XGBoost

Giới thiệu

Mục tiêu bài học

- Hiểu được **vấn đề kinh doanh thực tế** trong marketing trực tiếp.
- Nhận diện được lý do tại sao **dự đoán nhị phân (binary prediction)** lại quan trọng.
- Giới thiệu sơ lược về mô hình **XGBoost** – một thuật toán học máy mạnh mẽ cho các bài toán phân loại.
- Nhận diện các thách thức trong phân tích khách hàng truyền thống và cách ML có thể giải quyết.

1. Bối cảnh vấn đề: Dự báo trong Marketing Trực tiếp

Tình huống:

Một ngân hàng tại Bồ Đào Nha thực hiện chiến dịch tiếp thị qua điện thoại nhằm thuyết phục khách hàng đăng ký **tài khoản tiết kiệm**.

- Dữ liệu: danh sách khách hàng (giới tính, tuổi, tình trạng hôn nhân, nghề nghiệp,...)
- Kết quả: khách hàng **đồng ý (yes)** hoặc **không đồng ý (no)** với chiến dịch.

Câu hỏi đặt ra:

Liệu ta có thể dự đoán trước được ai sẽ nói "YES"?

> Nếu có thể, ngân hàng sẽ:
>
> - Tối ưu hóa chiến dịch
> - Giảm chi phí
> - Tăng tỷ lệ thành công

2. Tại sao việc dự đoán lại quan trọng?

1. **Customer Churn – Mất khách hàng:**

- Gọi nhầm người không hứng thú → khách hàng **hủy đăng ký, mất thiện cảm**.

2. **Opportunity Cost – Mất cơ hội:**

- Gọi nhầm người → **bỏ lỡ cơ hội gọi đúng người tiềm năng**.

3. **Relevance – Mất sự liên quan:**

- Gửi thông tin không phù hợp → họ **không quan tâm đến sản phẩm về sau**.

3. Chi phí và lợi nhuận trong B2B

- Trong môi trường B2B, **mỗi khách hàng mang lại doanh thu lớn**.
- Nhưng **chi phí tiếp cận ban đầu cao**, cần đầu tư nguồn lực.
- Gọi sai người → **lãng phí thời gian & công sức**.
- Gọi đúng người → **tăng khả năng chốt đơn & tiết kiệm chi phí**.

4. Phân tích khách hàng thủ công – Có vấn đề gì?

Ví dụ:

Phân nhóm khách hàng theo:

- Giới tính (Nam/Nữ)
- Tuổi (Trẻ/Trung niên/Già)
- Nghề nghiệp
- Trình độ học vấn
- Trạng thái tài chính
- Đã từng vay hay chưa,...

Bạn sẽ có **hàng trăm tổ hợp khác nhau** cần phân tích.Các vấn đề lớn:

1. **Độ phức tạp (Depth):** càng nhiều yếu tố → ma trận phân nhóm càng phức tạp.
2. **Tầm quan trọng (Importance):** không biết yếu tố nào **ảnh hưởng mạnh nhất**.
3. **Nhiễu (Noise):** có nhiều tổ hợp hiếm gặp → **khó nhìn thấy quy luật** bằng mắt thường.

5. Giải pháp: Sử dụng Machine Learning

Thay vì phân tích thủ công, ta sẽ:

- Sử dụng mô hình học máy như **XGBoost** để:
  - Tự động xác định yếu tố quan trọng
  - Dự đoán xác suất khách hàng sẽ đồng ý
  - Gợi ý danh sách khách hàng tiềm năng

Ưu điểm XGBoost:

- Hiệu suất cao
- Làm việc tốt với dữ liệu tabular (dữ liệu dạng bảng)
- Khả năng xử lý dữ liệu không cân bằng
- Có thể đánh giá mức độ quan trọng của từng đặc trưng (feature importance)

Đa cộng tuyến

Đa cộng tuyến (Multicollinearity) là hiện tượng trong mô hình hồi quy (hoặc mô hình học máy tuyến tính) khi **hai hoặc nhiều biến độc lập (predictor variables)** có mối quan hệ tuyến tính chặt chẽ với nhau — nghĩa là **có thể dự đoán được một biến từ một hoặc nhiều biến khác**.

Hiểu đơn giản:

> Khi các biến giải thích (X) lại **tự giải thích lẫn nhau**, mô hình sẽ **khó xác định chính xác ảnh hưởng riêng của từng biến** đến biến phụ thuộc (Y).

Hậu quả của đa cộng tuyến:

- Các hệ số hồi quy (coefficient) trở nên **không ổn định**, rất **nhạy cảm với dữ liệu**.
- **Ý nghĩa thống kê (p-value)** của các biến có thể sai lệch → dễ **hiểu nhầm** rằng biến không quan trọng.
- Dễ dẫn đến mô hình **overfitting** hoặc **dự đoán kém** khi dùng dữ liệu mới.

Ví dụ minh họa:

Giả sử bạn xây dựng một mô hình dự đoán **giá nhà** với các biến đầu vào sau:

Biến	Ý nghĩa
`house_size`	Diện tích ngôi nhà (m^2)
`num_bedrooms`	Số lượng phòng ngủ
`total_area`	Tổng diện tích bao gồm sân vườn, gara

Trong thực tế:

- `house_size` và `total_area` **gần như tuyến tính với nhau** (vì tổng diện tích = diện tích nhà + diện tích phụ).
- Điều này gây **đa cộng tuyến**.

→ Khi huấn luyện mô hình, thuật toán khó xác định chính xác:

- Liệu giá nhà tăng là do `house_size` hay `total_area`?

Dấu hiệu nhận biết đa cộng tuyến:

- **Hệ số hồi quy có dấu hiệu "lạ"** (ví dụ: âm khi đáng ra phải dương).
- Giá trị **p-value cao bất thường** mặc dù biến đó có vẻ quan trọng.
- Dùng chỉ số thống kê như:
  - **VIF (Variance Inflation Factor)**: nếu VIF > 5 hoặc 10 → có thể có đa cộng tuyến.

Cách xử lý đa cộng tuyến:

1. **Loại bỏ một trong các biến có tương quan cao**.
2. **Tổng hợp các biến lại** (dùng PCA, hoặc tạo một biến trung gian).
3. **Chuẩn hóa dữ liệu** (scaling) có thể giúp phần nào.
4. **Sử dụng mô hình ít nhạy cảm với đa cộng tuyến**, như:
   - **Ridge Regression** (hồi quy co giãn)
   - **Tree-based models** (Random Forest, XGBoost...)

Kết luận:

Đa cộng tuyến là một "kẻ thù thầm lặng" của các mô hình tuyến tính. Khi xây dựng mô hình hồi quy, bạn nên kiểm tra sự tương quan giữa các biến đầu vào để đảm bảo tính chính xác và ổn định cho mô hình.

Nếu bạn muốn, mình có thể giúp bạn tạo một mô hình hồi quy đơn giản bằng Python để **minh họa trực tiếp hiện tượng đa cộng tuyến**. Bạn có muốn thử không?

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

XGBoost trong Phân tích Dự báo

Mục tiêu bài học

- Hiểu được **tại sao lại chọn XGBoost** cho bài toán phân tích.
- Nắm rõ **ưu điểm vượt trội của XGBoost** so với các thuật toán khác.
- Biết được **cách thức hoạt động cơ bản của XGBoost** và sự khác biệt giữa hai dạng thuật toán: **tree-based** và **linear**.
- Chuẩn bị nền tảng để bước vào phần trực quan hóa thuật toán XGBoost ở bài học sau.

1. Tại sao chọn XGBoost?

Việc chọn một thuật toán máy học không phải ngẫu nhiên. Nó phụ thuộc vào:

- Bản chất của **vấn đề kinh doanh**.
- Mục tiêu của mô hình là gì (phân loại, hồi quy, v.v).
- Các **yêu cầu phức tạp** và **hạn chế** của dữ liệu.

> Trong bài toán tiếp thị trực tiếp của chúng ta (telemarketing), chúng ta cần **phân loại nhị phân**: khách hàng sẽ trả lời "Yes" hay "No" với chiến dịch?

2. Ưu điểm nổi bật của XGBoost

Ưu điểm	Mô tả
Tầm quan trọng của đặc trưng (feature importance)	XGBoost cho bạn biết đặc trưng nào có ảnh hưởng lớn nhất đến kết quả. Điều này rất hữu ích để điều chỉnh chiến lược kinh doanh.
Độ chính xác cao	XGBoost thường vượt trội hơn so với Logistic Regression, Random Forest, và thậm chí cả Deep Learning trong nhiều trường hợp.
Xử lý song song	XGBoost hỗ trợ xử lý song song ngay trong thư viện — giúp tăng tốc huấn luyện.
Tối ưu lặp lại (iterative learning)	Mô hình học dần từ sai lầm trong từng vòng lặp, cải thiện chính xác dần theo thời gian.

3. XGBoost là gì?

- **Tên đầy đủ**: eXtreme Gradient Boosting.
- **Là thuật toán dạng "ensemble"**: Tức là mô hình cuối cùng là sự kết hợp của nhiều mô hình nhỏ (base learners).
- **Sử dụng được cho**:
  - Hồi quy (regression): khi biến mục tiêu là **liên tục**.
  - Phân loại (classification): khi biến mục tiêu là **rời rạc** (như "Yes" / "No").

4. Tree-based vs Linear – Phân biệt 2 cách tiếp cận

Linear (Tuyến tính):

- Phù hợp khi dữ liệu có quan hệ tuyến tính.
- Ví dụ: như đường thẳng trong mô hình hồi quy.

Tree-based (Dạng cây quyết định):

- Dựa vào việc **chia nhánh** theo điều kiện.
- Ví dụ minh họa:
  - Có vỏ bánh không? □ → Không phải là bánh pie.
  - Có viền bánh không? □ → Là bánh pie.

→ Đây chính là cách mà **thuật toán cây** đưa ra quyết định phân loại. Trực quan, dễ hiểu, và rất hiệu quả với các dữ liệu phức tạp, không tuyến tính.

5. Tại sao chọn tree-based cho XGBoost?

- Cho kết quả chính xác hơn trong thực tế.
- Phù hợp với các dữ liệu đa chiều, nhiều điều kiện phức tạp.
- Thực thi tương tự như linear trong mã nguồn, chỉ cần cấu hình một vài tham số khác biệt.

Kết luận

- XGBoost là công cụ cực mạnh cho cả phân loại và hồi quy.
- Trong trường hợp của chúng ta (phân loại khách hàng trả lời “Yes” hoặc “No”), XGBoost sẽ giúp đưa ra mô hình chính xác và khả thi để áp dụng trong kinh doanh.
- Ở video tiếp theo, chúng ta sẽ **hiểu sâu hơn về trực giác bên trong của thuật toán XGBoost** — với đồ thị và ví dụ minh họa quá trình học của mô hình.

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Hoạt Động của XGBoost

MỤC TIÊU BÀI HỌC

Sau bài học này, học viên sẽ:

- Hiểu rõ cơ chế hoạt động của XGBoost.
- Biết cách XGBoost sử dụng trọng số để cải thiện dự đoán.
- Hiểu khái niệm **ensemble learning** và cách XGBoost xử lý **đa cộng tuyến (multicollinearity)**.
- Biết cách áp dụng XGBoost qua ví dụ minh họa.

I. CÁCH XGBOOST HOẠT ĐỘNG

1. Ví dụ Minh Họa Cơ Bản

Giả sử bạn có bảng dữ liệu nhỏ:

Quan sát	Đặc trưng (X)	Kết quả thật (y)
1	0.2	1
2	0.8	0
3	0.5	1
4	0.4	0

2. Vòng Lặp 1 (Cây đầu tiên)

- Mô hình khởi đầu: Mỗi quan sát có **trọng số bằng nhau**.
- Mô hình dự đoán: đúng quan sát 1 và 2, sai quan sát 3 và 4.
- **Kết quả**:
  - Quan sát đúng → giảm trọng số.
  - Quan sát sai → tăng trọng số.

XGBoost học từ sai lầm bằng cách ưu tiên học tốt hơn ở các điểm đã sai.

3. Vòng Lặp 2 (Cây thứ hai)

- Dựa trên trọng số mới.
- Mô hình học tập trung vào các điểm sai (ví dụ: quan sát 3 và 4).

- Lặp lại quá trình: cập nhật mô hình → đánh giá → điều chỉnh trọng số.

Mỗi cây mới **sửa lỗi của cây trước**.

II. ENSEMBLE & SUBSAMPLING

1. Không dùng toàn bộ dữ liệu

- XGBoost **không sử dụng toàn bộ quan sát** trong mỗi vòng lặp → gọi là **subsampling**.
- Ví dụ: Cây 1 bỏ qua quan sát số 3, cây 2 bỏ qua số 4…

2. Không dùng toàn bộ đặc trưng

- Mỗi cây học với một tập con của các đặc trưng.
- Gọi là **column subsampling**.

Nhờ đó, XGBoost tạo ra **nhiều mô hình nhỏ**, mỗi mô hình học trên dữ liệu khác nhau → tổng hợp lại tạo mô hình mạnh mẽ.

Đây chính là **Ensemble Learning**.

III. ƯU ĐIỂM CỦA XGBOOST

Tự động sửa lỗi qua từng vòng
Tránh quá khớp (overfitting) nhờ dùng một phần dữ liệu
Xử lý tốt đa cộng tuyến (Multicollinearity)
Hiệu quả cao cả về tốc độ và độ chính xác

IV. VÍ DỤ THỰC TẾ ĐƠN GIẢN

Bài toán: Dự đoán khách hàng có mua hàng không (1 = mua, 0 = không)

Tuổi	Số lần truy cập	Kết quả
25	3	1
40	5	0
30	2	1
45	6	0

1. **Cây đầu tiên**:
   - Dự đoán đúng 2, sai 2.
   - Cập nhật trọng số.

2. **Cây thứ hai**:
   - Tập trung học các điểm sai.
   - Giảm sai sót tổng thể.
3. **Cây thứ ba trở đi**:
   - Lặp lại quá trình.

Kết quả cuối cùng: Dự đoán chính xác cao hơn nhờ tổ hợp các cây nhỏ.

VI. KẾT LUẬN

- XGBoost học qua từng vòng → cải thiện kết quả dần dần.
- Sử dụng kỹ thuật **tổ hợp** và **lấy mẫu ngẫu nhiên** để tăng hiệu suất.
- Là một trong những thuật toán mạnh mẽ nhất hiện nay cho bài toán phân loại và hồi quy.

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Những Điểm Đặc Biệt (Quirks) của XGBoost

MỤC TIÊU BÀI HỌC

Sau bài học này, bạn sẽ:

- Biết được 3 đặc điểm quan trọng khi sử dụng XGBoost.
- Hiểu cách xử lý biến phân loại, giá trị thiếu (NA), và mối quan hệ phi tuyến.
- Biết cách tránh sai lầm khi tiền xử lý dữ liệu cho XGBoost.

I. GIỚI THIỆU NGẮN GỌN

XGBoost là một thuật toán mạnh mẽ nhưng có một số **đặc điểm riêng biệt (quirks)** mà người dùng cần hiểu rõ để sử dụng hiệu quả. Cụ thể có **3 điểm chính** như sau:

BIẾN PHÂN LOẠI (Categorical Variables)

Vấn đề:

- **XGBoost không hiểu kiểu dữ liệu "character" hay "factor"** (chuỗi hoặc phân loại).
- Bạn cần **chuyển đổi chúng thành biến giả (dummy variables)** trước khi đưa vào mô hình.

Ví dụ:

Giả sử bạn có cột **"Giới tính"** với 2 giá trị:

- Nam
- Nữ

Bạn cần chuyển thành:

Giới tính_Nam	Giới tính_Nữ
1	0
0	1

> Tuy nhiên, bạn **không nên giữ cả 2 cột** này cùng lúc!

Đây là cái gọi là **“Dummy Variable Trap”**, vì việc giữ cả 2 cột sẽ tạo ra **đa cộng tuyến (multicollinearity)**.

Giải pháp:

- Giữ lại 1 cột duy nhất (ví dụ chỉ giữ "Giới tính_Nam")
- Cột còn lại sẽ được suy ra.

II. GIÁ TRỊ THIẾU (Missing Values - NA)

Điểm đặc biệt của XGBoost:

- **Không cần loại bỏ hay thay thế NA**.
- XGBoost **coi giá trị NA là một thông tin riêng biệt**.

Ví dụ minh họa:

Giả sử bạn có cột “Thu nhập”:

ID	Thu nhập
1	5 triệu
2	NA
3	7 triệu
4	NA

Nếu bạn đang dùng hồi quy tuyến tính hay Random Forest, bạn **phải thay thế NA** bằng trung bình hoặc loại bỏ dòng.

 Nhưng với XGBoost:

- Nó **xem NA là một nhánh riêng trong cây quyết định**.
- Giúp giữ lại ý nghĩa của việc “không có thông tin”.

Tại sao điều này quan trọng?

Trong thực tế:

- Nhiều khách hàng không cung cấp đầy đủ dữ liệu.
- Việc giữ nguyên NA sẽ giúp mô hình hiểu rằng **“việc không cung cấp thông tin” là một tín hiệu riêng biệt**.

III. MỐI QUAN HỆ PHI TUYẾN (Non-linearity)

Lợi thế của XGBoost:

- Hiểu được mối quan hệ **phi tuyến (non-linear)** giữa biến độc lập và biến phụ thuộc.

Ví dụ:

Bạn có thể gặp:

- Mối quan hệ hình chữ U (U-shape)
- Mối quan hệ hình chữ S (S-shape)

Với các mô hình hồi quy tuyến tính, bạn phải:

- Biến đổi biến thủ công (thêm x2x^2x2, log, v.v.)

□□ Nhưng XGBoost thì:

- **Tự động học được mối quan hệ phức tạp đó** thông qua cây quyết định.

IV. TỔNG HỢP QUA VÍ DỤ THỰC TẾ

Giả sử bạn có tập dữ liệu về khách hàng:

Tuổi	Giới tính	Thu nhập	Mua hàng
25	Nam	5 triệu	1
32	Nữ	NA	0
40	Nam	10 triệu	1

Bạn cần làm gì?

1. **Giới tính** → chuyển thành biến giả: giữ "Giới_tính_Nam"
2. **Thu nhập NA** → giữ nguyên (không thay thế)
3. Dùng XGBoost để học mô hình → mô hình vẫn hoạt động tốt

KẾT LUẬN

Đặc điểm	Mô tả	Xử lý như thế nào
Biến phân loại	Không hỗ trợ trực tiếp	Dùng dummy variables, tránh trap
NA (giá trị thiếu)	XGBoost tự xử lý	Không cần thay thế
Phi tuyến	Học tốt các mối quan hệ phi tuyến	Không cần biến đổi thủ công

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Bẫy biến giả(Dummy Variable Trap) và Các Bước Tiền Xử Lý Dữ Liệu Trong Machine Learning

MỤC TIÊU BÀI HỌC

- Hiểu rõ hiện tượng **Dummy Variable Trap** là gì.
- Biết cách xử lý biến phân loại khi dùng trong Machine Learning.
- Nắm được các bước khởi đầu để chuẩn bị dữ liệu huấn luyện cho mô hình (Step-by-step).

I. Bẫy biến giả(DUMMY VARIABLE TRAP) LÀ GÌ?

Khái niệm:

Biến giả(**Dummy Variable):** là các biến nhị phân (chỉ nhận giá trị 0 hoặc 1) được tạo ra từ biến phân loại (categorical variable).

Bẫy biến giả xảy ra khi **có quá nhiều biến giả liên quan đến cùng một đặc tính**, gây ra hiện tượng **đa cộng tuyến hoàn hảo** (perfect multicollinearity), dẫn đến lỗi hoặc mô hình hoạt động không chính xác.

Ví dụ cụ thể:

Giả sử bạn có biến **“Thương hiệu nước ngọt ưa thích”** với 3 giá trị:

- Coca-Cola
- Pepsi
- 7Up

Bạn tạo các biến giả:

CocaCola	Pepsi	7Up
1	0	0
0	1	0
0	0	1
1	0	0

Bây giờ nếu bạn **dùng cả 3 biến giả này cùng lúc**, thì một biến có thể **dự đoán được** từ hai biến còn lại:

```
7Up = 1 - CocaCola - Pepsi
```

Điều này gây ra **đa cộng tuyến hoàn hảo** → mô hình học máy như hồi quy tuyến tính hoặc một số thuật toán sẽ **bị lỗi hoặc giảm hiệu suất**.

Vậy nên làm gì?

Giải pháp: **Loại bỏ 1 biến giả** → gọi là "biến chuẩn" hay **baseline**.

Ví dụ:

- Giữ lại: `CocaCola`, `Pepsi`
- Loại bỏ: `7Up`

Thông tin của `7Up` vẫn còn:
Khi cả CocaCola và Pepsi đều bằng 0 → người đó thích 7Up.

TẠI SAO KHÔNG MẤT THÔNG TIN?

Không mất thông tin vì:

- Biến bị loại bỏ (baseline) vẫn có thể suy ra từ các biến còn lại.
- Trong hồi quy: thông tin này được mã hóa trong **hệ số chặn (intercept)**.
- Trong XGBoost: baseline là điểm so sánh gốc để các cây quyết định phân tách.

TÓM TẮT PHẦN 1

Vấn đề	Giải pháp
Dummy variable trap	Loại bỏ 1 biến giả để tránh đa cộng tuyến
Mất thông tin?	□ Không → thông tin được mã hóa ở baseline

II. BẮT ĐẦU HƯỚNG DẪN TỪNG BƯỚC TIỀN XỬ LÝ

Bước 1: Xác định bài toán & Tạo bộ dữ liệu

Phải có giả thuyết/hướng nghiên cứu rõ ràng (hypothesis-driven)

- Mỗi bài toán nên tự xây dựng dataset phù hợp thay vì chỉ lấy sẵn từ nguồn khác.

> Ví dụ: Nếu bạn muốn dự đoán liệu khách hàng có mua hàng không, bạn cần xác định những yếu tố nào có thể ảnh hưởng đến hành vi này (tuổi, thu nhập, kênh tiếp cận,...)

Bước 2: Chuyển đổi biến phân loại thành biến giả (dummy variables)

- **Sử dụng thư viện như `pandas.get_dummies()` trong Python.**
- **Nhớ tránh Dummy Variable Trap bằng cách loại bỏ 1 biến**

Ví dụ

```
import pandas as pd

df = pd.DataFrame({
    'drink': ['CocaCola', 'Pepsi', '7Up', 'CocaCola']
})

dummies = pd.get_dummies(df['drink'], drop_first=True)
print(dummies)
```

Kết quả:

Pepsi	7Up
0	0
1	0
0	1
0	0

CocaCola là baseline.

Bước 3: Chia dữ liệu thành tập huấn luyện và kiểm tra

- Tập huấn luyện (train): ~70–80%
- Tập kiểm tra (test): ~20–30%

```
from sklearn.model_selection import train_test_split

X = df_features  # tập biến đầu vào
y = df_target    # biến mục tiêu

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

KẾT LUẬN

Bước	Mô tả
1	Xác định giả thuyết và tạo dữ liệu phù hợp
2	Chuyển đổi các biến phân loại thành dummy, tránh dummy trap
3	Chia dữ liệu thành tập huấn luyện và kiểm tra

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Hiểu về RMSE – Root Mean Squared Error trong XGBoost

1. RMSE là gì?

RMSE (Root Mean Squared Error) – là **căn bậc hai của trung bình bình phương sai số**, dùng để đo **mức độ sai lệch giữa giá trị thực tế và giá trị dự đoán của mô hình**.

> “ □□ RMSE càng thấp → mô hình càng chính xác.

2. Khi nào dùng RMSE?

RMSE thường được dùng khi bạn giải quyết bài toán **hồi quy** – tức là khi **biến mục tiêu (Y)** là **liên tục**, ví dụ:

- Dự đoán giá nhà
- Dự đoán doanh thu
- Dự đoán nhiệt độ, v.v.

3. Công thức RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Trong đó:

- y_i: Giá trị thực tế
- $\hat{y}_i$: Giá trị dự đoán
- n: Tổng số điểm dữ liệu

4. Trực quan hóa RMSE

Hãy tưởng tượng:

- Bạn có trục X và trục Y.
- Đường thẳng màu xanh biểu diễn **mô hình dự đoán**.

- Các **dấu chấm** là **giá trị thực tế**.
- **Mũi tên từ chấm tới đường thẳng** chính là **sai số (error)**.
- RMSE là **trung bình độ dài của các mũi tên đó**, rồi lấy căn bậc hai.

5. Ví dụ cụ thể

Giả sử bạn đang dự đoán điểm thi của học sinh, và bạn có:

Học sinh	Điểm thực tế (Y)	Điểm dự đoán ($\hat{Y}$)
A	8.0	7.5
B	6.5	7.0
C	9.0	8.5

Tính RMSE:

$$\text{RMSE} = \sqrt{\frac{(8.0-7.5)^2 + (6.5-7.0)^2 + (9.0-8.5)^2}{3}} = \sqrt{\frac{0.25+0.25+0.25}{3}} = \sqrt{0.25} = 0.5$$

👉 **RMSE = 0.5 điểm**, tức là mô hình sai lệch trung bình 0.5 điểm so với thực tế.

6. RMSE bao nhiêu là tốt?

Không có "một con số tuyệt đối" nào cho RMSE tốt, vì:

- Phụ thuộc vào **phạm vi dữ liệu** (ví dụ: sai lệch 10 là lớn nếu điểm thi từ 0–10, nhưng nhỏ nếu doanh thu tính bằng triệu đô).
- Phụ thuộc vào **độ khó của bài toán**.

→ □□ Hãy so sánh RMSE của **mô hình hiện tại với các mô hình khác** để đánh giá.

7. RMSE trong XGBoost

XGBoost tự động **tối ưu hóa để giảm RMSE** (hoặc một hàm lỗi tương tự) trong quá trình huấn luyện khi bạn dùng cho bài toán hồi quy.

Tóm tắt:

Nội dung	Ý nghĩa
RMSE	Đo sai số trung bình giữa thực tế và dự đoán
Thích hợp cho	Bài toán hồi quy

Nội dung	Ý nghĩa
RMSE nhỏ hơn → mô hình tốt hơn	Đúng, nhưng cần phụ thuộc vào ngữ cảnh
Dùng để	So sánh giữa các mô hình

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Overfitting và Underfitting

Khi xây dựng mô hình học máy (Machine Learning), một trong những **thách thức lớn nhất** là **làm sao để mô hình học "vừa đủ" từ dữ liệu**. Nếu mô hình học **quá ít** hoặc **quá nhiều**, ta sẽ gặp phải 2 hiện tượng:

- **Underfitting** (quá đơn giản)
- **Overfitting** (quá phức tạp)

1. Trực quan hóa: Dữ liệu và mô hình

Dữ liệu:

Giả sử bạn có một tập dữ liệu đơn giản như sau (trên đồ thị):

- **Trục X**: Biến đầu vào
- **Trục Y**: Biến cần dự đoán (target)
- Các **dấu chấm**: dữ liệu thực tế

Ba mô hình khác nhau:

Mô hình	Đặc điểm	Loại lỗi
Đường thẳng đơn giản	Không mô tả hết xu hướng dữ liệu	Underfitting
Đường cong cực phức tạp	Đi qua tất cả các điểm dữ liệu	Overfitting
Đường cong vừa phải	Bắt đúng xu hướng, chấp nhận một số sai lệch	Tối ưu

2. UNDERFITTING là gì?

- Mô hình quá đơn giản → không học được mối quan hệ trong dữ liệu
- **Biểu hiện**:
  - Sai số cao trên cả **training set** và **test set**
- **Nguyên nhân**:
  - Dùng mô hình quá đơn giản (VD: hồi quy tuyến tính cho quan hệ phi tuyến)
  - Không đủ thời gian huấn luyện
  - Sử dụng quá ít đặc trưng (features)

Ví dụ:

> Bạn dùng một đường thẳng để mô tả quan hệ giữa lượng học và điểm thi, nhưng dữ liệu thực tế cho thấy quan hệ cong. Mô hình bỏ qua xu hướng cong → sai số cao → underfitting.

3. OVERFITTING là gì?

- Mô hình quá phức tạp → học cả **nhiễu (noise)** trong dữ liệu
- **Biểu hiện**:
  - Sai số thấp trên **training set** nhưng cao trên **test set**
- **Nguyên nhân**:
  - Mô hình quá phức tạp
  - Dữ liệu huấn luyện không đủ đa dạng
  - Không có regularization (phạt phức tạp)

Ví dụ:

> Bạn cho mô hình học "thuộc lòng" từng điểm dữ liệu học sinh (ví dụ: điểm, tuổi, chiều cao), mô hình sẽ dự đoán đúng tuyệt đối trên tập huấn luyện nhưng sai lệch nghiêm trọng khi gặp dữ liệu mới.

Ví dụ về Bias (Underfitting):

> Khi còn nhỏ, bạn nghĩ rằng mọi vấn đề đều có thể giải quyết theo một cách duy nhất, do bạn **thiếu trải nghiệm và hiểu biết**. Bạn giống như một cái **búa** và nghĩ mọi vấn đề đều là **cái đinh** → đó là **thiên lệch (bias) cao**.

Ví dụ về Variance (Overfitting):

> Bạn có 40 loại gia vị trong nhà bếp, và cố trộn tất cả lại để tạo ra món ăn "hoàn hảo". Nhưng thay vì ngon, món ăn lại trở thành **lộn xộn** → quá nhiều lựa chọn, dẫn đến lỗi cao → **phương sai (variance) cao**.

Kết luận

Vấn đề	Biểu hiện	Giải pháp chính
Underfitting	Sai số cao ở cả train/test	Mô hình phức tạp hơn, nhiều feature

Vấn đề	Biểu hiện	Giải pháp chính
Overfitting	Sai số thấp ở train, cao ở test	Giảm độ phức tạp, dùng cross-validation
Mục tiêu	Mô hình cân bằng	Tối ưu bias-variance trade-off

Bias - Variance Tradeoff

1. Định nghĩa cơ bản

Bias (Thiên lệch)

- Bias là **độ lệch giữa dự đoán của mô hình và giá trị thực tế**.
- Bias **cao** xảy ra khi mô hình **quá đơn giản**, không thể học đúng mối quan hệ trong dữ liệu.
- **Hậu quả**: Dự đoán sai lệch có hệ thống.

◫ Ví dụ:

> Bạn dùng một đường thẳng để dự đoán kết quả học tập, trong khi mối quan hệ thực sự là phi tuyến (cong). Mô hình không đủ khả năng để học mối quan hệ này → kết quả luôn sai → bias cao.

Variance (Phương sai)

- Variance là **mức độ mà dự đoán của mô hình thay đổi khi dữ liệu huấn luyện thay đổi**.
- Variance **cao** xảy ra khi mô hình **quá nhạy cảm với dữ liệu huấn luyện**, học thuộc lòng dữ liệu.
- **Hậu quả**: Mô hình kém ổn định và dự đoán kém trên dữ liệu mới.

◫ Ví dụ:

> Một mô hình Decision Tree cực kỳ sâu có thể dự đoán chính xác tất cả điểm huấn luyện, nhưng khi gặp dữ liệu mới sẽ hoạt động rất tệ vì không có khả năng tổng quát.

2. Tổng lỗi (Total Error)

Khi huấn luyện mô hình, **Tổng lỗi = Bias² + Variance + Noise**

- **Bias²**: lỗi do mô hình đơn giản hóa quá mức
- **Variance**: lỗi do mô hình quá phức tạp
- **Noise**: yếu tố ngẫu nhiên trong dữ liệu không thể tránh

3. Tradeoff – Tại sao cần đánh đổi?

- **Không thể cùng lúc đạt Bias thấp và Variance thấp**
- Nếu giảm Bias (mô hình phức tạp hơn) → tăng Variance
- Nếu giảm Variance (mô hình đơn giản hơn) → tăng Bias

Mục tiêu:

> Tìm mô hình vừa đủ — không quá đơn giản cũng không quá phức tạp → Tổng lỗi thấp nhất

Ví dụ thực tế minh họa

Ví dụ 1: Học sinh làm bài toán

- **Bias cao**:
  - Học sinh mới học lớp 1, chỉ biết cộng trừ → gặp bài nâng cao thì làm sai hoàn toàn
- **Variance cao**:
  - Học sinh học thuộc lòng đáp án → đổi đề một chút là không biết làm

Học sinh giỏi sẽ hiểu bản chất, không học vẹt → Bias thấp, Variance thấp

Ví dụ 2: Nấu ăn với gia vị

- **Bias cao**: chỉ dùng mỗi muối và tiêu → món ăn luôn nhạt → đơn giản hóa quá mức
- **Variance cao**: có 50 loại gia vị, trộn lung tung mỗi lần khác nhau → không ổn định, vị không lặp lại

Người nấu ăn giỏi biết **chọn lọc gia vị hợp lý** → tạo ra món ngon ổn định

4. Làm sao để giải quyết Tradeoff?

Kỹ thuật	Tác dụng
Cross-validation	Ước lượng lỗi thực tế để tránh overfitting
Regularization (L1/L2)	Phạt độ phức tạp để giảm variance
Giảm chiều dữ liệu	Loại bỏ feature gây nhiễu
Ensemble methods	Kết hợp nhiều mô hình để giảm variance
Tăng dữ liệu	Giảm overfitting nhờ thêm dữ liệu mới

Tối ưu hóa mô hình XGBoost & Bias-Variance Tradeoff

Mục tiêu bài học

- Hiểu các tham số cần tinh chỉnh trong mô hình XGBoost.
- Nắm được ý nghĩa của từng tham số và ảnh hưởng của chúng đến **Bias** và **Variance**.
- Hiểu khái niệm và vai trò của **Cross-Validation** trong việc chọn tham số tối ưu.

I. Các tham số cần tinh chỉnh trong XGBoost

> Dưới đây là 8 tham số phổ biến ảnh hưởng đến hiệu suất của mô hình:

---

1. `n_estimators` / `num_boost_round` – Số vòng boosting

- Mô tả: Số cây (trees) được huấn luyện trong mô hình.
- Tăng quá cao → mô hình overfitting (variance cao).
- Quá thấp → mô hình underfitting (bias cao).

Ví dụ:

```
xgb.XGBClassifier(n_estimators=100)  # Thử 100 vòng boosting
```

2. `eta` – Learning Rate (tốc độ học)

- Mô tả: Mỗi cây đóng góp bao nhiêu vào kết quả cuối cùng.
- Giá trị thấp → mô hình học chậm → variance cao.
- Giá trị cao → mô hình học nhanh → dễ bị bias cao.

Gợi ý: Dùng `eta` thấp (0.01–0.3) + tăng số vòng.

Ví dụ:

```
xgb.XGBClassifier(eta=0.1)  # Tốc độ học vừa phải
```

3. `min_child_weight` – Trọng số tối thiểu cho mỗi leaf

- Mô tả: Quy định ngưỡng tối thiểu để tách cây.
- Ngăn không cho cây học từ các quan sát không đủ "trọng số".

Ví dụ:

```
xgb.XGBClassifier(min_child_weight=5)  # Tránh cây quá phức tạp
```

4. `max_depth` – Độ sâu tối đa của cây

- Mô tả: Cây càng sâu càng học kỹ → dễ overfitting.
- Tăng `max_depth` nếu `eta` thấp.

Ví dụ:

```
xgb.XGBClassifier(max_depth=6)  # Cây vừa phải
```

5. `gamma` – Ngưỡng để tách (split) cây

- Mô tả: Cây chỉ tách nếu đem lại lợi ích vượt qua gamma.
- Giá trị lớn → khó chia → giảm overfitting.

Ví dụ:

```
xgb.XGBClassifier(gamma=0.2)  # Thận trọng khi chia
```

6. `subsample` – Tỷ lệ mẫu lấy ra từ tập dữ liệu

- Mô tả: Ngẫu nhiên lấy 1 phần dữ liệu cho mỗi vòng.
- Giá trị thấp → giúp giảm variance.
- Giá trị cao (≈ 1.0) → học toàn bộ dữ liệu.

Ví dụ:

```
xgb.XGBClassifier(subsample=0.8)
```

7. `colsample_bytree` – Tỷ lệ cột sử dụng cho mỗi cây

- Mô tả: Giảm số lượng đặc trưng (features) dùng để huấn luyện mỗi cây.
- Tác dụng tương tự như `subsample` nhưng trên **features**.

Ví dụ:

```
xgb.XGBClassifier(colsample_bytree=0.7)
```

8. Tổng quan ảnh hưởng đến Bias vs. Variance

Tham số	Giá trị thấp → Ảnh hưởng	Giá trị cao → Ảnh hưởng
`eta`	Variance ↑	Bias ↑
`max_depth`	Bias ↑	Variance ↑
`min_child_weight`	Variance ↑	Bias ↑
`gamma`	Variance ↑	Bias ↑
`subsample`	Variance ↑	Bias ↑
`colsample_bytree`	Variance ↑	Bias ↑

II. Kỹ thuật Cross-Validation – Tối ưu tham số

Mục tiêu:

> Tìm **bộ tham số tốt nhất** bằng cách **chia nhỏ dữ liệu huấn luyện** thành nhiều phần → thử nghiệm mô hình trên từng phần → trung bình kết quả.

Ý tưởng chính

- Chia dữ liệu huấn luyện thành **k phần (folds)**.
- Dùng (k-1) phần để huấn luyện, 1 phần để kiểm tra.
- Lặp lại k lần → lấy trung bình độ chính xác.

Ưu điểm:

- Đảm bảo mô hình không chỉ tốt trên một tập dữ liệu cụ thể.
- Giúp chọn ra tham số tổng quát hóa tốt.

Ví dụ: Sử dụng `GridSearchCV` để tìm tham số

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

params = {
    'max_depth': [3, 6],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.7, 1],
}
```

```
model = XGBClassifier()
grid = GridSearchCV(estimator=model, param_grid=params, cv=5)
grid.fit(X_train, y_train)


print("Best Params:", grid.best_params_)
```

Ghi nhớ

Vấn đề	Cách giải quyết
Overfitting (Variance cao)	Tăng `min_child_weight`, tăng `gamma`, giảm `max_depth`, `subsample`, `colsample_bytree`
Underfitting (Bias cao)	Giảm `min_child_weight`, giảm `gamma`, tăng `max_depth`, `n_estimators`

Tổng kết

- Bạn nên tinh chỉnh tham số qua **GridSearch hoặc RandomizedSearch kết hợp Cross-Validation**.
- Luôn cân nhắc **Bias – Variance Tradeoff** khi chọn giá trị.
- XGBoost hỗ trợ song song hóa → xử lý Cross-Validation rất hiệu quả.

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft