

Blog Tuần 3 – Module 3

Decision Tree, Excel và Cloud

Từ trực quan hóa mô hình đến phân tích dữ liệu trên nền tảng đám mây

Tác giả: GRID034

Tuần thứ ba của Module 3 mở rộng kiến thức theo hai hướng song song: một bên là **thuật toán học máy** tập trung vào cây quyết định và các kỹ thuật ensemble, bên kia là **công cụ hỗ trợ phân tích dữ liệu thực tiễn** như Excel và các dịch vụ Cloud. Sự kết hợp này giúp người học vừa nắm vững lý thuyết thuật toán, vừa rèn luyện kỹ năng thao tác và triển khai trong môi trường doanh nghiệp.

Cụ thể, blog tuần này sẽ bao gồm các chủ đề:

1. **Decision Tree – Cây quyết định** Cấu trúc, nguyên lý hoạt động, cách chọn thuộc tính phân chia dựa trên Gini Impurity và Entropy – Information Gain.
2. **Pruning và so sánh với KNN** Cách cắt tỉa cây để tránh overfitting, và điểm khác biệt cơ bản giữa Decision Tree và KNN.
3. **Excel trong phân tích dữ liệu** Các thao tác xử lý, trực quan hóa cơ bản, so sánh ưu – nhược điểm của Excel và Python, cũng như cách kết hợp cả hai để tối ưu công việc.
4. **Cloud Computing với AWS** Giới thiệu các dịch vụ trọng yếu: IAM, S3, Glue, Athena, CloudWatch. Minh họa quy trình xử lý, lưu trữ và phân tích dữ liệu trên nền tảng đám mây.

Giá trị nhận được sau khi đọc Blog

- Hiểu và triển khai được Decision Tree.
- Biết cách áp dụng pruning để cải thiện chất lượng mô hình.
- So sánh ưu – nhược điểm giữa Decision Tree và KNN.
- Sử dụng Excel song song với Python cho phân tích dữ liệu.
- Nắm vững nền tảng AWS cơ bản để xử lý dữ liệu trên Cloud.

Khám phá thuật toán Decision Tree: Từ lý thuyết đến các phương pháp nâng cấp

Vũ Thái Sơn

Hành trình tìm hiểu thuật toán ”hỏi hàng xóm” trong học máy

1. Giới thiệu: Khám phá Decision Tree

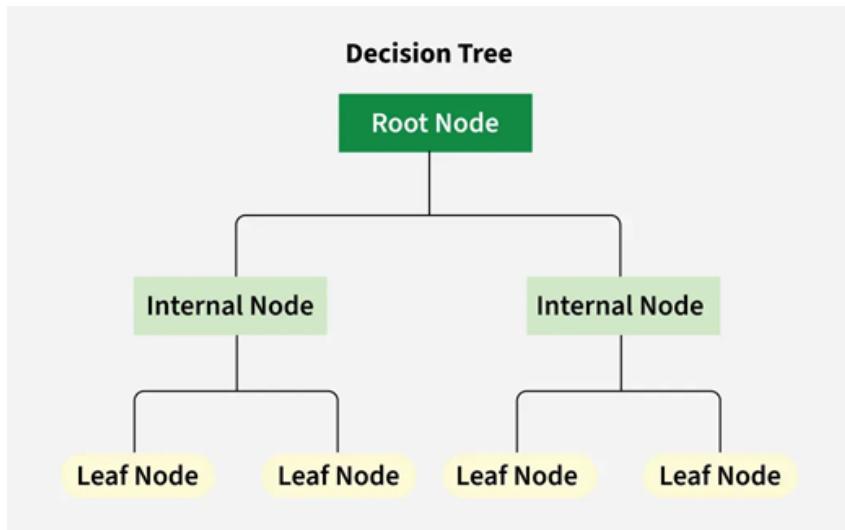
Decision Tree (Cây quyết định) là một trong những thuật toán nền tảng và dễ hiểu nhất trong lĩnh vực học máy. Hình dung đơn giản: Decision Tree mô phỏng quá trình con người đặt câu hỏi và đưa ra quyết định dựa trên các thuộc tính quan sát được. Khi học về cây quyết định, bạn sẽ thấy cách dữ liệu được phân loại từng bước, gần gũi như cách chúng ta suy nghĩ – từ tổng quát đến chi tiết, đến khi ra quyết định cuối cùng [1].

2. Quy tắc hoạt động của Decision Tree

2.1 Cấu trúc cây quyết định

Một cây quyết định bao gồm:

- **Nút gốc (Root Node):** Nút đầu tiên chứa toàn bộ dữ liệu.
- **Nút trong (Internal Node):** Đại diện cho các điều kiện hoặc câu hỏi trên từng thuộc tính.
- **Nút lá (Leaf Node):** Cho ta kết quả cuối cùng sau các phép phân tách.



Hình 1: Cấu trúc một cây quyết định đơn giản.

2.2 Cách thuật toán chọn nhánh (splitting)

Mục đích chia nhánh là làm các nút lá càng “thuần khiết” càng tốt, tức dữ liệu trong mỗi nút có cùng một lớp (hoặc giá trị dự đoán) nhiều nhất có thể.

Các phép đo mức độ thuần khiết:

- **Gini impurity:** Đo lường xác suất một mẫu bị gán nhãn sai nếu chọn ngẫu nhiên.

$$Gini(D) = 1 - \sum_{i=1}^k p_i^2$$

- **Entropy:** Đo lường tính ngẫu nhiên trong dữ liệu.

$$H(X) = - \sum_{i=1}^c p_i \log_2 p_i$$

Qua từng bước tách, thuật toán sẽ chọn thuộc tính giúp giảm độ không thuần khiết nhiều nhất.

3. Ứng dụng Decision Tree: Phân loại thực tế

Để hiểu rõ hơn về cách Decision Tree hoạt động, chúng ta hãy áp dụng nó vào một ví dụ thực tế. Giả sử chúng ta có một bộ dữ liệu về hành vi của khách hàng tiềm năng và muốn dự đoán liệu họ có đăng ký khóa học hay không. Dưới đây là bộ dữ liệu chi tiết hơn được sử dụng để xây dựng cây quyết định trong phần thực hành Python.

ID	Giờ xem	Số click	Webinar	Kết quả
1	1.5	3	Có	Có
2	2.0	1	Không	Không
3	0.5	5	Có	Có
4	3.0	2	Không	Không
5	1.0	4	Có	Có
6	2.5	1	Không	Không
7	1.8	4	Có	Có
8	2.4	2	Không	Có
9	2.9	3	Có	Không
10	1.2	2	Có	Không
11	0.8	3	Không	Không
12	2.6	5	Có	Có
13	0.7	2	Không	Không

Bảng 1: Bộ dữ liệu khách hàng mở rộng được dùng trong ví dụ này

3.1 Diễn giải quá trình phân tách của cây quyết định

Bước 1: Tính Gini Impurity của Nút Gốc

Nút gốc đại diện cho toàn bộ 13 mẫu dữ liệu. Chúng ta đếm được 7 khách hàng có kết quả **Không** (lớp 0) và 6 khách hàng có kết quả **Có** (lớp 1).

Chúng ta sử dụng công thức Gini Impurity để đo lường độ không thuần khiết của nút này.

$$Gini_{\text{root}} = 1 - \left(\left(\frac{7}{13} \right)^2 + \left(\frac{6}{13} \right)^2 \right) = 1 - \left(\frac{49}{169} + \frac{36}{169} \right) = 1 - \frac{85}{169} \approx 0.497$$

Giá trị $Gini \approx 0.497$ khớp với thông tin hiển thị tại nút gốc trên hình ảnh cây quyết định.

Bước 2: Chọn thuộc tính phân tách tối ưu: so_click ≤ 3.5

Thuật toán sẽ đánh giá tất cả các thuộc tính (gio_xem, so_click, webinar) để tìm ngưỡng phân tách giúp giảm Gini Impurity nhiều nhất.

Cây quyết định đã chọn ngưỡng so_click ≤ 3.5 là tối ưu nhất.

- Nhánh bên trái (điều kiện True):

- Bao gồm 9 mẫu dữ liệu có so_click ≤ 3.5 .
- Phân bố: 7 mẫu thuộc lớp **Không** và 2 mẫu thuộc lớp **Có**.
- Gini Impurity của nhánh này được tính là:

$$Gini_{\text{left}} = 1 - \left(\left(\frac{7}{9} \right)^2 + \left(\frac{2}{9} \right)^2 \right) = 1 - \left(\frac{49}{81} + \frac{4}{81} \right) = 1 - \frac{53}{81} \approx 0.346$$

- Nhánh bên phải (điều kiện False):

- Bao gồm 4 mẫu dữ liệu có so_click > 3.5 .
- Phân bố: 0 mẫu thuộc lớp **Không** và 4 mẫu thuộc lớp **Có**.
- Gini Impurity của nhánh này là:

$$Gini_{\text{right}} = 1 - \left(\left(\frac{0}{4} \right)^2 + \left(\frac{4}{4} \right)^2 \right) = 1 - (0 + 1) = 0$$

Bước 3: Tiếp tục phân tách

Vì nhánh bên trái ($Gini \approx 0.346$) vẫn chưa thuần khiết, quá trình phân tách tiếp tục. Thuật toán tìm ra điểm phân tách tối ưu tiếp theo là gio_xem ≤ 1.35 . Quá trình này sẽ lặp lại cho đến khi các nút lá đạt Gini Impurity bằng 0 hoặc đạt đến giới hạn độ sâu của cây (trong trường hợp này là max_depth = 3 như đã thiết lập trong mã nguồn).

4. Thực hành với Python

Mình họa xây dựng cây quyết định bằng scikit-learn và trực quan hóa.

```

1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier, plot_tree
3 import matplotlib.pyplot as plt
4
5 data = {
6     'gio_xem': [1.5, 2.0, 0.5, 3.0, 1.0, 2.5, 1.8, 2.4, 2.9, 1.2, 0.8, 2.6,
7     0.7],
8     'so_click': [3, 1, 5, 2, 4, 1, 4, 2, 3, 2, 3, 5, 2],

```

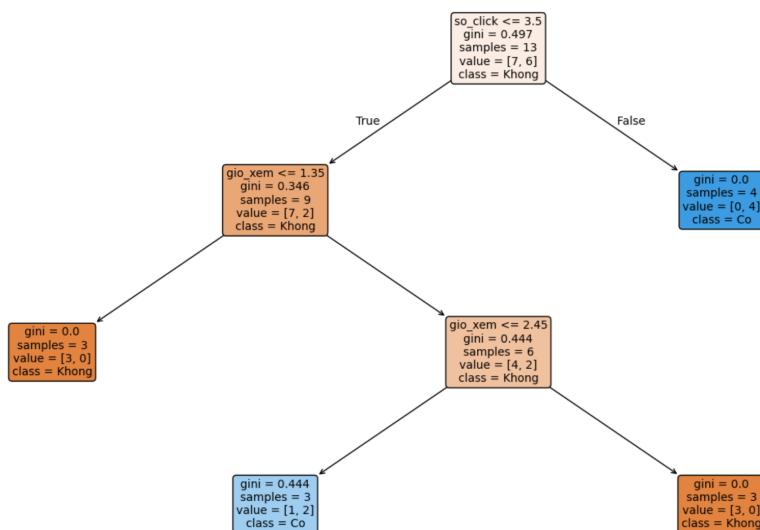
```

8     'webinar': [ 'Co', 'Khong', 'Co', 'Khong', 'Co', 'Khong', 'Co', 'Khong',
9      'Co', 'Co', 'Khong', 'Co', 'Khong'],
10     'ket_qua': [ 'Co', 'Khong', 'Co', 'Khong', 'Co', 'Khong', 'Co', 'Co',
11      'Khong', 'Khong', 'Co', 'Khong']
12 }
13 df = pd.DataFrame(data)
14 df[ 'webinar' ] = df[ 'webinar' ].map({ 'Co': 1, 'Khong': 0})
15 df[ 'ket_qua' ] = df[ 'ket_qua' ].map({ 'Co': 1, 'Khong': 0})
16 X = df[[ 'gio_xem', 'so_click', 'webinar']]
17 y = df[ 'ket_qua' ]
18
19 clf = DecisionTreeClassifier(max_depth=3, criterion='gini', random_state
20 =42)
21 clf.fit(X, y)
22
23 plt.figure(figsize=(14,8))
24 plot_tree(
25     clf,
26     feature_names=[ 'gio_xem', 'so_click', 'webinar'],
27     class_names=[ 'Khong', 'Co'],
28     filled=True,
29     rounded=True,
30     fontsize=10
31 )
32 plt.title("Decision tree: customer classification with many branches",
33             fontsize=14)
34 plt.tight_layout()
35 plt.show()

```

Listing 1: Xây dựng Decision Tree với scikit-learn

Cây quyết định: phân loại khách hàng với nhiều tầng nhánh



Hình 2: Trực quan cây quyết định trên dữ liệu khách hàng.

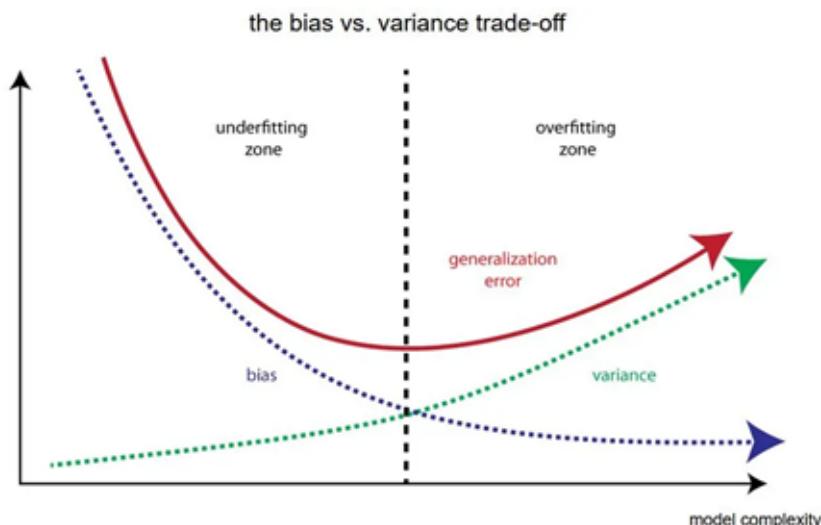
5. Ưu điểm và hạn chế của Decision Tree

5.1 Ưu điểm

- Dễ hiểu, dễ diễn giải: Sơ đồ cây trực quan hoá logic ra quyết định.
- Xử lý được cả dữ liệu số và phân loại mà không cần nhiều bước tiền xử lý.
- Áp dụng cho cả phân loại lẩn hồi quy.

5.2 Hạn chế

- Dễ bị overfitting với dữ liệu nhiều hoặc nhỏ – mô hình quá khớp với dữ liệu huấn luyện.
- Ít ổn định: Chỉ cần thay đổi nhỏ trong dữ liệu có thể tạo ra cây rất khác nhau.



Hình 3: Minh họa vấn đề Overfitting: cây quá phức tạp học thuộc dữ liệu huấn luyện.

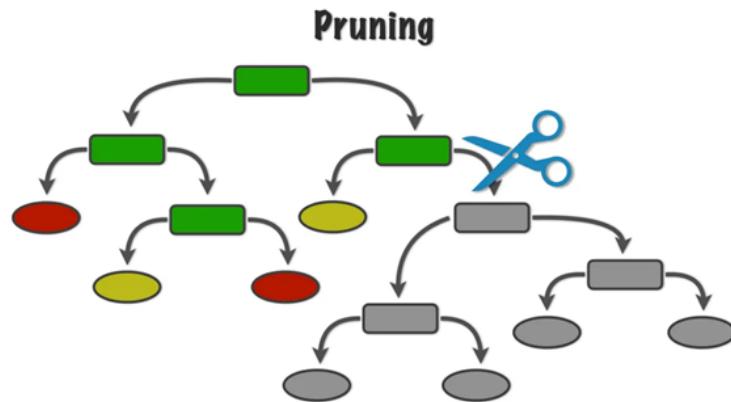
6. Cách khắc phục overfitting

Để kiểm soát độ phức tạp cây, có hai kỹ thuật chính:

- **Pre-Pruning:** Đặt giới hạn cho cây khi xây dựng bằng cách dùng các tham số như max_depth, min_samples_leaf, min_samples_split.
- **Post-Pruning:** Xây cây hoàn chỉnh rồi cắt bớt các nhánh không cần thiết dựa trên kiểm thử.

Ví dụ các tham số khi huấn luyện:

```
1 clf = DecisionTreeClassifier(max_depth=3, min_samples_leaf=2)
```



Hình 4: Hình minh họa cây quyết định trước và sau khi cắt tỉa.

7. Ứng dụng đa lĩnh vực

Decision Tree được sử dụng rộng rãi [1]:

1. **Y tế:** Hỗ trợ chẩn đoán bệnh từ các thuộc tính lâm sàng.
2. **Tài chính ngân hàng:** Đánh giá tín dụng, dự đoán khả năng vỡ nợ.
3. **Giáo dục:** Phân loại học viên tiềm năng, dự báo kết quả học tập.

8. So sánh Decision Tree với các thuật toán phân loại khác

Thuật toán	Mức độ trực quan	Hiệu năng với dữ liệu lớn	Độ dễ dùng
Decision Tree	Rất cao	TB	Rất dễ
KNN	TB	Thấp	TB
Logistic Regression	TB	Cao	TB

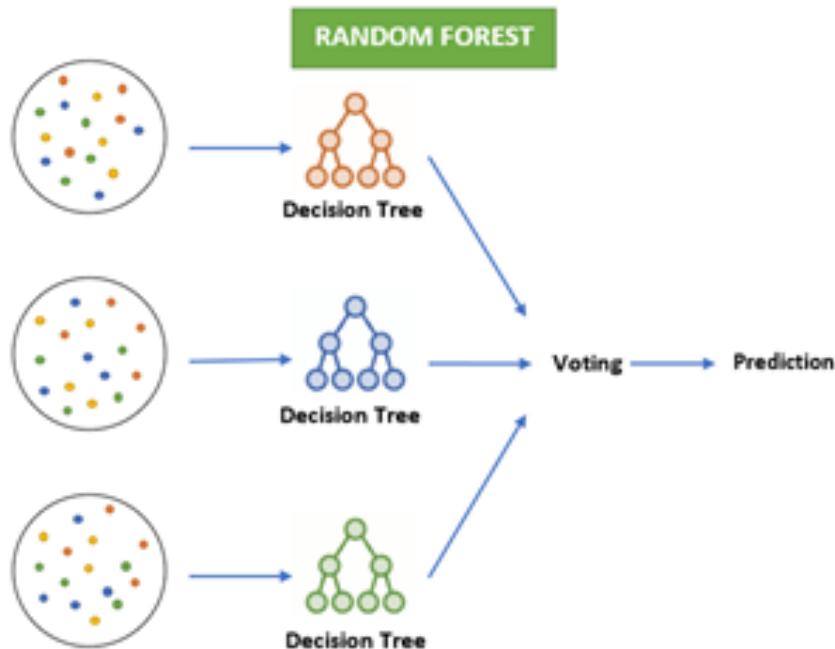
Bảng 2: So sánh Decision Tree với KNN, Logistic Regression

Nhận xét: - Decision Tree có ưu điểm lớn về tính trực quan và dễ hiểu nhất cho người mới học.

9. Các phương pháp nâng cấp: Ensemble cơ bản

9.1 Giới thiệu Random Forest

Random Forest kết hợp nhiều cây quyết định nhỏ (cây con), mỗi cây huấn luyện trên một tập dữ liệu con (bagging). Khi dự đoán, các cây cùng “bỏ phiếu”, kết quả là đa số phiếu hoặc trung bình. Phương pháp này giúp giảm overfitting, tăng tính ổn định & chính xác so với cây đơn lẻ [2].



Hình 5: Minh họa Random Forest và quá trình “bỏ phiếu” tổng hợp kết quả.

10. Giải thích mô hình: Tại sao Decision Tree “interpretable”

Cấu trúc Decision Tree cho phép bạn đọc dễ dàng lý giải vì sao một dự đoán được đưa ra (nhờ các bộ quy tắc từ gốc tới lá). Tính minh bạch này đặc biệt quan trọng trong các ngành tài chính, y tế, nơi cần giải thích rõ quyết định cho người dùng. Phương pháp hiện đại như SHAP, LIME còn cho phép giải thích cả các mô hình phức tạp hơn dựa trên nguyên lý Decision Tree.

11. Tổng kết

Decision Tree là bước khởi đầu lý tưởng cho người học AI. Bạn dễ dàng vừa học lý thuyết, vừa thực hành. Qua các ví dụ, bạn có thể ứng dụng vào nhiều lĩnh vực. Năm vững Decision Tree sẽ là nền móng vững chắc để tiếp cận các mô hình phát triển hơn như Random Forest[2], Gradient Boosting[3].

Tài liệu

- [1] A. Vietnam, “A simple guide to decision trees,” *Internal Study Materials*, 2025.
- [2] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [3] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

Khám Phá Decision Tree trong Bài Toán Phân Loại và Dự Đoán

Đàm Nguyễn Khánh

Tóm tắt nội dung

Decision Tree là một trong những mô hình học máy trực quan và dễ hiểu nhất, được sử dụng rộng rãi cho cả **phân loại (classification)** và **dự đoán (regression)**. Với cấu trúc dạng cây bao gồm **Root Node**, **Internal Nodes** và **Leaf Nodes**, mô hình này mô phỏng cách con người đưa ra quyết định dựa trên các điều kiện tuần tự.

Trong bài toán phân loại, Decision Tree lựa chọn thuộc tính phân chia tốt nhất dựa trên các chỉ số như **Gini Impurity** hoặc **Entropy – Information Gain**, giúp dữ liệu được tách thành những nhóm thuần khiết hơn. Trong bài toán dự đoán, mô hình sử dụng tiêu chí **Sum of Squared Residuals (SSR)** hoặc **Mean Squared Error (MSE)** để xây dựng hàm dự đoán bậc thang (*piecewise constant function*), từ đó ước lượng các giá trị liên tục như lương, giá nhà hay giá cổ phiếu.

Bên cạnh đó, các kỹ thuật mở rộng như **Pruning**, **Regularization** và các phương pháp **Ensemble** (Random Forest, Gradient Boosting, XGBoost) giúp mô hình khắc phục hạn chế về overfitting, cải thiện hiệu năng và mở rộng khả năng ứng dụng.

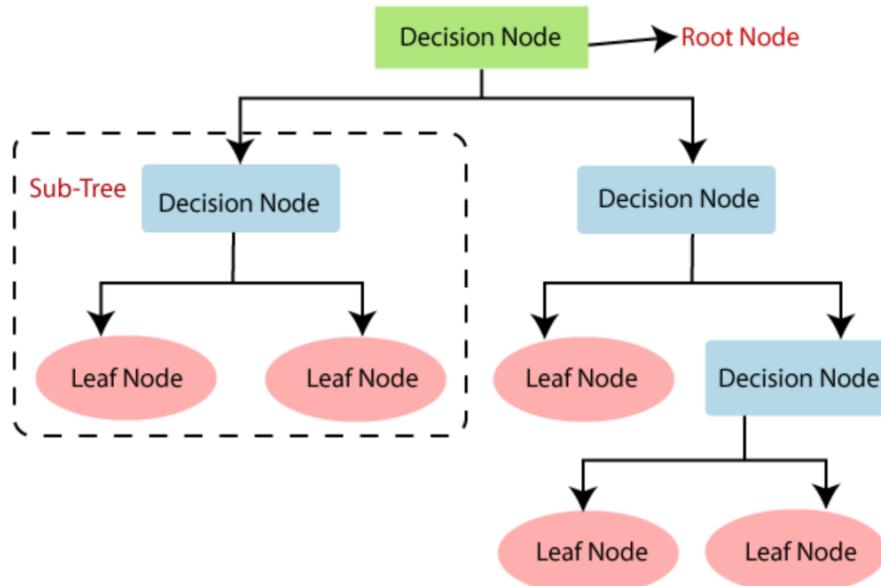
Nhờ vào sự minh bạch trong cách ra quyết định, Decision Tree không chỉ mạnh mẽ trong học máy mà còn đặc biệt hữu ích ở những lĩnh vực cần tính giải thích cao như **y tế**, **tài chính**, **pháp lý**.

1. Mở đầu

Trong học máy (Machine Learning), bài toán phân loại là một trong những ứng dụng quan trọng nhất, từ nhận diện khuôn mặt, phân loại email spam đến phát hiện gian lận tín dụng.

Một trong những mô hình phổ biến, trực quan và dễ hiểu nhất để giải quyết phân loại là **Decision Tree (Cây quyết định)**.

Không giống như KNN (K-Nearest Neighbors) – vốn đòi hỏi tính toán khoảng cách trên toàn bộ dữ liệu dẫn đến tốc độ chậm khi dataset lớn – Decision Tree cho phép **chia nhỏ dữ liệu thành các tập con** dựa trên các thuộc tính, từ đó đưa ra quyết định nhanh chóng.

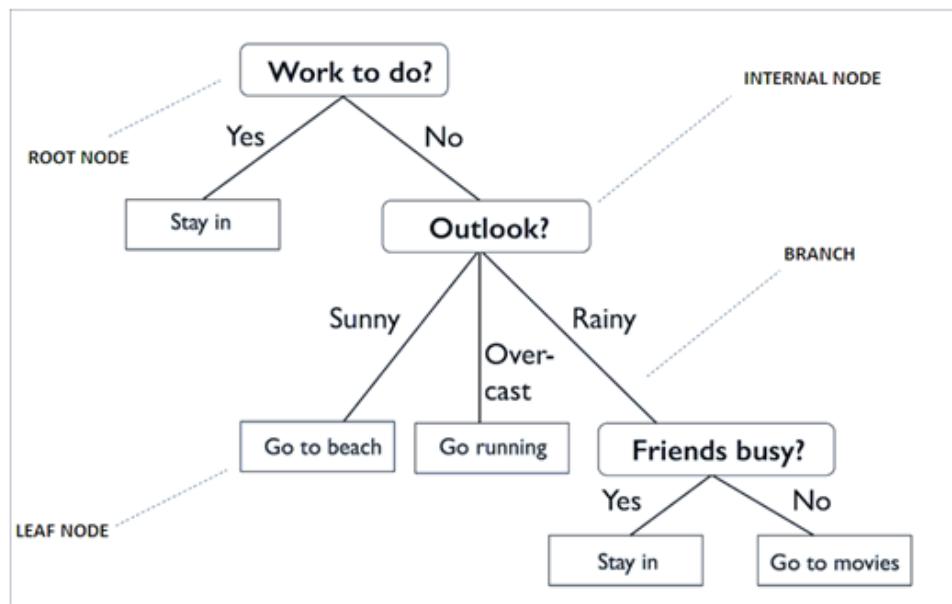


Hình 6: Cấu trúc cơ bản của một cây quyết định: **Root Node** (nút gốc) là điểm bắt đầu, các **Decision Node** (nút quyết định) phân nhánh dữ liệu dựa trên điều kiện, **Leaf Node** (nút lá) chứa nhãn phân loại cuối cùng, và các nhóm nhánh nhỏ có thể được xem như một **Sub-tree**.

2. Cấu trúc của Decision Tree

Một cây quyết định (*Decision Tree*) có cấu trúc phân cấp, được xây dựng từ các nút (*nodes*) và các nhánh (*branches*). Mỗi nhánh thể hiện một điều kiện chia dữ liệu, trong khi mỗi nút mang một vai trò cụ thể. Ba thành phần chính của cây quyết định bao gồm:

- **Root Node (Nút gốc):** Đây là điểm bắt đầu của toàn bộ cây, chứa **toàn bộ tập dữ liệu** ban đầu. Tại nút gốc, thuật toán sẽ lựa chọn một thuộc tính (feature) tốt nhất để chia dữ liệu thành các nhóm nhỏ hơn dựa trên chỉ số đánh giá như *Gini Impurity* hoặc *Entropy*. Việc chọn đúng Root Node quyết định chất lượng và độ chính xác của toàn bộ cây.
- **Internal Nodes (Nút trung gian):** Đây là các nút phân nhánh nằm giữa Root Node và các nút lá. Mỗi Internal Node chứa một điều kiện kiểm tra (ví dụ: *Age < 30?*, *Income = High?*). Khi dữ liệu đi qua nút này, nó sẽ được chia nhỏ hơn nữa thành các nhánh con. Quá trình phân chia tiếp tục cho đến khi dữ liệu đủ "thuần khiết" (ít hỗn tạp) hoặc đạt đến giới hạn độ sâu của cây.
- **Leaf Nodes (Nút lá):** Đây là điểm dừng của quá trình phân loại. Mỗi Leaf Node chứa **kết quả dự đoán cuối cùng** hoặc **nhãn phân loại** mà mô hình đưa ra (ví dụ: *Spam / Not Spam*, *Good Credit / Bad Credit*). Các Leaf Nodes không có nhánh con nào nữa, vì vậy chúng được coi là kết quả cuối cùng của đường đi trong cây.



Hình 7: Ví dụ minh họa cây quyết định trong thực tế: **Root Node** (nút gốc) bắt đầu bằng câu hỏi *Work to do?*, các **Internal Node** (nút trung gian) như *Outlook?* hay *Friends busy?* đưa ra điều kiện phân nhánh, **Branches** (nhánh) thể hiện các lựa chọn có thể, và **Leaf Nodes** (nút lá) đưa ra kết quả cuối cùng như *Stay in*, *Go to beach*, *Go running* hay *Go to movies*.

3. Làm thế nào để chọn thuộc tính tốt nhất?

3.1 Khái niệm Impurity (Độ hỗn tạp)

Trong một cây quyết định, mục tiêu của việc chia dữ liệu tại mỗi nút là làm cho các tập con trở nên **thuần khiết** hơn, tức là mỗi tập con chỉ chứa dữ liệu thuộc về một lớp duy nhất. Để đo lường mức độ "thuần khiết" hay "hỗn tạp" của dữ liệu tại một nút, người ta sử dụng khái niệm **Impurity (độ hỗn tạp)**.

Ý nghĩa

- Nếu tất cả các mẫu trong một nút đều thuộc cùng một lớp \Rightarrow độ hỗn tạp = 0 (nút thuần khiết).
- Nếu các mẫu được chia đều cho nhiều lớp \Rightarrow độ hỗn tạp cao (khó phân loại).
- Mục tiêu khi xây dựng cây: **chọn thuộc tính giúp giảm độ hỗn tạp nhiều nhất.**

Hai chỉ số phổ biến dùng để đo độ hỗn tạp là:

- **Gini Impurity:** đo xác suất chọn ngẫu nhiên một phần tử bị phân loại sai.

$$Gini = 1 - \sum_{i=1}^k p_i^2$$

Trong đó p_i là xác suất phần tử thuộc lớp i . Gini = 0 nghĩa là nút hoàn toàn thuần khiết.

- **Entropy – Information Gain:** Entropy đo độ hỗn loạn của dữ liệu:

$$\text{Entropy} = - \sum_{i=1}^k p_i \log_2 p_i$$

Dựa trên Entropy, ta tính được **Information Gain (IG)** – mức độ giảm Entropy khi tách dữ liệu bằng một thuộc tính. Thuộc tính nào cho IG cao nhất sẽ được chọn làm Root Node.

3.2 Gini Impurity

Công thức tính:

$$Gini = 1 - \sum_{i=1}^k p_i^2$$

Trong đó p_i là xác suất dữ liệu thuộc về lớp i .

- Gini = 0: dữ liệu thuần khiết (chỉ 1 lớp).
- Gini cao: dữ liệu lẫn lộn nhiều lớp.

3.3 Entropy và Information Gain

Entropy là một khái niệm xuất phát từ lý thuyết thông tin, dùng để đo mức độ hỗn tạp hoặc sự không chắc chắn của dữ liệu tại một nút trong cây quyết định. Nếu dữ liệu càng lẫn lộn giữa nhiều lớp thì giá trị Entropy càng cao, ngược lại nếu dữ liệu chỉ chứa một lớp duy nhất thì Entropy bằng 0.

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

Trong đó:

- k : số lớp trong tập dữ liệu.
- p_i : xác suất một phần tử trong tập S thuộc lớp i .

Ý nghĩa của Entropy

- Entropy = 0: dữ liệu hoàn toàn thuần khiết (chỉ có 1 lớp).
- Entropy cao: dữ liệu hỗn tạp, khó phân loại.
- Entropy đạt cực đại khi các lớp có phân phối đồng đều (ví dụ: 50% lớp A, 50% lớp B).

Information Gain (IG) là chỉ số thể hiện mức độ giảm Entropy sau khi tách dữ liệu theo một thuộc tính nào đó. Nói cách khác, IG đo lường “thông tin mới” mà một thuộc tính mang lại cho việc phân loại.

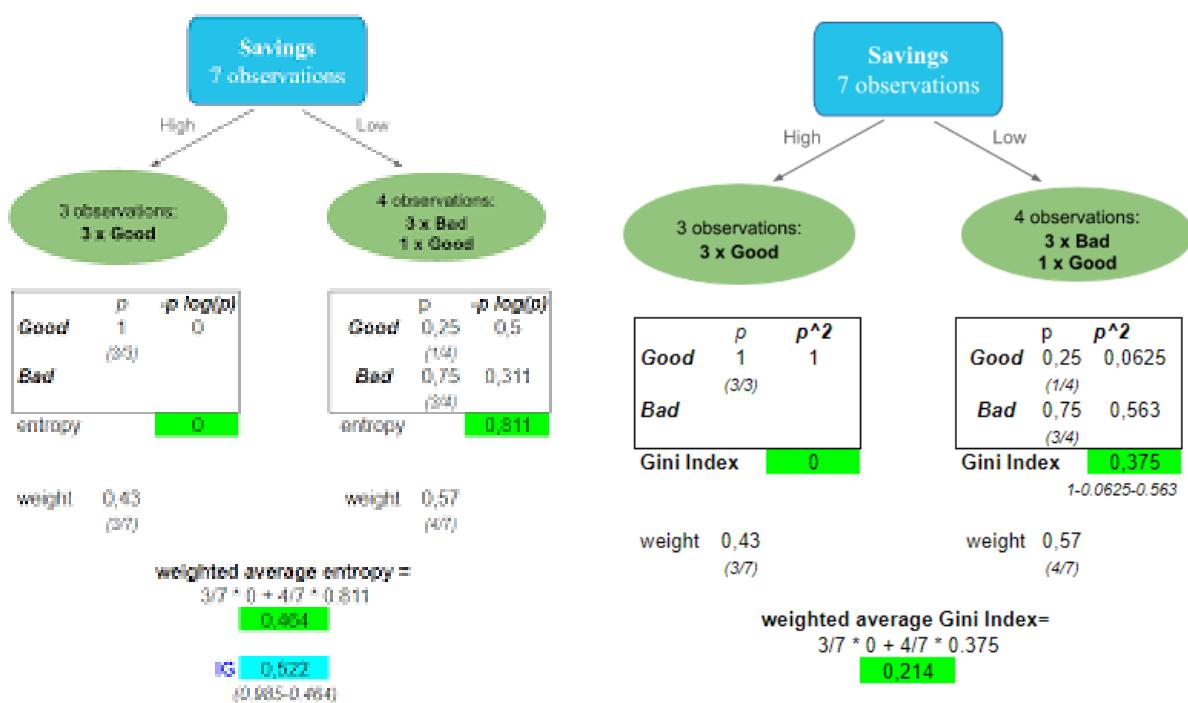
$$IG(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

Trong đó:

- S : tập dữ liệu ban đầu.
- A : thuộc tính được chọn để chia.
- S_v : tập con của S khi $A = v$.
- $\frac{|S_v|}{|S|}$: tỷ lệ kích thước tập con.

Ý nghĩa của Information Gain

- IG càng cao \Rightarrow thuộc tính càng “tốt” cho việc chia dữ liệu.
- Root Node được chọn là thuộc tính có IG cao nhất.



(a) Tính toán Entropy và Information Gain cho thuộc tính *Savings*

(b) Tính toán Gini Index cho thuộc tính *Savings*

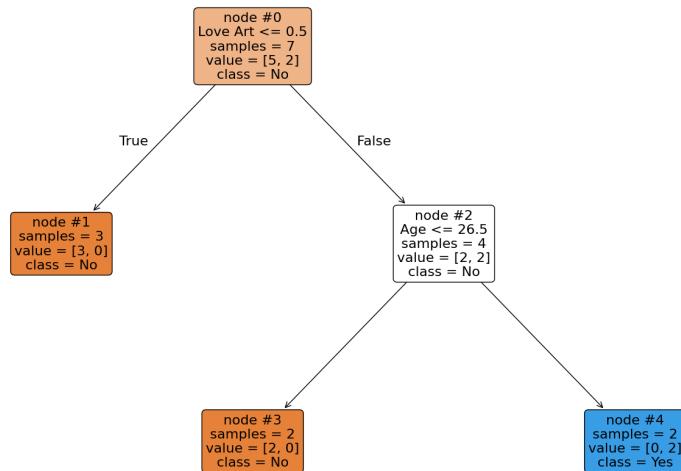
Hình 8: So sánh hai phương pháp chọn thuộc tính: Entropy (Information Gain) và Gini Index. Cả hai đều đo lường độ hỗn tạp (impurity) để xác định nút gốc tốt nhất, tuy nhiên cách tính khác nhau: Entropy dùng logarit trong khi Gini dựa trên xác suất bình phương.

4. Xây dựng Decision Tree – Ví dụ minh họa

Giả sử ta có dataset nhỏ về sở thích:

Love Math	Love Art	Age	Love AI
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Sau khi tính toán, ta thấy “Love Art” là Root Node tốt nhất. Tiếp theo, phân chia theo “Age < 26.5” để đưa ra quyết định cuối cùng.



Hình 9: Cây quyết định huấn luyện từ dữ liệu nhỏ về sở thích: nút gốc là thuộc tính *Love Art*, sau đó tiếp tục phân chia theo *Age*. Các nút hiển thị số mẫu (samples), phân bố giữa hai lớp (value), và nhãn dự đoán cuối cùng (class).

Cây quyết định được huấn luyện từ dataset nhỏ về sở thích. Dưới đây là diễn giải chi tiết cho từng node trong cây:

- **Root Node (node #0):**

- Điều kiện tách: Love Art ≤ 0.5 . Đây là kết quả của OneHotEncoder, trong đó cột Love Art_Yes được mã hoá thành 0/1 và ngưỡng 0.5 để phân tách Yes/No.
- Ý nghĩa: Nếu Love Art=No \Rightarrow đi nhánh trái, nếu Love Art=Yes \Rightarrow đi nhánh phải.
- Tổng số mẫu: 7 (5 “No”, 2 “Yes”).
- Nhãn dự đoán chính: **No** (đa số).

- **Node #1 (nhánh trái của Root):**

- Điều kiện: Love Art=No.
- Tổng số mẫu: 3, tất cả đều “No”.

- Đây là nút lá thuần khiết.
- Kết quả dự đoán: **No**.
- **Node #2 (nhánh phải của Root, tức Love Art=Yes):**
 - Tổng số mẫu: 4 (2 “No”, 2 “Yes”), rất hỗn tạp.
 - Cây tiếp tục chia bằng thuộc tính $\text{Age} \leq 26.5$.
 - Nhãn dự đoán tại node: **No** (do cân bằng, sklearn ưu tiên lớp đầu).
- **Node #3 (nhánh trái của Node #2):**
 - Điều kiện: $\text{Age} < 26.5$.
 - Tập con gồm 2 mẫu, cả hai đều “No”.
 - Nút lá thuần khiết, kết quả dự đoán: **No**.
- **Node #4 (nhánh phải của Node #2):**
 - Điều kiện: $\text{Age} \geq 26.5$.
 - Tập con gồm 2 mẫu, cả hai đều “Yes”.
 - Nút lá thuần khiết, kết quả dự đoán: **Yes**.

5. Vấn đề Overfitting và Giải pháp

Một trong những thách thức lớn khi sử dụng cây quyết định là hiện tượng **Overfitting** – mô hình học quá kỹ dữ liệu huấn luyện, ghi nhớ cả những chi tiết và nhiễu ngẫu nhiên thay vì nắm được quy luật tổng quát. Khi đó, cây quyết định có thể dự đoán chính xác trên tập huấn luyện nhưng lại hoạt động kém trên dữ liệu mới.

Dấu hiệu Overfitting

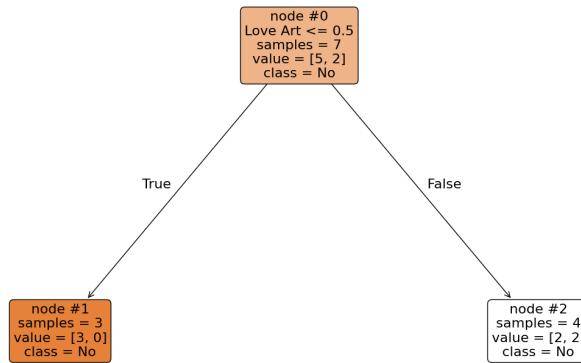
- Cây có quá nhiều tầng, nhánh phân chia phức tạp.
- Độ chính xác trên tập train cao bất thường, nhưng accuracy trên tập test thấp.
- Các nhánh cuối cùng chỉ còn rất ít mẫu (thậm chí 1 mẫu) nhưng vẫn được chia nhỏ.

Các giải pháp khắc phục

- **Pruning (Cắt tia):** Giới hạn độ sâu (`max_depth`) hoặc số node lá, loại bỏ các nhánh không quan trọng. Ví dụ: hình ?? cho thấy cây sau khi cắt tia (`max_depth=2`) trở nên gọn nhẹ hơn, giảm rủi ro overfitting.
- **Regularization:** Sử dụng các tham số điều chỉnh như `min_samples_split`, `min_samples_leaf`, hoặc `max_features` để kiểm soát việc phân chia. Ý tưởng: không cho phép một node được tách ra nếu số mẫu quá ít.
- **Ensemble Methods:** Thay vì dựa vào một cây duy nhất (dễ overfit), ta kết hợp nhiều cây:
 - **Random Forest:** xây nhiều cây độc lập, dựa trên mẫu bootstrap khác nhau, sau đó bỏ phiếu số đông.

- **Gradient Boosting / XGBoost:** xây các cây tuần tự, mỗi cây mới học từ lỗi của cây trước, dần cải thiện mô hình.

Cây quyết định sau khi cắt tỉa (max_depth=2)



Hình 10: Cây quyết định sau khi được **cắt tỉa (pruning)** với max_depth=2. Việc giới hạn độ sâu giúp giảm độ phức tạp của cây, tránh hiện tượng *overfitting*, đồng thời làm cho mô hình dễ hiểu và khái quát hơn.

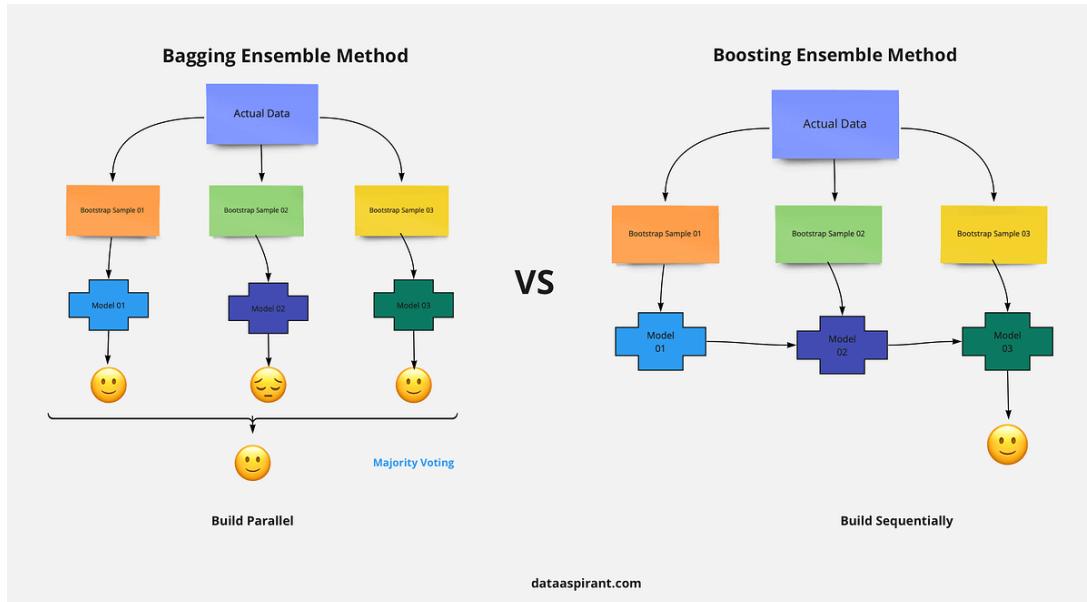
6. Ứng dụng thực tế

6.1 Phát hiện gian lận thẻ tín dụng

Phát hiện gian lận thẻ tín dụng là một trong những ứng dụng quan trọng và thực tế nhất của các mô hình phân loại. Các công ty tài chính cần nhanh chóng nhận diện các giao dịch bất thường để bảo vệ khách hàng và giảm thiểu thiệt hại.

- **Dataset:** hơn 280,000 giao dịch của khách hàng châu Âu (trong 2 ngày), trong đó chỉ ~0.17% là gian lận (492 mẫu gian lận). Đây là một tập dữ liệu **mất cân bằng nghiêm trọng**, vì số giao dịch hợp lệ quá lớn so với số giao dịch gian lận.
- **Thách thức:**
 - Nếu chỉ dự đoán tất cả đều là giao dịch hợp lệ, mô hình cũng đạt độ chính xác rất cao (hơn 99%), nhưng vô nghĩa về mặt thực tế.
 - Cần các thước đo khác ngoài accuracy, như **Precision**, **Recall**, **F1-score**, đặc biệt quan trọng là Recall (tỷ lệ phát hiện được giao dịch gian lận).
- **Giải pháp với Decision Tree và biến thể:**
 - **Decision Tree:** dễ giải thích, có thể phát hiện ra các quy luật từ đặc trưng (ví dụ: số tiền lớn bất thường, quốc gia hiếm gặp, tần suất giao dịch cao trong thời gian ngắn).
 - **Random Forest:** giảm overfitting, tăng khả năng khái quát hóa, hoạt động tốt khi số chiều dữ liệu lớn.

- **Gradient Boosting / XGBoost:** tối ưu dần trên lỗi của mô hình trước, đặc biệt hiệu quả trong bài toán dữ liệu mất cân bằng nhờ khả năng điều chỉnh trọng số (weights) cho các mẫu gian lận hiếm.



Hình 11: So sánh hai phương pháp Ensemble phổ biến: **Bagging** (ví dụ: Random Forest) xây dựng nhiều cây song song trên các mẫu bootstrap độc lập và kết hợp bằng *majority voting*; trong khi **Boosting** (ví dụ: AdaBoost, XGBoost) xây dựng các mô hình tuần tự, mỗi mô hình mới tập trung vào việc sửa lỗi của mô hình trước, từ đó cải thiện dần độ chính xác.

7. Decision Tree cho Dự đoán (Regression Tree)

Bên cạnh bài toán phân loại, **Decision Tree** còn có thể áp dụng cho các bài toán **dự đoán giá trị số liên tục** (*regression problems*). Trong trường hợp này, thay vì trả về một nhãn lớp (Yes/No, Spam/Ham, v.v.), cây quyết định dự đoán ra một giá trị số như *giá nhà*, *mức lương*, *hiệu quả vaccine* hay *giá cổ phiếu*.

7.1 Nguyên lý hoạt động

- **Khác biệt chính so với Classification Tree:**
 - Với **Classification Tree**, tiêu chí chọn thuộc tính dựa trên *Gini Impurity* hoặc *Entropy*.
 - Với **Regression Tree**, tiêu chí chọn thuộc tính dựa trên *lỗi dự đoán số*, thường được đo bằng **Sum of Squared Residuals (SSR)** hoặc **Mean Squared Error (MSE)**.
- **Cách tính tại một node:**

$$SSR = \sum_{i=1}^n (y_i - \hat{y})^2$$

trong đó:

- y_i : giá trị thực tế của quan sát thứ i .
- \hat{y} : giá trị trung bình của tất cả y_i trong node đó.

- **Nguyên tắc tách node:** thuật toán sẽ duyệt qua tất cả các thuộc tính và các ngưỡng có thể, chọn ra cách tách giúp **giảm SSR nhiều nhất**. Nói cách khác, mô hình luôn cố gắng tìm ngưỡng phân chia sao cho dữ liệu trong mỗi nhánh có giá trị y càng đồng nhất càng tốt.

Trực quan hóa

Với bài toán hồi quy, mỗi lá (Leaf Node) không phải là một nhãn, mà là **một giá trị trung bình**. Khi có một mẫu mới đi qua cây, nó sẽ rơi vào một nút lá, và giá trị dự đoán chính là trung bình của các mẫu huấn luyện trong nút lá đó.

$$\hat{y}_{leaf} = \frac{1}{n_{leaf}} \sum_{i=1}^{n_{leaf}} y_i$$

7.2 Ví dụ minh họa

Ví dụ 1: Dự đoán hiệu quả vaccine. Giả sử ta có dữ liệu về *liều lượng tiêm (unit)*, *tuổi (age)* và *giới tính (sex)* của bệnh nhân, cùng hiệu quả vaccine đo được (

- Nếu chọn Unit làm Root Node, cây sẽ tách dữ liệu theo các ngưỡng của liều lượng.
- Ví dụ: khi Unit=5, mô hình dự đoán hiệu quả $\approx 44\%$, vì đó chính là giá trị trung bình của các mẫu trong nhánh này.

Ví dụ 2: Dự đoán lương (Salary Prediction). Một công ty muốn dự đoán mức lương dựa trên vị trí công việc (position level). Regression Tree sẽ phân chia các khoảng giá trị của position level (ví dụ: Junior, Mid, Senior) và gán cho mỗi khoảng một mức lương trung bình.

- Nếu position level = 3 (Mid-level), cây có thể dự đoán mức lương $\sim 50,000$ USD.
- Nếu position level = 7 (Senior), cây có thể dự đoán mức lương $\sim 120,000$ USD.

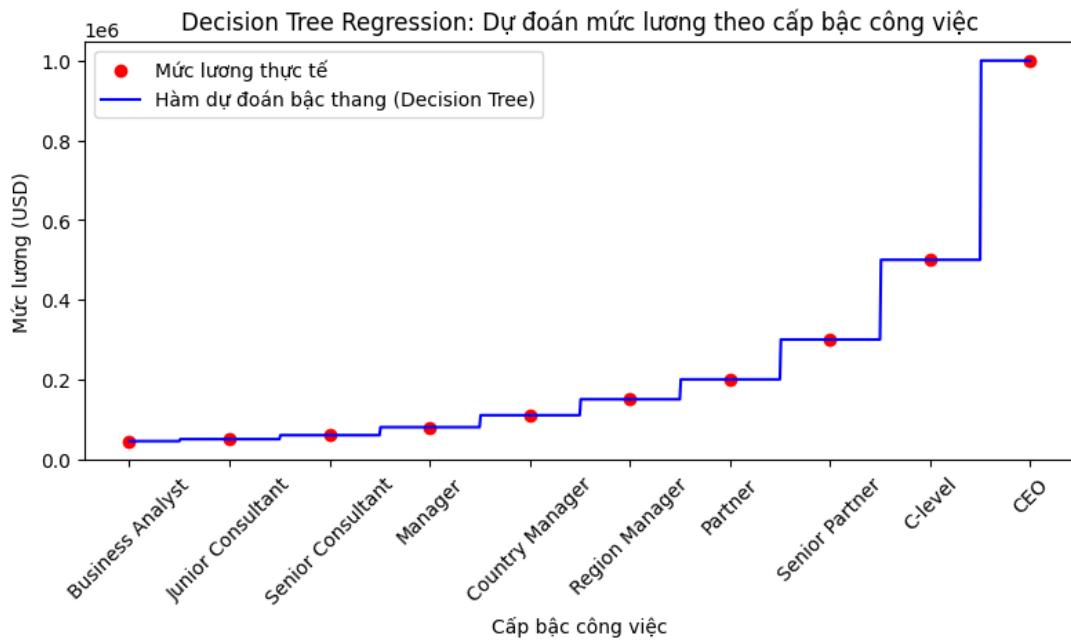
Ví dụ 3: Dự đoán giá cổ phiếu Microsoft (MSFT). Lấy dữ liệu lịch sử từ Yahoo Finance, Regression Tree có thể được huấn luyện để dự đoán giá cổ phiếu dựa trên các thuộc tính như ngày, volume, open/close price. Mỗi nhánh trong cây chia dữ liệu theo các ngưỡng giá hoặc khối lượng giao dịch, và giá dự đoán cuối cùng tại lá sẽ là giá trung bình trong khoảng đó.

7.3 Vấn đề Overfitting trong Regression Tree

Cũng như Classification Tree, Regression Tree rất dễ gặp hiện tượng **overfitting** khi cây phát triển quá sâu. Khi đó, mô hình có thể dự đoán hoàn hảo dữ liệu huấn luyện, nhưng lại dự đoán kém trên dữ liệu mới.

Giải pháp:

- Giới hạn độ sâu cây (max_depth).
- Đặt số mẫu tối thiểu để tách (min_samples_split) hoặc số mẫu tối thiểu tại một lá (min_samples_leaf).
- Sử dụng **Ensemble Methods** như Random Forest Regression hoặc Gradient Boosting Regression.



Hình 12: Ví dụ Decision Tree Regression cho dự đoán mức lương theo cấp bậc công việc. Các dấu tròn đỏ biểu diễn **mức lương thực tế**, trong khi đường xanh thể hiện **hàm dự đoán dạng bậc thang** của Decision Tree. Ta thấy rằng thay vì vẽ một đường thẳng như Linear Regression, Decision Tree tạo ra các đoạn hằng số (piecewise constant), phù hợp với các mức lương thực tế vốn thường nhảy bậc theo vị trí công việc.

8. Kết luận

Qua toàn bộ nội dung, có thể thấy rằng **Decision Tree** là một trong những mô hình đơn giản nhưng đầy sức mạnh trong học máy, đặc biệt với các bài toán phân loại và dự đoán. Điểm mạnh lớn nhất của mô hình này nằm ở sự *trực quan* và khả năng *giải thích kết quả*, mà nhiều mô hình phức tạp khác (như Neural Network) khó đạt được.

- Cây quyết định khắc phục được hạn chế về tốc độ và tính toán phức tạp của KNN, đồng thời hoạt động hiệu quả trên dữ liệu kích thước vừa và lớn.
- Các chỉ số **Gini Impurity** và **Entropy – Information Gain** giúp lựa chọn thuộc tính phân chia tốt nhất, đảm bảo dữ liệu được tách thành những tập con thuận lợi hơn trong **bài toán phân loại**.
- Với **bài toán hồi quy**, cây quyết định dùng tiêu chí **Sum of Squared Residuals (SSR)** hoặc **Mean Squared Error (MSE)** để chọn ngưỡng phân chia, giúp dự đoán các giá trị liên tục như lương, giá nhà hay giá cổ phiếu.
- Với các kỹ thuật mở rộng như **Random Forest**, **Gradient Boosting**, hay **XGBoost**, cây quyết định có thể đạt hiệu năng vượt trội, giảm thiểu overfitting và xử lý tốt các tập dữ liệu lớn, phức tạp.

Thông điệp cuối cùng

Không chỉ là một công cụ học máy, cây quyết định còn là một phương pháp **giải thích quyết định rõ ràng**, giúp con người hiểu được “*tại sao*” mô hình đưa ra dự đoán đó. Chính sự minh bạch này làm cho Decision Tree đặc biệt hữu ích trong các lĩnh vực nhạy cảm như y tế, tài chính và pháp lý, nơi mà cả **dự đoán số liệu** và **phân loại tình huống** đều cần sự tin cậy và minh bạch.

Quản Lý Dữ Liệu Trên Cloud – Từ Nền Tảng Đến Ứng Dụng Thực Tiễn

Dàm Nguyên Khánh

Tóm tắt nội dung

Trong kỷ nguyên dữ liệu, việc quản lý và khai thác thông tin hiệu quả đóng vai trò then chốt trong thành công của doanh nghiệp. Bài viết này cung cấp một cái nhìn toàn diện về **Data Engineering** và **Cloud Data Management**, từ các khái niệm nền tảng như kiểu dữ liệu, Data Warehouse, Data Lake, đến sự khác biệt giữa ETL và ELT. Ngoài ra, bài viết còn giới thiệu các dịch vụ đám mây quan trọng của AWS bao gồm IAM, S3, Glue, Athena và CloudWatch. Thông qua nội dung này, người đọc không chỉ nắm được kiến thức cơ bản mà còn hiểu rõ cách áp dụng vào thực tiễn để xây dựng hệ thống dữ liệu hiện đại, linh hoạt và có khả năng mở rộng.

1. Giới thiệu

Trong kỷ nguyên số, dữ liệu chính là “dầu mỏ mới” của thế giới. Khối lượng dữ liệu mà doanh nghiệp cần xử lý ngày càng khổng lồ, đa dạng và phức tạp. Do đó, **Data Engineering** và **Cloud Data Management** trở thành hai mảnh ghép quan trọng giúp tổ chức khai thác dữ liệu hiệu quả.

Bài blog này sẽ đưa bạn đi từ những khái niệm cơ bản như *Data Engineering*, *Cloud Computing*, cho đến các dịch vụ cloud cụ thể như **AWS S3**, **Glue**, **Athena**, **CloudWatch**.

2. Data Engineering là gì?

2.1 Vai trò của Data Engineer

Data Engineer đóng vai trò như một “kỹ sư hạ tầng dữ liệu”, giúp dữ liệu từ nhiều nguồn khác nhau được thu thập, biến đổi và sẵn sàng cho việc phân tích. Các nhiệm vụ chính bao gồm:

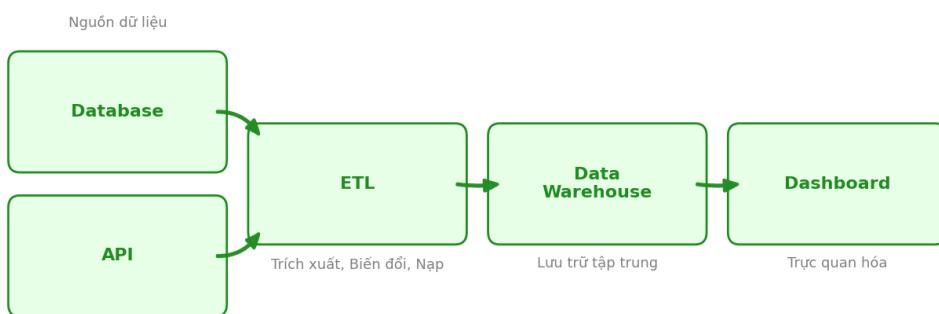
- **Xây dựng và duy trì Data Pipelines (ETL/ELT):** Thiết kế các luồng xử lý dữ liệu tự động từ nguồn (Database, API, log files) để trích xuất, biến đổi và nạp vào hệ thống lưu trữ. Nhờ đó, dữ liệu luôn được cập nhật liên tục và nhất quán.
- **Thiết kế và triển khai Data Warehouse:** Xây dựng kho dữ liệu tập trung, nơi dữ liệu được tổ chức có cấu trúc, dễ dàng cho việc phân tích, báo cáo và hỗ trợ ra quyết định của doanh nghiệp.
- **Đảm bảo chất lượng và độ tin cậy dữ liệu:** Kiểm tra, làm sạch, loại bỏ dữ liệu trùng lặp hoặc sai lệch. Đảm bảo dữ liệu được chuẩn hóa để người dùng cuối (Data Analyst, Data Scientist) có thể sử dụng mà không cần xử lý thêm.
- **Tối ưu hóa hạ tầng xử lý dữ liệu:** Sử dụng công nghệ Big Data và Cloud để cải thiện tốc độ xử lý, giảm chi phí và nâng cao khả năng mở rộng. Điều này giúp hệ thống đáp ứng tốt ngay cả khi dữ liệu tăng trưởng rất lớn.

2.2 Kỹ năng cần thiết

Để trở thành một Data Engineer giỏi, cần trang bị nhiều kỹ năng đa dạng, kết hợp giữa lập trình, cơ sở dữ liệu và công nghệ hiện đại. Một số kỹ năng quan trọng bao gồm:

- **Lập trình (Python):** Python là ngôn ngữ phổ biến nhất trong lĩnh vực dữ liệu nhờ thư viện phong phú (Pandas, PySpark, Airflow). Data Engineer cần dùng Python để xử lý dữ liệu, viết script tự động và triển khai các pipeline.
- **Kiến thức cơ sở dữ liệu:** Hiểu cách làm việc với cả cơ sở dữ liệu quan hệ (SQL Server, PostgreSQL, MySQL) và phi quan hệ (MongoDB, Cassandra). Đây là nền tảng để thiết kế hệ thống lưu trữ dữ liệu hiệu quả và tối ưu cho việc truy vấn.
- **Công nghệ Big Data:** Thành thạo các công cụ xử lý dữ liệu lớn như Hadoop, Spark, Kafka. Những công nghệ này giúp xử lý dữ liệu với dung lượng hàng terabyte hoặc petabyte một cách nhanh chóng và phân tán.
- **Điện toán đám mây (Cloud):** Hiểu và sử dụng dịch vụ của các nhà cung cấp cloud như AWS, Azure, Google Cloud. Các kỹ năng quan trọng gồm: lưu trữ dữ liệu trên S3, xây dựng pipeline với Glue, phân tích với Athena, hoặc triển khai hệ thống trên Kubernetes.

Sơ đồ Pipeline dữ liệu: Database/API → ETL → Data Warehouse → Dashboard



Hình 13: Sơ đồ Pipeline dữ liệu: dữ liệu từ **Database** hoặc **API** sẽ được trích xuất, biến đổi và nạp qua quy trình **ETL**, sau đó lưu trữ tập trung trong **Data Warehouse** và cuối cùng được trực quan hóa qua **Dashboard**.

3. Các loại dữ liệu

Trong lĩnh vực dữ liệu, ta thường gặp ba loại chính: dữ liệu có cấu trúc, phi cấu trúc và nửa cấu trúc. Mỗi loại dữ liệu có đặc điểm và cách lưu trữ, xử lý khác nhau:

- **Structured data (Dữ liệu có cấu trúc):** Là loại dữ liệu được tổ chức theo hàng và cột trong bảng, có *schema* rõ ràng. Dữ liệu này rất dễ lưu trữ trong cơ sở dữ liệu quan hệ (RDBMS) và có thể truy vấn trực tiếp bằng ngôn ngữ SQL. *Ví dụ:* Bảng khách hàng trong SQL với các cột ID, Name, Age.

- **Unstructured data (Dữ liệu phi cấu trúc):** Không tuân theo một định dạng hay cấu trúc nào cụ thể, khó phân tích trực tiếp bằng SQL. Thông thường cần sử dụng AI/ML hoặc NLP để xử lý. *Ví dụ:* Email, văn bản tự do, video, hình ảnh, file âm thanh.
- **Semi-structured data (Dữ liệu nửa cấu trúc):** Nằm giữa structured và unstructured. Dữ liệu không được lưu trong bảng hàng-cột, nhưng vẫn có các thẻ hoặc cặp key-value để định nghĩa cấu trúc. Điều này giúp việc xử lý và phân tích dễ dàng hơn so với unstructured. *Ví dụ:* JSON, XML, YAML, log files.

Bảng 3: So sánh ba loại dữ liệu: Structured, Semi-structured và Unstructured

Loại dữ liệu	Đặc điểm	Ví dụ
Structured (Có cấu trúc)	<ul style="list-style-type: none"> - Tổ chức chặt chẽ theo schema (hàng, cột). - Dễ lưu trữ, truy vấn và phân tích bằng SQL. 	CSDL quan hệ (MySQL, PostgreSQL), bảng Excel.
Semi-structured (Nửa cấu trúc)	<ul style="list-style-type: none"> - Không tuân thủ schema nghiêm ngặt. - Có thẻ/tags hoặc thuộc tính giúp tổ chức dữ liệu. - Linh hoạt nhưng vẫn có khả năng phân tích. 	JSON, XML, YAML, log files.
Unstructured (Phi cấu trúc)	<ul style="list-style-type: none"> - Không có tổ chức rõ ràng. - Khó lưu trữ và phân tích trực tiếp. - Thường cần AI/ML hoặc NLP để xử lý. 	Văn bản tự do, email, hình ảnh, video, âm thanh.

4. Data Warehouse vs Data Lake

Hai khái niệm thường được so sánh trong quản lý dữ liệu là **Data Warehouse** và **Data Lake**. Chúng đều là nơi lưu trữ dữ liệu tập trung, nhưng cách tổ chức và mục đích sử dụng lại khác nhau:

- **Data Warehouse:** Là kho dữ liệu truyền thống, nơi chỉ lưu trữ dữ liệu đã được *làm sạch, biến đổi* và tổ chức theo **schema rõ ràng** trước khi nạp vào hệ thống (**schema-on-write**). Dữ liệu trong Data Warehouse thường ở dạng bảng (structured), phù hợp cho phân tích BI (Business Intelligence), báo cáo, và ra quyết định chiến lược. *Ví dụ:* Lưu trữ báo cáo doanh thu theo tháng từ hệ thống bán hàng.
- **Data Lake:** Là kho dữ liệu hiện đại, có khả năng lưu trữ **tất cả các loại dữ liệu** ở dạng thô: structured, semi-structured, unstructured. Schema chỉ được áp dụng khi dữ liệu được đọc hoặc phân tích (**schema-on-read**). Điều này cho phép Data Lake linh hoạt hơn, phù hợp cho phân tích dữ liệu lớn, machine learning và nghiên cứu chuyên sâu. *Ví dụ:* Lưu log website, dữ liệu IoT, video, hình ảnh để phân tích hành vi người dùng.

Bảng 4: So sánh Data Lake và Data Warehouse

Tiêu chí	Data Lake	Data Warehouse
Kiểu dữ liệu	Lưu trữ mọi loại dữ liệu: structured, semi-structured, unstructured (dạng thô).	Chủ yếu lưu trữ dữ liệu structured, đã được xử lý và chuẩn hóa.
Cách lưu trữ	<i>Schema-on-read</i> : Áp dụng schema khi truy vấn.	<i>Schema-on-write</i> : Dữ liệu phải tuân thủ schema trước khi nạp.
Chi phí lưu trữ	Thường rẻ hơn, do dùng công nghệ lưu trữ phân tán (ví dụ: S3).	Cao hơn vì yêu cầu xử lý, tối ưu hóa và phần cứng mạnh.
Tốc độ truy vấn	Truy vấn chậm hơn vì dữ liệu chưa được xử lý.	Truy vấn nhanh, tối ưu cho báo cáo và phân tích BI.
Người dùng chính	Data Scientist, Data Engineer (phân tích nâng cao, AI/ML).	Business Analyst, nhà quản trị (báo cáo, dashboard).
Trường hợp sử dụng	Lưu trữ log, dữ liệu IoT, hình ảnh, video, dữ liệu raw để phân tích.	Lưu trữ dữ liệu đã chuẩn hóa phục vụ báo cáo tài chính, KPI, dashboard.

5. ETL vs ELT

Trong quá trình xây dựng **pipeline dữ liệu**, có hai mô hình phổ biến để trích xuất và xử lý dữ liệu: **ETL** và **ELT**. Chúng có điểm giống nhau ở việc đều bao gồm 3 bước *Extract*, *Transform*, *Load*, nhưng khác nhau ở *thứ tự thực hiện* và *cách xử lý dữ liệu*:

- **ETL (Extract – Transform – Load):** Đây là cách tiếp cận truyền thống. Dữ liệu được **trích xuất (Extract)** từ nguồn (database, API), sau đó **biến đổi (Transform)** trên một máy chủ trung gian để làm sạch, chuẩn hóa, và cuối cùng mới **nạp (Load)** vào kho dữ liệu (Data Warehouse). *Ví dụ:* Trước khi nạp dữ liệu bán hàng vào Data Warehouse, hệ thống loại bỏ bản ghi trùng, đổi đơn vị tiền tệ về cùng chuẩn, rồi mới lưu vào bảng phân tích doanh thu.
- **ELT (Extract – Load – Transform):** Đây là cách tiếp cận hiện đại, thường dùng trong môi trường **cloud**. Dữ liệu được **trích xuất (Extract)** và **nạp trực tiếp (Load)** vào Data Lake hoặc Data Warehouse ở dạng thô, sau đó mới **biến đổi (Transform)** ngay bên trong hệ thống lưu trữ. Cách làm này tận dụng khả năng tính toán phân tán và mở rộng của cloud. *Ví dụ:* Tải log website thô lên Amazon S3, rồi sử dụng AWS Athena để chạy truy vấn SQL nhằm làm sạch và phân tích.

Bảng 5: So sánh ETL (Extract–Transform–Load) và ELT (Extract–Load–Transform)

Tiêu chí	ETL	ELT
Trình tự xử lý	Extract → Transform → Load	Extract → Load → Transform
Vị trí xử lý	Dữ liệu được biến đổi trên máy chủ trung gian trước khi lưu vào kho dữ liệu.	Dữ liệu được tải trực tiếp vào hệ thống lưu trữ (Data Lake/Warehouse) rồi mới xử lý.
Ứng dụng	Phù hợp với Data Warehouse truyền thống, dữ liệu có cấu trúc rõ ràng.	Phù hợp với hệ thống cloud hiện đại, hỗ trợ dữ liệu đa dạng (structured, semi-structured, unstructured).
Hiệu năng và chi phí	Có thể tồn thời gian khi dữ liệu lớn vì phải biến đổi trước khi lưu.	Tận dụng khả năng mở rộng của cloud, xử lý nhanh hơn và chi phí tối ưu hơn.

6. Cloud Computing

6.1 Định nghĩa

Điện toán đám mây (Cloud Computing) là mô hình cung cấp tài nguyên công nghệ thông tin (như máy chủ, bộ nhớ, cơ sở dữ liệu, mạng, phần mềm) qua Internet theo nhu cầu. Thay vì phải *mua, cài đặt và duy trì* hạ tầng vật lý, doanh nghiệp có thể thuê dịch vụ đám mây và chỉ trả tiền theo mức sử dụng thực tế (**pay-as-you-go**).

Ví dụ: Một startup thương mại điện tử có thể thuê máy chủ ảo trên AWS để chạy website bán hàng, thay vì mua cả hệ thống server vật lý tốn kém.

6.2 Lý do sử dụng Cloud

Các doanh nghiệp hiện nay chuyển dịch sang Cloud bởi nhiều lợi ích nổi bật:

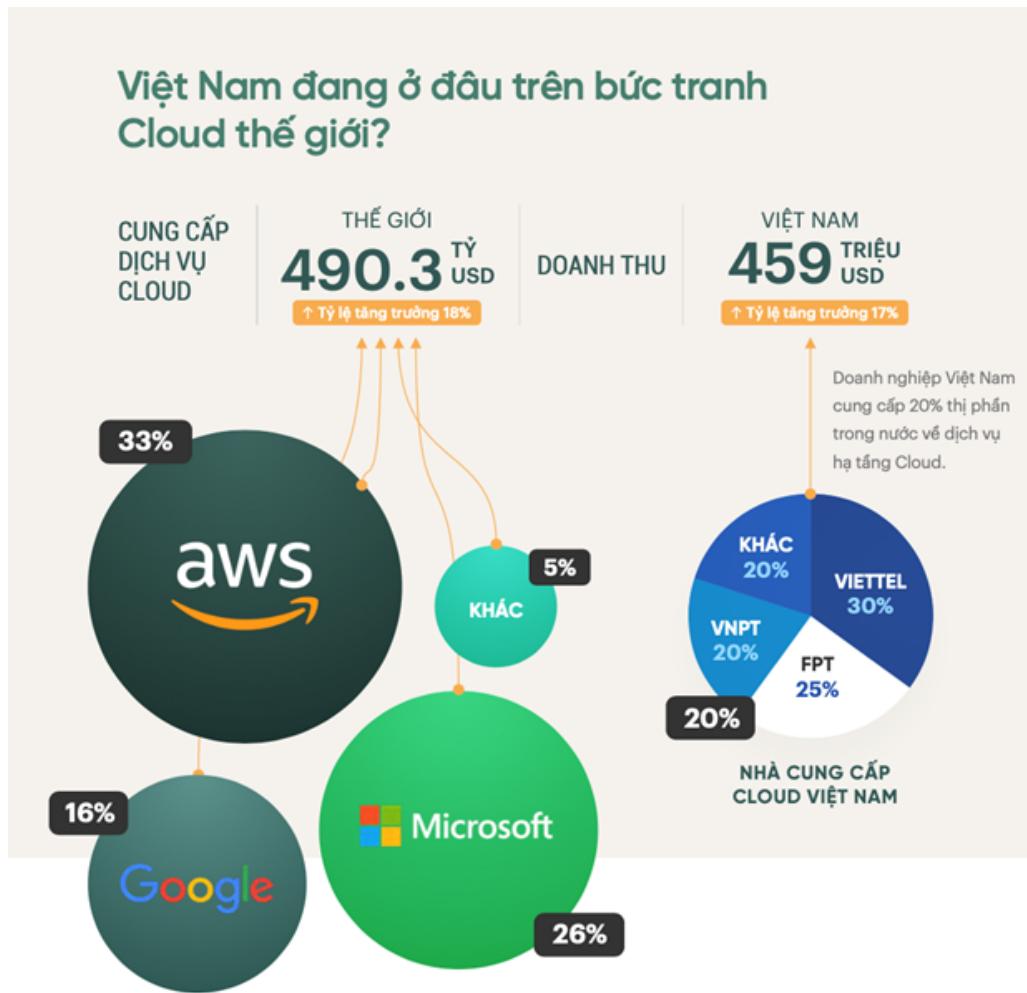
- Chi phí thấp:** Không cần đầu tư lớn vào phần cứng, giảm chi phí bảo trì. Doanh nghiệp chỉ trả tiền cho tài nguyên họ thực sự dùng.
- Bảo mật và sẵn sàng cao:** Các nhà cung cấp Cloud có trung tâm dữ liệu phân tán toàn cầu, đảm bảo dữ liệu luôn được sao lưu và bảo mật theo chuẩn quốc tế.
- Scalability (Khả năng mở rộng):** Dịch vụ có thể tự động tăng hoặc giảm tài nguyên (autoscaling) tùy theo lưu lượng người dùng, phù hợp với các ứng dụng có lượng truy cập biến động.
- Agility (Tính linh hoạt):** Doanh nghiệp có thể nhanh chóng triển khai hoặc gỡ bỏ một dịch vụ chỉ trong vài phút, giúp thử nghiệm ý tưởng mới dễ dàng.

6.3 Nhà cung cấp phổ biến

Trên thế giới, các “ông lớn” về Cloud chiếm lĩnh thị phần:

- Quốc tế:** Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Alibaba Cloud, IBM Cloud, Oracle Cloud.

- **Tại Việt Nam:** Viettel Cloud, FPT Smart Cloud, VNPT Cloud là những đơn vị chủ lực cung cấp hạ tầng đám mây cho doanh nghiệp trong nước.



Hình 14: Thị phần dịch vụ Cloud: Trên thế giới, AWS (33%), Microsoft (26%), Google (16%) dẫn đầu thị trường với doanh thu 490.3 tỷ USD (tăng trưởng 18%). Tại Việt Nam, tổng doanh thu đạt 459 triệu USD (tăng trưởng 17%), trong đó Viettel (30%), FPT (25%), VNPT (20%) và các nhà cung cấp khác chiếm 20%.

7. Command Line Interface (CLI) với Cloud

Bên cạnh giao diện web (Console), hầu hết các dịch vụ Cloud đều hỗ trợ thao tác qua **Command Line Interface (CLI)**. Đây là cách giao tiếp với hệ thống bằng cách gõ lệnh thay vì dùng chuột. Thông qua CLI, người dùng có thể **tạo, quản lý và giám sát tài nguyên trên Cloud** một cách nhanh chóng, đặc biệt hữu ích khi tự động hóa (automation) hoặc triển khai hạ tầng bằng script.

Ví dụ lệnh cơ bản trong Linux

Dưới đây là một số lệnh thường dùng trong Linux, nền tảng cơ bản khi làm việc với Cloud:

- pwd – Hiển thị đường dẫn của thư mục hiện tại (Print Working Directory).
- ls – Liệt kê tất cả các file và thư mục trong thư mục hiện tại.

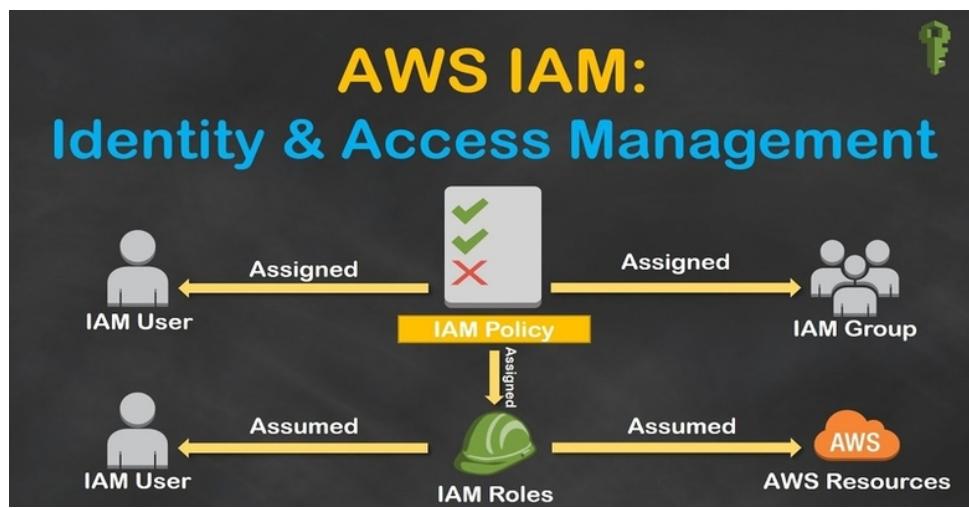
- cd <tên_thư_mục> – Di chuyển sang một thư mục khác.
- mkdir <tên_thư_mục> – Tạo thư mục mới.
- rm <tên_file> – Xóa file.
- cp <file1> <file2> – Sao chép file1 sang file2.
- mv <file1> <file2> – Di chuyển hoặc đổi tên file.
- cat <tên_file> – In toàn bộ nội dung file ra màn hình.
- head <tên_file> – Hiển thị 10 dòng đầu tiên của file.
- tail <tên_file> – Hiển thị 10 dòng cuối cùng của file.
- grep <tù_khóa> <tên_file> – Tìm kiếm từ khóa trong file.

8. AWS IAM

IAM (Identity and Access Management) là dịch vụ cốt lõi của AWS, cho phép doanh nghiệp quản lý người dùng và quyền truy cập vào các tài nguyên AWS một cách an toàn. IAM giúp đảm bảo rằng chỉ những người (hoặc ứng dụng) có quyền phù hợp mới được phép thực hiện hành động trên hệ thống.

Các thành phần chính trong IAM

- **User (Người dùng):** Đại diện cho một cá nhân hoặc một ứng dụng cụ thể. Mỗi user có thông tin đăng nhập riêng (username, password hoặc access key) để truy cập AWS.
- **Group (Nhóm):** Tập hợp nhiều user lại để dễ dàng gán quyền chung. Thay vì cấp quyền riêng lẻ cho từng người, ta có thể gán policy cho nhóm (ví dụ: nhóm Developers có quyền truy cập S3 và EC2).
- **Role (Vai trò):** Cung cấp quyền truy cập tạm thời cho user, ứng dụng hoặc dịch vụ khác. Ví dụ: một ứng dụng chạy trên EC2 có thể sử dụng Role để truy cập vào S3 mà không cần lưu trữ access key trong code.
- **Policy (Chính sách):** Là tập hợp các quy tắc được viết bằng JSON, định nghĩa rõ ai được làm gì trên tài nguyên nào. Ví dụ: một policy có thể cho phép user đọc dữ liệu từ S3 nhưng không được xóa file.



Hình 15: Mô hình AWS IAM (Identity & Access Management): IAM User hoặc IAM Group được gán quyền qua IAM Policy. Người dùng cũng có thể tạm thời đảm nhận IAM Roles để truy cập vào AWS Resources. IAM cho phép quản lý danh tính và phân quyền chi tiết, đảm bảo tính bảo mật và kiểm soát trong hệ thống AWS.

9. AWS S3 (Simple Storage Service)

9.1 Đặc điểm

Amazon S3 là dịch vụ lưu trữ đám mây phổ biến nhất của AWS, cho phép lưu trữ và truy xuất dữ liệu ở bất kỳ đâu qua Internet. Một số đặc điểm nổi bật:

- **Lưu trữ không giới hạn:** Có thể chứa từ vài kilobytes đến hàng petabytes dữ liệu mà không lo hết dung lượng.
- **Độ bền 99.99999999% (11 số 9):** Dữ liệu được sao chép và phân tán trên nhiều thiết bị, nhiều khu vực trong cùng một vùng (region), đảm bảo độ an toàn cực cao.
- **Trả phí theo mức dùng:** Người dùng chỉ trả tiền cho dung lượng lưu trữ và băng thông sử dụng thực tế, giúp tối ưu chi phí.

9.2 Ứng dụng

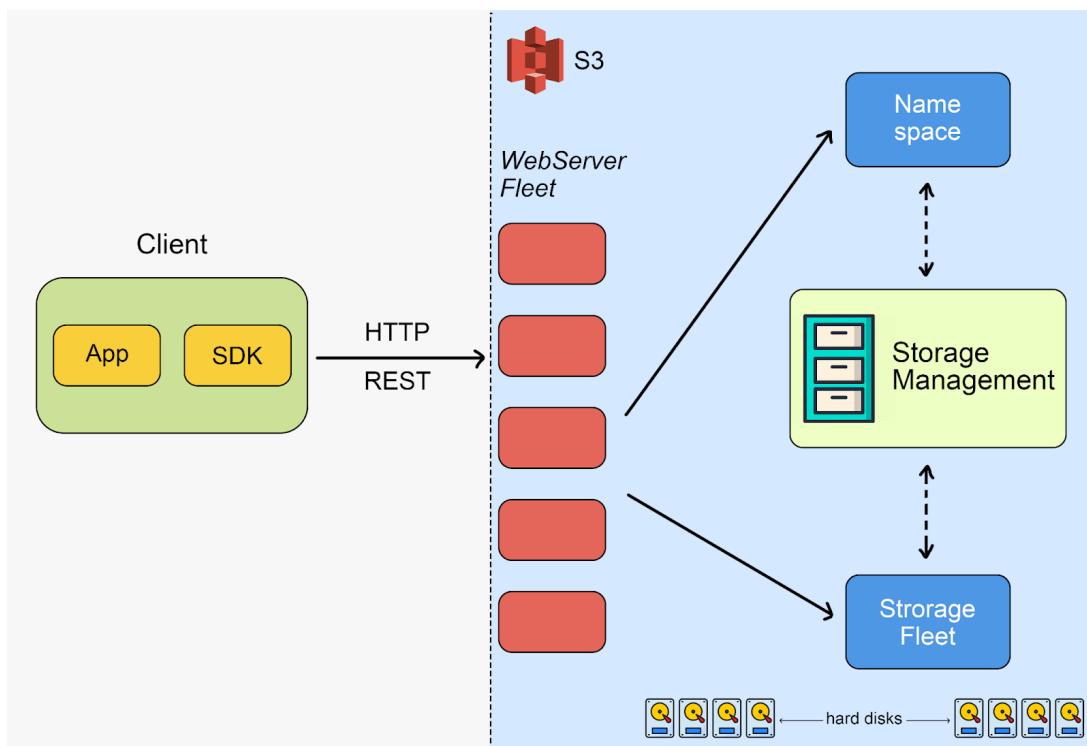
S3 có nhiều ứng dụng trong thực tế, đặc biệt trong việc quản lý và phân tích dữ liệu:

- **Data Lake:** Dùng để tập trung lưu trữ dữ liệu thô (raw data) từ nhiều nguồn, phục vụ phân tích sau này.
- **Backup & Disaster Recovery:** Lưu trữ bản sao dữ liệu dự phòng, giúp khôi phục nhanh chóng khi hệ thống gặp sự cố.
- **Static Website Hosting:** Có thể dùng S3 để triển khai website tĩnh (HTML, CSS, JS) mà không cần máy chủ web phức tạp.

9.3 Khái niệm chính

Một số khái niệm quan trọng trong S3:

- **Bucket:** Là “thư mục gốc” để lưu trữ dữ liệu trong S3. Mỗi bucket phải có tên duy nhất toàn cầu và được tạo tại một khu vực địa lý (region) cụ thể.
- **Object:** Là đơn vị lưu trữ cơ bản trong S3, có thể là bất kỳ loại file nào (ảnh, video, tài liệu...). Mỗi object gồm:
 - **Data:** Nội dung file.
 - **Key:** Tên duy nhất trong bucket.
 - **Metadata:** Thông tin bổ sung (ví dụ: loại file, ngày tạo).
 - **Version ID:** Định danh phiên bản, nếu bucket bật tính năng versioning.



Hình 16: Kiến trúc cơ bản của Amazon S3: Client (ứng dụng hoặc SDK) gửi yêu cầu qua giao thức HTTP/REST tới **WebServer Fleet**. Các request này được điều phối đến **Storage Management**, nơi quản lý **Name space** và **Storage Fleet**. Cuối cùng, dữ liệu được lưu trữ trên hệ thống **hard disks** phân tán, đảm bảo khả năng mở rộng, độ bền và tính sẵn sàng cao.

10. AWS Glue

AWS Glue là một dịch vụ **ETL (Extract – Transform – Load)** serverless do AWS cung cấp. Điều này có nghĩa là người dùng không cần quản lý máy chủ, mà chỉ tập trung vào việc chuẩn bị và biến đổi dữ liệu để phục vụ cho phân tích và machine learning.

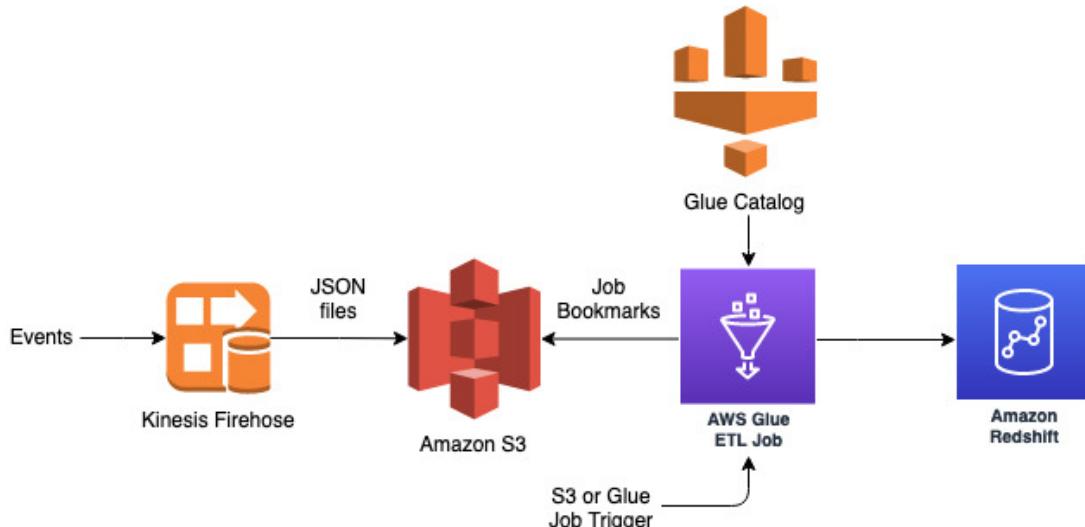
Các thành phần chính

- **Data Catalog:** Đây là kho siêu dữ liệu (metadata repository) trung tâm, lưu trữ thông tin về dữ liệu như vị trí, schema, định dạng. Data Catalog giúp các dịch vụ khác của AWS (như Athena, Redshift, EMR) có thể dễ dàng tìm kiếm và hiểu cấu trúc dữ liệu. *Ví dụ:* Khi dữ

liệu log được lưu dưới dạng file Parquet trên S3, Data Catalog sẽ ghi nhận tên bảng, cột và kiểu dữ liệu để Athena có thể truy vấn bằng SQL.

- **Crawlers:** Là các chương trình tự động quét dữ liệu từ nhiều nguồn (S3, RDS, DynamoDB...), nhận diện cấu trúc (schema inference), sau đó ghi lại thông tin đó vào Data Catalog. Nhờ Crawlers, người dùng không cần thủ công định nghĩa schema cho từng dataset. *Ví dụ:* Một crawler có thể phát hiện file JSON trên S3 và tự động ánh xạ thành bảng có các cột tương ứng.
- **ETL Jobs:** Là các tác vụ xử lý dữ liệu, nơi người dùng định nghĩa logic ETL. AWS Glue hỗ trợ nhiều cách tạo ETL Job:
 - **Giao diện trực quan (Glue Studio):** Kéo-thả các bước xử lý dữ liệu mà không cần code.
 - **Sinh code tự động:** Glue có thể tạo sẵn code bằng Python/Scala dựa trên dữ liệu đầu vào và đầu ra.
 - **Viết code tùy chỉnh:** Người dùng có thể viết script Python/Scala chạy trên nền tảng Apache Spark phân tán.

Ví dụ: Một ETL Job có thể đọc dữ liệu CSV từ S3, làm sạch dữ liệu (loại bỏ giá trị null), chuyển sang định dạng Parquet, rồi ghi ngược lại vào S3 để phục vụ Athena.



Hình 17: Quy trình xử lý dữ liệu với AWS Glue: **Events** được thu thập bởi **Kinesis Firehose** và lưu dưới dạng **JSON files** trong **Amazon S3**. Từ đây, **AWS Glue ETL Job** (kích hoạt bởi S3 hoặc trigger) sẽ thực hiện trích xuất, biến đổi và nạp dữ liệu, sử dụng **Glue Catalog** để quản lý metadata. Cuối cùng, dữ liệu đã qua xử lý được nạp vào **Amazon Redshift** để phục vụ phân tích.

11. AWS Athena

Amazon Athena là dịch vụ **query serverless** cho phép người dùng sử dụng ngôn ngữ SQL để phân tích dữ liệu trực tiếp trên **Amazon S3** mà không cần thiết lập máy chủ hay cơ sở dữ liệu riêng. Athena tự động mở rộng theo nhu cầu, và chi phí chỉ tính dựa trên lượng dữ liệu được quét trong mỗi truy vấn.

Đặc điểm nổi bật

- **Không cần hạ tầng:** Hoàn toàn serverless, người dùng chỉ tập trung vào câu lệnh SQL.
- **Tích hợp với Glue:** Sử dụng AWS Glue Data Catalog để định nghĩa schema, bảng và cột, giúp truy vấn dữ liệu dễ dàng.
- **Thanh toán theo truy vấn:** Mỗi lần chạy SQL, Athena chỉ tính phí dựa trên số GB dữ liệu được quét.
- **Hỗ trợ nhiều định dạng dữ liệu:** CSV, JSON, ORC, Avro, Parquet...

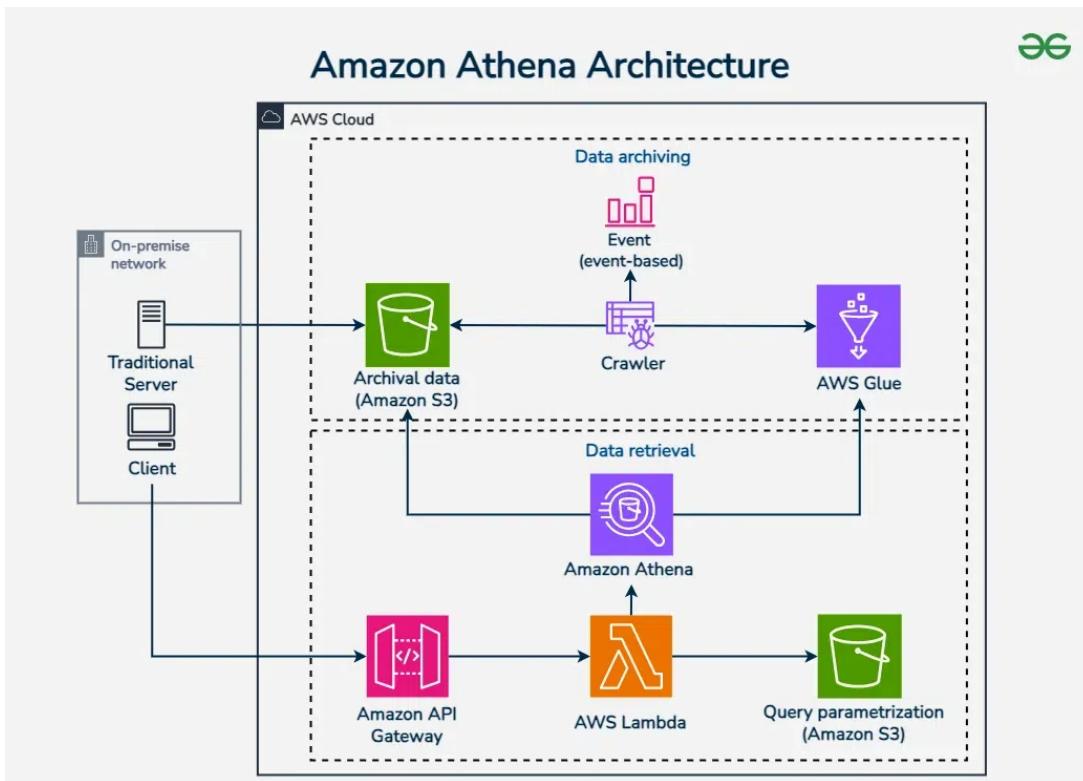
Các trường hợp sử dụng

- **Phân tích log:** Truy vấn trực tiếp log hệ thống hoặc ứng dụng lưu trên S3, phục vụ giám sát hoặc phát hiện sự cố.
- **Business Intelligence (BI):** Kết hợp Athena với công cụ trực quan hóa (như QuickSight, Tableau) để tạo dashboard theo thời gian thực.
- **Truy vấn Data Lake:** Khai thác dữ liệu thô hoặc bán cấu trúc trong Data Lake mà không cần di chuyển dữ liệu sang hệ quản trị cơ sở dữ liệu khác.

Ví dụ minh họa

Giả sử bạn có log website lưu dưới dạng file CSV trong S3, bạn có thể truy vấn số lượng truy cập theo ngày bằng Athena:

```
SELECT date, COUNT(*) AS total_visits  
FROM web_logs  
GROUP BY date  
ORDER BY date DESC;
```



Hình 18: Kiến trúc Amazon Athena: Dữ liệu từ **Traditional Server** hoặc **Client** được lưu trữ trong **Amazon S3**. Quá trình **Data Archiving** sử dụng **Crawler** và **AWS Glue** để lập danh mục và chuẩn hóa dữ liệu. Người dùng có thể truy vấn trực tiếp bằng **Amazon Athena**, dữ liệu được xử lý thông qua **AWS Lambda** và **Amazon API Gateway**, với tham số truy vấn được quản lý trong **Amazon S3**. Điều này cho phép phân tích dữ liệu nhanh chóng, serverless và linh hoạt.

12. AWS CloudWatch

Amazon CloudWatch là dịch vụ giám sát và quan sát hệ thống trong AWS. Nó cung cấp dữ liệu và thông tin chi tiết giúp người dùng theo dõi hiệu năng, độ tin cậy và mức sử dụng tài nguyên. CloudWatch đóng vai trò như một “trung tâm điều khiển”, thu thập dữ liệu từ nhiều dịch vụ khác nhau để hỗ trợ giám sát toàn diện.

Các tính năng chính

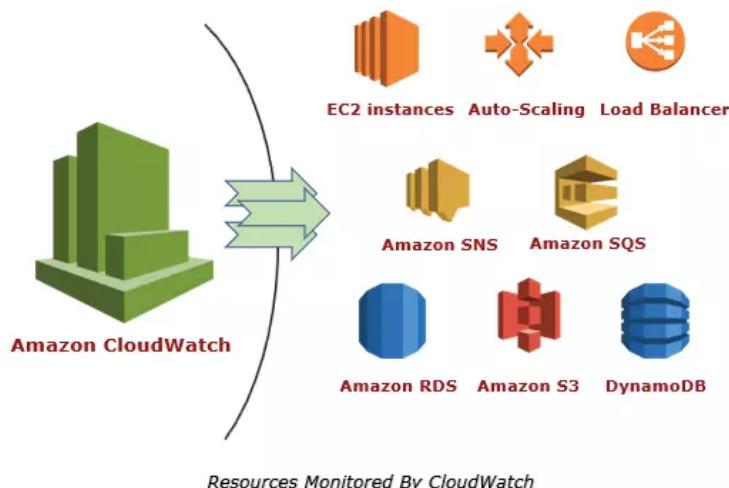
- **Metrics (Chỉ số):** CloudWatch tự động thu thập các chỉ số từ dịch vụ AWS (như CPUUtilization của EC2, số lượng request của S3). Người dùng cũng có thể tạo custom metrics từ ứng dụng của mình.
- **Logs:** CloudWatch Logs cho phép tập trung toàn bộ log từ EC2, Lambda hoặc hệ thống on-premises, giúp dễ dàng phân tích, tìm kiếm và đặt cảnh báo khi có lỗi bất thường.
- **Events:** CloudWatch Events (hiện tích hợp trong Amazon EventBridge) ghi nhận các sự kiện thay đổi trạng thái của tài nguyên. Ví dụ: khi một instance EC2 thay đổi trạng thái từ running sang stopped, sự kiện này có thể kích hoạt một Lambda function.
- **Alarms (Cảnh báo):** Người dùng có thể thiết lập ngưỡng cho bất kỳ metric nào và nhận cảnh báo (qua SNS, email, SMS) khi vượt ngưỡng. Ví dụ: gửi cảnh báo khi CPU của EC2

vượt 80%.

- **Dashboards:** CloudWatch Dashboards cho phép tùy chỉnh giao diện trực quan, tập hợp nhiều biểu đồ và chỉ số trên một màn hình, hỗ trợ quản trị hệ thống theo thời gian thực.

Ví dụ minh họa

- Giám sát CPU và RAM của EC2 để đảm bảo hiệu năng hệ thống.
- Thu thập log từ Lambda để debug ứng dụng serverless.
- Tự động kích hoạt scale-out khi traffic web tăng cao.



Hình 19: Amazon CloudWatch giám sát nhiều tài nguyên trong hệ sinh thái AWS, bao gồm **EC2 instances**, **Auto-Scaling**, **Load Balancer**, dịch vụ nhắn tin như **SNS** và **SQS**, các dịch vụ lưu trữ và cơ sở dữ liệu như **RDS**, **S3**, và **DynamoDB**. CloudWatch cung cấp dữ liệu metrics, logs, alarms để theo dõi hiệu năng và độ ổn định toàn hệ thống.

13. Kết luận

Trong bài viết này, chúng ta đã cùng tìm hiểu những kiến thức cốt lõi trong quản lý và xử lý dữ liệu trên nền tảng đám mây. Từ **Data Engineering** với vai trò xây dựng pipeline dữ liệu, đến **Cloud Computing** như một hạ tầng linh hoạt, và cách thao tác bằng **CLI**, tất cả đều là nền móng quan trọng để khai thác dữ liệu hiệu quả.

Bên cạnh đó, chúng ta cũng đã làm quen với các dịch vụ AWS then chốt:

- **IAM** giúp kiểm soát truy cập và đảm bảo bảo mật.
- **S3** mang lại khả năng lưu trữ không giới hạn và chi phí tối ưu.
- **Glue** hỗ trợ xây dựng pipeline ETL/ELT linh hoạt và tự động.
- **Athena** cho phép phân tích dữ liệu nhanh chóng chỉ với SQL.
- **CloudWatch** giúp giám sát và đảm bảo hệ thống hoạt động ổn định.

Điện toán đám mây không chỉ giúp doanh nghiệp **giảm chi phí đầu tư hạ tầng**, mà còn mang lại **khả năng mở rộng linh hoạt, độ tin cậy cao và bảo mật toàn diện**. Với việc nắm vững những công cụ và kiến thức trên, bạn đã có nền tảng vững chắc để:

- Xây dựng hệ thống dữ liệu hiện đại và dễ mở rộng.
- Đáp ứng nhu cầu phân tích, báo cáo, và machine learning.
- Tận dụng tối đa sức mạnh của dữ liệu để đưa ra quyết định kinh doanh.

“Nắm dữ liệu trong tay, bạn nắm trong tay nội lực để kiến tạo tương lai.”

Excel cho Phân Tích Dữ Liệu – Phần II

Đàm Nguyễn Khánh

Tóm tắt nội dung

Excel không chỉ là công cụ bảng tính quen thuộc mà còn là một nền tảng mạnh mẽ cho phân tích dữ liệu. Bài viết này tập trung vào ba khía cạnh quan trọng: (1) **Trực quan hóa dữ liệu** giúp phát hiện xu hướng, truyền đạt thông tin và hỗ trợ ra quyết định; (2) **Kiểm định giả thuyết & A/B Testing** cung cấp công cụ thống kê để đánh giá sự khác biệt, xác minh hiệu quả chiến lược kinh doanh; (3) **Tiền xử lý dữ liệu & Hồi quy tuyến tính** làm sạch dữ liệu, loại bỏ sai lệch và xây dựng mô hình dự báo. Thông qua các ví dụ và tình huống thực tế, bài viết cho thấy cách Excel có thể được khai thác như một công cụ phân tích dữ liệu hiện đại, giúp doanh nghiệp và cá nhân đưa ra quyết định dựa trên dữ liệu (*data-driven decisions*).

Giới thiệu

Excel thường được biết đến như một công cụ văn phòng quen thuộc để nhập số liệu và tính toán. Tuy nhiên, ít ai nhận ra rằng Excel cũng là một **công cụ phân tích dữ liệu mạnh mẽ**, có thể hỗ trợ trực tiếp trong các bài toán từ cơ bản đến nâng cao.

Trong bài viết này, chúng ta sẽ đi qua ba mảng kiến thức chính:

- **Trực quan hóa dữ liệu (Data Visualization):** Đây là bước quan trọng giúp biến các con số khô khan thành hình ảnh trực quan như biểu đồ cột, đường, tròn, heatmap hay biểu đồ thác nước. Nhờ vậy, bạn có thể nhanh chóng nhận diện xu hướng, so sánh các nhóm dữ liệu và truyền đạt kết quả một cách sinh động.
- **Kiểm định giả thuyết & A/B Testing (Hypothesis Testing & A/B Testing):** Đây là phần giúp ta *kiểm tra và chứng minh bằng số liệu*. Thay vì ra quyết định theo cảm tính, bạn có thể dựa trên các kiểm định thống kê (t-test, ANOVA, chi-square, ...) để xác định sự khác biệt có thật sự đáng kể hay chỉ là ngẫu nhiên. A/B Testing là ví dụ điển hình: so sánh hai phiên bản trang web hoặc chiến dịch marketing để chọn ra phương án hiệu quả hơn.
- **Tiền xử lý dữ liệu & Hồi quy tuyến tính (Data Preprocessing & Regression Analysis):** Dữ liệu thường không hoàn hảo: có giá trị thiếu, ngoại lệ, hay định dạng không thống nhất. Tiền xử lý sẽ giúp làm sạch và chuẩn hóa dữ liệu trước khi phân tích. Sau đó, với hồi quy tuyến tính, bạn có thể xây dựng mô hình để *dự đoán* và *giải thích mối quan hệ* giữa các yếu tố kinh doanh (ví dụ: chi phí marketing ảnh hưởng thế nào đến doanh thu).

Với ba nội dung trên, bạn sẽ thấy Excel không chỉ dừng lại ở những phép tính cơ bản, mà thực sự là một công cụ hỗ trợ đắc lực cho việc **ra quyết định dựa trên dữ liệu (data-driven decisions)**.

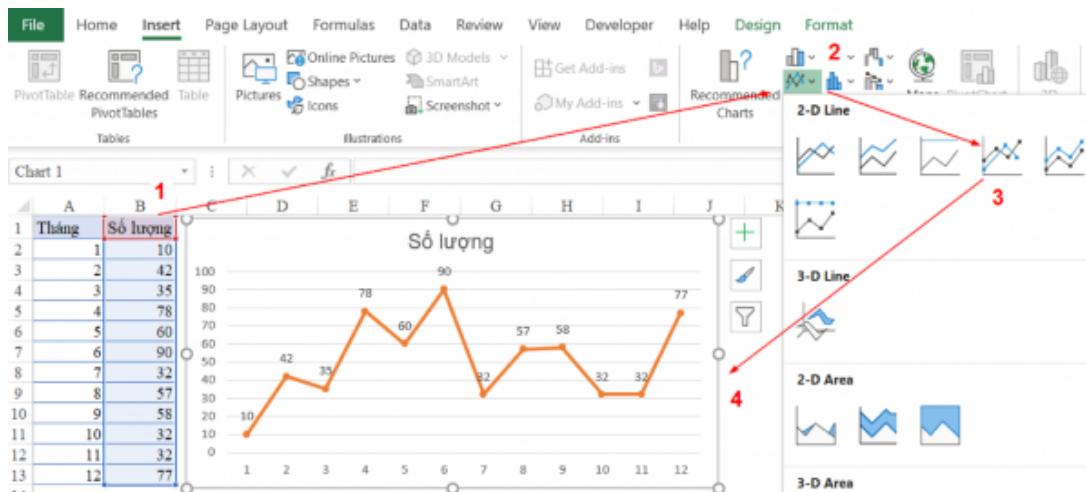
1. Trực quan hóa dữ liệu

1.1 Vì sao cần trực quan hóa?

Trực quan hóa dữ liệu là bước quan trọng biến những con số khô khan trong bảng tính thành hình ảnh dễ hiểu và dễ ghi nhớ. Thay vì phải đọc hàng trăm dòng dữ liệu, người dùng chỉ cần

nhìn vào một biểu đồ là có thể nắm bắt ý chính.

- **Giúp nhận diện xu hướng nhanh chóng:** Khi theo dõi doanh số bán hàng theo tháng, một bảng số liệu dài sẽ rất khó phát hiện xu hướng. Nhưng với *biểu đồ đường (Line Chart)*, bạn sẽ ngay lập tức thấy được giai đoạn nào doanh số tăng mạnh, giai đoạn nào sụt giảm.
- **Truyền đạt thông điệp hiệu quả:** Một báo cáo dày đặc con số có thể mất 15 phút để giải thích, nhưng một *dashboard* với biểu đồ trực quan chỉ cần 2-3 phút để người xem hiểu toàn cảnh. Điều này đặc biệt hữu ích trong các cuộc họp, khi thời gian trình bày rất hạn chế.
- **Hỗ trợ ra quyết định chính xác:** Các loại biểu đồ như *heatmap* hay *scatter plot* giúp phát hiện mối quan hệ giữa các biến. Ví dụ: heatmap thể hiện mức độ lợi nhuận của từng sản phẩm theo khu vực; scatter plot cho thấy mối liên hệ giữa chi phí quảng cáo và doanh thu. Từ đó, nhà quản lý có thể đưa ra chiến lược phân bổ nguồn lực hợp lý.



Hình 20: Các bước tạo biểu đồ đường trong Excel: (1) Chọn vùng dữ liệu; (2) Chọn tab *Insert*; (3) Chọn nhóm biểu đồ *Line*; (4) Chèn biểu đồ đường để trực quan hóa xu hướng dữ liệu theo thời gian.

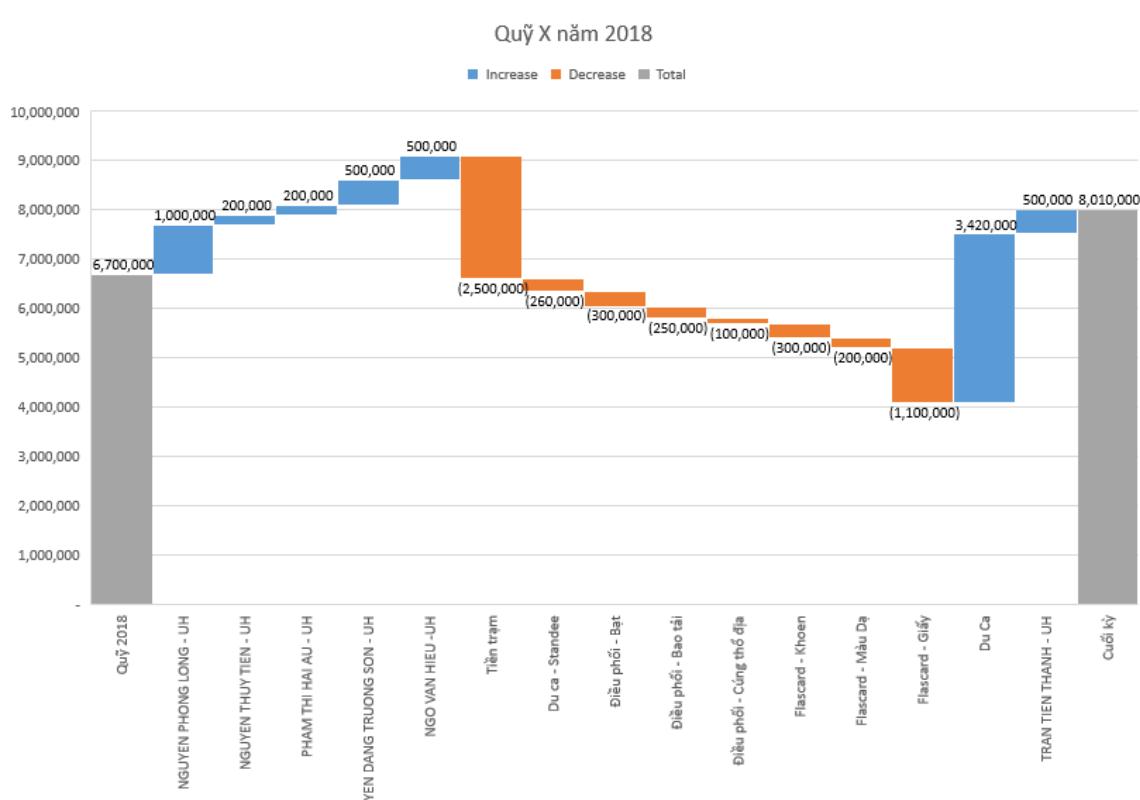
1.2 Các loại biểu đồ thường dùng

Trong Excel, có nhiều loại biểu đồ phục vụ cho những mục đích khác nhau. Dưới đây là các loại biểu đồ phổ biến nhất và ý nghĩa của chúng:

- **Biểu đồ cột (Bar Chart):** Dùng để so sánh giá trị giữa các nhóm khác nhau. Ví dụ: So sánh doanh số giữa các chi nhánh trong cùng một quý.
- **Biểu đồ đường (Line Chart):** Thể hiện xu hướng thay đổi theo thời gian. Ví dụ: Doanh thu theo từng tháng trong năm.
- **Biểu đồ tròn (Pie Chart):** Hiển thị tỷ lệ phần trăm các thành phần trong một tổng thể. Ví dụ: Tỷ lệ thị phần của các hãng điện thoại trên thị trường. Lưu ý: Nên dùng khi số lượng phần tử ≤ 7 , tránh rối mắt.
- **Histogram (Biểu đồ tần suất):** Cho thấy phân phối dữ liệu, giúp kiểm tra dữ liệu có phân bố chuẩn hay không. Ví dụ: Phân phối điểm thi của sinh viên trong một lớp học.

- **Heatmap:** Biểu diễn dữ liệu bằng màu sắc để phát hiện nhanh khu vực mạnh – yếu. Ví dụ: Bảng lợi nhuận theo sản phẩm và khu vực, trong đó màu đậm thể hiện doanh thu cao.
- **Biểu đồ thác nước (Waterfall Chart):** Thể hiện sự thay đổi lũy kế của một chỉ số qua nhiều giai đoạn. Ví dụ: Theo dõi biến động quỹ tài chính qua các khoản thu, chi và kết quả cuối kỳ.
- **Biểu đồ phân tán (Scatter Plot):** Thể hiện mối quan hệ giữa hai biến số. Ví dụ: Xem chi phí quảng cáo có ảnh hưởng như thế nào đến doanh thu bán hàng.

Gợi ý chèn hình: Mỗi loại biểu đồ có thể minh họa bằng một ví dụ nhỏ, chẳng hạn Bar Chart cho doanh số chi nhánh, Line Chart cho xu hướng doanh thu, Pie Chart cho thị phần, Scatter Plot cho mối quan hệ chi phí – doanh thu.



Hình 21: Biểu đồ Waterfall thể hiện biến động quỹ X trong năm 2018. Các cột màu xanh dương biểu diễn khoản **tăng** (Increase), màu cam biểu diễn khoản **giảm** (Decrease), và cột màu xám biểu diễn **tổng cộng** (Total) tại các giai đoạn. Biểu đồ giúp nhận diện rõ yếu tố nào đóng góp tích cực hoặc tiêu cực vào kết quả tài chính cuối kỳ.

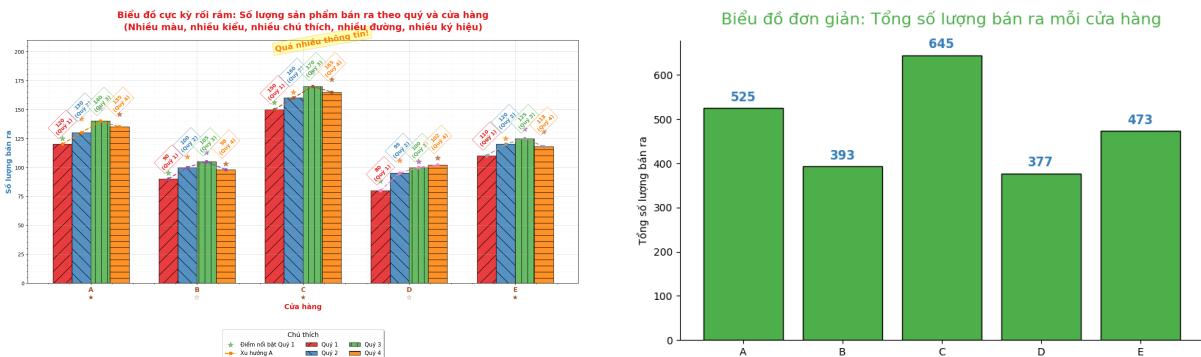
1.3 Tips để trực quan hóa hiệu quả

Một biểu đồ đẹp chưa chắc đã hiệu quả, nhưng một biểu đồ hiệu quả luôn phải rõ ràng và dễ hiểu. Dưới đây là một số mẹo quan trọng khi trực quan hóa dữ liệu trong Excel:

1. **Chọn đúng loại biểu đồ:** Mỗi loại biểu đồ phù hợp cho một mục đích khác nhau. Ví dụ: biểu đồ cột để so sánh doanh số giữa các chi nhánh, biểu đồ đường để theo dõi xu hướng doanh thu theo tháng, biểu đồ tròn để thể hiện tỷ lệ thị phần.
2. **Đơn giản hóa nội dung, loại bỏ chi tiết dư thừa:** Một biểu đồ quá nhiều chi tiết sẽ khiến

người xem mất tập trung. Hãy loại bỏ gridline thừa, hạn chế số màu sắc, và sắp xếp dữ liệu theo thứ tự hợp lý (tăng/giảm hoặc theo thời gian).

3. **Ghi nhãn rõ ràng, dùng màu sắc nhất quán:** Tiêu đề ngắn gọn giúp người xem hiểu ngay mục đích biểu đồ. Nhãn trực X, Y cần rõ ràng và dễ đọc. Màu sắc nên giữ nhất quán: ví dụ cùng một nhóm sản phẩm thì luôn dùng một màu xuyên suốt các biểu đồ trong báo cáo.



(a) Biểu đồ rối răm: Quá nhiều màu, nhãn, ký hiệu và chú thích khiến người xem khó tập trung vào thông điệp chính.

(b) Biểu đồ đơn giản: Tập trung vào thông tin quan trọng (tổng số lượng bán ra), dễ đọc và dễ so sánh.

Hình 22: So sánh giữa một biểu đồ rối răm và một biểu đồ đơn giản. Bài học: *Đơn giản hóa nội dung, loại bỏ chi tiết thừa, và chỉ giữ lại thông điệp chính mà biểu đồ cần truyền đạt.*

2. Kiểm định giả thuyết & A/B Testing

2.1 Khái niệm cơ bản

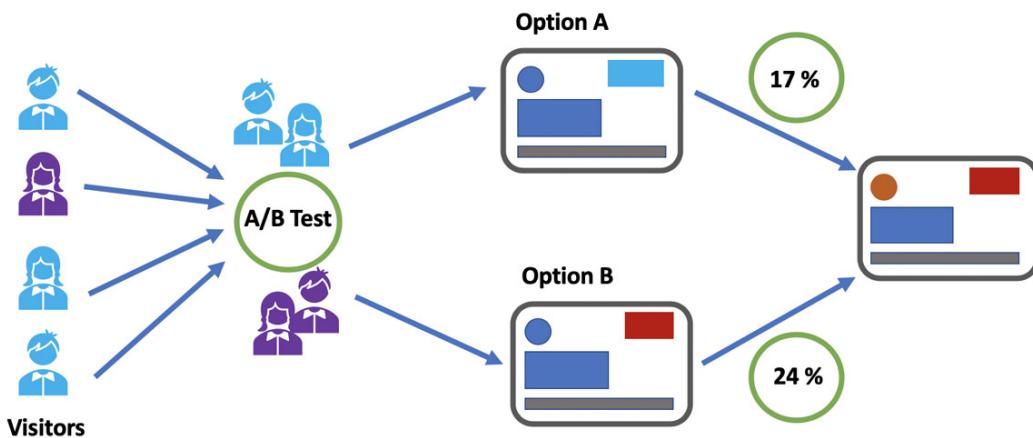
Kiểm định giả thuyết là một quy trình thống kê nhằm kiểm tra xem một giả thuyết đặt ra có được dữ liệu ủng hộ hay không. Thay vì dựa vào cảm tính, chúng ta dựa vào bằng chứng số liệu để đưa ra kết luận.

Trong kiểm định giả thuyết, thường có bốn khái niệm quan trọng:

- **Giả thuyết gốc (H_0):** Giả định ban đầu, thường là “không có sự khác biệt hoặc tác động nào”. Ví dụ: Doanh thu trung bình trước và sau khi chạy chiến dịch marketing là như nhau.
- **Giả thuyết thay thế (H_1):** Khẳng định có sự khác biệt hoặc tác động. Ví dụ: Doanh thu trung bình sau chiến dịch marketing **cao hơn** so với trước đó.
- **p-value:** Xác suất để thu được kết quả quan sát (hoặc cực đoan hơn) nếu H_0 là đúng. - Nếu p-value nhỏ (ví dụ < 0.05), dữ liệu mang lại bằng chứng mạnh để bác bỏ H_0 . - Nếu p-value lớn (≥ 0.05), ta chưa đủ cơ sở để bác bỏ H_0 .
- **Mức ý nghĩa (α):** Ngưỡng quyết định, thường chọn 5% ($\alpha = 0.05$). Điều này nghĩa là chúng ta chấp nhận rủi ro 5% có thể kết luận sai khi bác bỏ H_0 .

Ví dụ thực tế: Một công ty muốn biết chiến dịch quảng cáo mới có thật sự làm tăng doanh số.

- H_0 : Doanh thu trung bình trước và sau chiến dịch là như nhau.
- H_1 : Doanh thu trung bình sau chiến dịch cao hơn.
- Nếu kết quả kiểm định cho $p\text{-value} = 0.02 < 0.05$, ta có đủ bằng chứng để bác bỏ H_0 và kết luận rằng chiến dịch có hiệu quả.



Hình 23: Minh họa quy trình A/B Testing: Người dùng (visitors) được chia ngẫu nhiên thành hai nhóm, trải nghiệm hai phiên bản giao diện (Option A và Option B). Kết quả cho thấy Option A có tỷ lệ chuyển đổi 17%, trong khi Option B đạt 24%. Phân tích thống kê từ kết quả này giúp lựa chọn phiên bản tối ưu.

2.2 Ví dụ kinh doanh

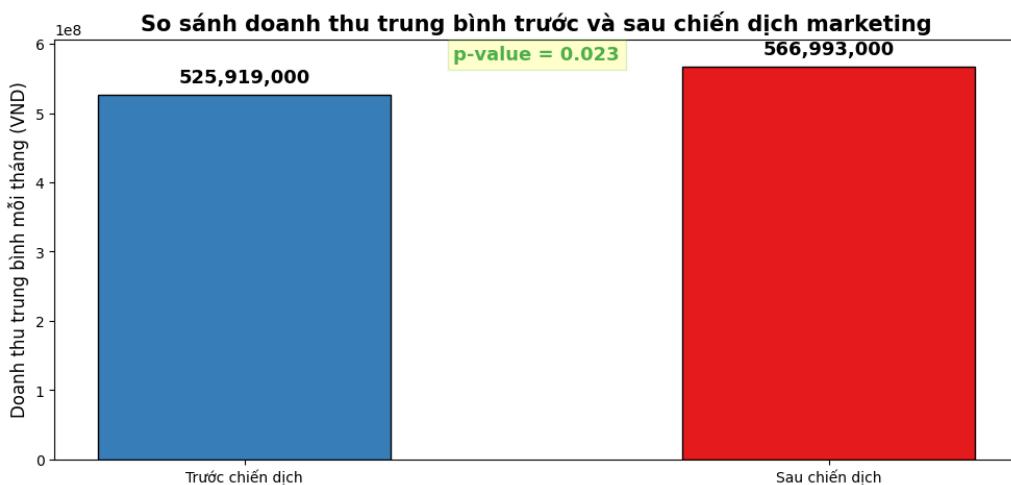
Giả sử một công ty tiến hành chiến dịch marketing mới và muốn biết liệu chiến dịch này có giúp tăng doanh số hay không.

- **Trước chiến dịch:** Doanh thu trung bình mỗi tháng là 520 triệu đồng.
- **Sau chiến dịch:** Doanh thu trung bình tăng lên 567 triệu đồng.
- **Kết quả kiểm định:** Giá trị $p\text{-value}$ thu được là 0.023.

Vì $p\text{-value} = 0.023 < 0.05 = \alpha$, ta bác bỏ giả thuyết H_0 (“không có sự khác biệt”). **Kết luận:** Với mức ý nghĩa 5%, có đủ bằng chứng thống kê để khẳng định chiến dịch marketing mới **thực sự giúp tăng doanh số trung bình**.

Bảng 6: Doanh thu trước và sau chiến dịch marketing (Đơn vị: VND)

Tháng	Trước chiến dịch	Sau chiến dịch
Tháng 1	52,993,400	58,532,300
Tháng 2	51,723,500	53,790,800
Tháng 3	53,295,400	54,205,200
Tháng 4	55,046,100	56,763,000
Tháng 5	51,531,700	55,771,800
Tháng 6	51,531,700	58,691,300
Tháng 7	55,158,400	56,002,300
Tháng 8	53,534,900	54,892,900
Tháng 9	51,061,100	61,224,400
Tháng 10	53,085,100	57,503,300
Tháng 11	51,073,200	58,148,600
Tháng 12	51,068,500	54,865,600



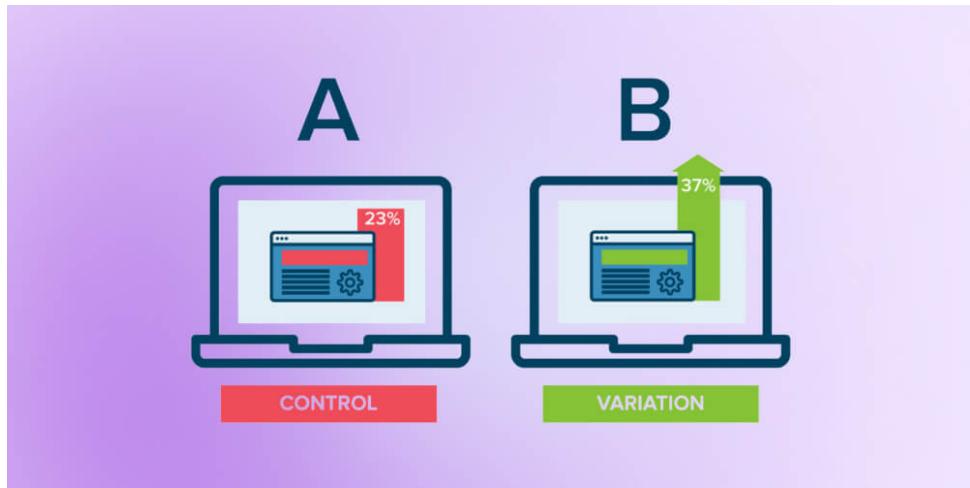
Hình 24: So sánh doanh thu trung bình trước và sau chiến dịch marketing. Kết quả cho thấy doanh thu trung bình tăng từ **525,919,000 VND** lên **566,993,000 VND** sau chiến dịch. Giá trị *p*-value = 0.023 (< 0.05) chứng tỏ sự khác biệt này có ý nghĩa thống kê, tức là chiến dịch marketing thực sự mang lại hiệu quả.

2.3 A/B Testing

A/B Testing là một ứng dụng phổ biến của kiểm định giả thuyết trong thực tế. Ý tưởng chính: so sánh hai phiên bản khác nhau để xem phiên bản nào hiệu quả hơn.

- **Phiên bản A (Control):** Banner quảng cáo lớn, chiếm 70% màn hình đầu tiên, kèm nút “Mua ngay”.
- **Phiên bản B (Variation):** Banner nhỏ hơn, kèm đánh giá 5 sao từ khách hàng, và hiển thị sản phẩm bán chạy ngay phía dưới.
- **Quy trình:** Người dùng được chia ngẫu nhiên thành hai nhóm để đảm bảo tính khách quan.

- **Kết quả đo lường:** Tỷ lệ chuyển đổi (conversion rate) được tính toán cho cả hai phiên bản.
- **Phân tích:** Nếu p-value < 0.05, ta bác bỏ H_0 và chọn phiên bản có tỷ lệ chuyển đổi cao hơn.



Hình 25: Kết quả A/B Testing: Phiên bản A (Control) đạt tỷ lệ chuyển đổi 23%, trong khi phiên bản B (Variation) đạt 37%. Sự khác biệt này cho thấy phiên bản B có hiệu quả cao hơn và được ưu tiên triển khai.

2.4 Các loại kiểm định phổ biến

Trong phân tích dữ liệu, việc chọn đúng loại kiểm định thống kê phụ thuộc vào đặc điểm dữ liệu và mục tiêu nghiên cứu. Một số kiểm định phổ biến trong Excel và thống kê ứng dụng gồm:

- **t-Test (Kiểm định t):** Dùng để so sánh *giá trị trung bình của 2 nhóm*. Ví dụ: so sánh doanh thu trung bình của cửa hàng A và cửa hàng B. Trong Excel có các dạng: *one-sample*, *paired*, và *independent two-sample*.
- **ANOVA (Analysis of Variance):** Dùng để so sánh *trung bình của nhiều nhóm cùng lúc* (lớn hơn 2 nhóm). Ví dụ: so sánh doanh số trung bình giữa 3 khu vực *Miền Bắc, Miền Trung, Miền Nam*. Nếu ANOVA cho kết quả có khác biệt, ta cần làm thêm kiểm định hậu nghiệm (post-hoc test) để biết nhóm nào khác biệt.
- **Chi-Square Test (Kiểm định Chi-bình phương):** Dùng cho dữ liệu phân loại, nhằm kiểm tra xem *hai biến có độc lập với nhau hay không*. Ví dụ: kiểm tra xem việc khách hàng chọn loại sản phẩm (điện thoại, laptop, tablet) có độc lập với giới tính hay không.
- **Non-parametric tests (Kiểm định phi tham số):** Dùng khi dữ liệu không tuân theo phân phối chuẩn hoặc kích thước mẫu nhỏ.
 - *Mann-Whitney U*: thay thế cho t-test khi so sánh 2 nhóm độc lập.
 - *Wilcoxon Signed-Rank*: thay thế cho paired t-test (so sánh dữ liệu theo cặp).
 - *Kruskal-Wallis*: thay thế cho ANOVA khi so sánh nhiều nhóm.

3. Tiền xử lý dữ liệu & Hồi quy tuyến tính

3.1 Tại sao cần tiền xử lý?

Trong thực tế, dữ liệu thu thập được thường không hoàn hảo. Có thể tồn tại giá trị bị thiếu, số liệu nhập sai, ký hiệu không đồng nhất hoặc những điểm ngoại lệ bất thường. Nếu không xử lý trước khi phân tích, những lỗi này sẽ dẫn đến kết quả sai lệch, làm giảm độ tin cậy của mô hình.

Tiền xử lý dữ liệu (*data preprocessing*) đóng vai trò quan trọng trong mọi bài toán phân tích, với các lợi ích chính:

- Xử lý giá trị thiếu:** Ví dụ, một cột doanh thu bị bỏ trống ở vài tháng. Nếu không bổ sung hoặc ước lượng, kết quả trung bình sẽ sai lệch.
- Phát hiện và loại bỏ ngoại lệ (outliers):** Những giá trị quá lớn/nhỏ bất thường (ví dụ: doanh thu âm, hoặc tăng đột biến không hợp lý) có thể làm méo phân tích.
- Chuẩn hóa định dạng dữ liệu:** Cùng một thông tin nhưng được ghi khác nhau (“MB” và “Miền Bắc”) cần được đồng nhất.
- Tạo biến giả (Dummy variables):** Khi làm việc với dữ liệu phân loại (ví dụ: khu vực Miền Bắc, Miền Nam, Miền Trung), cần chuyển đổi thành dạng số học (0/1) để có thể đưa vào mô hình hồi quy.
- Tăng độ chính xác của phân tích:** Dữ liệu sau khi được làm sạch và chuẩn hóa sẽ giúp mô hình hồi quy hoạt động ổn định hơn, cho kết quả dễ tin cậy và dễ giải thích.

Bảng 7: Bảng dữ liệu trước và sau xử lý

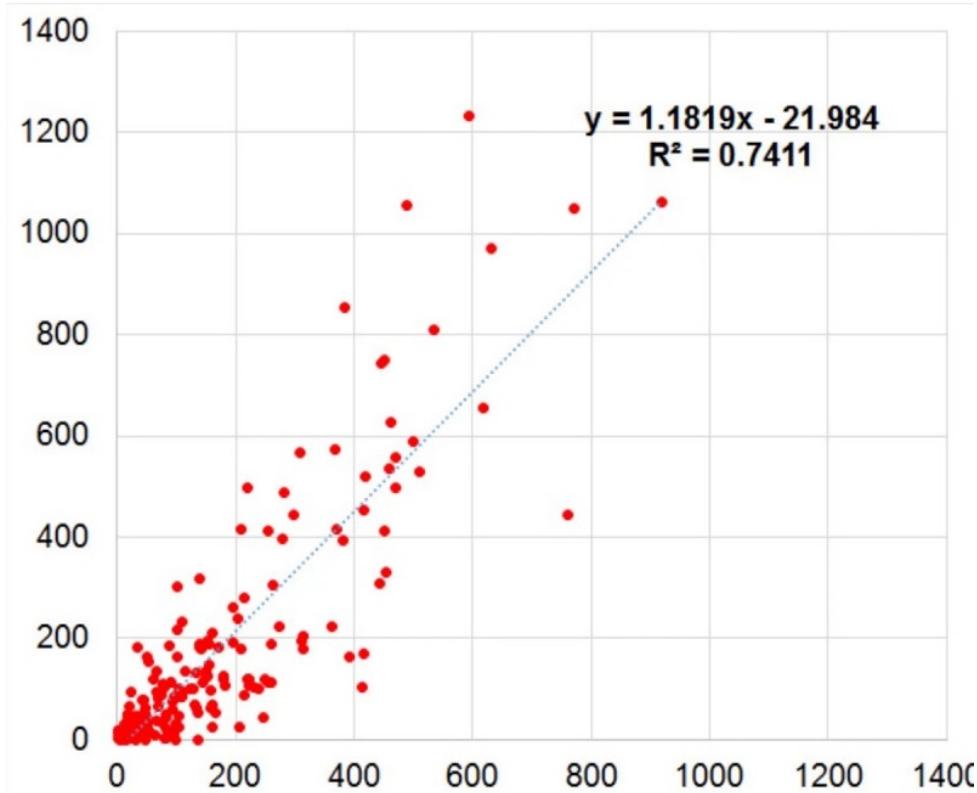
Dữ liệu thô (trước xử lý)				Dữ liệu sau xử lý		
Tháng	Doanh số (triệu VND)	Khu vực	Ghi chú	Tháng	Doanh số (triệu VND)	
T1/2023	45.2	Miền Bắc	Đầy đủ	T1/2023	45.2	
T2/2023	NULL	Miền Nam	Thiếu dữ liệu	T2/2023	48.95	
T3/2023	52.7	Miền Bắc	Đầy đủ	T3/2023	52.7	
T4/2023	198.5	Miền Bắc	Nghi ngờ sai sót	T4/2023	54.8	
T5/2023	54.8	MB	Đầy đủ	T5/2023	54.8	
T6/2023	-12.3	Miền Trung	Giá trị âm	T6/2023	53.7	

3.2 Các kỹ thuật phổ biến trong Excel

Trong Excel, có nhiều công cụ và hàm hỗ trợ cho việc tiền xử lý dữ liệu. Một số kỹ thuật quan trọng thường dùng là:

- Xử lý giá trị thiếu:** Sử dụng các hàm IFERROR(), IF(ISBLANK()), AVERAGE(), MEDIAN() để thay thế giá trị bị thiếu. Ví dụ: nếu một tháng doanh thu bị bỏ trống, ta có thể điền bằng giá trị trung bình của các tháng liền kề.
- Phát hiện ngoại lệ (Outliers):** Dùng hàm QUARTILE() để tính các tứ phân vị, kết hợp với quy tắc IQR để phát hiện điểm bất thường. Ngoài ra có thể chuẩn hóa bằng Z-score hoặc dùng biểu đồ Box Plot để trực quan hóa ngoại lệ.

- **Tạo biến giả (Dummy Variables)**: Khi có dữ liệu phân loại (ví dụ: khu vực *Miền Bắc*, *Miền Trung*, *Miền Nam*), ta cần chuyển thành biến nhị phân (0/1) để đưa vào phân tích hồi quy. Excel có thể thực hiện bằng IF(), SWITCH() hoặc Power Query.
- **Chuẩn hóa dữ liệu**: Để đưa các biến về cùng một thang đo, có thể dùng hàm STANDARDIZE(), hoặc biến đổi dữ liệu bằng LOG(), SQRT() khi phân phối dữ liệu bị lệch nhiều.
- **R^2 (hệ số xác định)**: Cho biết mức độ mô hình giải thích được biến động của biến phụ thuộc. Ví dụ: $R^2 = 0.8$ nghĩa là 80% sự thay đổi của doanh thu có thể được giải thích bởi các biến đầu vào (như marketing, khuyến mãi...).
- **p-value**: Dùng để đánh giá mức ý nghĩa thống kê của từng biến độc lập. - Nếu p-value < 0.05 , biến có ảnh hưởng đáng kể đến Y . - Nếu p-value ≥ 0.05 , ảnh hưởng của biến không rõ ràng, có thể bỏ qua trong mô hình.
- **Hồi quy đơn biến (Simple Linear Regression)**: Mô hình chỉ xét một biến độc lập. Ví dụ: phân tích mối quan hệ giữa **chi phí marketing** và **doanh thu**. Kết quả có thể cho thấy: cứ tăng 1 triệu đồng chi phí marketing thì doanh thu trung bình tăng thêm khoảng 50 triệu đồng.
- **Hồi quy đa biến (Multiple Linear Regression)**: Mô hình xét nhiều biến độc lập cùng lúc để dự đoán biến phụ thuộc. Ví dụ: doanh thu chịu ảnh hưởng đồng thời bởi **chi phí marketing**, **tỉ lệ khuyến mãi**, và **vị trí cửa hàng**. Mô hình này phản ánh thực tế hơn, vì kết quả kinh doanh hiếm khi chỉ phụ thuộc vào một yếu tố.



Hình 26: Biểu đồ Scatter Plot kèm đường hồi quy tuyến tính. Mỗi điểm đỏ biểu diễn một cặp giá trị (x, y) . Đường gạch chấm là đường hồi quy với phương trình $y = 1.1819x - 21.984$ và hệ số xác định $R^2 = 0.7411$, cho thấy khoảng 74.1% biến động của y có thể được giải thích bởi x .

Kết luận

Qua bài viết, chúng ta đã lần lượt khám phá ba khía cạnh quan trọng trong phân tích dữ liệu với Excel:

- **Trực quan hóa dữ liệu:** Giúp biến những con số khô khan thành biểu đồ, từ đó nhanh chóng nhận diện xu hướng và truyền đạt thông tin một cách trực quan.
- **Kiểm định giả thuyết & A/B Testing:** Cung cấp cơ sở thống kê để ra quyết định khách quan, thay thế cảm tính bằng bằng chứng định lượng, đồng thời ứng dụng trực tiếp trong các tình huống thực tế như so sánh chiến dịch marketing.
- **Tiền xử lý dữ liệu & Hồi quy tuyến tính:** Làm sạch dữ liệu, loại bỏ sai lệch và xây dựng mô hình dự báo, từ đó phân tích mối quan hệ nhân quả và hỗ trợ lập kế hoạch chiến lược.

Từ góc nhìn này, Excel không chỉ dừng lại ở vai trò một công cụ bảng tính, mà thực sự trở thành một **nền tảng phân tích dữ liệu toàn diện**. Khi nắm vững các kỹ năng từ trực quan hóa, kiểm định thống kê đến xây dựng mô hình hồi quy, bạn hoàn toàn có thể sử dụng Excel để đưa ra **quyết định dựa trên dữ liệu (data-driven decisions)** thay vì dựa trên cảm tính hay kinh nghiệm chủ quan.