Module 5 - Tuần 3 - XAI (SHAP)

Time-Series Team

Ngày 20 tháng 10 năm 2025

1 Tại sao cần XAI (Explainable AI)

1.1 Vấn đề mở đầu: Khi mô hình đúng mà vẫn sai

Trong học máy (Machine learning), ta thường quan tâm đến các chỉ số như **accuracy**, **F1-score**, **RMSE**, **AUC**,... và xem mô hình nào đạt điểm cao nhất thì "tốt nhất".

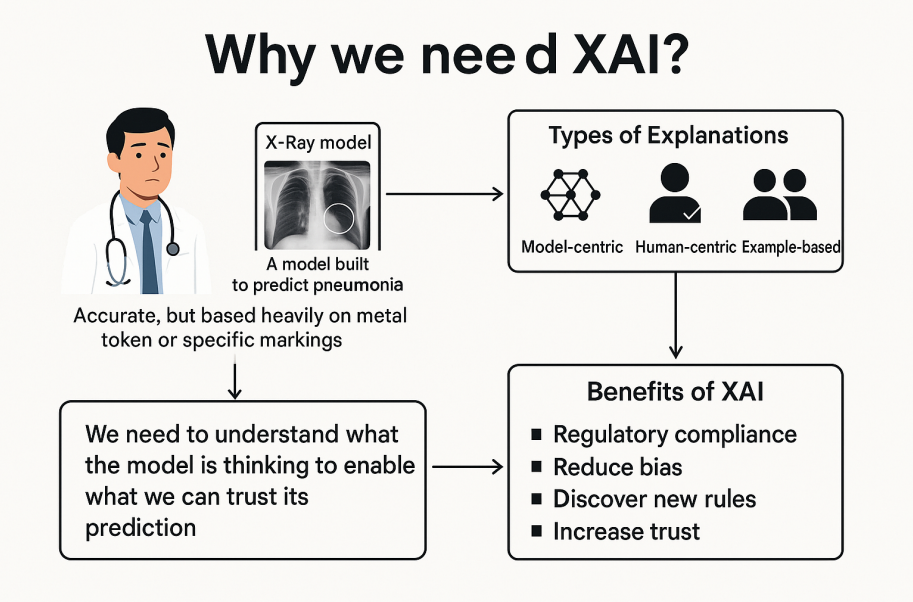
Nhưng nếu chỉ dừng ở đó, ta mới chỉ thấy *đầu ra*, chứ chưa hiểu *cách suy nghĩ bên trong* của mô hình.

Ví dụ thực tế: Mô hình chẩn đoán viêm phổi

Một mô hình Deep Learning được huấn luyện để phát hiện **viêm phổi từ ảnh X-quang ngực** đạt tới **95% accuracy**. Tuy nhiên, khi dùng công cụ XAI để trực quan hóa vùng mà mô hình "chú ý", người ta phát hiện:

- Mô hình **không hề nhìn vào vùng phổi**.
- Nó dựa vào **mảnh kim loại nhỏ** hoặc **ký hiệu của máy X-ray di động**.

Nguyên nhân: trong tập dữ liệu huấn luyện, phần lớn bệnh nhân nặng được chụp bằng máy X-ray di động có gắn token kim loại đó. Do đó mô hình học sai quy luật: *"Nếu có token kim loại → bệnh nặng"*.



Kết luận: Mô hình đúng về mặt thống kê, nhưng sai về mặt y học. Đây là lý do ta cần **XAI (Explainable AI)** để nhìn vào bên trong "hộp đen" và hiểu vì sao mô hình đưa ra quyết định như vậy.

1.2 Mục tiêu thật sự của XAI

XAI không chỉ là công cụ vẽ heatmap, mà là **tư tưởng khoa học về sự minh bạch trong trí tuệ nhân tạo**.

Khía cạnh	**Học máy truyền thống**	**Học máy có XAI**
Trọng tâm	Đạt accuracy cao nhất	Hiểu và kiểm soát được quyết định
Người sử dụng	Nhà khoa học dữ liệu	Cả chuyên gia, người dùng, cơ quan quản lý
Cách nhìn mô hình	"Hộp đen" không giải thích được	"Hộp trong suốt" có thể phân tích
Phản ứng khi sai	Thử tham số, thêm dữ liệu	Tìm hiểu nguyên nhân và sửa đúng chỗ

Học máy trả lời "mô hình đúng không?", còn XAI trả lời "vì sao nó đúng (hoặc sai)?"

1.3 Các loại giải thích trong XAI

XAI được chia thành ba nhóm chính:

a. Model-centric Explanation

Tập trung vào bản thân mô hình: cấu trúc, trọng số, attention, rule… Phù hợp với người nghiên cứu kỹ thuật.

- Ví dụ: trọng số trong Logistic Regression, attention map trong Transformer, feature importance trong Random Forest.

b. Human-centric Explanation

Chuyển thông tin kỹ thuật sang ngôn ngữ dễ hiểu với người dùng. Giúp chuyên gia hoặc người dùng cuối hiểu *vì sao* hệ thống đưa ra quyết định.

- Ví dụ: "Khoản vay bị từ chối vì thu nhập thấp và lịch sử nợ xấu".

c. Example-based Explanation

Giải thích bằng cách đưa ra các ví dụ tương tự trong dữ liệu. Dạng lý luận "case-based reasoning".

- Ví dụ: "Trong dữ liệu, 100 khách hàng giống bạn đều được duyệt vay vì họ có đặc điểm A, B".

Thực tế: Ba hướng này thường kết hợp với nhau. Ví dụ: SHAP là model-centric ở tầng tính toán, nhưng khi hiển thị bằng Waterfall Plot lại là human-centric.

1.4 XAI giúp sửa mô hình thông minh hơn

Nếu không có XAI, khi mô hình sai, ta chỉ biết "nó sai". Với XAI, ta biết *vì sao nó sai*, và từ đó sửa đúng chỗ.

Quan sát từ XAI	**Hướng điều chỉnh**
Mô hình dựa vào ký hiệu máy chụp thay vì nội dung ảnh	Làm sạch dữ liệu, loại bỏ ký hiệu, tái huấn luyện
Mô hình ưu tiên "income" quá mạnh	Cân bằng lại dữ liệu, thêm regularization hoặc constraint
Mô hình bỏ qua "giới tính" dù có ý nghĩa	Xem xét thêm feature đó nếu hợp lý về đạo đức

XAI biến việc huấn luyện từ "thử sai" thành "học có định hướng".

1.5 Lợi ích cốt lõi của XAI

1. **Tuân thủ quy định (Regulatory Compliance)**: Trong y tế, tài chính, pháp luật, mọi quyết định cần được giải thích hợp pháp. XAI giúp mô hình minh bạch, dễ kiểm tra và audit.
2. **Giảm thiên lệch (Bias Reduction)**: XAI giúp phát hiện yếu tố bị lệch (ví dụ: mô hình phân biệt giới tính hoặc chủng tộc), từ đó điều chỉnh dữ liệu huấn luyện.
3. **Khám phá quy luật mới (Knowledge Discovery)**: Phân tích SHAP value có thể tiết lộ mối quan hệ ẩn mà trước đây chưa được nhận ra.
4. **Tăng niềm tin (Trust)**: Người dùng, chuyên gia và tổ chức tin tưởng hơn khi hiểu được lý do mô hình đưa ra dự đoán.

6. Ví dụ tổng hợp (theo sơ đồ minh họa)

1. Bác sĩ sử dụng mô hình dự đoán viêm phổi → mô hình dự đoán "bị bệnh".
2. Dùng XAI heatmap → phát hiện mô hình chỉ chú ý vào token kim loại.
3. Hiểu ra lỗi → làm sạch dữ liệu, huấn luyện lại mô hình.
4. Sau khi sửa → mô hình chú ý đúng vùng phổi.
5. Kết quả → bác sĩ tin tưởng hơn, mô hình được chấp nhận trong hệ thống bệnh viện.

Kết luận

Thông điệp chính

XAI không phải là công cụ trang trí, mà là yếu tố nền tảng để xây dựng **niềm tin giữa con người và trí tuệ nhân tạo**.
Nó giúp ta:

- Hiểu cách mô hình suy nghĩ.
- Biết khi nào nên tin, khi nào nên nghi ngờ.
- Biến "hộp đen" thành "hộp trong suốt".

2 Shapley Value – Cơ sở lý thuyết của SHAP

2.1 Xuất phát điểm: Chia phần thưởng công bằng trong trò chơi

Ý tưởng của **Shapley Value** bắt nguồn từ một vấn đề trong **Lý thuyết trò chơi** (Game Theory), do nhà toán học **Lloyd Shapley** đề xuất năm 1953.

Bài toán gốc

Có một nhóm người chơi hợp tác để tạo ra một "phần thưởng" chung (ví dụ: tiền, điểm số, năng suất, v.v.).
Câu hỏi đặt ra: *Làm sao chia phần thưởng đó sao cho công bằng, tương ứng với đóng góp thật sự của từng người?*

Ví dụ, có 3 người chơi **A, B, C** cùng tạo ra tổng phần thưởng là 100 điểm. Hỏi: mỗi người nên nhận bao nhiêu điểm?

2.2 Ví dụ trực quan: Đội bóng và sức mạnh cá nhân

Giả sử có đội bóng gồm 3 cầu thủ:

- Cầu thủ 1: Tiền đạo (Striker)
- Cầu thủ 2: Tiền vệ (Midfielder)
- Cầu thủ 3: Hậu vệ (Defender)

Ta đo hiệu suất của từng nhóm cầu thủ khi chơi cùng nhau:

Tổ hợp cầu thủ	**Giá trị đội (V)**
{1}	20
{2}	10
{3}	15
{1,2}	35
{1,3}	40
{2,3}	25
{1,2,3}	50

Ta thấy tổng giá trị không bằng tổng cá nhân – các cầu thủ tạo ra **hiệu ứng hợp tác (synergy)**. Câu hỏi: Ai là người đóng góp nhiều nhất cho sức mạnh chung của đội?

2.3 Đóng góp biên (Marginal Contribution)

Đóng góp biên của một người chơi trong một tập hợp (coalition) được định nghĩa là:

$$\Delta_i(S) = v(S \cup \{i\}) - v(S)$$

tức là: giá trị tăng thêm khi người chơi i được thêm vào nhóm S.

Ví dụ: Nếu xét tập $\{2, 3\}$ có giá trị $v(\{2, 3\}) = 25$ và khi thêm cầu thủ 1 vào, ta được $v(\{1, 2, 3\}) = 50$, thì đóng góp biên của cầu thủ 1 là:

$$\Delta_1(\{2, 3\}) = 50 - 25 = 25$$

Ta lặp lại quá trình này cho tất cả các tổ hợp có hoặc không có cầu thủ 1, rồi lấy trung bình. Giá trị trung bình đó chính là **Shapley Value** của cầu thủ 1.

2.4 Shapley Value

Với tập người chơi $N = \{1, 2, \ldots, n\}$ và hàm giá trị $v(S)$, Shapley Value của người chơi i được định nghĩa là:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} [v(S \cup \{i\}) - v(S)]$$

- S: mọi tập con không chứa i.
- $v(S \cup \{i\}) - v(S)$: đóng góp biên của i trong nhóm S.
- Hệ số $\frac{|S|!(n-|S|-1)!}{n!}$: xác suất xảy ra của tập S trong mọi hoán vị có thể.

Trực giác công thức

Shapley Value là **trung bình đóng góp biên của người chơi i trên tất cả các thứ tự có thể mà người đó tham gia vào cuộc chơi.**

2.5 Các tính chất của Shapley Value

Shapley Value là quy tắc phân bổ duy nhất thỏa mãn 4 tính chất công bằng sau:

Tính chất	**Ý nghĩa**
Efficiency	Tổng Shapley của tất cả người chơi bằng tổng phần thưởng của nhóm.
Symmetry	Hai người chơi có đóng góp như nhau thì nhận phần bằng nhau.
Dummy	Nếu một người chơi không ảnh hưởng đến giá trị nhóm, thì Shapley của họ bằng 0.
Additivity	Nếu có hai trò chơi f và g, thì $\phi_i(f + g) = \phi_i(f) + \phi_i(g)$.

Kết luận: Shapley Value được xem là "chuẩn vàng" của công bằng trong các bài toán phân bổ tài nguyên và ra quyết định hợp tác.

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} [v(S \cup \{i\}) - v(S)]$$

- S: mọi tập con không chứa i.
- $v(S \cup \{i\}) - v(S)$: đóng góp biên của i trong nhóm S.
- Hệ số $\frac{|S|!(n-|S|-1)!}{n!}$: xác suất xảy ra của tập S trong mọi hoán vị có thể.

Trực giác công thức

Shapley Value là **trung bình đóng góp biên của người chơi i trên tất cả các thứ tự có thể mà người đó tham gia vào cuộc chơi.**

2.6 Các tính chất của Shapley Value

Shapley Value là quy tắc phân bổ duy nhất thỏa mãn 4 tính chất công bằng sau:

Tính chất	**Ý nghĩa**
Efficiency	Tổng Shapley của tất cả người chơi bằng tổng phần thưởng của nhóm.
Symmetry	Hai người chơi có đóng góp như nhau thì nhận phần bằng nhau.
Dummy	Nếu một người chơi không ảnh hưởng đến giá trị nhóm, thì Shapley của họ bằng 0.
Additivity	Nếu có hai trò chơi f và g, thì $\phi_i(f+g) = \phi_i(f) + \phi_i(g)$.

Kết luận: Shapley Value được xem là "chuẩn vàng" của công bằng trong các bài toán phân bổ tài nguyên và ra quyết định hợp tác.

2.7 Từ lý thuyết trò chơi sang Machine Learning

Trong bối cảnh Machine Learning:

- **Game** → Quá trình mô hình tạo ra một dự đoán.
- **Players** → Các **feature** của mô hình (ví dụ: diện tích, số phòng ngủ, khu vực...).
- **Value (v)** → Dự đoán đầu ra của mô hình ($f(x)$).

Khi đó, câu hỏi trở thành:

"Mỗi đặc trưng (feature) đóng góp bao nhiêu vào dự đoán cuối cùng của mô hình?"

Một dự đoán cụ thể $f(x)$ được xem như **phần thưởng**, và Shapley Value cho ta biết feature nào góp phần làm tăng hoặc giảm giá trị dự đoán đó.

2.8 Ví dụ chi tiết: Bài toán dự đoán giá nhà

Giả sử ta có mô hình dự đoán giá nhà (triệu đồng) dựa trên 3 đặc trưng:

Ký hiệu	**Đặc trưng**	**Mô tả**
x_1	Diện tích	(m^2)
x_2	Số phòng ngủ	Số nguyên
x_3	Khu vực	City center / Suburb

Ta muốn biết: trong dự đoán của mô hình, "diện tích", "phòng ngủ", và "khu vực" ảnh hưởng bao nhiêu.

2.8.1 Ý tưởng của hàm giá trị $v(S)$

Khi chỉ giữ lại một tập feature S (ví dụ: $\{x_1, x_2\}$) ta tính **giá trị trung bình của mô hình** khi thay ngẫu nhiên các feature còn lại bằng dữ liệu thật trong tập huấn luyện.

Ví dụ tính $v(S)$

Giữ nguyên x_1, x_2, và thay x_3 (khu vực) ngẫu nhiên:

Khu vực	**Dự đoán $f(x_1, x_2, x_3)$ (triệu đồng)**
Trung tâm	3.0
Ngoại ô	2.0
Ven đô	2.5

$$v(S) = \frac{3.0 + 2.0 + 2.5}{3} = 2.5$$

2.8.2 Diễn giải ý nghĩa

- $v(S)$: Giá trị mô hình khi chỉ biết feature trong S.
- $f(x)$: Giá trị mô hình khi biết tất cả feature.
- $v(S \cup \{i\}) - v(S)$: Mức tăng giá trị khi thêm feature i.

Kết luận: Shapley Value của một feature chính là **độ quan trọng trung bình của nó** trong mọi kịch bản mô hình có thể (khi các feature khác được giữ cố định hoặc ngẫu nhiên).

2.9 Tính toán thực tế của Shapley Value

Số lượng tổ hợp tăng theo cấp số mũ:

$$2^n \text{ tổ hợp cần tính cho } n \text{ feature.}$$

- $n = 4 \Rightarrow 2^4 = 16$ tổ hợp (chấp nhận được)
- $n = 32 \Rightarrow 2^{32} \approx 17$ tỷ tổ hợp (bất khả thi)

Do đó, ta cần phương pháp **xấp xỉ Shapley Value** bằng cách **lấy mẫu ngẫu nhiên (sampling)** các permutation. Sau này, phương pháp **Kernel SHAP** ra đời để thực hiện việc này nhanh và chính xác hơn (phần 3 sẽ trình bày).

—

2.10 Dự đoán cộng tính (Additive Explanation)

Trong Machine Learning, Shapley Value giúp biểu diễn mô hình dự đoán theo dạng **cộng tính**:

$$f(x) = E[f(X)] + \sum_i \phi_i$$

Trong đó:

- $E[f(X)]$: Dự đoán trung bình của mô hình (Base Value).
- ϕ_i: Shapley Value của feature i.

Ví dụ:

$$f(x) = 0.3 + 0.1 + 0.2 - 0.05 = 0.55$$

Diễn giải: Xác suất được duyệt vay tăng 0.1 nhờ thu nhập, tăng 0.2 nhờ tuổi, nhưng giảm 0.05 do nợ xấu.

2.11 Trực giác cuối cùng: "Chia bánh công bằng"

Hình dung dễ hiểu

Nếu một dự đoán $f(x)$ là **một chiếc bánh** , thì Shapley Value là cách **chia chiếc bánh đó cho từng feature** sao cho:

- Không ai bị thiệt phần.
- Tổng phần chia đúng bằng tổng bánh.
- Mỗi feature nhận phần tương xứng với ảnh hưởng của nó.

Tổng kết

Khía cạnh	**Nội dung chính**
Nền tảng	Lý thuyết trò chơi – chia phần thưởng công bằng.
Công thức	Trung bình đóng góp biên trên mọi tổ hợp feature.
Trong ML	Mỗi feature là người chơi, mỗi dự đoán là phần thưởng.
Ví dụ	Mô hình giá nhà: tính trung bình dự đoán trên các vùng khác nhau.
Kết quả	$f(x) = \text{Base Value} + \sum \phi_i$.
Ứng dụng	Đo lường ảnh hưởng từng đặc trưng, giúp mô hình minh bạch.

3 Kernel SHAP – Cách xấp xỉ Shapley Value bằng hồi quy tuyến tính có trọng số

3.1 Vấn đề đặt ra: Shapley Value chính xác là bất khả thi khi số lượng đặc trưng lớn

Ở phần trước, ta đã thấy công thức Shapley Value yêu cầu tính trung bình trên tất cả các tập con của tập đặc trưng.

Với n đặc trưng, ta phải tính 2^n tổ hợp. Điều này trở nên bất khả thi trong thực tế.

Số feature (n)	**Số tổ hợp 2^n**	**Ghi chú**
4	16	Có thể tính được
10	1,024	Bắt đầu tốn kém
32	17 tỷ	Bất khả thi

Vì vậy, cần một phương pháp **xấp xỉ nhanh** nhưng vẫn giữ được tinh thần công bằng của Shapley. Phương pháp đó chính là **Kernel SHAP**.

3.2 Ý tưởng cốt lõi của Kernel SHAP

Thay vì duyệt qua toàn bộ 2^n tổ hợp, Kernel SHAP sử dụng một mô hình hồi quy tuyến tính có trọng số để **xấp xỉ Shapley Value**.

Giả sử ta cần giải thích dự đoán $f(x)$, Kernel SHAP xem dự đoán này như một mô hình tuyến tính cục bộ:

$$f(z') \approx \phi_0 + \sum_{i=1}^{M} \phi_i z'_i$$

- $f(z')$: Dự đoán của mô hình gốc khi chỉ bật các feature có $z'_i = 1$.
- ϕ_0: Base value (giá trị dự đoán trung bình).
- ϕ_i: Đóng góp của feature thứ i (Shapley Value).
- z': Vector nhị phân thể hiện feature nào đang bật hoặc tắt.

Mục tiêu: Tìm các ϕ_i sao cho mô hình tuyến tính trên mô phỏng được cách $f(x)$ thay đổi theo các tập feature.

3.3 Quy trình tổng quan của Kernel SHAP

Quy trình của Kernel SHAP được tóm tắt như sau:

Bước	**Mô tả**
1	Chọn một mẫu x cần giải thích.
2	Chọn một tập **background** (tập nền) từ dữ liệu huấn luyện.
3	Tạo nhiều tổ hợp bật/tắt feature (biểu diễn bằng vector nhị phân z').
4	Với mỗi tổ hợp, thay feature "tắt" bằng giá trị thật từ background set, rồi chạy mô hình để lấy $f(x)$.
5	Hồi quy tuyến tính có trọng số để tìm các hệ số ϕ_i.

3.4 Vai trò của Background Set

Mô hình học máy không thể chạy nếu bị thiếu feature. Ví dụ, không thể dự đoán giá nhà chỉ với "diện tích" mà thiếu "khu vực".

Vì vậy, Kernel SHAP sử dụng một tập **background** lấy từ dữ liệu thật. Khi một feature bị "tắt", nó được thay bằng giá trị ngẫu nhiên từ tập nền này.

- Background càng đa dạng, kết quả SHAP càng ổn định.
- Thường chọn từ 100–1000 mẫu nền trong tập huấn luyện.

3.5 Trọng số trong Kernel SHAP

Khác với Shapley Value gốc, Kernel SHAP cần gán trọng số khác nhau cho mỗi tổ hợp. Trọng số này đảm bảo tính công bằng khi chỉ xét một phần nhỏ các tập con.

Công thức trọng số:

$$w_C = \frac{(M-1)}{\binom{M}{|C|} \cdot |C| \cdot (M - |C|)}$$

Trong đó:

- M: Tổng số feature.
- $|C|$: Số feature đang bật trong tổ hợp.
- $\binom{M}{|C|}$: Số cách chọn $|C|$ feature từ M.

Ý nghĩa:

- Các tổ hợp có ít hoặc gần đầy đủ feature sẽ có trọng số lớn hơn.
- Những trường hợp biên (rất ít hoặc gần đủ feature) chứa thông tin ranh giới quan trọng nhất để mô hình học.

Ví dụ (với $M = 5$):

Kích thước tổ hợp $\|C\|$	**Trọng số w_C**
1	0.20
2	0.06
3	0.06
4	0.20

3.6 Mô hình tuyến tính cục bộ

Sau khi có các cặp $(z', f(z'))$ và trọng số $w_{z'}$, Kernel SHAP giải một bài toán hồi quy tuyến tính có trọng số:

$$\min_{\phi} \sum_{z'} w_{z'} \cdot \left(f(z') - \phi_0 - \sum_i \phi_i z'_i \right)^2$$

Kết quả ϕ_i chính là **Shapley Value xấp xỉ**. Vì hồi quy tuyến tính rất nhanh, Kernel SHAP cho phép ta tính được giá trị SHAP cho hàng nghìn mẫu trong thời gian ngắn.

3.7 Ví dụ minh họa cụ thể

Giả sử mô hình gốc là:

$$f(F_1, F_2, F_3, F_4) = 10 + 2F_1 + 3F_2 + F_3^2 - 0.5F_4$$

Mẫu cần giải thích:

$$X = [F_1 = 3, F_2 = 4, F_3 = 2, F_4 = 10]$$

Tập nền trung bình:

$$E[X_{bg}] = [1, 1, 1, 4]$$

Tính Base Value:

$$\phi_0 = f(1, 1, 1, 4) = 10 + 2 + 3 + 1 - 2 = 14$$

Giá trị thực của mô hình:

$$f(X) = 10 + 6 + 12 + 4 - 5 = 27$$

Vì vậy:

$$14 + \phi_1 + \phi_2 + \phi_3 + \phi_4 = 27$$

Sau khi hồi quy tuyến tính có trọng số, Kernel SHAP cho kết quả:

$$\phi_1 = 4, \quad \phi_2 = 9, \quad \phi_3 = 3, \quad \phi_4 = -3$$

$$14 + 4 + 9 + 3 - 3 = 27$$

Feature F_2 có tác động mạnh nhất, giúp mô hình tăng giá trị dự đoán nhiều nhất.

3.8 Trực giác của hồi quy tuyến tính cục bộ

Kernel SHAP giả định rằng trong vùng lân cận của mẫu cần giải thích, mối quan hệ giữa đầu ra và các feature là **xấp xỉ tuyến tính**.

Nhờ vậy:

- Không cần hiểu toàn bộ mô hình (có thể là phi tuyến như Neural Network).
- Chỉ cần xấp xỉ một mô hình tuyến tính nhỏ quanh điểm x.

Do đó, Kernel SHAP được xem là một **Local Surrogate Explainer** – bộ giải thích cục bộ cho các mô hình "hộp đen".

3.9 So sánh giữa Shapley Value gốc và Kernel SHAP

Tiêu chí	**Shapley gốc**	**Kernel SHAP**
Độ chính xác	Chính xác tuyệt đối	Xấp xỉ bằng hồi quy tuyến tính có trọng số
Tốc độ	Rất chậm, phải duyệt 2^n tổ hợp	Rất nhanh, chỉ cần vài trăm mẫu
Yêu cầu mô hình	Cần biết cấu trúc mô hình $f(x)$	Chỉ cần truy cập đầu ra mô hình (model-agnostic)
Ứng dụng thực tế	Ít dùng do tốn tài nguyên	Là lõi của thư viện SHAP hiện đại

3.10 Kết quả đầu ra của Kernel SHAP

Sau khi tính xong, Kernel SHAP trả về các giá trị:

$$\phi_0,\ \phi_1,\ \phi_2,\ \ldots,\ \phi_M$$

Mô hình được viết lại theo dạng cộng tính:

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i$$

- ϕ_0: Giá trị trung bình (base value).
- $\phi_i > 0$: Feature làm tăng dự đoán.
- $\phi_i < 0$: Feature làm giảm dự đoán.

Từ đó, ta có thể biểu diễn kết quả bằng **Waterfall Plot**, **Beeswarm Plot**, hoặc **Bar Chart** để minh họa tầm quan trọng của từng feature.

3.11 Sơ đồ Pipeline tóm tắt

1. Chọn mẫu x cần giải thích.
2. Chọn background từ dữ liệu huấn luyện.
3. Tạo các phiên bản bật/tắt feature (z').
4. Thay feature "tắt" bằng giá trị nền.
5. Tính $f(z')$, gán trọng số $w(z')$.
6. Hồi quy tuyến tính để tìm ϕ_i.

Kết quả cuối cùng là bảng giá trị SHAP thể hiện mức độ đóng góp của từng feature.

Kết luận

Thông điệp chính

Kernel SHAP là cầu nối giữa lý thuyết Shapley và thực tế tính toán. Nó giúp:

- Giải thích mọi mô hình (Random Forest, XGBoost, Neural Network, v.v.).
- Giữ được tính công bằng trong phân bổ ảnh hưởng giữa các feature.
- Tính nhanh, chính xác, và dễ mở rộng cho dữ liệu lớn.

4 SHAP Pipeline và Minh Họa Từng Bước

4.1 Tổng quan mục tiêu của SHAP Pipeline

Ở phần Kernel SHAP, ta đã biết SHAP sử dụng mô hình tuyến tính cục bộ để ước lượng đóng góp của từng đặc trưng. Phần này mô tả cụ thể **các bước tuần tự mà SHAP thực hiện** khi tính giá trị cho một mẫu cần giải thích.

Bước	Mô tả	Đầu ra chính
1	Chọn mẫu cần giải thích	Vector $x = [x_1, x_2, ..., x_M]$
2	Chọn tập background (nền)	Mẫu đại diện dữ liệu huấn luyện
3	Tạo tổ hợp bật/tắt feature	Vector nhị phân $z' \in \{0,1\}^M$
4	Tính dự đoán mô hình $f(z')$	Dự đoán khi chỉ bật feature trong z'
5	Gán trọng số kernel $w(z')$	Cân bằng giữa các tổ hợp
6	Hồi quy tuyến tính có trọng số	Tìm $\phi_0, \phi_1, ..., \phi_M$

Bước 1 – Chọn mẫu cần giải thích Chọn một mẫu cụ thể x trong tập kiểm thử mà ta muốn hiểu vì sao mô hình đưa ra kết quả như vậy.

Feature	Giá trị
Diện tích (m^2)	80
Số phòng ngủ	3
Khu vực	Trung tâm
Giá dự đoán	3.2 tỷ

Đây là mẫu mà SHAP sẽ "mổ xẻ" để biết mỗi đặc trưng đã đóng góp bao nhiêu.

Bước 2 – Chọn Background Set Vì SHAP cần "tắt" và "bật" các feature, nên cần dữ liệu thay thế cho phần bị tắt. **Background Set** đóng vai trò làm mức tham chiếu trung bình của dữ liệu thật.

Diện tích	Phòng ngủ	Khu vực
60	2	Ven đô
90	4	Ngoại ô
70	3	Trung tâm

Khi một feature bị "tắt", SHAP sẽ thay bằng giá trị tương ứng trong background (trung bình hoặc theo từng mẫu nền).

Bước 3 – Tạo các tổ hợp bật/tắt Feature SHAP sinh ra các vector nhị phân $z' = [z'_1, z'_2, ..., z'_M]$:

- $z'_i = 1$: feature được bật (giá trị thật từ mẫu cần giải thích)
- $z'_i = 0$: feature bị tắt (thay bằng background)

Tổ hợp	Vector z'	Ý nghĩa
1	[0,0,0]	Không dùng feature nào
2	[1,0,0]	Chỉ bật diện tích
3	[0,1,0]	Chỉ bật số phòng ngủ
4	[1,1,0]	Bật diện tích và phòng ngủ
5	[1,1,1]	Bật tất cả feature (mẫu thật)

Kernel SHAP không cần tính toàn bộ 2^M tổ hợp, mà chỉ lấy mẫu ngẫu nhiên vài trăm tổ hợp đủ để hồi quy chính xác.

Bước 4 – Tính giá trị mô hình $f(z')$ cho từng tổ hợp Với mỗi tổ hợp z':

- SHAP tạo một phiên bản "pha trộn" giữa mẫu x và background.
- Mô hình được gọi để tính $f(z')$.

Tổ hợp z'	**Giá trị $f(z')$ (tỷ đồng)**
$[0, 0, 0]$	2.0
$[1, 0, 0]$	2.4
$[0, 1, 0]$	2.3
$[1, 1, 0]$	2.6
$[1, 1, 1]$	3.2

Bước 5 – Tính trọng số Kernel $w(z')$ Trọng số trong Kernel SHAP được định nghĩa là:

$$w_C = \frac{(M-1)}{\binom{M}{|C|} \cdot |C| \cdot (M - |C|)}$$

- M: tổng số feature.
- $|C|$: số feature bật trong tổ hợp.
- $\binom{M}{|C|}$: số cách chọn $|C|$ feature trong M.

Ý nghĩa:

- Các tổ hợp ở rìa (ít hoặc gần đầy đủ feature) có trọng số cao.
- Các tổ hợp trung gian có trọng số nhỏ hơn.

Kích thước tổ hợp $|C|$	**Trọng số w_C (xấp xỉ)**
1	0.20
2	0.06
3	0.06
4	0.20

Bước 6 – Hồi quy tuyến tính có trọng số Sau khi có các bộ dữ liệu $(z', f(z'))$ và trọng số tương ứng $w(z')$, Kernel SHAP giải bài toán hồi quy tuyến tính:

$$\min_{\phi} \sum_{z'} w(z') \cdot \left(f(z') - \phi_0 - \sum_i \phi_i z'_i \right)^2$$

- ϕ_i: giá trị đóng góp (Shapley Value xấp xỉ).
- ϕ_0: Base Value (giá trị trung bình của mô hình).

4.2 Ví dụ hoàn chỉnh – Mô hình dự đoán giá nhà

Giả sử mô hình dự đoán giá nhà có:

- 3 feature: diện tích, phòng ngủ, khu vực.
- Base Value: $\phi_0 = 2.0$ tỷ.
- Dự đoán thực tế: $f(x) = 3.2$ tỷ.

Kết quả SHAP sau khi hồi quy:

Feature	**Giá trị SHAP** ϕ_i	**Ý nghĩa**
Diện tích	+0.6	Làm tăng giá 0.6 tỷ.
Phòng ngủ	+0.4	Làm tăng giá 0.4 tỷ.
Khu vực	+0.2	Làm tăng giá 0.2 tỷ.

Kiểm tra tổng:

$$2.0 + 0.6 + 0.4 + 0.2 = 3.2$$

→ SHAP giải thích hoàn hảo dự đoán của mô hình.

4.3 Diễn giải bằng biểu đồ trực quan

SHAP hỗ trợ nhiều dạng biểu đồ:

- **Waterfall Plot**: hiển thị từng feature cộng/trừ vào base value.
- **Force Plot**: dạng thanh động minh họa hướng tác động (đỏ: tăng, xanh: giảm).
- **Beeswarm Plot**: thể hiện phân bố SHAP values của toàn bộ tập mẫu.

Ví dụ:

- Base Value: 2.0 tỷ.
- Diện tích: +0.6 → tăng giá mạnh.
- Phòng ngủ: +0.4 → tăng thêm.
- Khu vực: +0.2 → tăng nhẹ.

Không có feature âm → mô hình dự đoán "an toàn" và hợp lý.

4.4 SHAP Pipeline trong thư viện Python

Trong thư viện `shap`:

1. Khởi tạo background bằng `shap.maskers.Independent()`.
2. Sinh các mẫu bật/tắt feature.
3. Gọi mô hình `predict()` cho từng mẫu.
4. Gán trọng số kernel.
5. Giải hồi quy tuyến tính (weighted least squares).
6. Trả về ϕ_i và ϕ_0.
7. Trực quan hóa bằng `shap.plots.waterfall()`, `shap.plots.bar()`, `shap.plots.force()`.

4.5 Ý nghĩa thực tiễn của Pipeline

- Giải thích từng dự đoán riêng biệt (local explanation).
- Tổng hợp mức ảnh hưởng trung bình (global explanation).
- Phát hiện bias hoặc overfitting trong dữ liệu.

Ví dụ:

- Mô hình tín dụng đánh giá quá cao "địa chỉ cư trú" $\rightarrow$ có dấu hiệu bias vùng miền.
- Feature có $|\phi_i|$ nhỏ $\rightarrow$ có thể loại bỏ để giảm nhiễu.

4.6 Tóm tắt toàn bộ SHAP Pipeline

Bước	**Hành động**	**Mục tiêu**
1	Chọn mẫu x	Xác định quyết định cần giải thích
2	Chọn background	Thiết lập giá trị tham chiếu
3	Bật/tắt feature	Tạo các tổ hợp dữ liệu
4	Chạy mô hình	Thu giá trị $f(z')$
5	Tính trọng số kernel	Cân bằng tổ hợp
6	Hồi quy tuyến tính	Ước lượng ϕ_i (SHAP Value)

$$f(x) = \phi_0 + \sum_i \phi_i$$

Diễn giải: Feature i đóng góp ϕ_i vào quyết định của mô hình đối với mẫu x.

Kết luận

Thông điệp chính

SHAP Pipeline là trung tâm của Explainable AI hiện đại. Nó giúp:

- Biến mô hình "hộp đen" thành mô hình có thể giải thích được.
- Minh chứng mối liên hệ nhân quả giữa dữ liệu và dự đoán.
- Cung cấp cơ sở toán học vững chắc cho các quyết định quan trọng trong tài chính, y tế, pháp lý, và công nghiệp.