** SVM tìm kiếm siêu phẳng có khoảng cách lớn nhất (margin) tới các điểm dữ liệu gần nhất của mỗi lớp. Các điểm dữ liệu nằm trên margin được gọi là support vectors
- **Phân loại dữ liệu mới:** khi có một điểm dữ liệu mới, SVM xác định vị trí của điểm dữ liệu đó so với siêu phẳng đã tìm được và phân loại nó vào lớp tương ứng

b. Ưu điểm

- **Hiệu quả cao với dữ liệu có biên phân tách rõ:** mô hình thường đạt độ chính xác cao khi dữ liệu có ranh giới phân lớp tương đối rõ ràng (vd: khi hành vi học viên

hài lòng và không hài lòng thể hiện qua các đặc trưng học tập khác biệt (số bài hoàn thành, điểm trung bình, tần suất truy cập,...)

- **Khả năng mô hình hóa quan hệ phi tuyến với kernel trick:** SVM sử dụng hàm kernel (RBF, polynomial, sigmoid,...) để ánh xạ dữ liệu sang không gian đặc trưng có chiều cao hơn, giúp mô hình nắm bắt các quan hệ phi tuyến phức tạp.
- **Linh hoạt với nhiều dạng dữ liệu đầu vào:** với các kernel khác nhau, SVM có thể thích ứng với nhiều loại đặc trưng từ định lượng đến định tính.

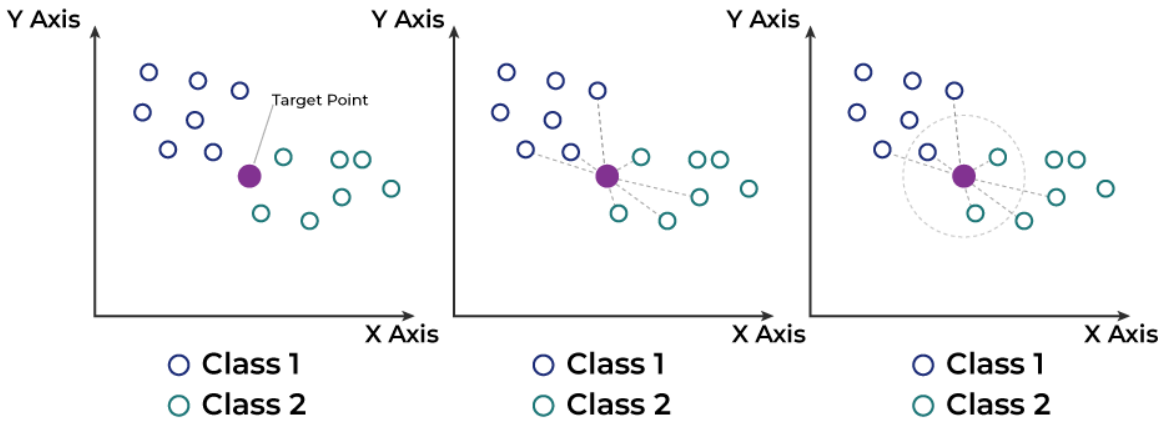
c. Nhược điểm

- **Không phù hợp cho dữ liệu quá lớn:** độ phức tạp huấn luyện cao ($O(n^2)$ đến $O(n^3)$)
- **Khó mở rộng và tốn bộ nhớ:** khi số mẫu tăng, số vector hỗ trợ (support vectors) tăng tương ứng, khiến mô hình chiếm nhiều bộ nhớ và dự đoán chậm
- **Khó chọn kernel và tham số tối ưu:** việc chọn kernel phù hợp (RBF, polynomial, linear, v.v.) cùng các siêu tham số như C và gamma cần nhiều thử nghiệm

1.4. KNN

K-Nearest Neighbors (KNN) là một thuật toán học có giám sát và phi tham số. Trong đó nó sử dụng độ gần (proximity) giữa các điểm dữ liệu để phân loại và hồi quy. Vì vậy nó

được sử dụng phổ biến cho các bài toán phân loại, hồi quy, phát hiện bất thường, gợi ý,..



a. Cách hoạt động

- Lưu trữ toàn bộ tập dữ liệu huấn luyện
- Khi có một học viên mới cần dữ đoán, thuật toán sẽ tính toán khoảng cách từ học viên này đến tất cả các học viên khác trong tập huấn luyện
- Chọn ra K học viên gần nhất và dự đoán mức độ hài lòng dựa trên lớp đa số (hoặc trung bình của K hàng xóm)
- **Các cách tính khoảng cách:**

Khoảng cách Euclidean (p = 2)	$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$
Khoảng cách Manhattan (p = 1)	$d(x,y) = \sum_{i=1}^{m} \lvert x_i - y_i \rvert$
Khoảng cách Minkowski	$d(x,y) = \left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{\frac{1}{p}}$
Khoảng cách Hamming	$D_H = \sum_{i=1}^{k} \lvert x_i - y_i \rvert$ $x = y \Rightarrow D = 0$ $x \neq y \Rightarrow D = 1$

b. Ưu điểm

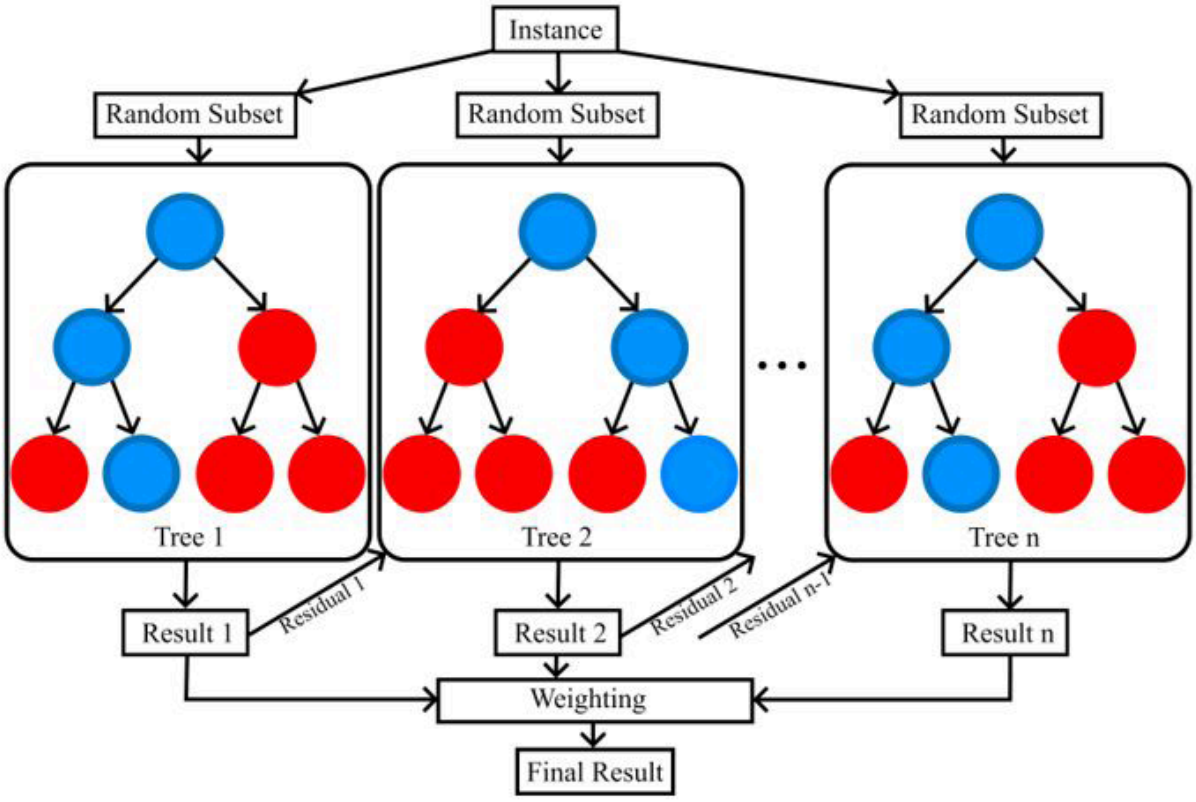
- **Dễ hiểu và dễ triển khai:** KNN hoạt động dựa trên nguyên tắc: "học viên có hành vi giống nhau sẽ có mức độ hài lòng tương tự" và gán nhãn theo số đông.
- **Không yêu cầu giả định về phân phối dữ liệu:** KNN là mô hình phi tham số nên không cần giả định dữ liệu tuân theo phân phối nào.
- **Khả năng thích ứng linh hoạt với các đặc trưng khác nhau:** KNN có thể áp dụng nhiều hàm đo khoảng cách khác nhau: Euclidean, Manhattan, Cosine,.. giúp linh hoạt điều chỉnh cho phù hợp với loại dữ liệu.
- **Cập nhật dễ dàng:** khi có dữ liệu mới, chỉ cần thêm vào bộ dữ liệu mà không cần huấn luyện lại toàn bộ mô hình.

c. Nhược điểm

- **Không phù hợp với dữ liệu lớn:** Với dữ liệu lớn, việc tính toán trở nên rất chậm và tốn bộ nhớ.
- **Hiệu năng dự đoán kém khi dữ liệu nhiễu hoặc không chuẩn hóa:** KNN rất nhạy cảm với thang đo và nhiễu dữ liệu. Nếu các đặc trưng (có đơn vị khác nhau mà chưa được chuẩn hóa, khoảng cách Euclid sẽ bị sai lệch → mô hình cho kết quả không chính xác.
- **Không thể hiện được quan hệ thời gian:** KNN chỉ dựa trên khoảng cách trong không gian đặc trưng tĩnh, mô hình không học được chuỗi hành vi theo thời gian (phase 1 → 4)
- **Khó lựa chọn số lượng láng giềng k:** Nếu chọn k quá nhỏ, mô hình dễ **o**verfitting; nếu k quá lớn, mô hình dễ underfitting.

1.5. XGBoost

XGBoost (Extreme Gradient Boosting) là một phiên bản tối ưu hóa của Gradient Boosting, được thiết kế để tập trung vào tốc độ tính toán và độ chính xác của mô hình. Nó là một mô hình ensemble hiệu suất cao, xây dựng mô hình dự đoán bằng cách kết hợp tuần tự nhiều cây quyết định (decision trees).



a. Cách hoạt động

Nền tảng của XGBoost là Gradient Boosting, trong đó các cây quyết định được xây dựng tuần tự để sửa chữa phần lỗi (phần dư) của tổng mô hình trước đó.

- **Tối ưu hóa gradient (Đạo hàm bậc 1 và 2):** XGBoost sử dụng cả đạo hàm bậc một (gradient) và đạo hàm bậc hai (hessian) của hàm mất mát giúp thuật toán tìm ra cách sửa lỗi tối ưu nhanh và chính xác hơn.

- **Điều chỉnh tích hợp:** Cơ chế chống overfitting được tích hợp trực tiếp vào hàm mục tiêu:
    - **Phạt khi thêm nút (gamma / min_split_loss):** Kiểm soát độ phức tạp của cây bằng cách chỉ thực hiện chia nút nếu mức giảm mất mát kỳ vọng vượt quá ngưỡng gamma.
    - **Phạt trọng số lá (L1: reg_alpha / L2: reg_lambda):** Điều chỉnh trọng số của các lá bằng L1 và L2, giúp làm mượt mô hình và ngăn các trọng số trở nên quá lớn.
- **Tối ưu hóa Hệ thống:** Hỗ trợ tính toán song song, sử dụng thuật toán gần đúng dựa trên Histogram để tối ưu bộ nhớ và tốc độ tìm điểm chia

b. Ưu điểm

- **Hiệu năng và độ chính xác cao:** XGBoost thường cho kết quả dự đoán chính xác hơn nhiều mô hình khác nhờ kỹ thuật boosting và tối ưu hàm mất mát.
- **Khả năng tổng quát hóa tốt, giảm overfitting:** XGBoost sử dụng regularization (L1 và L2) giúp kiểm soát độ phức tạp của cây, từ đó giảm hiện tượng overfitting.
- **Hỗ trợ tính toán song song và phân tán:** Mô hình được thiết kế tối ưu cho hiệu năng, có thể chạy đa lõi CPU/GPU, giúp rút ngắn đáng kể thời gian huấn luyện..
- **Giải thích được kết quả dự đoán:** XGBoost cung cấp chỉ số feature importance và có thể kết hợp với công cụ SHAP (SHapley Additive exPlanations) để diễn giải rõ ràng đặc trưng nào ảnh hưởng mạnh đến dự đoán mức độ hài lòng

c. Nhược điểm

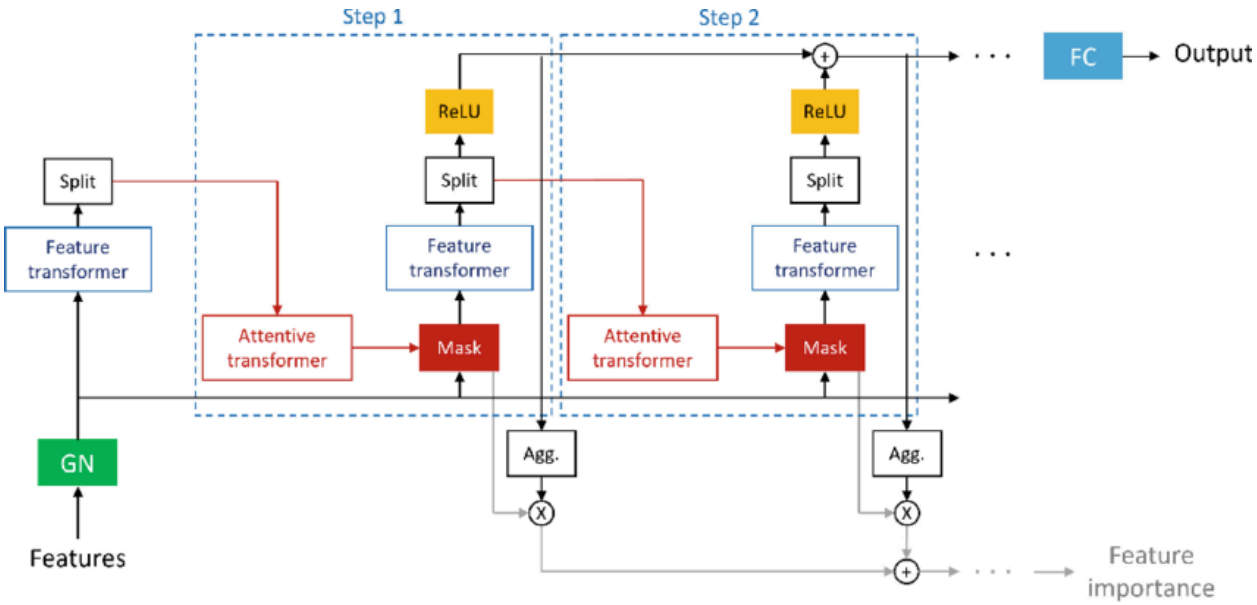
- **Khó tinh chỉnh tham số:** XGBoost có nhiều tham số quan trọng (learning_rate, n_estimators, max_depth, subsample, colsample_bytree, lambda, gamma,…). Việc tìm bộ tham số tối ưu cần thử nghiệm nhiều lần hoặc sử dụng GridSearchCV / Optuna / Bayesian Optimization, gây tốn thời gian.

- **Đòi hỏi tài nguyên tính toán cao:** Dù hiệu năng tốt, việc huấn luyện trên dữ liệu rất lớn vẫn ngốn nhiều RAM và GPU.
- **Khó nắm bắt chuỗi thời gian (temporal dependency):** Mặc dù có thể dùng đặc trưng phase (week 1–4), XGBoost không mô hình hóa trực tiếp quan hệ tuần tự giữa các phase, do đó không phản ánh được quá trình học tiến triển theo thời gian.
- **Có thể thiên lệch khi đặc trưng mạnh áp đảo:** Nếu một vài đặc trưng (như số bài tập hoàn thành hoặc điểm trung bình) có giá trị nổi trội, mô hình dễ tập trung quá mức vào các đặc trưng này, làm giảm khả năng nhận diện tín hiệu nhỏ nhưng quan trọng (như mức tương tác video, bình luận,…).

2. Về các mô hình học sâu

2.1. Tabnet Classifier

TabNet (Tabular Neural Network) là một mô hình học sâu được thiết kế đặc biệt cho dữ liệu dạng bảng (tabular data). TabNet sử dụng cơ chế "sequential attention" để học cách chọn lọc các đặc trưng quan trọng tại từng bước ra quyết định, tương tự như cách con người tập trung vào các yếu tố nổi bật khi phân tích dữ liệu.



a. Cách hoạt động

TabNet gồm ba thành phần chính:

- **Feature Transformer:** biến đổi các đặc trưng đầu vào thành không gian biểu diễn sâu, học các mối quan hệ phi tuyến.
- **Attentive Transformer:** sử dụng cơ chế chú ý (attention) để chọn ra những đặc trưng quan trọng nhất ở mỗi bước ra quyết định.
- **Decision Step:** kết hợp các đặc trưng được chọn và tích lũy thông tin từ nhiều bước tuần tự (decision steps), giúp mô hình vừa học sâu vừa giữ được khả năng diễn giải.

Điểm đặc biệt của TabNet là khả năng tự động "học trọng số chú ý" cho từng đặc trưng, giúp mô hình vừa mạnh mẽ như deep learning, vừa diễn giải được như cây quyết định.

b. Ưu điểm

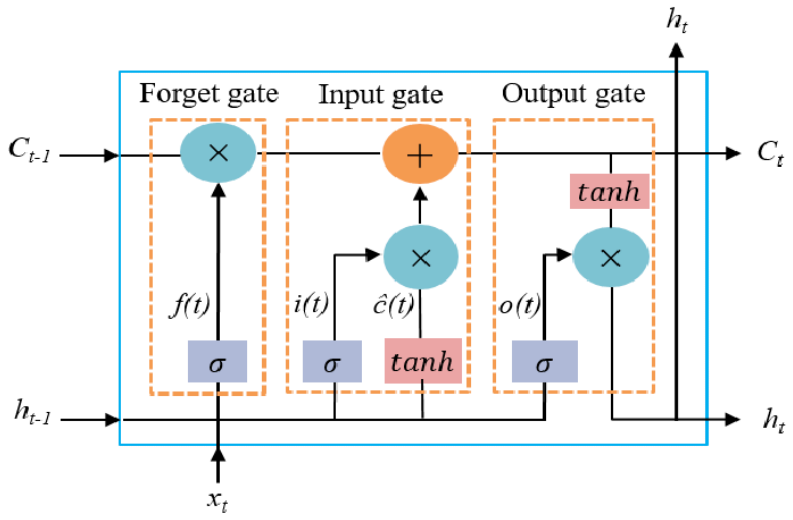
- **Xử lý trực tiếp dữ liệu bảng mà không cần nhiều tiền xử lý:** TabNet hoạt động trực tiếp trên dữ liệu dạng bảng không cần chuyển đổi sang dạng embedding hoặc chuẩn hóa phức tạp, giúp giảm công sức tiền xử lý.
- **Khả năng tự động chọn đặc trưng thông minh:** Cơ chế Attentive transformer cho phép TabNet chọn lọc những cột quan trọng nhất cho từng bước ra quyết định, giảm nhiễu và cải thiện khả năng khái quát.
- **Diễn giải mô hình rõ ràng:** TabNet cung cấp feature masks thể hiện mức độ đóng góp của từng đặc trưng trong từng quyết định.
- **Tối ưu hóa bằng mini-batch và GPU:** TabNet sử dụng mini-batch gradient descent nên dễ mở rộng cho dữ liệu rất lớn.

c. Nhược điểm

- **Huấn luyện phức tạp và nhạy cảm với siêu tham số:** Mô hình có nhiều siêu tham số (số bước quyết định, kích thước embedding, hệ số gamma, hệ số sparsity, learning rate,…) nên cần chú ý trong việc lựa chọn params.
- **Tốn tài nguyên tính toán:** TabNet yêu cầu GPU và bộ nhớ lớn, đặc biệt khi dữ liệu có nhiều đặc trưng hoặc cần nhiều bước attention.
- **Không tối ưu với dữ liệu nhiễu hoặc quá mất cân bằng:** Mặc dù có regularization, TabNet vẫn có thể bị ảnh hưởng khi dữ liệu chứa nhiều nhiễu hoặc mất cân bằng nghiêm trọng giữa các mức độ hài lòng
- **Khó triển khai với dữ liệu phân tán hoặc dạng streaming:** Vì cơ chế attention hoạt động tuần tự, TabNet chưa tối ưu cho các luồng dữ liệu đến liên tục (vd: cập nhật hành vi học viên theo thời gian thực).

2.2. LSTM

LSTM là một biến thể nâng cao của Recurrent Neural Network (RNN), được thiết kế để ghi nhớ thông tin trong chuỗi dữ liệu dài và khắc phục nhược điểm mất nhớ dần (vanishing gradient) của RNN truyền thống. Trong bài toán dự đoán mức độ hài lòng, LSTM có thể mô hình hóa chuỗi hành vi học theo thời gian (phase 1 → 4), từ đó học được xu hướng thay đổi và dự đoán sớm sự không hài lòng của học viên.



a. Cách hoạt động

LSTM hoạt động dựa trên ba cổng điều khiển trong mỗi "ô nhớ" (memory cell):

- **Forget Gate:** quyết định phần thông tin nào cần quên.
- **Input Gate:** xác định thông tin mới nào sẽ được ghi nhớ.
- **Output Gate:** xác định giá trị đầu ra tại bước hiện tại.

Nhờ vậy, mô hình có khả năng học mối quan hệ dài hạn giữa các phase hoặc tuần học.

b. Ưu điểm

- Nắm bắt tốt mối quan hệ theo thời gian và xu hướng học tập (vd: học viên giảm điểm, giảm lượt comments tốt, không xem video,... qua từng phase → dự đoán không hài lòng).
- Có khả năng học dữ liệu phi tuyến, phức tạp, không cần đặc trưng thủ công.
- Phù hợp với dữ liệu hành vi có chuỗi cố định (phase 1–4).
- Có thể kết hợp với dữ liệu tĩnh (đặc điểm cá nhân, khóa học) qua các mạng phụ.

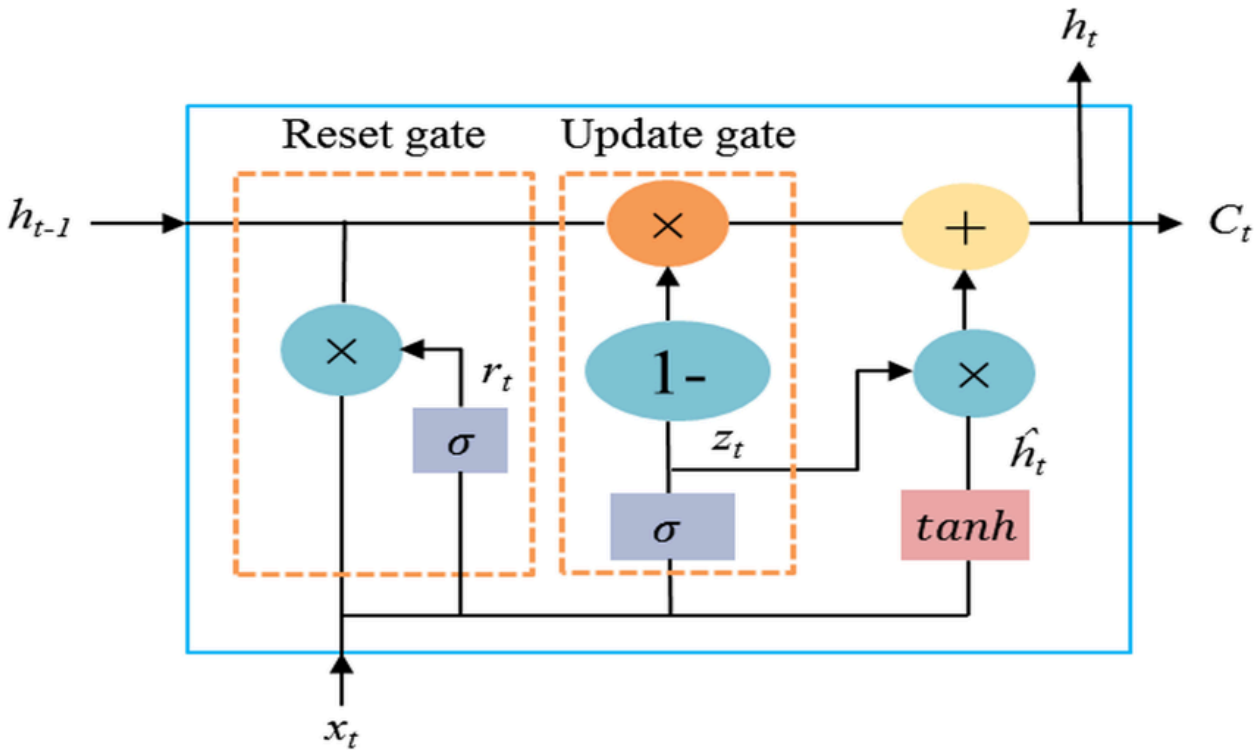
c. Nhược điểm

- Huấn luyện chậm, đặc biệt với dữ liệu dài hoặc nhiều chuỗi.
- Khó song song hóa → tốn thời gian huấn luyện hơn CNN hoặc Transformer.
- Dễ overfitting nếu không có regularization (dropout, early stopping).
- Khó diễn giải mô hình (không rõ yếu tố nào ảnh hưởng mạnh nhất).

2.3. GRU

GRU là biến thể gọn hơn của LSTM, đơn giản hóa cấu trúc mà vẫn giữ khả năng học chuỗi dài hạn. GRU gộp Forget Gate và Input Gate thành Update Gate, giảm số lượng tham số, giúp mô hình nhẹ hơn và hội tụ nhanh hơn LSTM. GRU hoạt động tốt khi chuỗi

thời gian ngắn và có ít bước, đồng thời vẫn học được mối quan hệ giữa các giai đoạn học tập của học viên.



a. Ưu điểm:

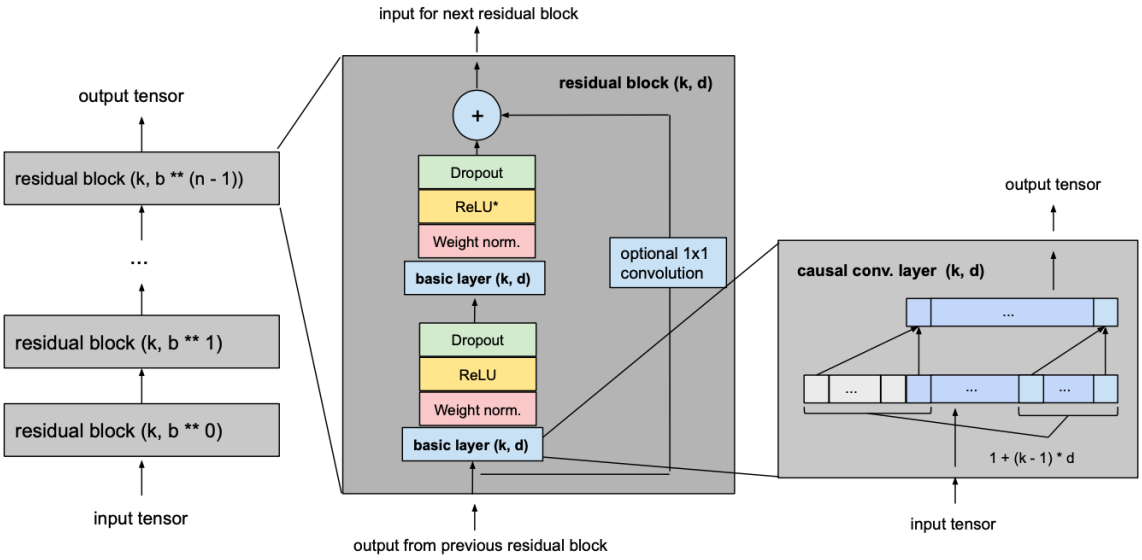
- Cấu trúc đơn giản, huấn luyện nhanh hơn LSTM.
- Giữ được khả năng học phụ thuộc dài hạn trong chuỗi.
- Phù hợp cho dữ liệu có độ dài ngắn.
- Cần ít tài nguyên hơn, dễ mở rộng cho dữ liệu lớn.

b. Nhược điểm

- Có thể mất thông tin chi tiết hơn LSTM trong chuỗi dài.
- Khó diễn giải như các mô hình RNN nói chung.
- Vẫn khó song song hóa, nên không lý tưởng cho dữ liệu cực lớn.
- Hiệu quả phụ thuộc vào việc chuẩn hóa và sắp xếp dữ liệu chuỗi.

2.4. TCN

TCN là mô hình học sâu dựa trên CNN, nhưng được thiết kế đặc biệt cho chuỗi thời gian. Thay vì lan truyền trạng thái tuần tự như RNN/LSTM, TCN dùng các lớp tích chập 1D với dilated convolution để mở rộng phạm vi quan sát trong chuỗi mà không làm tăng chiều dài tính toán. Nhờ vậy, TCN có thể nắm bắt quan hệ dài hạn trong chuỗi mà vẫn song song hóa và huấn luyện nhanh hơn nhiều so với RNN/LSTM.



a. Ưu điểm

- Huấn luyện nhanh, dễ song song hóa trên GPU.
- Giữ được mối quan hệ dài hạn giữa các phase học (nhờ dilated convolutions).
- Ít tham số hơn LSTM/GRU, ít overfitting hơn.
- Hoạt động tốt với chuỗi cố định độ dài (ví dụ 4 phase, 8 tuần,...).

b. Nhược điểm

- Cần chọn đúng độ trễ (dilation rate) để nắm bắt đúng khoảng thời gian.
- Khó giải thích vì là mô hình CNN-based (black-box).
- Không linh hoạt nếu chuỗi có độ dài thay đổi (cần padding hoặc cắt).
- Hiệu suất giảm nếu dữ liệu nhiễu hoặc không tuần tự rõ ràng.

CÁC ĐỘ ĐO ĐÁNH GIÁ HIỆU SUẤT MÔ HÌNH AI

Để đánh giá hiệu quả của các mô hình dự đoán mức độ hài lòng của học viên, đề tài sử dụng nhiều độ đo khác nhau phù hợp với tính chất bài toán phân loại đa lớp (thang đo 5 mức độ). Các độ đo này được tính dựa trên ma trận nhầm lẫn (confusion matrix), thể hiện sự khác biệt giữa nhãn thực tế và nhãn mô hình dự đoán.

1. **Accuracy**

$$\textbf{Accuracy}=(TN+TP)/(TP+TN+FP+FN)$$

Trong đó: **TP**: True Positives

TN: True Negatives

FP: False Positives

FN: False Negatives

- **Ý nghĩa:** accuracy phản ánh tỷ lệ số mẫu được mô hình dự đoán đúng trên tổng số mẫu quan sát. Đây là thước đo cơ bản nhất để đánh giá hiệu quả mô hình. Accuracy đơn giản, dễ diễn giải nhưng có lúc phản ánh không chính xác khi dữ liệu bị mất cân bằng lớp.

2. **Precision**

$$\textbf{Precision} = TP/(TP+FP)$$

- **Ý nghĩa:** Precision cho biết trong số các mẫu được mô hình dự đoán là “hài lòng”, có bao nhiêu mẫu thực sự là hài lòng. Nếu Precision cao mà Recall thấp, mô hình có thể bỏ sót nhiều trường hợp thật sự hài lòng.

3. **Recall**

$$\textbf{Recall} = TP/(TP+FN)$$

- **Ý nghĩa:** Recall phản ánh khả năng phát hiện đúng các mẫu thuộc lớp "hài lòng" trong tập dữ liệu thực tế. Recall phù hợp cho mục tiêu phát hiện sớm học viên không hài lòng, tránh bỏ sót nhưng cũng có thể dự đoán dư thừa, làm giảm độ chính xác.

4. **F1-score**

$$F1 = 2*((Precision*Recall)/(Precision+Recall))$$

- **Ý nghĩa:** Là trung bình điều hòa giữa Precision và Recall, giúp cân bằng giữa độ chính xác và khả năng bao phủ, phù hợp khi dữ liệu mất cân bằng nhưng không thể hiện sự phân bố của từng lớp cụ thể.

=> F1-Score được chọn là độ đo chính để so sánh giữa các mô hình

5. **AUC-ROC**

AUC-ROC đánh giá khả năng mô hình xếp hạng một mẫu tích cực (positive sample) cao hơn một mẫu tiêu cực (negative sample) được chọn ngẫu nhiên.
Giá trị: Nằm trong khoảng [0, 1].

- 1: Mô hình phân loại hoàn hảo.
- 0.5: Mô hình hoạt động không tốt hơn dự đoán ngẫu nhiên.

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(t))\, dt$$

Trong đó: **True Positive Rate (TPR) / Recall:** Tỷ lệ dự đoán đúng của lớp positive.

False Positive Rate (FPR): Tỷ lệ dự đoán sai lớp negative thành positive.

- AUC-ROC Không bị ảnh hưởng nhiều bởi sự mất cân bằng nhãn (Imbalance) vì nó chỉ quan tâm đến thứ hạng tương đối, không phải số lượng tuyệt đối của các nhãn.