 **TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**ĐỒ ÁN 3**

**ĐỀ TÀI:**

**GRADIENT BOOSTING DECISION TREE**

|  |  |
| --- | --- |
| **GVHD: Nguyễn Thiên Bảo** |  |
| **Lê Mạnh Hùng** | **15110058** |
| **Nguyễn Đức Quan** | **15110105** |

**Tp. Hồ Chí Minh, tháng 11 năm 2018**

**MỤC LỤC**

[I. Giới thiệu sơ lược về cây quyết định (Decision tree) 1](#_Toc531590687)

[*1. Giới thiệu chung* 1](#_Toc531590688)

[*2. Các kiểu cây quyết định* 1](#_Toc531590689)

[*3. Ví dụ thực tế* 2](#_Toc531590690)

[*4. Ưu điểm cuả cây quyết định* 5](#_Toc531590691)

[*5. Nhược điểm của cây quyết định* 6](#_Toc531590692)

[II. Cây quyết định tăng hiệu quả cao (Highly Efficient Gradient Boosting Decision Tree) 7](#_Toc531590693)

[*1. Giới thiệu về Gradient Boosting Decision Tree (GBDT)* 7](#_Toc531590694)

[*2. Hai thuật toán XGBoost và LightGBM* 7](#_Toc531590695)

[2.1. XGBoost 7](#_Toc531590696)

[2.2. LightGBM 8](#_Toc531590697)

[2.3. Tính năng của LightGBM 8](#_Toc531590698)

[2.3.1. Tối ưu hóa tốc độ và cách sử dụng bộ nhớ 8](#_Toc531590699)

[2.3.2. Tối ưu hóa thưa thớt 9](#_Toc531590700)

[2.3.3. Tối ưu hóa chính xác 9](#_Toc531590701)

[2.3.4. Tối ưu hóa trong giao tiếp mạng 10](#_Toc531590702)

[2.3.5. Tối ưu hóa trong học song song 10](#_Toc531590703)

[2.3.6. Ứng dụng và số liệu 11](#_Toc531590704)

**I. Giới thiệu sơ lược về cây quyết định (Decision tree)**

***1. Giới thiệu chung***

Trong lĩnh vực [máy học](https://vi.wikipedia.org/wiki/M%C3%A1y_h%E1%BB%8Dc), **cây quyết định** là một kiểu mô hình dự báo (*predictive model*), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng. Mỗi một nút trong (*internal node*) tương ứng với một biến; đường nối giữa nó với nút con của nó thể hiện một giá trị cụ thể cho biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu, cho trước các giá trị của các biến được biểu diễn bởi đường đi từ nút gốc tới nút lá đó. Kỹ thuật học máy dùng trong cây quyết định được gọi là **học bằng cây quyết định**, hay chỉ gọi với cái tên ngắn gọn là **cây quyết định**.

Học bằng cây quyết định cũng là một phương pháp thông dụng trong [khai phá dữ liệu](https://vi.wikipedia.org/wiki/Khai_ph%C3%A1_d%E1%BB%AF_li%E1%BB%87u). Khi đó, cây quyết định mô tả một cấu trúc cây, trong đó, các lá đại diện cho các phân loại còn cành đại diện cho các kết hợp của các thuộc tính dẫn tới phân loại đó[1]. Một cây quyết định có thể được học bằng cách chia [tập hợp](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p) nguồn thành các tập con dựa theo một kiểm tra giá trị thuộc tính [1]. Quá trình này được lặp lại một cách đệ quy cho mỗi tập con dẫn xuất. Quá trình [đệ quy](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%87_quy) hoàn thành khi không thể tiếp tục thực hiện việc chia tách được nữa, hay khi một phân loại đơn có thể áp dụng cho từng phần tử của tập con dẫn xuất. Một bộ phân loại [rừng ngẫu nhiên](https://vi.wikipedia.org/w/index.php?title=R%E1%BB%ABng_ng%E1%BA%ABu_nhi%C3%AAn&action=edit&redlink=1) (*random forest*) sử dụng một số cây quyết định để có thể cải thiện tỉ lệ phân loại.

Cây quyết định cũng là một phương tiện có tính mô tả dành cho việc tính toán các [xác suất có điều kiện](https://vi.wikipedia.org/wiki/X%C3%A1c_su%E1%BA%A5t_c%C3%B3_%C4%91i%E1%BB%81u_ki%E1%BB%87n).

Cây quyết định có thể được mô tả như là sự kết hợp của các kỹ thuật toán học và tính toán nhằm hỗ trợ việc mô tả, phân loại và tổng quát hóa một tập dữ liệu cho trước.

Dữ liệu được cho dưới dạng các bản ghi có dạng:

***(x, y) = (x1, x2, x3..., xk, y)***

Biến phụ thuộc (*dependant variable*) *y* là biến mà chúng ta cần tìm hiểu, phân loại hay tổng quát hóa. *x1*, *x2*, *x3*... là các biến sẽ giúp ta thực hiện công việc đó

***2. Các kiểu cây quyết định***

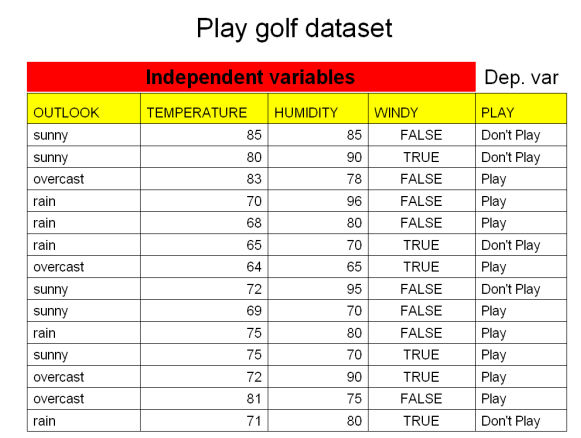
Cây quyết định còn có hai tên khác:

* **Cây hồi quy** (*Regression tree*) ước lượng các hàm giá có giá trị là số thực thay vì được sử dụng cho các nhiệm vụ phân loại. (ví dụ: ước tính giá một ngôi nhà hoặc khoảng thời gian một bệnh nhân nằm viện)
* **Cây phân loại** (*Classification tree*), nếu *y* là một biến phân loại như: giới tính (nam hay nữ), kết quả của một trận đấu (thắng hay thua).

***3. Ví dụ thực tế***

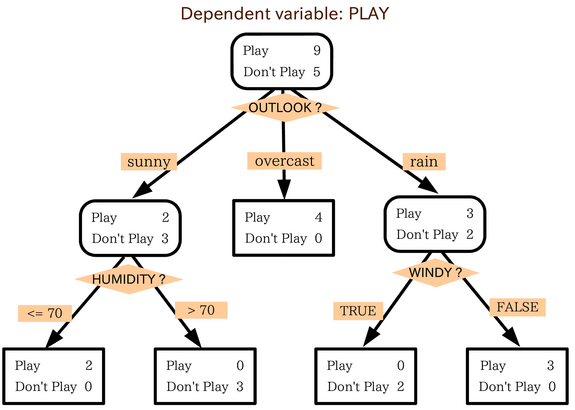
Ta sẽ dùng một ví dụ để giải thích về cây quyết định:

* David là quản lý của một câu lạc bộ đánh golf nổi tiếng. Anh ta đang có rắc rối chuyện các thành viên đến hay không đến. Có ngày ai cũng muốn chơi golf nhưng số nhân viên câu lạc bộ lại không đủ phục vụ. Có hôm, không hiểu vì lý do gì mà chẳng ai đến chơi, và câu lạc bộ lại thừa nhân viên.
* Mục tiêu của David là tối ưu hóa số nhân viên phục vụ mỗi ngày bằng cách dựa theo thông tin dự báo thời tiết để đoán xem khi nào người ta sẽ đến chơi golf. Để thực hiện điều đó, anh cần hiểu được tại sao khách hàng quyết định chơi và tìm hiểu xem có cách giải thích nào cho việc đó hay không.
* Vậy là trong hai tuần, anh ta thu thập thông tin về:
* Trời (*outlook*) (nắng (*sunny*),
* Và tất nhiên là số người đến chơi golf vào hôm đó. David thu được một bộ dữ liệu gồm 14 dòng và 5 cột.



Hình 1: Play golf dataset

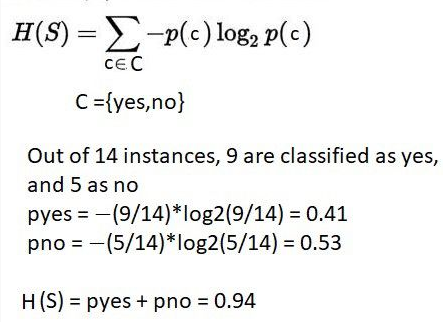
* Sau đó, để giải quyết bài toán của David, người ta đã đưa ra một mô hình cây quyết định:



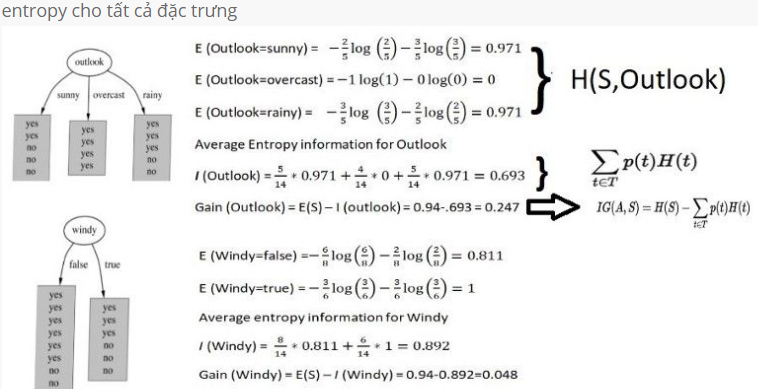
Hình 2: Dependent variable: PLAY.

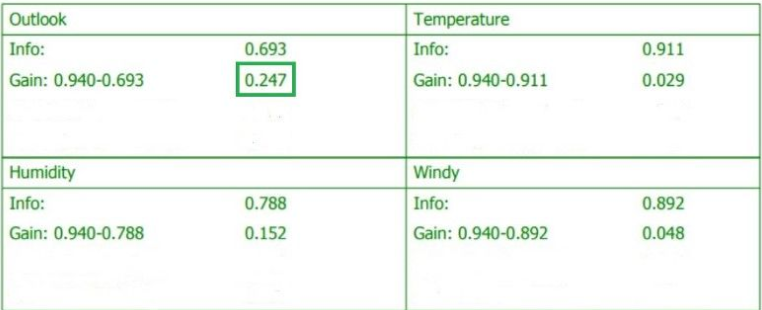
* Nhóm người chơi golf khi trời nắng, nhóm chơi khi trời nhiều mây, và nhóm chơi khi trời mưa.
* Kết luận thứ nhất: nếu trời nhiều mây, người ta luôn luôn chơi golf. Và có một số người ham mê đến mức chơi golf cả khi trời mưa.
* Tiếp theo, ta lại chia nhóm trời nắng thành hai nhóm con. Ta thấy rằng khách hàng không muốn chơi golf nếu độ ẩm lên quá 70%.
* Cuối cùng, ta chia nhóm trời mưa thành hai và thấy rằng khách hàng sẽ không chơi golf nếu trời nhiều gió.
* Và đây là lời giải ngắn gọn cho bài toán mô tả bởi cây phân loại. David cho phần lớn nhân viên nghỉ vào những ngày trời nắng và ẩm, hoặc những ngày mưa gió. Vì hầu như sẽ chẳng có ai chơi golf trong những ngày đó. Vào những hôm khác, khi nhiều người sẽ đến chơi golf, anh ta có thể thuê thêm nhân viên thời vụ để phụ giúp công việc.

Kết luận là cây quyết định giúp ta biến một biểu diễn dữ liệu phức tạp thành một cấu trúc đơn giản hơn rất nhiều.



Hình 3:





***4. Ưu điểm cuả cây quyết định***

So với các phương pháp [khai phá dữ liệu](https://vi.wikipedia.org/wiki/Khai_ph%C3%A1_d%E1%BB%AF_li%E1%BB%87u) khác, cây quyết định là phương pháp có một số ưu điểm:

* **Cây quyết định dễ hiểu**. Người ta có thể hiểu mô hình cây quyết định sau khi được giải thích ngắn.
* **Việc chuẩn bị dữ liệu cho một cây quyết định là cơ bản hoặc không cần thiết**. Các kỹ thuật khác thường đòi hỏi [chuẩn hóa dữ liệu](https://vi.wikipedia.org/wiki/Chu%E1%BA%A9n_h%C3%B3a_d%E1%BB%AF_li%E1%BB%87u), cần tạo các biến phụ (*dummy variable*) và loại bỏ các giá trị rỗng.
* **Cây quyết định có thể xử lý cả dữ liệu có giá trị bằng số và dữ liệu có giá trị là tên thể loại.** Các kỹ thuật khác thường chuyên để phân tích các bộ dữ liệu chỉ gồm một loại biến. Chẳng hạn, các luật quan hệ chỉ có thể dùng cho các biến tên, trong khi [mạng nơ-ron](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_n%C6%A1-ron) chỉ có thể dùng cho các biến có giá trị bằng số.
* **Cây quyết định là một mô hình hộp trắng.** Nếu có thể quan sát một tình huống cho trước trong một mô hình, thì có thể dễ dàng giải thích điều kiện đó bằng logic Boolean. Mạng nơ-ron là một ví dụ về mô hình hộp đen, do lời giải thích cho kết quả quá phức tạp để có thể hiểu được.
* **Có thể thẩm định một mô hình bằng các kiểm tra thống kê.** Điều này làm cho ta có thể tin tưởng vào mô hình.
* **Cây quyết định có thể xử lý tốt một lượng dữ liệu lớn trong thời gian ngắn**. Có thể dùng máy tính cá nhân để phân tích các lượng dữ liệu lớn trong một thời gian đủ ngắn để cho phép các nhà chiến lược đưa ra quyết định dựa trên phân tích của cây quyết định.

***5. Nhược điểm của cây quyết định***

* Khó giải quyết được những vấn đề có dữ liệu phụ thuộc thời gian liên tục.
* Dễ xảy ra lỗi khi có quá nhiều lớp chi phí tính toán để xây dựng mô hình cây quyết định cao.

**II. Cây quyết định tăng hiệu quả cao (Highly Efficient Gradient Boosting Decision Tree)**

***1. Giới thiệu về Gradient Boosting Decision Tree (GBDT)***

Gradient Boosting Decision Tree là một thuật toán học máy được sử dụng rộng rãi, do hiệu quả, độ chính xác và khả năng diễn giải của nó, chẳng hạn như phân loại nhiều lớp, nhấp vào dự đoán và học cách xếp hạng. Có một vài triển khai hiệu quả như XGBoost và pGBRT. Mặc dù nhiều tối ưu hóa kỹ thuật đã được áp dụng trong các triển khai này, hiệu quả và khả năng mở rộng vẫn không đạt yêu cầu khi thứ nguyên tính năng cao và kích thước dữ liệu lớn. Một lý do chính là đối với mỗi tính năng, họ cần phải quét tất cả các trường hợp dữ liệu để ước tính mức tăng thông tin của tất cả các điểm phân tách có thể, rất tốn thời gian. Để giải quyết vấn đề này, chúng tôi đề xuất hai kỹ thuật mới: Lấy mẫu một bên dựa trên nền (Gradient-based One-Side Sampling- GOSS) và Gói tính năng độc quyền (Exclusive Feature Bundling- EFB).

***2. Hai thuật toán XGBoost và LightGBM***

**2.1. XGBoost**

XGBoost từ trường đại học Washington và xuất bản năm 2016 giới thiệu hai kỹ thuật để cải thiện hiệu suất. Thứ nhất, ý tưởng về Phác thảo Định lượng Trọng số (Weighted Quantile Sketch), là một thuật toán xấp xỉ để xác định cách phân chia trong cây quyết định (phân tách ứng cử viên). Thứ hai là tìm kiếm phân tách nhận biết Sparsity hoạt động trên dữ liệu thưa thớt, dữ liệu có nhiều ô bị thiếu.

Phác thảo Định lượng Trọng số là một vấn đề khó trong việc thúc đẩy cho các tập dữ liệu lớn. Một ý tưởng cơ bản trong việc thúc đẩy là tăng cường dữ liệu mà các mô hình trước đó mắc lỗi. Vì vậy, trong việc thúc đẩy chúng tôi quan tâm đến việc tìm các điểm phân tách trong các mẫu có trọng số hoạt động tốt nhất. Các tác giả của bài báo XGBoost giải thích giải pháp của họ, một cấu trúc dữ liệu hỗ trợ các hoạt động hợp nhất và cắt tỉa trong một bài bổ sung.

Tìm kiếm phân tách nhận biết Sparsity giới thiệu một hướng mặc định trong mỗi nút cây. Khi một giá trị bị thiếu trong cột thưa thớt, mẫu được phân loại theo hướng mặc định này. Các hướng mặc định được học từ dữ liệu. Có hướng mặc định cho phép thuật toán bỏ qua các mẫu thiếu mục nhập cho đối tượng địa lý. Kỹ thuật này tận dụng lợi thế của sự thưa thớt trong tập dữ liệu, giảm tính toán thành tìm kiếm tuyến tính trên các mục không bị thiếu.

**2.2. LightGBM**

**LightGBM được phát hành từ Microsoft**, là một framework tăng cường sử dụng các thuật toán học tập dựa trên cây. Nó được thiết kế để phân phối và hiệu quả với những ưu điểm sau:

* Tốc độ đào tạo nhanh hơn và hiệu quả cao hơn.
* Giảm mức sử dụng bộ nhớ.
* Độ chính xác tốt hơn.
* Hỗ trợ học song song và GPU.
* Có khả năng xử lý dữ liệu có quy mô lớn.

**2.3. Tính năng của LightGBM**

**2.3.1. Tối ưu hóa tốc độ và cách sử dụng bộ nhớ**

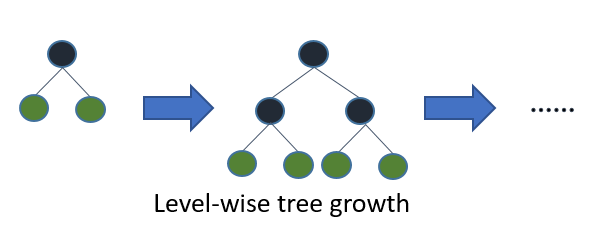
* **Giảm chi phí tính toán mức tăng cho mỗi lần chia**
* Các thuật toán dựa trên phân loại trước có độ phức tạp về thời gian O(#data)
* Tính toán biểu đồ có độ phức tạp về thời gian O(#data), nhưng điều này chỉ liên quan đến hoạt động tổng hợp nhanh. Khi biểu đồ được xây dựng, thuật toán dựa trên biểu đồ có độ phức tạp về thời gian O(#bins)và #binsnhỏ hơn nhiều so với #data.
* **Sử dụng phép trừ biểu đồ để tăng tốc thêm**
* Để có được biểu đồ của một lá trong cây nhị phân, hãy sử dụng phép trừ biểu đồ của cha mẹ và hàng xóm của nó
* Vì vậy, nó cần phải xây dựng biểu đồ cho chỉ một lá (với nhỏ #datahơn so với hàng xóm của nó). Sau đó nó có thể nhận được biểu đồ của hàng xóm của nó bằng phép trừ biểu đồ với chi phí nhỏ ( O(#bins))
* **Giảm mức sử dụng bộ nhớ**
* Thay thế các giá trị liên tục bằng các thùng rời rạc. Nếu #binsnhỏ, có thể sử dụng kiểu dữ liệu nhỏ, ví dụ uint8\_t, để lưu trữ dữ liệu huấn luyện
* Không cần phải lưu trữ thông tin bổ sung cho các giá trị tính năng sắp xếp trước
* Giảm chi phí truyền thông cho việc học song song

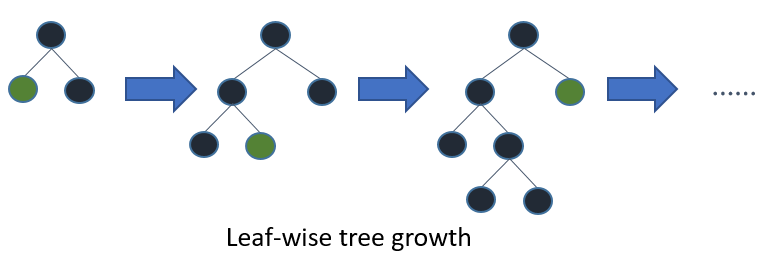
**2.3.2. Tối ưu hóa thưa thớt**

Chỉ cần xây dựng biểu đồ cho các tính năng thưa thớtO(2 \* #non\_zero\_data)

**2.3.3. Tối ưu hóa chính xác**

* **Leaf-wise (Best-first) Cây tăng trưởng**
* Hầu hết các thuật toán học cây quyết định đều phát triển cây theo cấp độ (chiều sâu), giống như hình dưới đây:



* LightGBM phát triển cây lá khôn ngoan (trees leaf-wise) (tốt nhất đầu tiên) . Nó sẽ chọn lá với mất đồng bằng tối đa để phát triển. Nắm giữ #leaf các thuật toán cố định, lá khôn ngoan (leaf-wise) có xu hướng đạt được tổn thất thấp hơn các thuật toán cấp độ khôn ngoan.
*  Lá khôn ngoan có thể gây ra quá phù hợp khi #datanhỏ, vì vậy LightGBM bao gồm các max\_depththam số để hạn chế chiều sâu cây. Tuy nhiên, cây vẫn mọc lá ngay cả khi max\_depthđược chỉ định.
* **Chia tối ưu cho các tính năng phân loại**
  + Nó là phổ biến để đại diện cho các tính năng phân loại với mã hóa một nóng, nhưng cách tiếp cận này là tối ưu hóa cho người học cây. Đặc biệt đối với các tính năng phân hạng cao, một cây được xây dựng trên các tính năng nóng có xu hướng không cân bằng và cần phát triển rất sâu để đạt được độ chính xác cao.
  + Thay vì mã hóa một nóng, giải pháp tối ưu là chia nhỏ tính năng phân loại bằng cách phân vùng các danh mục của nó thành 2 tập con. Nếu đối tượng địa lý có kcác danh mục, có thể có các phân vùng. Nhưng có một giải pháp hiệu quả cho cây hồi quy [8] . Nó cần tìm phân vùng tối ưu.2^(k-1) - 1O(k \* log(k))
* Ý tưởng cơ bản là sắp xếp các danh mục theo mục tiêu đào tạo ở từng phân chia. Cụ thể hơn, LightGBM sắp xếp biểu đồ (cho một tính năng phân loại) theo giá trị tích lũy của nó ( ) và sau đó tìm phân chia tốt nhất trên biểu đồ được sắp xếp.sum\_gradient / sum\_hessian

**2.3.4. Tối ưu hóa trong giao tiếp mạng**

Nó chỉ cần sử dụng một số thuật toán truyền thông tập thể, như “All reduce”, “All gather” và “Reduce scatter”, song song với việc học về LightGBM. LightGBM triển khai các thuật toán hiện đại. Các thuật toán truyền thống tập thể này có thể cung cấp hiệu suất tốt hơn nhiều so với giao tiếp điểm với điểm.

**2.3.5. Tối ưu hóa trong học song song**

LightGBM cung cấp các thuật toán học tập song song sau đây.

* Tính năng song song
* Thuật toán truyền thống
* Tính năng song song trong LightGBM
* Dữ liệu song song
* Thuật toán truyền thống
* Dữ liệu song song trong LightGBM
* Bỏ phiếu song song

**2.3.6. Ứng dụng và số liệu**

**LightGBM hỗ trợ các ứng dụng sau:**

* Hồi quy, chức năng mục tiêu là mất L2
* Phân loại nhị phân, hàm mục tiêu là logloss
* Đa phân loại
* Cross-entropy, hàm mục tiêu là logloss và hỗ trợ đào tạo trên các nhãn không phải nhị phân
* Lambdarank, hàm mục tiêu là lambdarank với NDCG

**LightGBM hỗ trợ các chỉ số sau:**

* Mất L1
* Mất L2
* Mất nhật ký
* Tỷ lệ lỗi phân loại
* AUC
* NDCG
* BẢN ĐỒ
* Mất đăng nhập nhiều lớp
* Tỷ lệ lỗi nhiều lớp
* Hội chợ
* Huber
* Poisson
* Định lượng
* MAPE
* Kullback-Leibler
* Gamma
* Tweedie

**2.4. Gradient-based One-Side Sampling- GOSS**