XGBoost - Từ cơ sở lý thuyết đến ứng dụng thực tế

1st Nguyễn Dương Thanh Dự
Trường Đại học Công nghiệp TP.HCM
Hồ Chí Minh, Việt Nam

2nd Nguyễn Đình Thanh
Trường Đại học Công nghiệp TP.HCM
Hồ Chí Minh, Việt Nam

3rd Cao Nguyễn Gia Hưng
Trường Đại học Công nghiệp TP.HCM
Hồ Chí Minh, Việt Nam

TÓM LƯỢC

Bài viết này tập trung vào giải thích lý thuyết của thuật toán XGBoost (xgb). XGBoost được xem như một phương pháp tăng cường (boosting), nghĩa là xây dựng một loạt các ước lượng yếu trên dữ liệu và tích lũy chúng qua nhiều lần lặp. Mỗi bộ phân loại trong XGBoost là cây CART (Classification And Regression Tree). Do đó, mô hình cây có hiệu quả tốt trong việc tạo ra các tương tác giữa các biến, nhưng cũng dễ bị quá khớp (overfitting).

Bên cạnh đó cũng giải thích qua quá trình điều chỉnh siêu tham số trong XGBoost. Quá trình tìm kiếm siêu tham số có thể được thực hiện thông qua lưới tìm kiếm (grid search) để đơn giản hóa quá trình điều chỉnh siêu tham số của XGBoost.

Keywords: XGBoost, Gradient Boosting, Grid Search, Tuning Parameter

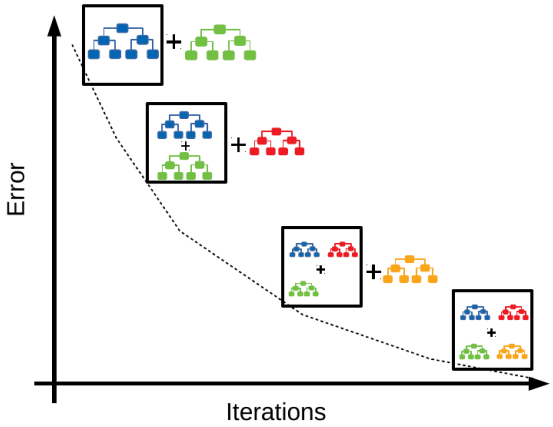
I. GIỚI THIỆU

Mặc dù GBDT đã áp dụng các biện pháp chính quy hóa như tỷ lệ học (learning rate), số lượng cây, tỷ lệ lấy mẫu con (subsample), và các tham số cấu trúc cây để tránh quá khớp, nhưng liệu có phương pháp chính quy hóa tốt hơn không? Ngoài ra, phương pháp huấn luyện tuần tự của GBDT tạo ra một chi phí tính toán lớn, liệu chúng ta có thể tối ưu từ các khía cạnh khác để tăng tốc quá trình huấn luyện mô hình không? Đáp án là có, vào năm 2016, Tianqi Chen đã công bố XGBoost [1], ông đã tối ưu hóa cả lý thuyết và thực hiện của thuật toán GBDT, nhưng bản chất của XGBoost vẫn là một mô hình tăng cường. Tiếp theo, tôi sẽ trình bày về hai khía cạnh của XGBoost: "Tối ưu lý thuyết" và "Tối ưu thực hiện".

XGBoost (Extreme Gradient Boosting) là một trong những thuật toán tăng cường (boosting) tiên tiến nhất trong Machine Learning và Data Science, nó được phát triển bởi Chen và Guestrin vào năm 2016. XGBoost là một phần của họ cây (Decision Tree, Random Forest, Bagging, Boosting, Gradient Boosting).

Yếu tố quan trọng nhất đằng sau sự thành công của XGBoost là khả năng mở rộng của nó trong mọi tình huống. Hệ thống này chạy nhanh hơn gấp mười lần so với các giải pháp phổ biến hiện có trên một máy duy nhất và chia tỷ lệ thành hàng tỷ ví dụ trong cài đặt phân tán hoặc giới hạn bộ nhớ. Khả năng mở rộng của XGBoost là do một số hệ thống quan trọng và tối ưu hóa thuật toán. Những đổi mới này bao gồm: một thuật toán học cây mới dùng để xử lý dữ liệu thưa thớt; một quy trình phác thảo lượng tử có trọng số hợp lý về mặt lý thuyết cho phép xử lý các trọng số của cá thể trong quá trình học cây gần đúng.

XGBoost cũng cung cấp một số tính năng quan trọng để kiểm soát quá khớp (overfitting) và tối ưu hóa hiệu suất. Các tính năng như chặn cắt cây (tree pruning), tỷ lệ học (learning rate), điều chỉnh tham số và kiểm tra tăng cường (regularization) giúp đảm bảo mô hình không quá phức tạp và đạt hiệu suất tốt trên tập dữ liệu kiểm tra. Thuật toán này áp dụng thành công trong nhiều lĩnh vực và bài toán khác nhau, bao gồm dự đoán giá cổ phiếu, phân loại email rác,



Hình 1: Mô hình cây của theo Boosting

xếp hạng trang web và nhiều bài toán khác. Nhờ hiệu suất ấn tượng và khả năng tùy chỉnh linh hoạt, XGBoost đã trở thành một công cụ quan trọng trong công cuộc khai phá dữ liệu và xây dựng mô hình học máy.

Mặc dù đã có nhiều công trình liên quan đến thuật toán này, nhưng các hướng đi như tính toán ngoài bộ nhớ, việc học nhạy với bộ nhớ đệm và tính thưa chưa được khám phá. Quan trọng hơn, một hệ thống end-to-end kết hợp tất cả các khía cạnh này mang lại một giải pháp mới cho các trường hợp sử dụng trong thực tế. Điều này cho phép các nhà khoa học dữ liệu cũng như các nhà nghiên cứu xây dựng các biến thể mạnh mẽ của thuật toán tăng cường cây [7, 8]. Bên cạnh những đóng góp chính này, chúng tôi cũng có những cải tiến bổ sung trong việc đề xuất một mục tiêu học tập được điều chỉnh, mà chúng tôi sẽ bao gồm để hoàn thiện. Phần còn lại của bài báo được tổ chức như sau. Chúng tôi sẽ trước tiên xem xét lại tăng cường cây và giới thiệu mục tiêu được điều chỉnh trong Mục 2. Sau đó, chúng tôi mô tả các phương pháp tìm điểm chia trong Mục 3 cũng như thiết kế hệ thống trong Mục 4, bao gồm các kết quả thực nghiệm khi có liên quan để cung cấp hỗ trợ định lượng cho mỗi tối ưu hóa chúng tôi mô tả. Công trình liên quan được thảo luận trong Mục 5. Đánh giá chi tiết từ đầu đến cuối được bao gồm trong Mục 6. Cuối cùng, chúng tôi kết luận bài báo trong Mục 7.

II. GIỚI THIỆU SƠ LƯỢC VỀ ENSEMBLE LEARNING

A. *Bagging*

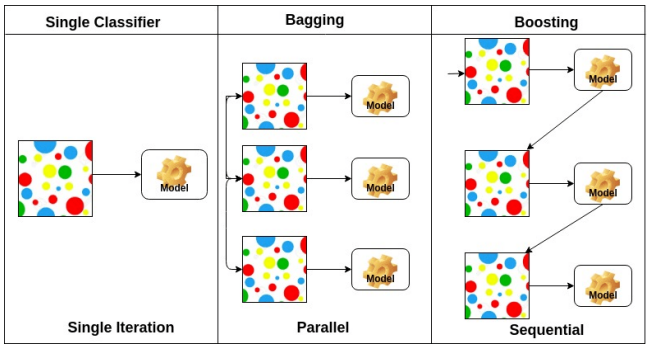
Các mô hình con được huấn luyện độc lập với nhau bằng cách sử dụng một hàm mục tiêu đơn giản như Cross-entropy loss hoặc Mean Squared Error. Khi kết hợp các dự đoán của các mô hình con lại, thường sử dụng phương pháp đa số đếm (majority voting) để đưa ra quyết định cuối cùng.

B. *Boosting*

Trong Boosting, các mô hình con được huấn luyện tuần tự, trong đó mỗi mô hình đặt trọng số cao hơn cho các điểm dữ liệu bị phân

loại sai bởi các mô hình trước đó. Việc chọn hàm mục tiêu trong trường hợp này rất quan trọng vì nó liên quan đến cách cập nhật trọng số và hướng dẫn cho quá trình tối ưu hoá.

Boosting thường được sử dụng để cải thiện hiệu suất dự đoán cho các mô hình yếu, yêu cầu phần lớn các mô hình con đóng vai trò thay thế. Nó tỏ ra rất hiệu quả khi áp dụng cho các thuật toán học máy yếu, chẳng hạn như cây quyết định (decision tree) hay k-nearest neighbors (k-NN), và có thể đưa ra dự đoán chính xác hơn với mức độ chính xác cao hơn.



Hình 2: Mô hình cây Ensemble của Bagging và Boosting.

C. *Stacking*

Trong Stacking, các mô hình con đầu tiên được huấn luyện trên toàn bộ tập dữ liệu và tạo ra các dự đoán riêng biệt.Thông thường, nó được sử dụng để kết hợp các mô hình có tính chất hoàn toàn khác nhau, đòi hỏi mạng lưới mô hình phức tạp và có khả năng học cả các đặc trưng tinh vi và phức tạp. Cụ thể, Stacking là phương pháp phù hợp để kết hợp dự đoán từ các mô hình khác nhau, đối với các bài toán dự đoán phức tạp mà một mô hình đơn lẻ không đáp ứng được.

III. TỔNG QUAN TĂNG CƯỜNG CÂY

A. *Regularied Learning Objective*

Cho một tập dữ liệu đã cho với n mẫu và m đặc trưng $D = \{f(x_i, y_i)\}$ $(|D| = n; x_i \in \mathbb{R}^m; y_i \in \mathbb{R})$. Một mô hình cây kết hợp ensemble model. Sử dụng K cây hồi quy để dự đoán đầu ra.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F \tag{1}$$

trong đó $F = \{f(x) = w_{q(x)}\}$ $(q : \mathbb{R}^m \to T; w \in \mathbb{R}^T)$ là không gian cây quyết định. Ở đây q đại diện cho cấu trúc của cây và $q(x)$ sẽ chả về index của nút lá trong cây đó. T là số lượng nút lá có trong cây. Mỗi hàm f_k phụ thuộc vào cấu trúc cây q và trọng số nút lá w. Khác với cây quyết định, Mỗi cây hồi quy có một số hạng trên mỗi nút là, ở đây ta sẽ sử dụng w_i, để đại diễn điểm số trên nút lá thứ i.

B. *Hàm mất mát của XGBoost*

Để xây dựng một mô hình ổn định, chúng ta cần xem xét cả sai số và phương sai của mô hình. Tuy nhiên, việc giảm tối thiểu hóa hàm mất mát chỉ giúp giảm thiểu sai số mà không đảm bảo rằng mô hình sẽ có khả năng tổng quát hóa trên các tập dữ liệu mới. Vì vậy, chúng ta cần sử dụng regularization để kiểm soát phương sai của mô hình. Cho nên hàm mục tiêu có thể định nghĩa như sau:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{2}$$

$$\begin{aligned} \text{trong đó } \Omega(f) &= \gamma T + \frac{1}{2}\lambda \|w\|_2^2 \\ &= \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \end{aligned}$$

Ở đây hàm l là một hàm mất mát lồi khả vi, đo lường sự khác biệt giữa dự đoán $\hat{y}_i$ và giá trị thực y_i. Biểu thức thứ hai $\Omega(f)$ đo độ phức tạp của cây. Hàm chính quy hóa (regularized objective) được đưa vào để tránh quá khớp (overfitting).

Nếu xây dựng các mô hình dự đoán ta cần tối ưu hóa hàm Traning loss. Nếu xây dựng mô hình dữ đoán đơn giản thì ta tối ưu hàm Regularized

C. *Tối ưu hàm mất mát*

Chúng ta cần f_t để tối ưu hàm mục tiêu, đặt $\hat{y}_i^{(t)}$ là dự đoán của trường hợp thứ i ở lần lặp thứ t. Và mô hình được đào tạo theo Addtive Training của phương pháp boosting theo lượt đồ sau:

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

$$\ldots$$

$$\Rightarrow \hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{3}$$

Trong vòng lặp thứ t, chúng ta chỉ quan tâm đến hàm mục tiêu của vòng lặp này, và giả định rằng kết quả huấn luyện của vòng lặp $t-1$ đã được biết trước là $y_i^{(t-1)}$. Do đó, hàm mục tiêu có thể được viết lại dưới dạng sau:

$$\begin{aligned} L^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{yi}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \end{aligned} \tag{4}$$

Để tìm giá trị tối ưu của hàm mục tiêu tại thời điểm này tương đương với việc tìm giá trị tối ưu của $f(x)$ là rất khó, vì vậy ta có thể sử dụng công thức triển khai Taylor theo biến thứ 2 để xấp xỉ hàm này:

$$f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2 \tag{5}$$

Ta có thể xấp xỉ hàm mục tiêu như sau:

$$\begin{aligned} L^t &\simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega f(t) \\ &\simeq \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega f(t) \end{aligned} \tag{6}$$

Lý do cho việc sử dụng giá trị $\hat{y}^{(t-1)}$ vì trong quá trình xây dựng chuỗi các cây quyết định, mỗi cây mới được huấn luyện để tối ưu hóa sai số còn lại giữa giá trị dự đoán và giá trị thật sự của y_i tức là $r_i^t = y_i - y_i^{t-1}$. Do đó, khi xây dựng cây mới tại bước lặp t, chúng ta sẽ sử dụng giá trị y_i^{t-1} làm đầu vào (input) để dự đoán sai số còn lại r_i^t, thay vì giá trị y_i ban đầu. Vì vậy, khi xấp xỉ hàm mất mát, chúng ta sử dụng giá trị y_i^{t-1} để tính toán sai số còn lại r_i^t

Lúc này:

$$g_i = f^{'}(x) = \frac{\partial l(y_i, \hat{y}_i^{t-1})}{\partial \hat{y}_i^{t-1}}$$
$$h_i = f^{''}(x) = \frac{\partial^2 l(y_i, \hat{y}_i^{t-1})}{\partial (\hat{y}_i^{t-1})^2}$$

Định nghĩa tập nút lá j là $I_j = \{i|q(x_i) = j\}$. Và mục đích của ta sau khi xấp xỉ hàm này về một hàm dễ tính toán hơn thì ta có thể quan sát thấy tại bước t, ta đã biết giá trị $y^{(t-1)}$ nên $l(y_i, y_i^{(t-1)})$ là 1 hằng số, mà hằng số thì không ảnh hưởng đến quá trình tìm nghiệm tối ưu của hàm mục tiêu nên có thể lược bỏ đi.

$$\begin{aligned} L^{(t)} &= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega f(t) \\ &= \sum_{i=1}^{n} [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \end{aligned} \tag{7}$$

Đặt $G_j = \sum_{i \in I_j} g_i$, $Hj = \sum_{i \in I_j} h_i$

$$L^{(t)} = \sum_{j=1}^{T} (G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 + \gamma T$$

Thực hiện đạo hàm $L^{(t)}$ theo biến w_j và cho đạo hàm bằng 0,

$$(L^{(t)})' = \sum_{j=1}^{T} (G_j + (H_j + \lambda) w_j) \tag{8}$$

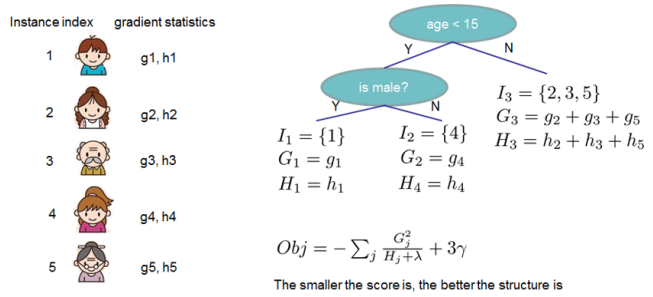
$\Rightarrow w_j^* = -\frac{G_j}{H_j + \lambda}$

Thế nghiệm tối ưu vào (8), ta có:

$$L^{(t)}(w_j^*) = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \tag{9}$$

Công thức (9) có thể được dùng để làm hàm để đo chất lượng của cấu trúc cây q. Điểm số này giống như độ vẩn đục (impurity) trong Cây Quyết Định (Decision Tree) khác ở cái nó được suy ra cho nhiều hàm mục tiêu hơn.

Công thức này thường được sử dụng trong thực tế để đánh giá các phần dư thừa cần tách.



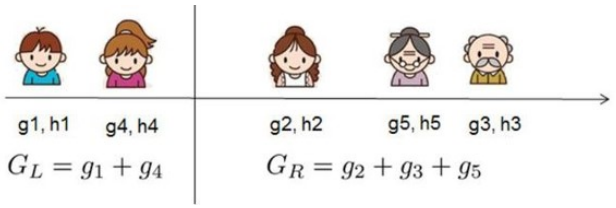
Hình 3: Tính toán điểm của cấu trúc cây.

IV. MỘT SỐ THUẬT TOÁN TÌM KIẾM PHẦN CẮT

A. *Thuật toán Basic Exact Greedy*

Đây là thuật toán cơ bản và chính xác để tìm điểm chia (split) trong quá trình xây dựng cây trong XGBoost. Thuật toán này tìm kiếm qua tất cả các giá trị của mỗi đặc trưng (feature) và tính toán tổng lỗi (loss) của cây con được tạo ra bằng cách chia dữ liệu theo từng giá trị. Sau đó, thuật toán chọn điểm chia có tổng lỗi nhỏ nhất để tiếp tục xây dựng cây. Công thức dưới đây mô tả việc tính toán tổng lỗi của cây con.

$$\begin{aligned} Gain &= L - (L_L + L_R) \\ &= \frac{1}{2} [\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda}] - \gamma \\ &= \frac{1}{2} [\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I_I} g_i)^2}{\sum_{i \in I_I} h_i + \lambda}] - \gamma \\ &= \frac{1}{2} [\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\sum_{i \in I_L} g_i + \sum_{i \in I_R} g_i}{\sum_{i \in I_L} h_i + \sum_{i \in I_R} h_i + \lambda}] - \gamma \\ &= \frac{1}{2} [\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma \end{aligned}$$



Hình 4: Tính toán Gain.

Algorithm 1 Exact Greedy Algorithm

Input: I, tập hợp các node lá hiện tại
Input: d, kích thước của đặc trưng
1: $Gain = 0$, $G = \sum_{i \in I} g_i$, H = $\sum_{i \in I} h_i$
2: **for** $k = 1$ to m **do**
3: $G_L = 0$, $H_L = 0$
4: **for** j in sorted(I, by x_{jk}) **do**
5: $G_L = G_L + g_j, H_L = H_L + h_j$
6: $G_R = G - G_L, H_R = H - H_L$
7: score = max(score, $\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}$)
8: **end for**
9: **end for**
Output: Phần tách có score lớn nhất

B. *Thuật toán Approximate*

Đối với các tập dữ liệu rất lớn, thuật toán cơ bản trên có thể trở nên chậm do tính toán tổng lỗi cho tất cả các giá trị của đặc trưng. Do đó, paper cung cấp một thuật toán xấp xỉ để tìm điểm chia nhanh hơn. Cách tiếp cận này sử dụng phương pháp lấy mẫu ngẫu nhiên một số điểm dữ liệu và tính toán tổng lỗi xấp xỉ từ mẫu này. Công thức xấp xỉ để tính toán tổng lỗi:

Algorithm 2 Approximate Algorithm

for $k = 1$ to m **do**
  Đề suất $S_k = s_{k1}, s_{k2}, ..., s_{kl}$ theo phần trăm trên đặc trưng k
  Đề suất có thể thực hiện trên mỗi cây (global), hoặc mỗi phần của cây (local)
end for
for $k = 1$ to m **do**
  $G_{kv} = \sum_{j \in \{j | s_{k,v} \leq x_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} = \sum_{j \in \{j | s_{k,v} \leq x_{jk} > s_{k,v-1}\}} h_j$
end for
Thực hiện theo bước tương tự như trong phần trước để chỉ tìm điểm tối đa trong số các phần tách được đề xuất.

C. *Weighted Quantile Sketch*

Trong quá trình tìm điểm cắt, việc tính toán các phân vị (quantile) của các giá trị đặc trưng là một phép toán phức tạp và tốn nhiều thời gian. Đặc biệt, khi có hàng triệu hoặc hàng tỷ mẫu dữ liệu, việc tính toán chính xác các phân vị trở nên không khả thi.

Weighted Quantile Sketch là một phương pháp xấp xỉ để tính toán các phân vị của dữ liệu. Nó sử dụng các kỹ thuật tóm tắt dựa trên trọng số để ước lượng các phân vị một cách nhanh chóng và chính xác đủ cho quá trình tìm điểm cắt. Phương pháp này giúp giảm đáng kể thời gian tính toán và tăng tốc quá trình xây dựng cây.

D. *Sparsity-aware Split Finding*

Khi dữ liệu có tính thưa (sparse), tức là có rất nhiều giá trị bằng không, việc xác định điểm chia truyền thống có thể làm mất nhiều thời gian và không hiệu quả. Với thuật toán này, mô hình sẽ tìm điểm cắt dựa trên việc nhận biết sự thưa thớt của dữ liệu. Để có thể làm cho thuật toán nhận biết được mô hình thưa thớt trong dữ liệu, ta thêm hướng mặc định trong mỗi nút của cây, khi một giá trị bị thiếu instance được phân loại theo hướng mặc định. Thay vì kiểm tra tất cả các giá trị, thuật toán chỉ xem xét những giá trị không bằng không và tính toán tổng lỗi tương ứng, từ đó tận dụng tính thưa thớt để tăng tốc quá trình tìm kiếm điểm cắt.

Algorithm 3 Sparsity-aware

Input: I, tập hợp gồm các node lá
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, kích thước của đặc trưng
$Gain = 0$, $G = \sum_{i \in I} g_i$, H = $\sum_{i \in I} h_i$
for $k = 1$ to m **do**
  $G_L = 0$, $H_L = 0$
  for j in sorted(I, thứ tự tăng theo x_{jk}) **do**
    $G_L = G_L + g_j, H_L = H_L + h_j$
    $G_R = G - G_L, H_R = H - H_L$
    score = max(score, $\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$)
  end for
  $G_R = 0$, $H_R = 0$
  for j in sorted(I, thứ tự giảm theo x_{jk}) **do**
    $G_R = G_R + g_j, H_R = H_R + h_j$
    $G_L = G - G_R, H_L = H - H_R$
    score = max(score, $\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$)
  end for
end for
Output: Split and default directions with max gain

V. ĐẶC ĐIỂM NỔI BẬT CỦA XGBOOST

A. *Khả năng mở rộng*

XGBoost linh hoạt cho các bài toán Machine Learning quy mô lớn. Nó sử dụng kỹ thuật tối ưu và tính toán song song để tận dụng tài nguyên tính toán trên nhiều nền tảng phần cứng, giảm thời gian huấn luyện mô hình đáng kể.

B. *Hiệu suất vượt trội*

XGBoost vượt trội so với các thuật toán Machine Learning khác, đặc biệt là trong xử lý dữ liệu lớn và phức tạp. Nó tạo ra mô hình dự đoán mạnh mẽ và chính xác, đồng thời giảm quá khớp và cải thiện khả năng tổng quát hóa.

C. *Xử lý mất cân bằng dữ liệu*

Tự động tối ưu hóa dữ liệu thưa thớt là một tính năng trong thư viện XGBoost được thiết kế để cải thiện hiệu suất và hiệu quả của các mô hình tăng cường độ dốc khi xử lý các bộ dữ liệu thưa thớt. Dữ liệu thưa đề cập đến các bộ dữ liệu trong đó phần lớn các tính năng có giá trị bằng 0 hoặc rất thấp, dẫn đến một ma trận có nhiều ô trống hoặc gần như trống.

Trong các triển khai tăng cường độ dốc truyền thống, chẳng hạn như XGBoost, các bộ dữ liệu thưa thớt có thể đưa ra những thách thức vì chúng yêu cầu bộ nhớ bổ sung để lưu trữ các ma trận thưa thớt và thời gian tính toán bổ sung để xử lý dữ liệu thưa thớt. Tuy nhiên, Tối ưu hóa dữ liệu thưa thớt tự động của XGBoost giải quyết những thách thức này bằng cách triển khai các thuật toán và kỹ thuật chuyên biệt.

D. *Đa dạng trong ứng dụng*

XGBoost được áp dụng thành công trong nhiều lĩnh vực, từ y tế, phân loại ảnh đến dự báo tài chính và xử lý ngôn ngữ tự nhiên.

VI. THỰC HIỆN DỰ ĐOÁN THỜI TIẾT NGÀY MAI CÓ MƯA

A. *Giới thiệu về dữ liệu*

Dữ liệu được thu thập từ tập tin *weatherAUS.csv* và bao gồm các trường thông tin liên quan đến thời tiết. Dưới đây là mô tả chi tiết về các trường dữ liệu:

Cột	**Ý nghĩa**
Date	Ngày ghi nhận dữ liệu
Location	Vị trí địa lý
MinTemp	Nhiệt độ tối thiểu trong ngày (được biết đến gần nhất)
MaxTemp	Nhiệt độ tối đa trong ngày (được biết đến gần nhất)
Rainfall	Lượng mưa trong 24 giờ đến 9 giờ sáng
Evaporation	Lượng bay hơi "Class A" trong 24 giờ đến 9 giờ sáng
Sunshine	Số giờ nắng trong 24 giờ đến nửa đêm
WindGustDir	Hướng gió mạnh nhất trong ngày
WindGustSpeed	Tốc độ gió mạnh nhất trong ngày
WindDir9am	Hướng gió lúc 9 giờ sáng
Humidity9am	Độ ẩm tương đối lúc 9 giờ sáng
Humidity3pm	Độ ẩm tương đối lúc 3 giờ chiều
Pressure9am	Áp suất khí quyển lúc 9 giờ sáng
Pressure3pm	Áp suất khí quyển lúc 3 giờ chiều
Cloud9am	Phần trăm bầu trời che phủ mây lúc 9 giờ sáng
Cloud3pm	Phần trăm bầu trời che phủ mây lúc 3 giờ chiều
Temp9am	Nhiệt độ lúc 9 giờ sáng
Temp3pm	Nhiệt độ lúc 3 giờ chiều
RainToday	Có mưa trong ngày hay không (Yes/No)
RainTomorrow	Dự báo có mưa vào ngày hôm sau hay không (Yes/No)

Dữ liệu trong tập tin *weatherAUS.csv* cung cấp thông tin về thời tiết được ghi nhận tại các địa điểm khác nhau. Bằng cách phân tích

các trường dữ liệu, chúng ta có thể tìm hiểu về xu hướng thời tiết, sự thay đổi của nhiệt độ, mưa, độ ẩm và nhiều yếu tố khác trong khoảng thời gian quan sát.

Qua việc khám phá và phân tích dữ liệu này, chúng ta có thể tìm ra các mô hình, xu hướng và thông tin quan trọng về thời tiết để phục vụ việc dự báo và hiểu hơn về các yếu tố liên quan đến thời tiết trong tương lai.

B. Phân tích dữ liệu

Sau khi thu thập dữ liệu từ tập tin "weatherAUS.csv", dữ liệu có 142193 dòng và có 24 cột, chúng tôi tiến hành phân tích chi tiết để hiểu rõ hơn về thông tin thời tiết và các yếu tố liên quan. Dưới đây là những kết quả chính mà chúng tôi đã thu được từ quá trình phân tích dữ liệu:

1. Xu hướng nhiệt độ:

- Nhiệt độ tối thiểu trong ngày (MinTemp) và nhiệt độ tối đa trong ngày (MaxTemp) có sự biến thiên đáng kể. Chúng ta đã quan sát được xu hướng tăng/giảm của nhiệt độ theo thời gian.
- Nhiệt độ tối thiểu trong ngày (MinTemp) và nhiệt độ tối đa trong ngày (MaxTemp) có sự biến thiên đáng kể. Chúng ta đã quan sát được xu hướng tăng/giảm của nhiệt độ theo thời gian.

2. Mưa và lượng mưa:

- Trong các 24 giờ đến 9 giờ sáng, có sự biến đổi về lượng mưa (Rainfall).
- Chúng ta đã xác định được các ngày có mưa và tính toán lượng mưa trung bình trong khoảng thời gian quan sát.

3. Độ ẩm:

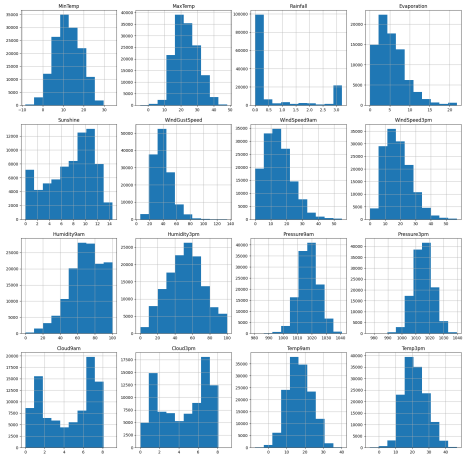
- Độ ẩm tương đối (Humidity) lúc 9 giờ sáng và 3 giờ chiều có sự thay đổi trong ngày.
- Chúng ta đã phân tích độ ẩm để hiểu tác động của nó đến khí hậu và thời tiết.

4. Áp suất khí quyển:

- Áp suất khí quyển (Pressure) lúc 9 giờ sáng và 3 giờ chiều có sự biến đổi.
- Chúng ta đã theo dõi áp suất để xác định các thay đổi trong hệ thống khí quyển.

5. Gió:

- Chúng tôi đã phân tích hướng gió mạnh nhất (WindGustDir) trong ngày và tốc độ gió mạnh nhất (WindGustSpeed).
- Việc theo dõi hướng gió và tốc độ gió giúp chúng tôi hiểu được sự ảnh hưởng của gió đến thời tiết và điều kiện môi trường.



Hình 5: Sự phân bố của dữ liệu

C. Các bước điều chỉnh các siêu tham số cho mô hình XGBoost

Quá trình điều chỉnh siêu tham số bao gồm các bước sau:

1. Xác định siêu tham số cần điều chỉnh: Trước hết, chúng ta cần xác định danh sách các siêu tham số quan trọng trong mô hình XGBoost. Một số siêu tham số quan trọng bao gồm learning rate, số lượng cây (n_estimators), độ sâu của cây (max_depth), và giá trị gamma.

2. Xây dựng lưới siêu tham số: Tiếp theo, chúng ta xác định các giá trị có thể cho mỗi siêu tham số và tạo ra một lưới siêu tham số để thử nghiệm. Việc này giúp chúng ta tìm ra bộ siêu tham số tốt nhất cho mô hình.

3. Áp dụng phương pháp tìm kiếm siêu tham số: Chúng ta có thể sử dụng các phương pháp tìm kiếm như tìm kiếm ngẫu nhiên (random search) hoặc tìm kiếm theo lưới (grid search) để tìm ra bộ siêu tham số tối ưu. Quá trình này đòi hỏi chúng ta đánh giá hiệu suất của mô hình với từng bộ siêu tham số để tìm ra bộ siêu tham số tối ưu dựa trên các tiêu chí như độ chính xác hay độ đo F1-score.

4 .Đánh giá kết quả và lựa chọn siêu tham số tốt nhất: Sau khi hoàn thành quá trình tìm kiếm siêu tham số, chúng ta đánh giá kết quả và chọn bộ siêu tham số tốt nhất dựa trên hiệu suất của mô hình trên tập dữ liệu kiểm tra.

5. Huấn luyện mô hình với siêu tham số tốt nhất: Cuối cùng, chúng ta huấn luyện mô hình XGBoost với bộ siêu tham số tốt nhất đã được xác định từ quá trình điều chỉnh.

D. So sánh với các mô hình khác

Model	Accuracy	ROC_AUC	Time Taken
XGBoost	0.863708	0.756724	62.371878
AdaBoost	0.839270	0.697255	44.050767
Random Forest	0.854179	0.727315	47.557297
Decision Tree	0.788002	0.701318	4.865300
Logistic Regression	0.845845	0.726394	3.259313
KNN	0.796793	0.638380	11.807977
Neural Network	0.847674	0.740840	288.409838

Hình 6: Kết quả So sánh XGboost với các model khác.

Dựa trên kết quả so sánh, mô hình XGBoost có kết quả tốt nhất với độ chính xác cao và giá trị ROC_AUC khá ổn định. Mô hình AdaBoost và Random Forest cũng cho kết quả tốt, trong khi mô hình Decision Tree, Logistic Regression, KNN và Neural Network có hiệu suất khá tốt nhưng thời gian huấn luyện khác nhau.

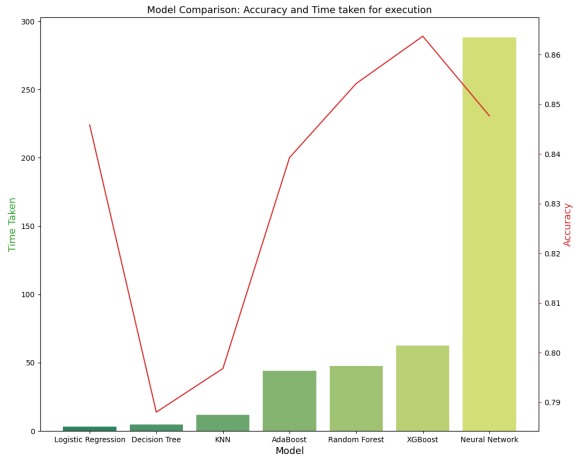
Dựa trên kết quả so sánh hiệu suất của các mô hình, ta có các nhận xét sau:

- Mô hình XGBoost đạt được hiệu suất tốt nhất với độ chính xác cao nhất và giá trị ROC_AUC ổn định. Đây là một lựa chọn tốt để dự đoán và phân loại dữ liệu.

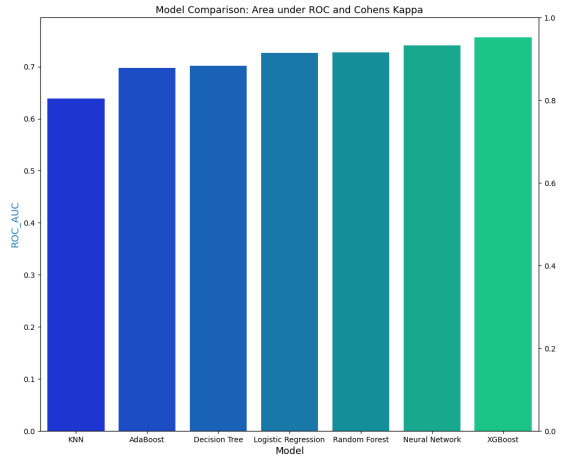
-Mô hình AdaBoost và Random Forest cũng cho kết quả tốt với độ chính xác và giá trị ROC_AUC gần như cao nhất. Đây là các mô hình có khả năng xử lý tốt cả dữ liệu phân loại và dữ liệu dự đoán.

- Mô hình Decision Tree, Logistic Regression và Neural Network đạt được hiệu suất tương đối tốt, nhưng có thời gian huấn luyện và giá trị ROC_AUC khác nhau. Các mô hình này có thể phù hợp cho các bài toán có yêu cầu về thời gian huấn luyện và độ chính xác khác nhau.

- Mô hình KNN cho kết quả độ chính xác thấp hơn so với các mô hình khác, tuy nhiên thời gian huấn luyện vẫn chấp nhận được. Mô hình này có thể được sử dụng cho các bài toán đơn giản và không yêu cầu độ chính xác cao.



Hình 7: Biểu đồ thời gian và độ chính xác của các model.



Hình 8: Biểu đồ roc của các model.

VII. Kết Luận

Trong báo cáo này, chúng tôi đã áp dụng XGBoost - một hệ thống tree boosting hiệu suất cao, để giải quyết bài toán "dự đoán tương lai thời tiết có mưa". Kết quả nghiên cứu đã chứng minh sự hiệu quả và khả năng mạnh mẽ của XGBoost trong việc xử lý bài toán này.

Sử dụng XGBoost, chúng tôi đã đạt được một mô hình dự đoán với độ chính xác cao và khả năng tổng quát tốt. XGBoost đã cho phép chúng tôi tạo ra các cây quyết định mạnh mẽ và tinh vi, tận dụng cả các thông tin quan trọng và không gian đặc trưng hiệu quả. Điều này đã cung cấp một phương pháp mạnh mẽ để dự đoán và phân loại dữ liệu "dự đoán tương lai thời tiết có mưa".

Đồng thời, XGBoost cũng đã cung cấp khả năng điều chỉnh siêu tham số linh hoạt, cho phép chúng tôi tối ưu hóa hiệu suất của mô hình dự đoán. Chúng tôi đã thực hiện việc điều chỉnh các siêu tham số và tìm ra các giá trị tối ưu, từ đó cải thiện kết quả dự đoán và giảm thiểu overfitting.

Kết quả nghiên cứu của chúng tôi đã xác nhận rằng XGBoost là một công cụ mạnh mẽ và hiệu quả cho các bài toán dự đoán và phân loại trong lĩnh vực dự đoán phân loại. XGBoost không chỉ cung cấp kết quả tốt mà còn cho phép chúng tôi hiểu rõ hơn về tác động của các đặc trưng và ảnh hưởng của các siêu tham số đến kết quả dự đoán.

Tổng quát, sử dụng XGBoost đã cung cấp cho chúng tôi một phương pháp mạnh mẽ để giải quyết bài toán "dự đoán tương lai thời tiết có mưa" trong nghiên cứu của chúng tôi. Điều này mở ra nhiều cơ hội cho ứng dụng XGBoost trong các lĩnh vực khác và tạo nền tảng cho sự tiến bộ trong các nghiên cứu tương lai.

VIII. Tài liệu tham khảo

[1]. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

[2]. XGBoost Documentation. (n.d.). Retrieved from https://xgboost.readthedocs.io/en/latest/

[3]. XGBoost GitHub Repository. (n.d.). Retrieved from https://github.com/dmlc/xgboost

[4]. Chen, T., & He, T. (2020). XGBoost: A Scalable Tree Boosting System. arXiv preprint arXiv:1603.02754.

[5]. XGBoost GitHub Wiki. (n.d.). Retrieved from https://github.com/dmlc/xgboost/wiki

[6]. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189-1232.

[7]. XGBoost R Package Documentation. (n.d.). Retrieved from https://cran.r-project.org/web/packages/xgboost/index.html

[8]. XGBoost Python Package Documentation. (n.d.). Retrieved from https://xgboost.readthedocs.io/en/latest/python/index.html

[9]. XGBoost Resources on Kaggle. (n.d.). Retrieved from https://www.kaggle.com/learn/xgboost