

**TỔNG CÔNG TY VIỄN THÔNG VIETTEL**

**KHỐI CÔNG NGHỆ THÔNG TIN**

**XÂY DỰNG MÔ HÌNH DỰ ĐOÁN BANKING POTENTIAL CHO NHÓM MERCHANT**

**Mã hiệu dự án:** **MyViettel\_0009**

**Mã hiệu tài liệu: MyViettel\_0009**

**<Địa điểm, Thời gian>**

**BẢNG GHI NHẬN TIẾN ĐỘ**

\*A - Tạo mới, M - Sửa đổi, D - Xóa bỏ

| **Ngày**  **bàn giao** | **Yêu cầu** | **A\***  **M, D** | **Trạng thái** | **Đầu mối KH** | **Mô tả** | **Ghi chú** |
| --- | --- | --- | --- | --- | --- | --- |
| 20/5/2023 | xây dựng mô hình dự đoán banking potential cho nhóm merchant |  | Hoàn thành | PTC, PKH | * xây dựng mô hình dự đoán độ phù hợp banking potential cho nhóm merchant |  |
|  |  |  |  |  |  |  |

**MỤC LỤC**

[**1. Tìm Hiểu Bài Toán 6**](#_heading=h.wddkan5ax8hb)

[1.1. Khảo sát bài toán 6](#_heading=h.otexc141hes8)

[1.2. Phân tích bài toán 6](#_heading=h.fn8hvz316etd)

[**2. Tìm Hiểu Dữ Liệu 7**](#_heading=h.4ekebi7s4l4y)

[2.1. Tìm hiểu về mô tả dữ liệu 7](#_heading=h.3qrcd6hsy2ew)

[2.2. Kiểm tra sơ bộ dữ liệu 8](#_heading=h.ifmhjieme53c)

[2.3. Phân tích dữ liệu 8](#_heading=h.z2e8u9lu4qb1)

[2.4. Sinh dữ liệu từ meta data 8](#_heading=h.eh2b0738579d)

[2.5. Kiểm tra missing value 8](#_heading=h.ulta7nmxxtu1)

[2.6. Xử lý missing value 9](#_heading=h.f1lxdx3kjilz)

[2.7. Chuẩn hóa dữ liệu 9](#_heading=h.c2jj3miqc0rl)

[2.8. Selection features 9](#_heading=h.i0yps1ib7pnw)

[2.9. Chia tập test set 9](#_heading=h.pu3dnsrro5i3)

[**3. Để Xuất Giải Pháp 10**](#_heading=h.4rp8l964egc5)

[3.1. Đề xuất các mô hình 10](#_heading=h.9w03czompztl)

[3.2. Chọn ra 3 - 5 mô hình 11](#_heading=h.ed6z7lu9awon)

[3.3. Thống nhất giải pháp xây dựng mô hình 11](#_heading=h.kqmcfsmccn4i)

[3.4. Mô Hình 11](#_heading=h.u9h2nyvb43mv)

[3.4.1. SVM 11](#_heading=h.xo6yqa7z7gq)

[3.4.2. XGBoost 14](#_heading=h.dsotq4utsmu3)

[3.4.3. RandomForest 17](#_heading=h.p6ijiw1qxatz)

[3.4.4. MLP 19](#_heading=h.d2ekl3riw0ro)

[**4. Kết quả 21**](#_heading=h.jbqtlc9ix8k1)

**TRANG KÝ**

Người lập: <Ngày>

<Chức danh>

Người xem xét: <Ngày>

<Chức danh>

Người xem xét: <Ngày>

<Chức danh>

Người phê duyệt: <Ngày>

<Chức danh>

# **Tìm Hiểu Bài Toán**

## Khảo sát bài toán

các mô hình dự đoán banking potential là các mô hình sử dụng các thuật toán học máy để dự đoán khả năng một khách hàng hoặc một nhóm khách hàng sẽ sử dụng các sản phẩm và dịch vụ ngân hàng. Các mô hình này giúp ngân hàng thu hút, giữ chân và phát triển những khách hàng có lợi nhất, cải thiện hoạt động và tối ưu hóa chiến lược kinh doanh.

Đối với nhóm merchant, có thể sử dụng các mô hình dự đoán banking potential sau:

• Mô hình phân loại: Mô hình này sử dụng các thuộc tính của merchant như doanh thu, số lượng giao dịch, lịch sử thanh toán, v.v. để phân loại merchant vào các nhóm khác nhau theo mức độ tiềm năng. Ví dụ: cao, trung bình, thấp. Mô hình này giúp ngân hàng xác định các merchant có nhu cầu và khả năng sử dụng các sản phẩm và dịch vụ ngân hàng cao nhất, và tập trung vào việc cung cấp các giải pháp phù hợp cho họ.

• Mô hình hồi quy: Mô hình này sử dụng các thuộc tính của merchant để dự đoán một giá trị liên tục cho banking potential. Ví dụ: tỷ lệ chuyển đổi, doanh số bán hàng, số tiền gửi, v.v. Mô hình này giúp ngân hàng ước tính được lợi nhuận hoặc rủi ro tiềm năng từ việc cung cấp các sản phẩm và dịch vụ ngân hàng cho merchant, và điều chỉnh chiến lược kinh doanh cho phù hợp.

• Mô hình phân cụm: Mô hình này sử dụng các thuật toán học không giám sát để phân chia merchant thành các nhóm có đặc điểm tương tự nhau theo banking potential. Ví dụ: nhóm merchant có nhu cầu về thanh toán điện tử cao, nhóm merchant có nhu cầu về vay vốn cao, v.v. Mô hình này giúp ngân hàng phát hiện được các mẫu hay xu hướng trong banking potential của merchant, và thiết kế các sản phẩm và dịch vụ ngân hàng theo từng nhóm.

## Phân tích bài toán

Bài toán đặt ra xây dựng mô hình dự đoán banking potential cho nhóm merchant cần phải phân loại với mỗi một người dùng thuộc những nhóm nào trong 9 nhóm:

* Bảo hiểm
* Ngân hàng
* Du lịch
* Giáo dục
* Tín dụng (vay nợ)
* Game
* Xổ số
* Lĩnh vực khác

**=> Bài toán cần xử lý là phân loại nhiều nhãn hay còn gọi classify multi label**

# **Tìm Hiểu Dữ Liệu**

## Tìm hiểu về mô tả dữ liệu

Dữ liệu đầu vào bảo gồm :

* Demographic: Tuổi, giới tính, vị trí (tỉnh)
* CCAI URL

Dữ liệu đầu ra mẫu:

| Isdn | Merchant | Score |
| --- | --- | --- |
| 086805511 | Bảo hiểm | 0.87 |
| 086805511 | Ngân hàng | 0.87 |
| 086805511 | Du lịch | 0.87 |
| 086805511 | Giáo dục | 0.87 |
| 086805511 | Tín dụng (vay nợ) | 0.87 |
| 086805511 | Game | 0.87 |
| 086805511 | Xổ số | 0.87 |

## Kiểm tra sơ bộ dữ liệu

Tổng quan dữ liệu bao gồm :

* 902 thuộc tính
* 2000 mẫu dữ liệu

Các kiểu dữ liệu trong dữ liệu :

* Biến định danh được mã hóa theo dạng numeric
* Biến liên tục
* Biến định danh nhị phân

## Phân tích dữ liệu

## Sinh dữ liệu từ meta data

Bộ dữ liệu được sinh tự động dựa trên số mẫu và số đặc trưng của từng mẫu do người dùng tùy ý truyền vào với tùy mục đích để đánh giá hiệu năng mô hình. Các kiểu dữ liệu của từng đặc trưng được random ngẫu nhiên với các kiểu: "int", "float", "string".

Người dùng có thể tùy chỉnh kiểu dữ liệu của các đặc trưng, số mẫu cũng như số đặc trưng của từng mẫu khi khởi tạo bộ dữ liệu. Bộ dữ liệu sẽ được lưu dưới dạng .csv sau khi khởi tạo với tên và đường dẫn tùy theo người dùng định nghĩa.

## Kiểm tra missing value

Kiểm tra missing value là một bước quan trọng trong quá trình phân tích dữ liệu, vì nó có thể ảnh hưởng đến kết quả và độ tin cậy của phân tích. Missing value là những giá trị bị thiếu hoặc không có trong dữ liệu, có thể do nhiều nguyên nhân khác nhau, như lỗi nhập liệu, lỗi thiết bị, không trả lời câu hỏi, v.v.

Chúng tôi sẽ kiểm tra giá trị **missing value** trên từng thuộc tính và xử lý trước khi đưa vào huấn luyện mô hình

## Xử lý missing value

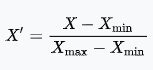
Quy trình xử lý missing value :

* B1:Loại bỏ các dòng chứa tất cả các giá trị bị null vì không mang giá trị
* B2:Loại bỏ các dòng chứa hơn 45% giá trị bị null vì thiếu quá nhiều thông tin
* B3:loại bỏ các thuộc tính chưa hơn 45% số mẫu bị null
* B4:Thay thế các giá trị null ở thuộc tính biến liên tục bằng các giá trị nội suy
* B5:Đánh dấu các giá trị null ở thuốc tính định danh

## Chuẩn hóa dữ liệu

Chúng tôi sử dụng phương pháp chuẩn hóa dữ liệu để cân bằng giữa các thuộc tính khi tránh tính trạng bias khi dữ liệu huấn luyện trong các mô hình SVM , MLO. Trong đó phương pháp được sử dụng là normalizer .

Phương pháp chuẩn hóa dữ liệu normalizer là một phương pháp scale dữ liệu từ miền giá trị bất kì sang miền giá trị nằm trong khoảng 0 đến 1. Phương pháp này yêu cầu chúng ta cần xác định được giá trị lớn nhất (max) và giá trị nhỏ nhất (min) của dữ liệu. Giá trị được normalize theo công thức sau:



Phương pháp **Normalizing** rất nhạy với biến có outliers, vì vậy nếu có outliers trong tập dữ liệu thì không nên sử dụng phương pháp này vì vậy cần phải loại bỏ outlier trước khi chuẩn hóa

## Selection features

Selection features là quá trình giảm số lượng các biến thông tin đầu vào trong mô hình Machine Learning của bạn, bằng cách chỉ sử dụng những dữ liệu liên quan, có ý nghĩa và loại bỏ các dữ liệu nhiễu Selection features được dùng vì một số lý do: đơn giản hóa các mô hình để giúp các nhà nghiên cứu/người dùng diễn dịch dễ dàng hơn, giảm thời gian huấn luyện, tránh lời nguyền chiều (curse of dimensionality), tăng cường tổng quát hóa bằng cách giảm sự quá khớp

## Chia tập test set

Chia tập test set là quá trình tách một phần dữ liệu ra khỏi tập dữ liệu ban đầu để kiểm tra hiệu năng của mô hình Machine Learning sau khi đã được huấn luyện. Chia tập test set giúp đánh giá mức độ thành công của mô hình, khi nào nên cập nhật mô hình, và tránh hiện tượng quá khớp (overfitting) . Có nhiều cách để chia tập test set, phụ thuộc vào kích thước và tính chất của bộ dữ liệu. Một số phương pháp phổ biến là: Hold-out, Train-Validation-Test Split và Cross Validation.

Phương pháp sử dụng :Train-Validation-Test Split

# **Để Xuất Giải Pháp**

## Đề xuất các mô hình

mô hình classify multi label là một loại bài toán phân loại nhiều nhãn, khi mỗi mẫu dữ liệu có thể có nhiều hơn một nhãn đầu ra. Các nhãn này có thể độc lập hoặc phụ thuộc vào nhau. Các mô hình classify multi label có thể được sử dụng cho nhiều ứng dụng như gán thẻ cho văn bản, ảnh, video, âm thanh, v.v.

có một số mô hình classify multi label label phổ biến như sau:

* Mô hình Binary Relevance: Mô hình này giải quyết bài toán phân loại nhiều nhãn bằng cách xây dựng nhiều mô hình phân loại nhị phân cho mỗi nhãn. Mỗi mô hình sẽ dự đoán xem một mẫu có thuộc về nhãn đó hay không. Mô hình này đơn giản và dễ triển khai, nhưng bỏ qua sự phụ thuộc giữa các nhãn và có thể gặp khó khăn khi số lượng nhãn lớn.
* Mô hình Classifier Chains: Mô hình này giải quyết bài toán phân loại nhiều nhãn bằng cách xây dựng nhiều mô hình phân loại nhị phân cho mỗi nhãn, nhưng với một thứ tự xác định trước. Mỗi mô hình sẽ dự đoán xem một mẫu có thuộc về nhãn đó hay không, và sử dụng kết quả dự đoán làm đầu vào cho mô hình tiếp theo. Mô hình này có thể khai thác được sự phụ thuộc giữa các nhãn, nhưng cũng phụ thuộc vào việc chọn thứ tự của các nhãn và có thể bị ảnh hưởng bởi lỗi tích lũy.
* Mô hình Label Powerset: Mô hình này giải quyết bài toán phân loại nhiều nhãn bằng cách biến đổi nó thành một bài toán phân loại đơn nhãn. Mỗi tập con của các nhãn sẽ được coi là một lớp mới và được gán cho các mẫu có chính xác các nhãn đó. Mô hình này có thể khai thác được sự phụ thuộc giữa các nhãn, nhưng cũng có thể gặp vấn đề với không gian lớp quá lớn và thiếu dữ liệu cho từng lớp.
* Mô hình deep learning : Mô hình này giải quyết bài toán phân loại nhiều nhãn bằng cách xây dựng mạng neural với đầy ra bằng số nhãn và sử dụng hàm kích hoạt đầu ra bằng softmax thể hiện tỉ lệ của nhãn rơi vào

## Chọn ra 3 - 5 mô hình

Chúng tôi đề xuất sử dụng nhóm mô hình xây dựng Classifier Chains nhằm thể hiện đc mối quan hệ giữa các nhãn và thử nghiệm mô hình deep learning và các mô hình tree-base. Sau quá trình nghiên cứu và tìm hiểu chúng tôi lựa chọn 5 mô hình :

* nhóm Classifier Chains:
  + SVM
  + KNN
  + Logistic Regression
* nhóm tree-base:
  + lightGBMboost
  + XGboost
  + RandomForest
* Deep learning
  + MLP

## Thống nhất giải pháp xây dựng mô hình

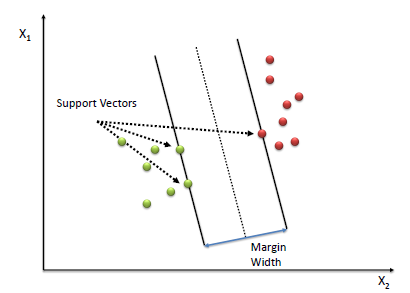
Qua tìm hiểu và nghiên cứu để tối ưu cho bài toán cũng như hiệu quả thời gian chạy và độ chính xác chúng tôi lựa chọn 4 mô hình để huấn luyện thực nghiệm cho bài toàn :

* + Classifier Chains -SVM
  + XGboost
  + RandomForest
  + MLP

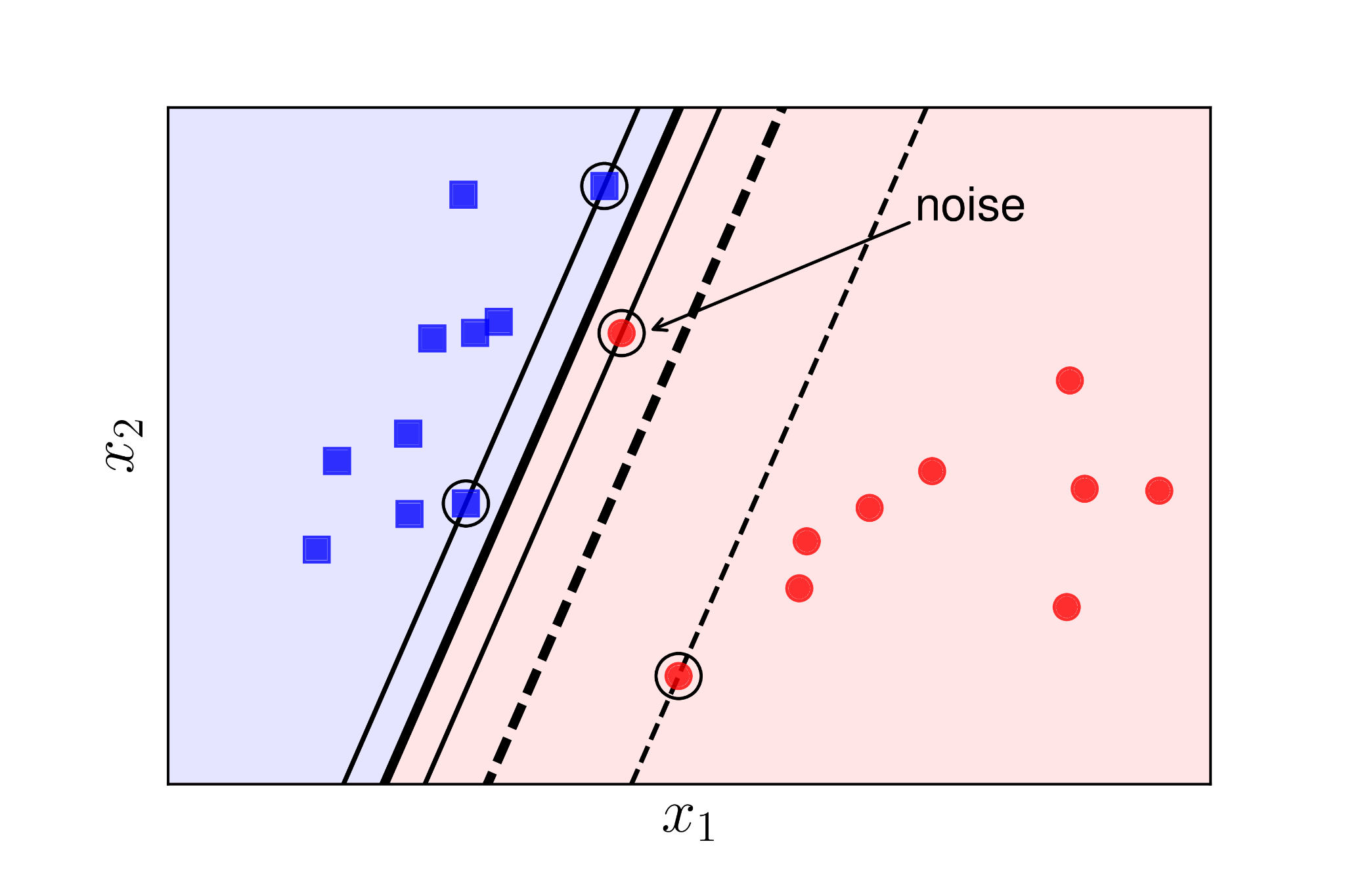
## Mô Hình

### SVM

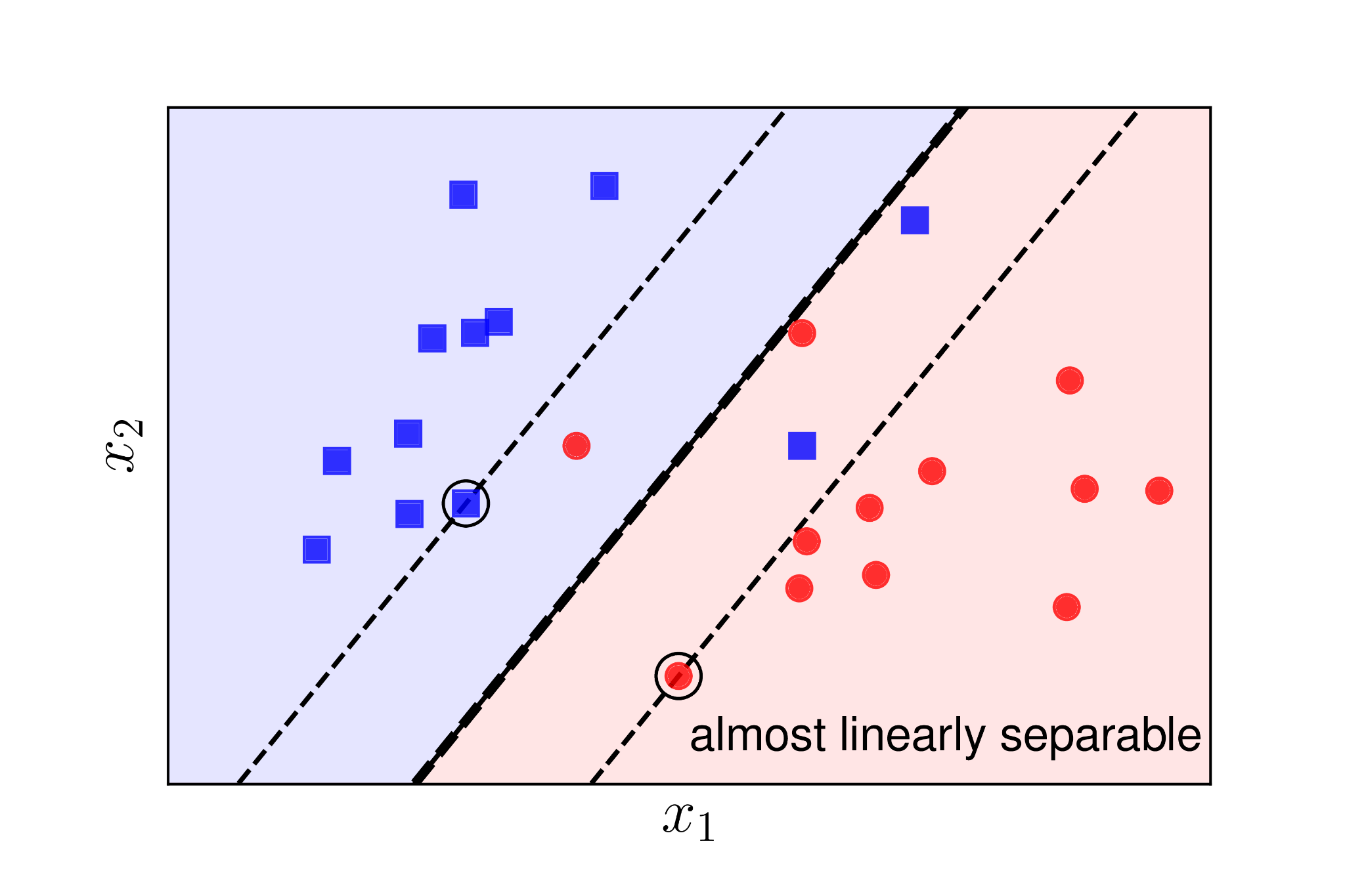
SVC (Support Vector Classifier) là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại.nó huấn luyện với dữ liệu nhằm trả về một siêu phẳng phù hợp nhất phân chia dữ liệu của bạn .Từ đó, sau khi nhận được siêu phẳng, bạn có thể cung cấp một số tính năng cho bộ phân loại của mình để xem lớp "dự đoán" là gì.



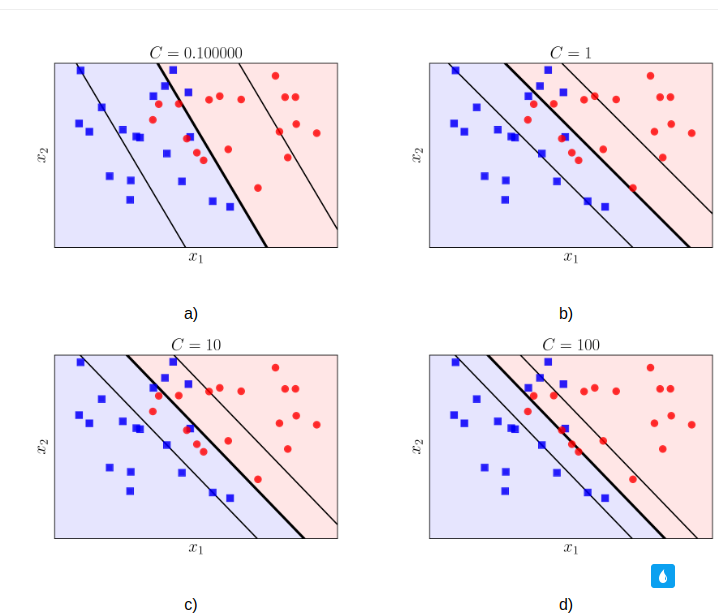
SVM là bài toán đi tìm mặt phân cách sao cho *margin* tìm được là lớn nhất trong đó *margin được hiểu là khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, Việc margin rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì sự phân chia giữa hai classes là rạch ròi hơn* , đồng nghĩa với việc các điểm dữ liệu *an toàn nhất* so với mặt phân cách.Nếu nếu để siêu phẳng quá fit với dữ liệu sẽ làm mất tính khái quát tổng thể lúc đó SVM làm việc không hiệu quả hoặc thậm chí không làm việc



Do đó chúng ta chấp nhận lỗi xảy ra ở một vài điểm dữ liệu. Lỗi này được xác định bằng khoảng cách từ điểm đó tới đường biên tương ứng để mô hình được tổng quát hóa hơn.



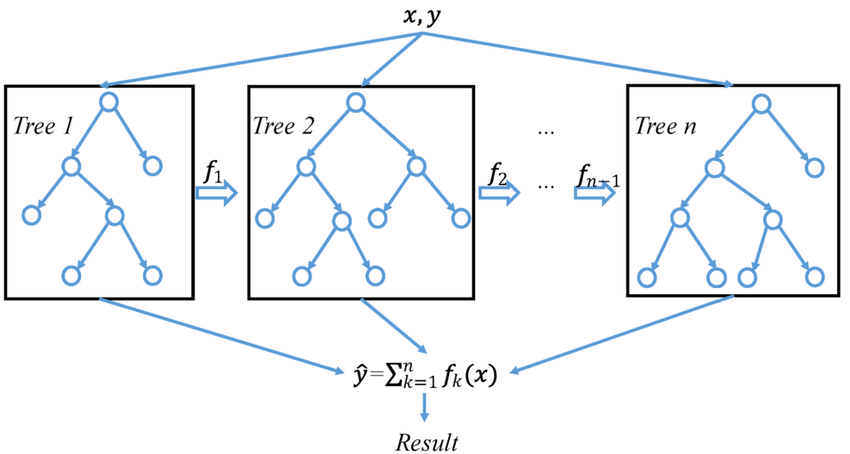
ta có thể điểm chỉnh tham số C với hằng số dương giúp cân đối độ lớn của *margin* và *sự hy sinh* của các điểm nằm trong vùng *không an toàn* Khi C càng lớn thì biên càng nhỏ và ngược lại .Khi C=∞hoặc rất lớn, Soft Margin SVM trở thành Hard Margin SVM.



### XGBoost

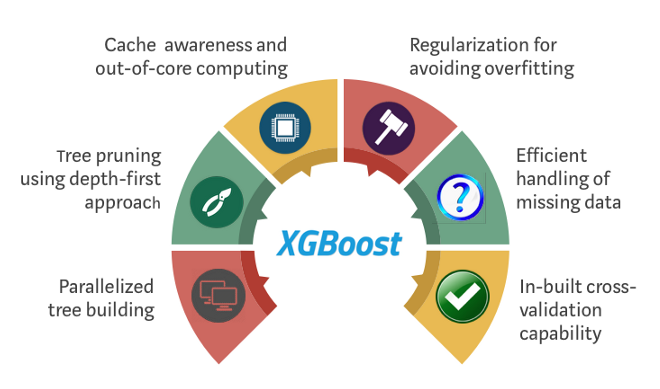
XGBoost là một triển khai cây quyết định Gradient Boosted. Các mô hình XGBoost chủ yếu chiếm ưu thế trong nhiều Cuộc thi Kaggle.

Trong thuật toán này, cây quyết định được tạo ở dạng tuần tự. Trọng lượng đóng một vai trò quan trọng trong XGBoost. Trọng số được gán cho tất cả các biến độc lập, sau đó được đưa vào cây quyết định để dự đoán kết quả. Trọng số của các biến được cây dự đoán sai được tăng lên và các biến này sau đó được đưa sang cây quyết định thứ hai. Sau đó, các bộ phân loại / dự đoán riêng lẻ này kết hợp với nhau để đưa ra một mô hình mạnh mẽ và chính xác hơn. Nó có thể hoạt động trên các vấn đề hồi quy, phân loại, xếp hạng và dự đoán do người dùng xác định.



Những cải tiến XGBoost:

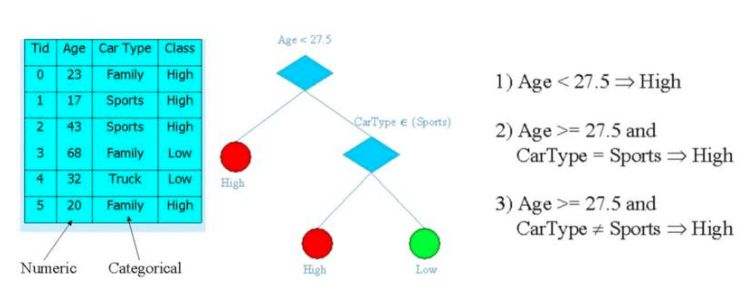
* Regularization:Vì sự tập hợp của các quyết định, cây đôi khi có thể dẫn đến rất phức tạp. XGBoost sử dụng cả quy trình điều chỉnh hồi quy Lasso và Ridge để xử phạt mô hình rất phức tạp.
* Parallelization and Cache block:Trong XGboost, chúng ta không thể huấn luyện nhiều cây song song, nhưng nó có thể tạo ra các nút khác nhau của cây song song. Vì vậy, dữ liệu cần được sắp xếp theo thứ tự. Để giảm chi phí sắp xếp, nó lưu trữ dữ liệu trong các khối. Nó lưu trữ dữ liệu ở định dạng cột nén, với mỗi cột được sắp xếp theo giá trị tính năng tương ứng. Công tắc này cải thiện hiệu suất thuật toán bằng cách bù đắp bất kỳ chi phí song song nào trong tính toán.
* Tỉa cây: XGBoost sử dụng tham số max\_depth như đã chỉ định tiêu chí dừng cho việc tách nhánh và bắt đầu tỉa cây lùi lại. Cách tiếp cận theo chiều sâu này cải thiện đáng kể hiệu suất tính toán.
* Cache-Awareness and Out-of-score computation::huật toán này đã được thiết kế để sử dụng hiệu quả tài nguyên phần cứng. Điều này được thực hiện nhờ nhận thức bộ nhớ cache bằng cách phân bổ bộ đệm nội bộ trong mỗi luồng để lưu trữ thống kê gradient. Các cải tiến khác như 'tính toán ngoài lõi tối ưu hóa dung lượng ổ đĩa có sẵn trong khi xử lý các khung dữ liệu lớn không vừa với bộ nhớ. Trong tính toán ngoài lõi, Xgboost cố gắng giảm thiểu tập dữ liệu bằng cách nén nó.
* Sparsity Awareness:XGBoost có thể xử lý dữ liệu thưa thớt có thể được tạo ra từ các bước tiền xử lý hoặc các giá trị bị thiếu. Nó sử dụng một thuật toán tìm kiếm phân tách đặc biệt được kết hợp với nó để có thể xử lý các loại mẫu thưa thớt khác nhau.
* Weighted Quantile Sketch:XGBoost đã tích hợp sẵn thuật toán phác thảo lượng tử có trọng số phân tán giúp dễ dàng tìm thấy hiệu quả các điểm phân tách tối ưu giữa các tập dữ liệu có trọng số.
* Cross-validation: Việc triển khai XGboost đi kèm với một phương pháp xác thực chéo được tích hợp sẵn. Điều này giúp thuật toán ngăn chặn việc trang bị quá mức khi tập dữ liệu không lớn như vậy,



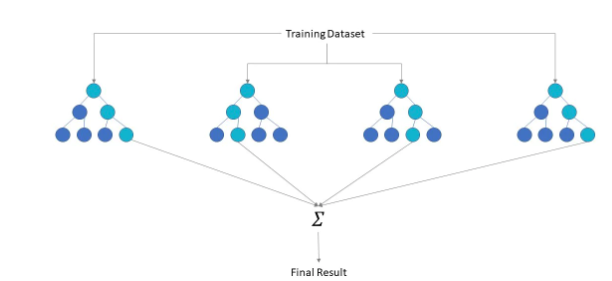
### RandomForest

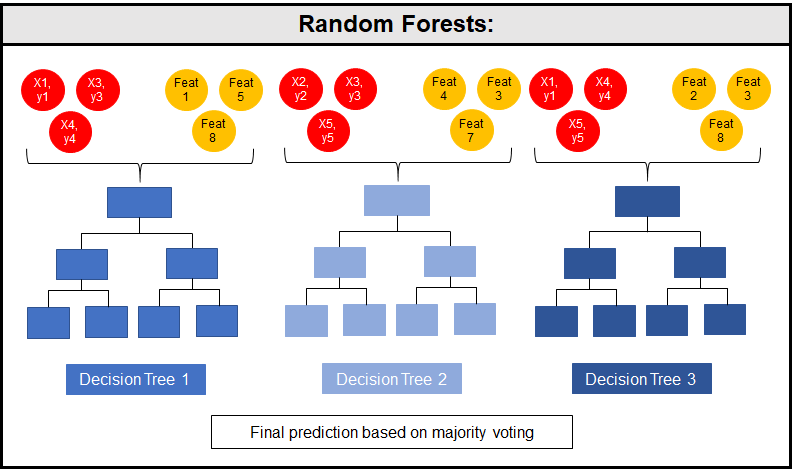
Random forest là một thuật toán học máy thường được sử dụng bởi Leo Breiman và Adele Cutler, kết hợp đầu ra của nhiều cây quyết định để đạt được một kết quả duy nhất. Tính dễ sử dụng và tính linh hoạt của nó đã thúc đẩy việc áp dụng nó, vì nó xử lý được cả vấn đề phân loại và hồi quy.

Với Cây quyết định được hiểu là cây mà mỗi nút biểu diễn một đặc trưng(tính chất), mỗi nhánh(branch) biểu diễn một quy luật(rule) và mỗi lá biểu diễn một kết quả (giá trị cụ thể hay một nhánh tiếp tục)



Trong khi cây quyết định là các thuật toán học tập có giám sát phổ biến, chúng có thể dễ gặp vấn đề, chẳng hạn như sai lệch và trang bị quá mức. Tuy nhiên, khi nhiều cây quyết định tạo thành một tập hợp trong thuật toán Random Forest, chúng dự đoán kết quả chính xác hơn, đặc biệt khi các cây riêng lẻ không tương quan với nhau. Các phương pháp học tập kết hợp được tạo thành từ một tập hợp các bộ phân loại ví dụ như cây quyết định và các dự đoán của chúng được tổng hợp để xác định kết quả phổ biến nhất. Các phương pháp tổng hợp nổi tiếng nhất là bagging, còn được gọi là tổng hợp bootstrap và tăng cường. Năm 1996, [Leo Breiman](https://link.springer.com/content/pdf/10.1007/BF00058655.pdf) đã giới thiệu trong phương pháp này, một mẫu dữ liệu ngẫu nhiên trong tập huấn luyện được chọn thay thế ,nghĩa là các điểm dữ liệu riêng lẻ có thể được chọn nhiều lần. Sau khi một số mẫu dữ liệu được tạo, các mô hình này sau đó được huấn luyện độc lập và tùy thuộc vào loại nhiệm vụ - tức là hồi quy hoặc phân loại - trung bình hoặc phần lớn các dự đoán đó mang lại ước tính chính xác hơn. Cách tiếp cận này thường được sử dụng để giảm phương sai trong một tập dữ liệu nhiễu.

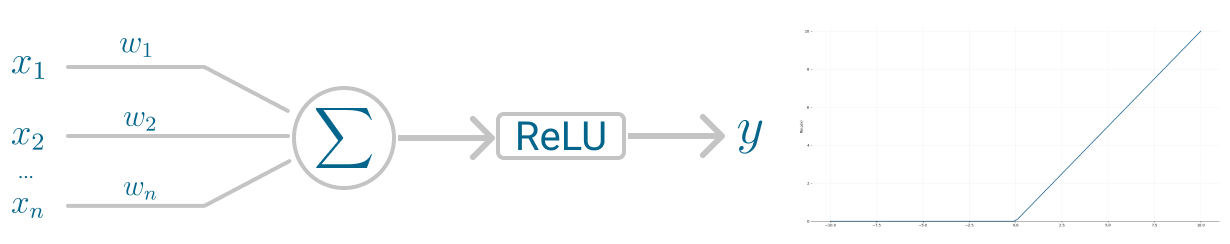
Thuật toán Random Forest là một phần mở rộng của phương pháp đóng bao vì nó sử dụng cả tính năng đóng bao và tính chất ngẫu nhiên để tạo ra một rừng cây quyết định không tương quan với nhau.Tính ngẫu nhiên của đối tượng.tạo ra một tập hợp con ngẫu nhiên của các đối tượng, đảm bảo tính tương quan thấp giữa các cây quyết định. Đây là điểm khác biệt chính giữa cây quyết định và rừng ngẫu nhiên. Trong khi cây quyết định xem xét tất cả các phân tách đối tượng có thể có, các khu rừng ngẫu nhiên chỉ chọn một tập hợp con của các đối tượng địa lý đó. Bằng cách tính đến tất cả các biến thể tiềm ẩn trong dữ liệu, RF có thể giảm overfitting , thiên vị và phương sai tổng thể, dẫn đến các dự đoán chính xác hơn.



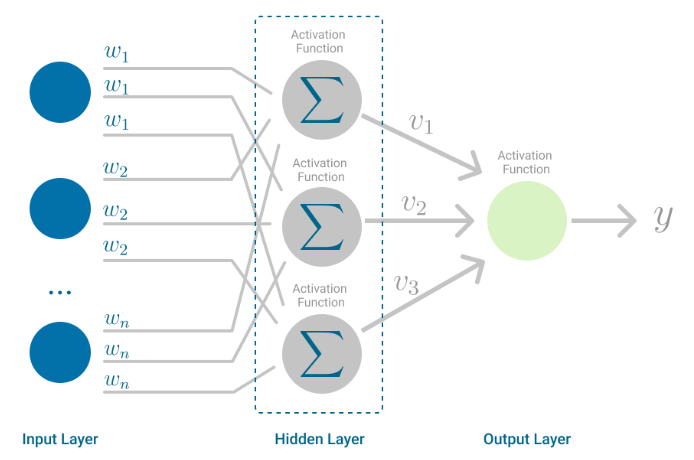
### MLP

Perceptron nhiều lớp ( MLP ) là một lớp được kết nối đầy đủ của mạng nơ-ron nhân tạo truyền thằng với mỗi Perceptron được cấu tạo bởi các hàm năng lượng và hàm phân loại đầu ra xác định xem nơron có hoạt động hay không.

Perceptron sử dụng SGD làm hàm tối ưu hóa.tập hợp trọng số giúp giảm thiểu khoảng cách giữa các điểm bị phân loại sai và ranh giới quyết định. Khi Stochastic Gradient Descent hội tụ, tập dữ liệu được tách thành hai vùng bởi một siêu phẳng tuyến tính.

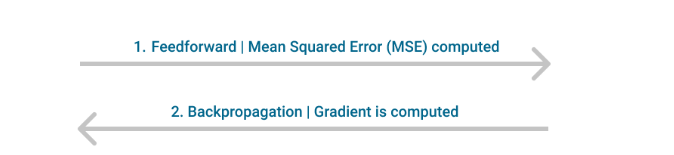


MLP có các lớp đầu vào và đầu ra, và một hoặc nhiều lớp ẩn với nhiều nơ-ron xếp chồng lên nhau. Và khi ở trong Perceptron, neuron phải có chức năng kích hoạt áp đặt ngưỡng, như ReLU hoặc sigmoid, neuron trong Perceptron nhiều lớp có thể sử dụng bất kỳ chức năng kích hoạt tùy ý nào.MLP thuộc loại thuật toán chuyển tiếp , bởi vì đầu vào được kết hợp với trọng số ban đầu thành một tổng có trọng số và chịu chức năng kích hoạt, giống như trong Perceptron. Nhưng sự khác biệt là mỗi kết hợp tuyến tính được truyền sang lớp tiếp theo.Mỗi lớp đang cung cấp cho lớp tiếp theo với kết quả tính toán của chúng, biểu diễn bên trong của chúng về dữ liệu. Điều này đi qua tất cả các lớp ẩn đến lớp đầu ra.



Nếu thuật toán chỉ tính tổng trọng số trong mỗi nơ-ron, truyền kết quả đến lớp đầu ra và dừng lại ở đó, nó sẽ không thể học các trọng số làm giảm thiểu hàm chi phí. Nếu thuật toán chỉ tính toán một lần lặp, sẽ không có quá trình học thực sự.

Backpropagation là cơ chế học tập cho phép Multilayer Perceptron điều chỉnh lặp đi lặp lại các trọng số trong mạng, với mục tiêu giảm thiểu hàm chi phí.Có một yêu cầu khó để nhân giống ngược hoạt động bình thường. Hàm kết hợp các đầu vào và trọng số trong một nơ-ron, chẳng hạn như tổng có trọng số và hàm ngưỡng, ví dụ như ReLU, phải có thể phân biệt được. Các hàm này phải có đạo hàm bị giới hạn , vì [Gradient Descent](https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32) thường là hàm tối ưu hóa được sử dụng trong MultiLayer Perceptron.



Trong mỗi lần lặp, sau khi tổng trọng số được chuyển tiếp qua tất cả các lớp, gradient của Lỗi bình phương trung bình được tính trên tất cả các cặp đầu vào và đầu ra. Sau đó, để lan truyền nó trở lại, trọng số của lớp ẩn đầu tiên được cập nhật với giá trị của gradient. Đó là cách các trọng số được truyền trở lại điểm bắt đầu của mạng nơ-ron.Quá trình này tiếp tục diễn ra cho đến khi gradient cho mỗi cặp đầu vào-đầu ra đã hội tụ, có nghĩa là gradient mới được tính toán không thay đổi nhiều hơn ngưỡng hội tụ được chỉ định , so với lần lặp trước đó.

# **Kết quả**