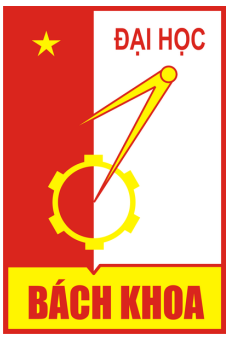
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



ĐỒ ÁN I

SỬ DỤNG THUẬT TOÁN STACKING ENSEMBLE LEARNING ĐỂ PHÁT HIỆN THAO TÚNG TRONG GIAO DỊCH CHỨNG KHOÁN

Chuyên ngành: Toán ứng dụng

Giảng viên hướng dẫn:	**TS. Nguyễn Thị Ngọc Anh**
Sinh viên thực hiện:	**Trương Việt Dũng**
MSSV:	**20206278**
Lớp:	**HTTT 01 - K65**

Hà Nội, tháng 6 năm 2023

NHẬN XÉT CỦA GIẢNG VIÊN

1. Mục tiêu

(a)

(b)

(c)

2. Nội dung

(a)

(b)

(c)

3. Đánh giá kết quả đạt được

(a)

(b)

(c)

Hà Nội, ngày ... tháng ... năm 2023

Giảng viên hướng dẫn

TS. NGUYỄN THỊ NGỌC ANH

Lời Cảm Ơn

Báo cáo này được thực hiện và hoàn thành tại Trường Đại học Bách Khoa Hà Nội, nằm trong nội dung học phần Đồ án I của kì học 2022-2.
Em xin được dành lời cảm ơn chân thành tới TS. Nguyễn Thị Ngọc Anh, là giảng viên đã trực tiếp hướng dẫn và gợi ý cho em đề tài rất thú vị này, đồng thời cô cũng đã giúp đỡ tận tình và có những góp ý, định hướng bổ ích để em hiểu hơn về đề tài, từ đó có thể hoàn thành báo cáo này một cách tốt nhất.

Hà Nội, tháng 06 năm 2023

Tác giả đồ án

Trương Việt Dũng

Tóm tắt nội dung đồ án

Trong đồ án này, em đã áp dụng các phương pháp học máy nhằm xây dựng nên một mô hình phát hiện thao túng trong thị trường chứng khoán. Qua việc ứng dụng và kết hợp các thuật toán học máy phân loại sao cho đạt hiệu quả tốt hơn, em đã xây dựng mô hình Stacking Ensemble Learning nhằm kết hợp nhiều mô hình học máy phân loại lại với nhau. Khi xem xét đến vấn đề dữ liệu hạn chế và mất cân bằng, em tiếp tục sử dụng phương pháp K-Fold Cross Validation khi chia dữ liệu để tránh bị overfitting và đánh giá mô hình đầy đủ và chính xác hơn, kỹ thuật lấy mẫu quá mức cho nhóm thiểu số tổng hợp ở Borderline (Borderline SMOTE) để lấy mẫu quá mức cho lớp thiểu số. Cuối cùng em dùng phương pháp F1-score để đánh giá hiệu suất của mô hình. Kết quả khi thực nghiệm cho thấy mô hình là khả quan nhưng cũng cần cải thiện hơn trong tương lai.

Mục lục

Chương 1

Giới thiệu

1.1 Động cơ phát triển đồ án

Thị trường chứng khoán tại Việt Nam đang ngày càng phát triển, điều đó cũng đồng nghĩa với việc chúng ta sẽ phải đối mặt với nhiều thách thức trong việc quản lý, giám sát để giữ cho thị trường trong sạch và ổn định. Đặc biệt là trong vấn đề thao túng thị trường, vì điều đó sẽ làm ảnh hưởng rất nhiều đến chức năng, trật tự của thị trường cũng như ảnh hưởng tới các nhà đầu tư.

Các thuật toán học máy đang dần được áp dụng rộng rãi trong việc giải quyết các vấn đề của xã hội và việc sử dụng chúng hứa hẹn sẽ giúp hỗ trợ và tiết kiệm công sức cho việc phát hiện những vi phạm trên. Vì vậy mà việc nghiên cứu xây dựng nên một mô hình hiệu quả sẽ mang đến thêm những giải pháp nhằm bảo vệ thị trường chưng khoán khỏi những hành vi thao túng, giúp thị trường ổn định và phát triển hơn.

1.2 Các nghiên cứu liên quan

Với ý nghĩa quan trọng của mình mà vấn đề phát hiện thao túng đã nhận được nhiều sự quan tâm nghiên cứu nhằm phát triển các phương pháp và công cụ phát hiện thao túng.Tiêu biểu phải kể đến "Detecting stock market manipulation via machine learning: Evidence from China Securities Regulatory Commission punish-

ment cases" của Qingbai Liu, Chuanjie Wang, Ping Zhang, Kaixin Zheng[2021]. Trong bài báo này các tác giả áp dụng các kỹ thuật học máy để xây dựng các mô hình phát hiện thao túng trên thị trường chứng khoán. Bằng cách kết hợp các trường hợp xử phạt của Ủy ban điều tiết chứng khoán Trung Quốc được thu thập thủ công từ năm 2014 đến năm 2016 với thông tin tài chính của các công ty niêm yết, các tác giả xây dựng một bộ đào tào và một bộ kiểm tra để so sánh khả năng phát hiện của máy vectơ hỗ trợ (SVM) và mô hình logistic. Họ còn kết hợp Kỹ thuật lấy mẫu quá mức cho nhóm thiểu số tổng hợp ở Borderline (Borderline SMOTE) để lấy mẫu quá mức cho lớp thiểu số và sau đó thấy rằng Borderline SMOTE–SVM hoạt động tốt hơn SVM và mô hình chuẩn trong việc phát hiện sự thao túng. Ngoài ra, Để nâng cao hiệu suất phát hiện của các mô hình, họ còn giới thiệu một cách sáng tạo các chỉ báo tâm lý thị trường được trích xuất từ các báo cáo xếp hạng của nhà phân tích, tin tức tài chính và nhận xét của Guba vào bộ chỉ báo của chúng tôi. Kết quả chỉ ra rằng các chỉ số mới tạo ra sự gia tăng biên đáng kể cho độ chính xác của mô hình[1].

Ngoài ra, bài báo "Market manipulation detection: A systematic literature review" của các tác giả Samira Khodabandehlou , Seyyed Alireza Hashemi Golpayegani[2022] cũng đã thực hiện tổng quan một cách có hệ thống các tài liệu về phát hiện thao túng thị trường từ năm 2010 đến năm 2020, và 52 nghiên cứu quan trọng nhất đã được xem xét và phân tích sâu sắc và toàn diện. Trong các nghiên cứu được chọn, một đánh giá đã được tiến hành về các định nghĩa và nguyên tắc phân loại của thao túng dựa trên thương mại; các mục tiêu; vấn đề được điều tra; phương pháp luận; điểm mạnh và điểm yếu; đề xuất giải pháp; các phương pháp, danh mục và kỹ thuật khai thác dữ liệu; và ngoài ra, về các loại thao tác và dị thường khác nhau được điều tra, và các công việc trong tương lai được đề xuất trong các nghiên cứu. Các kết quả quan trọng nhất của bài báo như : đánh giá chuyên sâu và phê bình các nghiên cứu, đề xuất các giải pháp mới và thiết thực để phát hiện thao túng, lập bản đồ các thao tác với các điểm bất thường, xác định các thách thức phải đối mặt trong phát hiện thao túng và các vấn đề mở trong lĩnh vực này, xác định các loại thao túng và dị thường, phương pháp và cách tiếp cận phát hiện,... Kết quả của bài báo giúp hiểu được các phương pháp thao túng và cách phát hiện chúng trong thị trường tài chính điện tử, đồng thời có thể là

kim chỉ nam cho các nhà nghiên cứu trong tương lai thực hiện các nghiên cứu cần thiết trong lĩnh vực này, đồng thời cũng là cơ sở để lựa chọn các phương pháp tiếp cận khoa học để giải quyết các vấn đề mở trong quá trình nghiên cứu xây dựng các mô hình, công cụ phát hiện gian lận và dự báo tài chính[2].

1.3 Mục tiêu của Đồ án

Mục tiêu của đồ án này là bước đầu xây dựng nên một mô hình phát hiện thao túng trong thị trường chứng khoán bằng cách sử dụng Stacking Ensemble Learning nhằm kết hợp các mô hình phân loại học máy. Mô hình này có thể dựa trên các dữ liệu trong quá khứ để đưa ra những dự báo, giúp cảnh báo và phát hiện các trường hợp vi phạm. Ngoài ra, em cũng cố gắng cải thiện hiệu suất mô hình và làm giảm tác động từ sự hạn chế của dữ liệu bằng phương pháp K-Fold Cross-validation và kỹ thuật Borderline-SMOTE với mục tiêu tránh overfitting và xử lý tình trạng mất cân bằng nhãn của dữ liệu.

1.4 Kết cấu Đồ án

Báo cáo đồ án I này sẽ được trình bày gồm 4 chương như sau:

- **Chương 1: Giới thiệu**
- **Chương 2: Mô hình nghiên cứu**
- **Chương 3: Kết quả thực nghiệm**
- **Chương 4: Kết luận và hướng phát triển đề tài**

Chương 2

Mô hình nghiên cứu

Với mục tiêu nhằm dự đoán và phát hiện các gian lận trong vi phạm chứng khoán, mô hình được xây dựng dựa trên các thật toán học máy phân loại và Stacking Ensemble Learning, kết hợp với các phương pháp xử lí mất cân bằng nhãn để cho ra một mô hình ban đầu, giúp đưa ra các dự đoán ở mức chấp nhận được và làm tiền để cho những bước phát triển sau. Các thuật toán và phương pháp đánh giá hiệu suất cũng đã được trình bày đầy đủ nhằm giải thích cho lý do và cách thức mà mô hình hoạt động.

2.1 Các thuật toán phân loại

Các thuật toán phân loại ngày càng đóng vai trò quan trọng trong phát hiện thao túng trên thị trường chứng khoán. Chúng giúp phân loại các giao dịch hoặc hành vi có khả năng gian lận hoặc thao túng trên thị trường chứng khoán. Các mô hình phân loại có thể học từ các mẫu dữ liệu huấn luyện chứa các trường hợp đã biết của gian lận và thao túng, sau đó sử dụng những kiến thức đã học để dự đoán các trường hợp tiềm năng khác.

Bài toán phân loại sẽ được thiết lập như sau: Giả sử ta có tập huấn luyện $(\mathbf{X}, \mathbf{y})$ với $\mathbf{X}$ là ma trận các điểm dữ liệu đặc trưng và $\mathbf{y}$ là vector nhãn (-1 hoặc 1). Từ tập dữ liệu này, ta cần tạo ra một hàm số ánh xạ mỗi phần tử từ tập $\mathbf{X}$ sang một

phần tử tương ứng của tập $\mathbf{y}$.

$$y_i \simeq f(X_i), \quad \forall i = 1, 2, \ldots, n. \tag{2.1}$$

Mục đích của bài toán là xấp xỉ hàm số $f()$ để với bộ dữ liệu mới ở tập kiểm tra (test data) ký hiệu là $\left(\mathbf{x}'_i, y'_i\right)$, mô hình có thể tính toán và dự báo được giá trị/nhãn tương ứng $\widehat{y'_i} = f\left(\mathbf{x}'_i\right)$.

2.1.1 Support Vecter Machine

SVM (Support Vector Machine) là một thuật toán học máy có giám sát phổ biến được sử dụng cho các nhiệm vụ phân loại và hồi quy. SVM được sử dụng rộng rãi trong nhiều lĩnh vực như nhận dạng mẫu, phân tích hình ảnh, phân loại văn bản và sinh học thông tin.

Mục tiêu chính của SVM là tìm một siêu phẳng tối ưu có thể phân tách các điểm dữ liệu thuộc các lớp khác nhau một cách tốt nhất. Siêu phẳng là ranh giới quyết định tối đa hóa khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất từ mỗi lớp, được gọi là các vector hỗ trợ.

Tổng quan về cách SVM hoạt động:

- Chuẩn bị dữ liệu: SVM yêu cầu dữ liệu huấn luyện được gán nhãn, trong đó mỗi điểm dữ liệu được liên kết với một nhãn lớp.
- Lựa chọn và tỷ lệ các đặc trưng: Việc chọn các đặc trưng có ý nghĩa và tỷ lệ chúng một cách phù hợp là rất quan trọng để đảm bảo hiệu suất tốt hơn của mô hình SVM.
- Khởi tạo siêu phẳng: SVM bắt đầu bằng cách chọn một siêu phẳng ban đầu phân tách các lớp, thường là với một khoảng cách lớn. Siêu phẳng được biểu diễn bằng một đường thẳng trong không gian 2D hoặc một siêu phẳng trong các chiều cao hơn.
- Tính toán khoảng cách giữa các lớp: SVM tính toán khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất từ mỗi lớp. Mục tiêu là tối đa hóa khoảng cách này.

- Xác định các vector hỗ trợ: SVM xác định các điểm dữ liệu, được gọi là các vector hỗ trợ, nằm gần ranh giới quyết định.
- Tối ưu hóa khoảng cách: Thuật toán SVM nhằm mục tiêu tối ưu hóa khoảng cách bằng cách điều chỉnh vị trí siêu phẳng. Mục tiêu là tìm ra siêu phẳng tối ưu hóa khoảng cách trong khi giảm thiểu số lõi phân loại.
- Kỹ thuật Kernel: Trong trường hợp dữ liệu không thể phân tách tuyến tính, SVM có thể sử dụng kỹ thuật Kernel để biến đổi các đặc trưng đầu vào thành không gian có số chiều cao hơn, nơi phân tách tuyến tính trở nên khả thi. Các hàm kernel thông thường bao gồm tuyến tính, đa thức, hạt nhân cơ sở hình quả tròn (RBF) và sigmoid.
- Phân loại: Khi siêu phẳng được xác định, SVM có thể phân loại các điểm dữ liệu mới chưa được quan sát dựa trên việc chúng nằm ở phía nào của siêu phẳng.

Ta có các cặp dữ liệu của tập huấn luyện là $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ với vector $x_i \in R^d$ thể hiện đầu vào của một điểm dữ liệu và y_i là nhãn của điểm dữ liệu đó, d là số chiều của dữ liệu và n là số điểm dữ liệu. Giả sử nhãn của mỗi điểm dữ liệu được xác định bởi $y_i = 1$ (class 1) hoặc $y_i = -1$ (class 2) và mặt $W^T\mathbf{X} + b = 0$ là mặt phẳng phân chia giữa 2 classes. Bài toán tối ưu trong SVM chính là bài toán tìm W và b sao cho:

$$(\mathbf{w}, b) = \arg\max_{w,b} \left\{ \min_n \frac{y_n \left(\mathbf{w}^T\mathbf{x}_n + b\right)}{\|\mathbf{w}\|_2} \right\} = \arg\max_{w,b} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n \left(\mathbf{w}^T\mathbf{x}_n + b\right) \right\}. \tag{2.2}$$

Sau một số phép biến đổi biểu thức, bài toán đưa ra ở phương trình trên có thể đưa về bài toán tối ưu có ràng buộc sau đây:

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},\mathrm{b}} \frac{1}{2}\|\mathbf{w}\|_2^2, \tag{2.3}$$

trong đó $1 - y_n \left(\mathrm{w}^T\mathrm{x}_n + b\right) \leq 0, \forall n = 1, 2, \ldots, N$

Sau khi tìm được mặt phân cách $\mathbf{W}^T\mathbf{X} + b = 0$, lớp của bất kỳ một điểm dữ liệu

nào sẽ được xác định bằng phương trình:

$$class(x) = \text{sgn}\left(\mathbf{W}^T\mathbf{X} + b\right). \tag{2.4}$$

SVM là một trong những thuật toán hoạt động khá hiệu quả trong lớp các bài toán phân loại hai lớp và dự báo của học có giám sát. Nhờ vậy mà SVM đã trở thành một trong những công cụ mạnh mẽ trong học máy, được áp dụng rộng rãi trong nhiều lĩnh vực như nhận dạng chữ viết tay, nhận dạng ảnh, phân loại văn bản, và đặc biệt là phát hiện gian lận. **Mã giả thuật toán SVM**

Input: Tập dữ liệu huấn luyện $(\mathbf{X}, \mathbf{y})$ với $\mathbf{X}$ là ma trận các điểm dữ liệu đặc trưng và $\mathbf{y}$ là vector nhãn ($+1$ hoặc -1).

Output: Vector trọng số $\mathbf{W}$ và giá trị điều chỉnh b cho siêu phẳng tối ưu.

1. Khởi tạo vector trọng số $\mathbf{W}$ và giá trị điều chỉnh b ban đầu.
2. Lặp lại các bước sau cho một số lượng epochs hoặc cho đến khi hội tụ:
   (a) Đặt độ lỗi (loss) bằng 0.
   (b) Lặp lại qua từng điểm dữ liệu $\mathbf{x}_i$ và nhãn tương ứng y_i trong tập huấn luyện:
   - Nếu $y_i(\mathbf{W}^T\mathbf{x}_i + b) \geq 1$, không có lỗi xảy ra, tiếp tục vòng lặp.
   - Nếu $y_i(\mathbf{W}^T\mathbf{x}_i + b) < 1$, có lỗi xảy ra, cập nhật độ lỗi và cập nhật vector trọng số và giá trị điều chỉnh:

$$\mathbf{W} = \mathbf{W} - \eta\left(2\lambda\mathbf{W} - y_i\mathbf{x}_i\right)$$
$$b = b + \eta y_i$$

   Trong đó, η là tỷ lệ học (learning rate) và λ là hệ số ràng buộc (regularization parameter).

   (c) Tính độ lỗi trung bình và kiểm tra điều kiện hội tụ (nếu đạt được điều kiện hội tụ thì thoát vòng lặp).
3. Trả về vector trọng số $\mathbf{W}$ và giá trị điều chỉnh b đã tối ưu.

2.1.2 Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (ANN) là một loại mô hình học máy được lấy cảm hứng từ cấu trúc và hoạt động của hệ thống nơ-ron trong não người. Nó là một mô hình tính toán song song và được sử dụng rộng rãi trong các nhiệm vụ như phân loại, dự đoán, và nhận dạng mẫu.

Mạng nơ-ron nhân tạo bao gồm một số lượng lớn các nơ-ron nhân tạo (unit) được tổ chức thành các lớp. Mỗi nơ-ron nhân tạo nhận đầu vào, thực hiện một phép tính toán đơn giản và truyền kết quả đến nơ-ron tiếp theo. Các nơ-ron thường được kết nối theo một cấu trúc mang tính sắp xếp, trong đó mỗi nơ-ron trong một lớp kết nối với các nơ-ron trong lớp liền kề.

Thuật toán ANN được chia thành hai giai đoạn chính: giai đoạn lan truyền thuận (feedforward) và giai đoạn lan truyền ngược (backpropagation).

Lan truyền thuận (Feedforward):

- Bước 1: Khởi tạo trọng số ngẫu nhiên cho các kết nối giữa các nơ-ron.
- Bước 2: Đưa dữ liệu đầu vào vào lớp đầu vào.
- Bước 3: Lan truyền dữ liệu qua các lớp ẩn, tính toán đầu ra của mỗi nơ-ron bằng cách áp dụng hàm kích hoạt lên tổng trọng số của các đầu vào.
- Bước 4: Lặp lại bước 3 cho đến khi dữ liệu đạt lớp đầu ra. Đầu ra của mạng nơ-ron là kết quả của các nơ-ron trong lớp đầu ra.

Lan truyền ngược (Backpropagation):

- Bước 1: Tính toán độ lỗi giữa đầu ra thực tế và đầu ra dự đoán của mạng nơ-ron.
- Bước 2: Cập nhật trọng số ngược dựa trên độ lỗi đó. Thuật toán lan truyền ngược sử dụng phương pháp gradient descent để điều chỉnh trọng số sao cho giảm thiểu độ lỗi.
- Bước 3: Lan truyền ngược độ lỗi từ lớp đầu ra qua các lớp ẩn, tính toán độ lỗi ẩn của mỗi nơ-ron.

- Bước 4: Lặp lại bước 2 và bước 3 cho đến khi cập nhật trọng số cho tất cả các kết nối trong mạng.

Như vậy, thuật toán ANN có thể được trình bày tổng quát lại như sau:

- Bước 1 (Khởi tạo): Khởi tạo các trọng số ngẫu nhiên cho các kết nối giữa các nút (neuron) trong mạng.

- Bước 2 (Feedforward): Đầu vào của mạng: $\mathbf{x} = [x_1, x_2, \ldots, x_n]$.
  Tính giá trị đầu ra của các nút ẩn (hidden layer) và nút đầu ra (output layer) thông qua phép tính trọng bình của đầu vào và trọng số, sau đó áp dụng hàm kích hoạt (activation function):

  Tính giá trị đầu ra của nút j ẩn tại lớp l:

$$z_j^l = \sum_{i=1}^{n} w_{ij}^l \cdot x_i + b_j^l. \tag{2.5}$$

  Áp dụng hàm kích hoạt $f(\cdot)$ cho giá trị z_j^l để tính giá trị đầu ra của nút j ẩn:

$$a_j^l = f(z_j^l). \tag{2.6}$$

- Bước 3 (Tính sai số Loss): So sánh giá trị dự đoán của mạng với giá trị thực tế (đầu ra mong muốn) để tính sai số (loss function - L).

- Bước 4 (Backpropagation): Sử dụng thuật toán lan truyền ngược để cập nhật các trọng số của mạng để giảm thiểu sai số (loss). Công thức cập nhật trọng số w_{ij}^l từ nút i ở lớp $l-1$ tới nút j ở lớp l:

$$w_{ij}^l = w_{ij}^l - \alpha \frac{\partial L}{\partial w_{ij}^l}, \tag{2.7}$$

  Trong đó, α là tỷ lệ học (learning rate) để điều chỉnh độ lớn của bước cập nhật.

- Bước 5 (Lặp lại): Lặp lại các bước Feedforward, Tính sai số, Backpropagation với dữ liệu huấn luyện nhiều lần để cập nhật và điều chỉnh các trọng số sao cho mạng học tốt hơn.

Tuy ANN là một công cụ mạnh mẽ, nhưng nó cũng có nhược điểm như cần một lượng lớn dữ liệu huấn luyện và đòi hỏi tính toán phức tạp, đặc biệt khi mô hình lớn. Sự phát triển và kết hợp các mô hình ANN với các phương pháp học máy khác đã dẫn đến những tiến bộ đáng kể trong lĩnh vực trí tuệ nhân tạo và ứng dụng thực tế.

Mã giả cho thuật toán ANN

Input: Tập dữ liệu huấn luyện $(\mathbf{X}, \mathbf{y})$ với $\mathbf{X}$ là ma trận các điểm dữ liệu đặc trưng và $\mathbf{y}$ là vector nhãn ($+1$ hoặc -1).
Output: Các trọng số của mạng $\mathbf{W}$ và các giá trị điều chỉnh (biases) b.

1. Khởi tạo các trọng số của mạng $\mathbf{W}$ và các giá trị điều chỉnh b ngẫu nhiên hoặc theo một cách khác.
2. Lặp lại các bước sau cho một số lượng epochs hoặc cho đến khi hội tụ:
   (a) Đặt độ lỗi (loss) bằng 0.
   (b) Lặp lại qua từng điểm dữ liệu $\mathbf{x}_i$ và nhãn tương ứng y_i trong tập huấn luyện:
   - Thực hiện quá trình feedforward để tính toán giá trị đầu ra của mạng:
   $$z_j^l = \sum_{i=1}^{n} w_{ij}^l \cdot x_i + b_j^l$$
   $$a_j^l = f(z_j^l)$$
   - Tính toán độ lỗi (loss) của điểm dữ liệu này bằng một hàm mất mát (loss function) như Cross-Entropy Loss hoặc Mean Squared Error (MSE).
   - Cập nhật độ lỗi tổng thể.
   - Tính toán độ lỗi của các nút đầu ra (output layer) và các nút ẩn (hidden layers) bằng lan truyền ngược (backpropagation).
   (c) Cập nhật các trọng số của mạng $\mathbf{W}$ và các giá trị điều chỉnh b bằng một thuật toán tối ưu hóa như Gradient Descent hoặc Stochastic Gradient Descent (SGD).

3. Trả về các trọng số của mạng $\mathbf{W}$ và các giá trị điều chỉnh b đã được huấn luyện.

2.1.3 Random Forest

Random Forest là một thuật toán học máy giám sát trong lĩnh vực của cây quyết định (decision tree) và tổ hợp (ensemble learning). Nó được sử dụng chủ yếu cho các nhiệm vụ phân loại và hồi quy.

Random Forest có một số ưu điểm, bao gồm khả năng làm việc với dữ liệu lớn, khả năng xử lý các đặc trưng có ý nghĩa khác nhau, khả năng xác định mức độ quan trọng của các đặc trưng, và khả năng giảm thiểu tác động của overfitting. Ngoài ra, Random Forest cũng có khả năng xử lý dữ liệu không cân bằng và có thể được sử dụng cho cả các bài toán phân loại và hồi quy.

Ý tưởng chính của Random Forest là xây dựng một tập hợp các cây quyết định độc lập và kết hợp kết quả của chúng để đưa ra dự đoán cuối cùng.

Các bước cơ bản của thuật toán Random Forest:

- Chuẩn bị dữ liệu: Tiền xử lý dữ liệu bằng cách chuẩn hóa, xử lý giá trị thiếu, chọn đặc trưng và phân chia tập huấn luyện và tập kiểm tra.
- Xây dựng các cây quyết định: Mỗi cây quyết định được xây dựng trên một tập dữ liệu con được lấy mẫu ngẫu nhiên từ tập huấn luyện. Với mỗi cây, tại mỗi nút trong cây, một số lượng ngẫu nhiên các đặc trưng được chọn để xem xét cho phân chia tốt nhất.
- Huấn luyện các cây quyết định: Với mỗi nút trong cây, tìm phân chia tốt nhất bằng cách tối ưu hóa một tiêu chí đo lường, ví dụ như hệ số Gini hoặc độ suy giảm thông tin. Tiếp tục phân chia cho đến khi đạt được một tiêu chí dừng.
- Kết hợp dự đoán: Khi có dữ liệu mới, Random Forest lấy dự đoán của các cây quyết định và kết hợp chúng để đưa ra dự đoán cuối cùng. Trong trường hợp phân loại, kết quả dự đoán cuối cùng được xác định bằng cách bầu chọn từ các cây quyết định. Trong trường hợp hồi quy, kết quả cuối cùng là giá trị trung bình của các dự đoán từ các cây.

Mã giả cho hoạt động của Random Forest :

1. Chọn ngẫu nhiên "k" features từ tập "m" features.(Để ý $k \ll m$)

2. từ tập "k" features, tính toán ra node "d" là tốt nhất cho Node phân loại.

3. Chia các node con theo node tốt nhất vừa tìm được

4. Lặp lại bước 1-3 cho đến khi đẹt đến **k** node

5. Lặp lại bước 1-4 để tạo ra "n" cây

Random Forest là một thuật toán khá mới, được sử dụng trong vòng 10 năm gần đây, và có giá trị lớn trong những thuật toán Surpervised Learning.

2.1.4 Hồi quy Logistic

Hồi quy Logistic (Logistic Regression) là một thuật toán học máy được sử dụng chủ yếu cho các bài toán phân loại. Mặc dù có tên là "hồi quy," nó thực chất là một thuật toán phân loại.

Ý tưởng cơ bản của hồi quy Logistic là xây dựng một mô hình tuyến tính, nhưng sau đó áp dụng một hàm kích hoạt phi tuyến (thường là hàm sigmoid) để chuyển đổi đầu ra thành một giá trị xác suất. Kết quả là một phân phối xác suất dự đoán cho các lớp khác nhau.

Dưới đây là các khía cạnh chính của hồi quy Logistic:

- Hàm kích hoạt Sigmoid: Hàm sigmoid được sử dụng để biến đổi đầu ra của mô hình tuyến tính thành một giá trị xác suất nằm trong khoảng từ 0 đến 1. Công thức của hàm sigmoid là: $sigmoid(z) = \frac{1}{1/(1+e^{(-z)})}$) trong đó z là tổng trọng số của các đặc trưng đầu vào.

- Hàm mất mát (Loss Function): Hồi quy Logistic Xây dựng các cây quyết định: Mỗi cây quyết định được xây dựng trên một hàm mất mát như hàm cross-entropy để đo lường sự sai khác giữa xác suất dự đoán và nhãn thực tế. Mục tiêu là tìm tập dữ liệu con được lấy mẫu ngẫu nhiên từ tập huấn luyện. Với mỗi cây, tại mỗi nút trong cây, một số lượng ngẫu nhiên các đặc trọưng số tối ưu để giảm thiểu hàm mất mát được chọn để xem xét cho phân chia

tốt nhất.

- Gradient Descent: Phương pháp tối ưu hóa thường được sử dụng trong Logistic Regression là gradient descent. Thuật toán này điều chỉnh các trọng số dựa trên đạo hàm của hàm mất mát, để di chuyển dần về phía hướng giảm mất mátHuấn luyện các cây quyết định: Với mỗi nút trong cây, tìm phân chia tốt nhất bằng cách tối ưu hóa một tiêu chí đo lường, ví dụ như hệ số Gini hoặc độ suy giảm thông tin. Tiếp tục phân chia cho đến khi đạt được một tiêu chí dừng.

- Đánh giá và dự đoán: Sau khi huấn luyện, mô hình Logistic Regression có thể được sử dụng để dự đoán xác suất thuộc về một lớp cụ thể dựa trên đầu vào mới. Ngưỡng (threshold) có thể được áp dụng để quyết định lớp cuối cùng dựa trên xác suất dự đoán. Kết hợp dự đoán: Khi có dữ liệu mới, Random Forest lấy dự đoán của các cây quyết định và kết hợp chúng để đưa ra dự đoán cuối cùng. Trong trường hợp phân loại, kết quả dự đoán cuối cùng được xác định bằng cách bầu chọn từ các cây quyết định. Trong trường hợp hồi quy, kết quả cuối cùng là giá trị trung bình của các dự đoán từ các cây.

Giả sử rằng xác suất để một điểm dữ liệu $\mathbf{x}$ rơi vào class 1 là $f\left(\mathbf{w}^T\mathbf{x}\right)$ và rơi vào class 0 là $1 - f\left(\mathbf{w}^T\mathbf{x}\right)$. Với mô hình được giả sử như vậy, với các điểm dữ liệu training (đã biết đầu ra y), ta có thể viết như sau:

$$P\left(y_i = 1 \mid \mathbf{x}_i; \mathbf{w}\right) = \quad f\left(\mathbf{w}^T\mathbf{x}_i\right), \tag{2.8}$$

$$P\left(y_i = 0 \mid \mathbf{x}_i; \mathbf{w}\right) = 1 - f\left(\mathbf{w}^T\mathbf{x}_i\right), \tag{2.9}$$

trong đó $P\left(y_i = 1 \mid \mathbf{x}_i; \mathbf{w}\right)$ được hiểu là xác suất xảy ra sự kiện đầu ra $y_i = 1$ khi biết tham số mô hình $\mathbf{w}$ và dữ liệu đầu vào $\mathbf{x}_i$. Bạn đọc có thể đọc thêm Xác suất có điều kiện. Mục đích của chúng ta là tìm các hệ số $\mathbf{w}$ sao cho $f\left(\mathbf{w}^T\mathbf{x}_i\right)$ càng gần với 1 càng tốt với các điểm dữ liệu thuộc class 1 và càng gần với 0 càng tốt với những điểm thuộc class 0 . Ký hiệu $z_i = f\left(\mathbf{w}^T\mathbf{x}_i\right)$ và viết gộp lại hai biểu thức bên trên ta có:

$$P\left(y_i \mid \mathbf{x}_i; \mathbf{w}\right) = z_i^{y_i}\left(1 - z_i\right)^{1-y_i} \tag{2.10}$$

Biểu thức này là tương đương với hai biểu thức ở trên vì khi $y_i = 1$, phần thứ hai của vế phải sẽ triệt tiêu, khi $y_i = 0$, phần thứ nhất sẽ bị triệt tiêu! Chúng ta muốn mô hình gần với dữ liệu đã cho nhất, tức xác suất này đạt giá trị cao nhất.

Xét toàn bộ training set với $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$ và $\mathbf{y} = [y_1, y_2, \ldots, y_N]$, chúng ta cần tìm $\mathbf{w}$ để biểu thức sau đây đạt giá trị lớn nhất:

$$P(\mathbf{y} \mid \mathbf{X}; \mathbf{w}).$$

Nói cách khác bài toán sẽ là tìm tham số để mô hình gần dữ liệu nhất:

$$\mathbf{w} = \arg\max_{\mathbf{w}} P(\mathbf{y} \mid \mathbf{X}; \mathbf{w}). \tag{2.11}$$

Như vậy, ta có thể kết luận sau khi tìm được mô hình, việc xác định class y cho một điểm dữ liệu $\mathbf{x}$ được xác định bằng việc so sánh hai biểu thức xác suất:

$$P(y = 1 \mid \mathbf{x}; \mathbf{w}); P(y = 0 \mid \mathbf{x}; \mathbf{w}) \tag{2.12}$$

Nếu biểu thức thứ nhất lớn hơn thì ta kết luận điểm dữ liệu thuộc class 1 , ngược lại thì nó thuộc class 0 . Vì tổng hai biểu thức này luôn bằng 1 nên một cách gọn hơn, ta chỉ cần xác định xem $P(y = 1 \mid \mathbf{x}; \mathbf{w})$ lớn hơn 0.5 hay không. Nếu có, class 1 . Nếu không, class 0 .

Mã giả cho thuật toán hồi quy Logistic

Input: Tập dữ liệu huấn luyện $(\mathbf{X}, \mathbf{y})$ với $\mathbf{X}$ là ma trận các điểm dữ liệu đặc trưng và $\mathbf{y}$ là vector nhãn (0 hoặc 1).
Output: Vector trọng số $\mathbf{W}$ cho mô hình logistic regression.

1. Khởi tạo vector trọng số $\mathbf{W}$ ban đầu.
2. Lặp lại các bước sau cho một số lượng epochs hoặc cho đến khi hội tụ:
   (a) Đặt độ lỗi (loss) bằng 0.
   (b) Lặp lại qua từng điểm dữ liệu $\mathbf{x}_i$ và nhãn tương ứng y_i trong tập huấn luyện:
   - Tính giá trị dự đoán $\hat{y}_i$ cho điểm dữ liệu $\mathbf{x}_i$:

$$\hat{y}_i = \sigma(\mathbf{W}^T \mathbf{x}_i).$$

- Tính độ lỗi (loss) của điểm dữ liệu này bằng hàm mất mát (loss function) như Cross-Entropy Loss:

$$\text{loss} = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)].$$

- Cập nhật độ lỗi tổng thể.
- Cập nhật vector trọng số $\mathbf{W}$ bằng thuật toán tối ưu hóa như Gradient Descent hoặc Stochastic Gradient Descent (SGD):

$$\mathbf{W} = \mathbf{W} - \alpha\nabla\text{loss},$$

Trong đó, α là tỷ lệ học (learning rate) để điều chỉnh độ lớn của bước cập nhật và ∇loss là gradient của hàm mất mát theo $\mathbf{W}$.

(c) Tính độ lỗi trung bình và kiểm tra điều kiện hội tụ (nếu đạt được điều kiện hội tụ thì thoát vòng lặp).

3. Trả về vector trọng số $\mathbf{W}$ đã được huấn luyện.

2.2 Borderline Smote

Khi tiến hành xem xét bài toán, chúng ta có thể dễ dàng nhận thấy dữ liệu đầu vào có rất ít nhãn được gắn là thao túng. Vì vậy mà kỹ thuật lấy mẫu quá mức đã được xem xét sử dụng. Kỹ thuật Smote được dùng để lấy mẫu quá mức cho lớp thiểu số bằng cách tạo ra các thể hiện tổng hợp. Tuy nhiên, SMOTE không xem xét thông tin danh mục của các mẫu lân cận gần nhất, dẫn đến hiệu quả phân loại kém. Còn Borderline Smote chỉ sử dụng các mẫu nhỏ trên đường biên để tổng hợp các mẫu mới. Do đó, khi tính đến các đặc tính của mẫu và hiệu quả của thuật toán, Borderline SMOTE đã được chọn để cân bằng các mẫu.

Borderline SMOTE lấy mẫu quá mức lớp thiểu số theo ba bước sau:

- Bước 1: Tính z hàng xóm gần nhất cho mọi mẫu trong lớp thiểu số và biểu thị số lượng mẫu đa số trong số z hàng xóm gần nhất là z'
- Bước 2: : Chia tất cả các mẫu thiểu số thành ba loại—tức là mẫu nhiễu $\big(z = z'\big)$, mẫu an toàn $(\big(0 \leq z' < z/2\big))$ và mẫu nguy hiểm $(z/2 \leq z' < z$). Chỉ mẫu nguy hiểm mới tham gia vào quá trình vượt mẫu sau đây

- Bước 3: Đối với một mẫu nguy hiểm x_i trong lớp thiểu số, chúng tôi chọn ngẫu nhiên một điểm mẫu $\widetilde{x_i}$ từ các điểm lân cận gần nhất của nó trong lớp thiểu số N và tạo ra một điểm mới x_{new} :

$$x_{\text{new}} = x_i + \text{rand}(0,1) \cdot \left(\widetilde{\text{x}_\text{i}} - \text{x}_\text{i}\right), \tag{2.13}$$

trong đó rand(0,1) đại diện cho một số ngẫu nhiên giữa các khoảng (0,1).

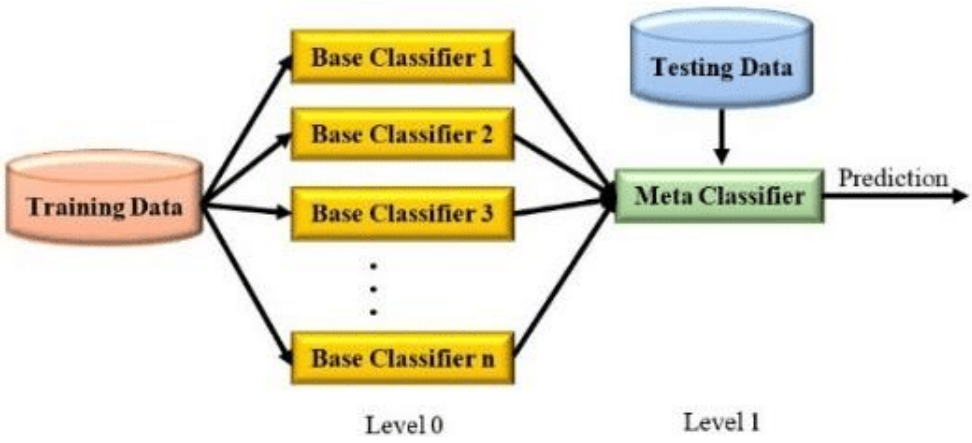
2.3 Stacking Ensemble Learning

Như chúng ta đã biết, việc kết hợp thật nhiều mô hình giống nhau giúp cải thiện kết quả dự đoán. Nhưng với nhiều mô hình học máy, mỗi mô hình có những điểm lợi và hại, và có những cách riêng để đưa ra dự đoán, vậy làm sao để có thể chọn ra mô hình nào tốt nhất, hay làm sao để có thể kết hợp các đặc tính tốt của các mô hình lại với nhau? Câu trả lời là Stacking. Stacking mang đến góc nhìn mới về cách kết hợp các mô hình học máy lại với nhau.

Mô hình Stacking cơ bản thường được phân thành 2 cấp là level-0 models và level-1 model:

- Level-0 Models (Base-Models): Mô hình cơ sở học trực tiếp từ bộ dữ liệu và đưa ra dự đoán cho mô hình level-1
- Level-1 Model (Meta-Model): Mô hình học từ các dự đoán của mô hình cơ sở (level-0)

Có nghĩa Meta-model được huấn luyện dựa trên đầu ra dự đoán của các base-modes, các outputs này kết hợp với nhãn của bài toán tạo thành cặp dữ liệu đầu vào - đầu ra trong quá trình huấn luyện Meta-model. Có thể thấy Meta-model không học trực tiếp từ tập dữ liệu huấn luyện, tuy nhiên, việc dùng dữ liệu bàn đầu thêm vào outputs của base-models vẫn hoàn toàn hợp lý, sẽ cấp thêm cho meta-model nhiều thông tin hơn về dữ liệu.



Hình 2.1: Mô hình Stacking Ensemble Learning

Mô hình Stacking Ensemble Learning được chia làm 3 bước chính:

- Bước 1: Sử dụng các base-models để học trên toàn bộ dữ liệu và đưa ra kết quả dự đoán ban đầu
- Bước 2: Xây dựng bộ dữ liệu mới dựa trên outputs của các base-models
- Bước 3: Huấn luyện Meta-model với bộ dữ liệu mới và đưa ra kết quả cuối cùng

Thuật toán chi tiết:

Input: $D = \left\{(x_i, y_i) \mid x_i \in X, y_i \in Y\right\}$

Output : Kết quả dự đoán **H**

1. **Step 1 :** Học trên toàn bộ dữ liệu và đưa ra kết quả dự đoán ban đầu
2. For $t \leftarrow 1$ to T do
3. Học một bộ phân loại cơ sở h, dựa trên D
4. **Step 2 :** Xây dựng tập dữ liệu mới từ D
5. For $i \leftarrow 1$ to m do
6. Xây dựng tập dữ liệu mới có chứa $\left\{x_i^{\text{new}}, y_i\right\}$, where $x_i^{\text{now}} = \left\{h\left(x_i\right) \text{ for } j = 1 \text{ to } T\right\}$
7. **Step 3 :** Học trên bộ phân loại cấp 2
8. Học trên bộ phân loại mới h^{new} dựa trên dữ liệu mới được tạo
9. **Return** $H(x) = h^{\text{new}}\ \left(h_1(x), h_2(x), \ldots, h_7(x)\right)$

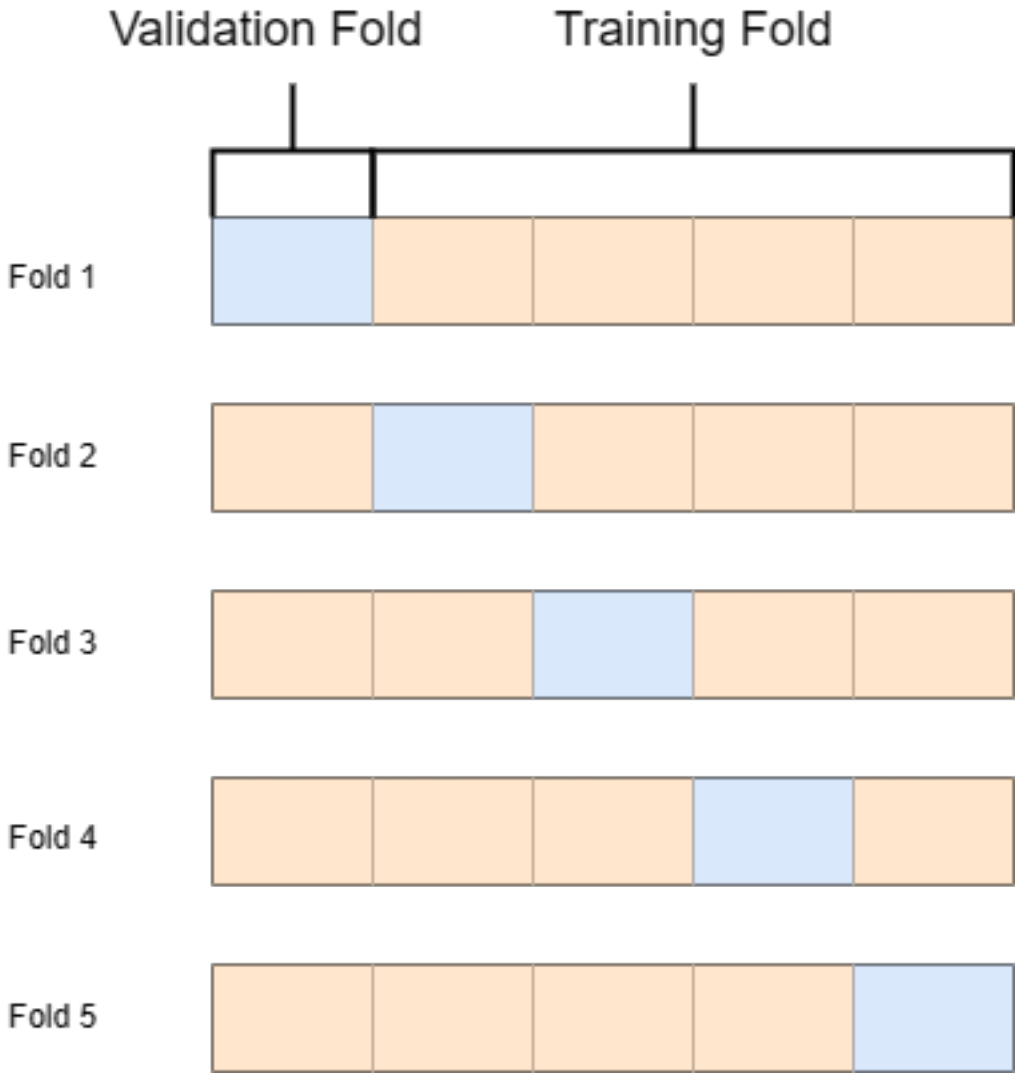
Trong đó X là dữ liệu các biến được dùng để dự đoán, Y là dữ liệu nhãn tương ứng với X và H là kết quả dự đoán ra các nhãn cho bộ dữ liệu được dùng để kiểm tra.

2.4 Stacking Ensemble với Cross-Validation

Tuy nhiên, dữ liệu thu thập được được để chạy mô hình là rất ít(chỉ 138 mẫu) và dữ liệu được dùng để huấn luyện sẽ còn ít hơn. Điều này dẫn đến mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên tập test nữa.

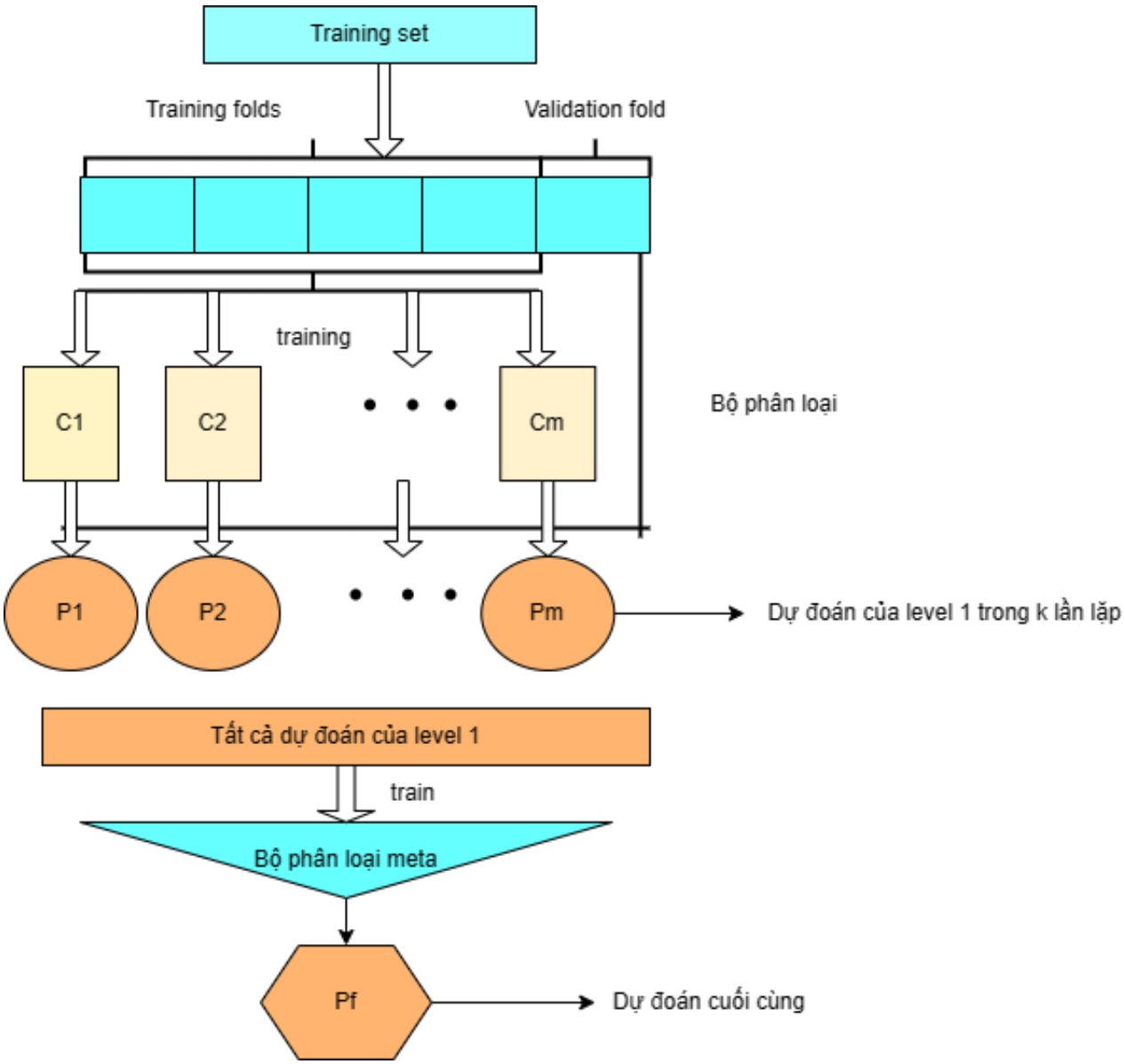
Để giải quyết vấn đề này, thay vì sử dụng phương pháp hold-out(hold-out sẽ chia bộ dữ liệu ra làm 2 phần là training và validation (testing) một cách ngẫu nhiên, khi đó từ một bộ dữ liệu, ta đã có 2 bộ có thể dùng để kiểm thử hiệu năng mô hình) chúng ta sẽ dùng Cross-Validation.

K-fold cross-validation cũng chia bộ dữ liệu ra thành 2 phần chính training và validation, tuy nhiên là một tập các validation set khác nhau được xây dựng từ bộ dataset ban đầu. k-fold nghĩa là bộ dữ liệu sẽ được chia ra làm k phần bằng nhau từ bộ dữ liệu, và mô hình sẽ được huấn luyện k-lần, mỗi lần huấn luyện đc gọi là 1 run. Trong mỗi run, (k-1) phần sẽ được dùng để xây dựng mô hình huấn luyện và phần còn lại sẽ được dùng làm validation set. Kết quả cuối cùng là trung bình các kết quả thu được.



Hình 2.2: Cách k-fold cross-validation chia dữ liệu

Kết hợp Stacking Ensemble Learning với k-fold cross-validation ta thu được mô hình mà khi đó meta-model được huấn luyện dựa trên các nhãn được tạo ra bởi base-models.



Hình 2.3: Mô hình Stacking Ensemble learning với k-fold cross-validation

Thuật toán Stacking Ensemble learning với k-fold cross-validation:

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m} (\mathbf{x}_i \in X, y_i \in Y)$

Output: Kết quả dự đoán H

Step 1: Dùng cross validation chuẩn bị tập huấn luyện cho bộ phân loại cấp 2

Chia ngẫu nhiên $\mathcal{D}$ thành K tập con bằng nhau: $\mathcal{D} = \left\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\right\}$

for $k \leftarrow 1$ to K do

    Step 1.1: Học trên bộ phân loại thứ nhất

    for $t \leftarrow 1$ to T do

    Học trên mô bộ phân loại h_{kt} từ $\mathcal{D} \backslash \mathcal{D}_k$

    end for

    Step 1.2: Xây dựng tập huấn luyện cho bộ phân loại cấp hai

    for $\mathbf{x}_i \in \mathcal{D}_k$ do

        Có bản ghi $\left\{\mathbf{x}'_i, y_i\right\}$, where $\mathbf{x}'_i = \left\{h_{k1}(\mathbf{x}_i), h_{k2}(\mathbf{x}_i), \ldots, h_{kT}(\mathbf{x}_i)\right\}$

    end for

end for

Step 2: Học trên bộ phân loại cấp 2

Học trên bộ phân loại mới h' từ $\left\{\mathbf{x}'_i, y_i\right\}$

Step 3: Học lại bộ phân loại cấp 1

for $t \leftarrow 1$ to T do

    Học trên bộ phân loại h_t dựa trên $\mathcal{D}$

end for

return $H(\mathbf{x}) = h'\left(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x})\right)$

Trong đó X là dữ liệu các biến được dùng để dự đoán, Y là dữ liệu nhãn tương ứng với X và H là kết quả dự đoán ra các nhãn cho bộ dữ liệu được dùng để test.

2.5 Phương pháp đánh giá hiệu suất mô hình

Như đã trình bày ở trên, bộ dữ liệu không cân bằng, vì vậy mà độ chính xác sẽ không được hợp lý. Ví dụ: trong 100 mẫu thì có 2 mẫu thao túng và 98 mẫu không thao túng. Tại thời điểm này, ngay cả khi tất cả các mẫu được xác định là mẫu không thao túng, tỷ lệ chính xác vẫn cao tới 98 %, điều này trái với mục tiêu xác định chứng khoán bị thao túng. Nên để có thể đánh giá toàn diện hiệu quả của mô hình, F1-score là một chỉ số đánh giá hợp lý vì điểm F là trung bình hài hòa của độ chính xác và độ nhạy , có thể đảm bảo rằng khi cả hai thành phần này càng cao thì hiệu suất được đánh giá càng cao.

Công thức cho điểm F1 tiêu chuẩn là trung bình hài hòa của độ chính xác và khả năng nhớ lại. Một mô hình hoàn hảo có điểm F là 1.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
$$= \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

Trong đó:

precision là phần nhỏ của các ví dụ tích cực thực sự trong số các ví dụ mà mô hình phân loại là tích cực. Nói cách khác, số dương tính thật chia cho số dương tính giả cộng với dương tính thật.

recall còn được gọi là độ nhạy, là phần nhỏ của các ví dụ được phân loại là tích cực, trong tổng số các ví dụ tích cực. Nói cách khác, số dương tính thật chia cho số dương tính thật cộng với âm tính giả.

tp: Số lượng dương tính thực sự được phân loại theo mô hình.

tn: Số lượng âm tính giả được phân loại theo mô hình.

fp: Số lượng dương tính giả được phân loại theo mô hình.

Chương 3

Kết quả thực nghiệm

3.1 Dữ liệu và kịch bản chạy chương trình

Dữ liệu được thu thập thủ công trên Vietstock trong khoảng thời gian từ năm 2015 đến năm 2020. Sau đó gắn nhãn để phân biệt các trường hợp thao túng và không thao túng (biến manipulation được gắn là 1 với các trường hợp thao túng và -1 với các trường hợp không thao túng). Dữ liệu sau khi thu thập sẽ có kích thước 20 cột và 138 dòng.

year	maCK	P/E	yeild	turnove	beta	abnor	concen	Divide	leverag	liquidit	profit o	EPS	EPS2	ROE	cash or	TATO	NPGR	growth	manipu
2015	AAA	9.7677	-0.00958	0.005579	0.515813	20	33.94	1	0.079301	11.04784	-0.09757	-1145	642	0.049481	-1.50446	0.573413	-2.44031	-0.44066	-1
2015	ADC	4.646	0.011346	0.000281	-0.30123	29	54.77	0	0.67384	1.262103	0.024529	3623	2580	0.197124	0	1.847829	0.23979	0.277451	-1
2020	AAA	9.7228	0.001081	0.010148	1.219468	24	50.64	1	0.44499	2.12623	0.134105	5179	1310	0.070372	1.970065	0.53106	0.417947	0.023693	-1
2015	ALT	9.7975	0.021572	7.98E-05	0.086972	47	41.66	1	0.594336	1.480355	0.073377	2479	1048	0.02952	3.162816	0.820277	0.171873	-0.20627	-1
2015	AAM	29.2986	-0.00033	0.000366	0.178396	18	35.01	1	0.300493	0.881453		0	226	0.010269	-0.79733	0	-460.574	-1	-1
2015	AMC	4.0413	0.000661	0.000307	-0.06072	35	40	1	0.548226	0.769159	-0.09394	0	3767	0.251136	1.031556	0.069752	-1.07146	-0.86003	-1
2015	AME	26.1604	0.002619	5.85E-05	-0.03577	24	65.5	0	0.834267	1.100478	0.036267	4077	109	0.009726	65.68888	1.319362	0.464877	0.487474	-1
2015	ABT	5.6675	0.005511	0.000137	-0.04073	22	78.09	1	0.877229	1.068679	0.143622	1803	5713	0.169046	1.723502	0.076851	0.895927	0.177086	-1
2015	AMV	-4.5516	0.001212	0.000339	0.269778	27	39.46	0	0.09832	7.646569	0.256933	6273	-309	-0.05821	0	0.804502	0.446694	0.153719	-1
2015	API	38.588	0.002847	0.012676	0.720705	32	30.98	0	0.633118	0.579097	0.045898	750	286	0.02616	-5.98184	0.635871	0.041036	0.014654	-1
2017	AAA	5.6066	0.000443	0.015706	1.028913	26	39.69	1	0.354979	2.179177	0.044436	419	3426	0.162073	-0.24739	0.538782	-0.53311	-0.40882	-1
2015	ACC	3.5941	0.000899	8.57E-05	0.305392	18	83.36	1	0.660468	1.075161	-0.10727	-1962	3112	0.174879	0.899479	0.55267	-2.80303	-0.08713	-1
2015	ARM	5.6165	0.022036	0.000598	-0.0268	15	56.83	1	0.648015	1.196937	0.001816	74	2737	0.182971	-0.05124	1.356194	-0.09726	0.468775	-1
2015	BBS	5.5904	0.015186	0.00034	0.110306	23	48.99	1	0.15133	1.054373	0.297651	2678	1989	0.100735	0	0.502988	0.086785	0.236106	-1
2015	ACL	2.3096	0.00825	0.000405	0.30611	20	33.21	1	0.297287	2.045468	0.073161	1707	1583	0.097005	-1.58493	0.981596	0.079065	0.069511	-1
2015	BCC	3.4107	0.001382	0.002397	1.106216	22	73.14	1	0.352968	1.206254	0.000662	16	2722	0.148858	2.149978	1.152812	-0.94555	0.021939	-1
2018	AAA	10.3444	-0.00155	0.018119	0.781763	19	54.1	1	0.627991	1.176943	0.043397	1280	1115	0.071185	0.15687	0.943481	0.450877	0.396417	-1
2015	BDB	7.2646	0.001867	6.09E-06	-0.04137	24	40.17	1	0.353684	1.572033	0.055403	3546	517	0.050632	0	1.507764	-0.21893	-0.07596	-1
2015	BED	11.4321	-0.00083	0.000583	-0.05148	21	60.99	1	0.376525	1.439422	0.039009	987	1534	0.111561	0.951229	0.674023	0.593363	0.154463	-1
2020	AAV	31.0891	0.006148	0.009225	1.075557	36	39	0	0.717482	0.562576	0.011567	880	419	0.037118	-1.33294	1.516349	-0.2009	0.064533	-1
2015	BII	30.056	0.001005	0.005294	0.905168	25	20.41	1	0.850213	1.081002	0.011526	525	278	0.017929	-4.58469	0.756143	-0.13405	0.09551	-1
2015	AGM	2.7457	9.73E-05	0.000173	0.487417	27	78.5	1	0.11229	4.331741	0.012771	2587	2602	0.123571	0.465436	5.814846	-0.27424	-0.31029	-1

Hình 3.1: Dữ liệu được thu thập và gắn nhãn

Các chỉ số được thu thập dựa trên một nghiên cứu đã được kiểm chứng và được định nghĩa như sau:

Tên biến	Định nghĩa
PE (Price - Earnings)	Tỷ lệ giá/thu nhập trước khi có thao túng đối với cổ phiếu bị thao túng hoặc vào cuối năm tài chính đối với các cổ phiếu khác. (Chính là giá thị trường của cổ phiếu đấy)
Yield	Lợi suất đầu tư, số % được tính theo thu nhập bằng tiền của người sở hữu chứng khoán.
Turnover	Tổng số cổ phiếu giao dịch tại mỗi tháng trên tổng số cổ phiếu đang lưu hành (tổng số cổ phiếu mà công ty được phép giao dịch) It is calculated by dividing the daily volume of a stock by the "float" of a stock (A stock float is the total number of shares that are available for public investors to buy and sell.), which is the number of shares available for sale by the general trading public.
Beta (Hệ số rủi ro)	Đo lường mức độ biến động/thước đo rủi ro hệ thống của một chứng khoán hay 1 danh mục đầu tư trong tương quan với toàn bộ thị trường.
Abnormal (Sự kiện bất thường)	Số lượng các sự kiện (lần) giá cổ phiếu biến động bất thường được công bố trong năm tài chính.
Concentration	Tỷ lệ sở hữu cổ phần của 3 cổ đông hàng đầu doanh nghiệp.
Dividend	Biến giả (chỉ dùng cho cổ phiếu bị thao túng), bằng 1 nếu DN có kế hoạch phân bổ cổ phiếu, cổ tức hoặc phát hành thêm cổ phiếu trong thông báo công bố tài chính hàng quý trước khi có thao túng; bằng 0 nếu ngược lại.
Leverage	Là hệ số nợ của DN, phản ánh năng lực sử dụng và quản lý nợ. Nếu tỷ số có giá trị càng cao, tài sản của DN hầu hết đến từ vốn vay, mức độ rủi ro tài chính lớn. Nếu tỷ số có giá trị quá thấp, DN sử dụng nợ không hiệu quả.
Liquidity (Tính thanh khoản)	Sử dụng Tỷ số thanh toán hiện hành (Current ration) để đánh giá, cho biết khả năng dùng các tài sản ngắn hạn như tiền mặt, hàng tồn kho hay các khoản phải thu để chi trả cho các khoản nợ ngắn hạn của DN.
Profit on Sales	Lợi nhuận ròng tính trên doanh thu bán hàng, hoạt động KD.
EPS (Earnings per share)	Là chỉ số Lãi cơ bản trên cổ phiếu (BCKQ hoạt động - BCTC), cho biết Lợi nhuận sau thuế phân bổ trên một cổ phiếu thông thường đang được lưu hành trên thị trường.
ROE	Tỷ suất lợi nhuận ròng trên vốn chủ sở hữu: Là chỉ số đo lường mức độ hiệu quả của việc sử dụng vốn chủ sở hữu trong doanh nghiệp
Cash on Sales	là thuật ngữ để mô tả một giao dịch trên sàn chứng khoán trong đó người mua hoặc người bán sử dụng một khoản tiền thay vì mua hoặc bán ký quỹ (on margin). Một giao dịch cash sale yêu cầu nhà đầu tư thanh toán chứng khoán trong vòng hai ngày kể từ việc mua được thực hiện. (giao dịch ký quỹ: là dịch vụ mà bạn vay tiền của công ty chứng khoán để đầu tư, thông qua việc thế chấp tài sản của bạn gồm tiền, chứng khoán, quyền mua cổ phiếu và những tài sản khác được công ty chứng khoán chấp nhận)
TATO	total assets turnover- hiệu suất sử dụng tổng tài sản (hay vòng quay tổng tài sản) là chỉ tiêu được đưa ra để đánh giá quá trình quản lý tài sản của doanh nghiệp có hiệu quả hay không
NPGR	tăng trưởng lợi nhuận ròng hàng năm
Growth	tăng trưởng doanh thu hàng năm

Hình 3.2: Định nghĩa biến

Dữ liệu được tải lên google colab để chạy bằng ngôn ngữ python. Sau đó thực hiện lần lượt các bước nhằm loại bỏ bản ghi trống, chia dữ liệu thành X và Y (trong đó X là dữ liệu các biến đầu vào còn Y là dữ liệu biến manipulation) và sau đó sẽ chuẩn hóa dữ liệu X bằng phương pháp chuẩn hóa min-max để đưa dữ liệu về phạm vi [-1; 1]. Sau cùng dữ liệu sẽ còn 138 dòng và được chia làm 2 phần là Y gồm cột biến manipulation và X gồm 17 cột là các biến được dùng để tính toán.

Mô hình sẽ được chạy trên hai kịch bản (kịch bản "0" là chạy mô hình đầy đủ bao gồm Stacking Ensemble Learning với cross-validation kết hợp Borderline-SMOTE, kịch bản "1" là chỉ chạy thuật toán Stacking Ensemble Learning với cross-validation mà không dùng Borderline-SMOTE). Do dữ liệu khá ít nên tốc độ chạy của mô hình ở vòng lặp chính dùng để huấn luyện chỉ là hơn 2 giây và tốc độ chạy ra kết quả dự đoán là chưa đến 1 giây.

3.2 Kết quả

	Không thao túng	Thao túng
Không thao túng	TP = 24	FP = 1
Thao túng	FN = 1	TN = 2

Hình 3.3: Ma trận nhầm lẫn biểu diễn các dương tính và tiêu cực đúng và giả

Qua kết quả dự đoán được biểu diễn bởi ma trận nhầm lẫn, ta có thể thấy được trong số 25 nhãn không thao túng và 3 nhãn thao túng, mô hình đã dự đoán chính xác được 24 trường hợp không thao túng và 2 trường hợp thao túng. Đồng thời cũng dự đoán sai 1 trường hợp không thao túng và 1 trường hợp thao túng.

	Precision	Recall	F1-score
0	96%	96%	66,66%
1	0%	0%	0%

Hình 3.4: Hiệu suất của Stacking Ensemble Learning

Qua bảng đánh giá hiệu suất của mô hình ta thấy được:

- Trong kịch bản "1" khi sử dụng mô hình Stacking Ensemble Learning với cross-validation vmà không dùng Borderline-SMOTE mô hình cho ra kết quả dự đoán trên tập test toàn bộ là không thao túng, chỉ số đánh giá hiệu suất F1-score = 0%.

- Trong kịch bản "0" khi sử dụng mô hình Stacking Ensemble Learning với cross-validation kết hợp Borderline-SMOTE mô hình cho ra kết quả dự đoán trên tập test với F1-score = 66.66%.(phát hiện đúng 24 nhãn không thao túng trên tổng 25 nhãn không thao túng và 2 nhãn thao túng trên tổng 3 nhãn thao túng)

3.3 Nhận xét

Mô hình cho ra kết quả triển vọng, có thể phát triển và tối ưu trong tương lai. Việc sử dụng thuật toán Stacking Ensemble Learning giúp nâng cao hiệu quả dự đoán so với các thuật toán đơn lẻ. Đặc biệt việc kết hợp với cross-validation và Borderline-SMOTE đã giúp khắc phục phần nào hạn chế về dữ liệu như quá ít và mất cân bằng.

Tuy vậy, dữ liệu vẫn còn đang rất hạn chế và số lượng nhãn vi phạm là rất ít. Điều đó có tác động rất lớn đến kết quả của mô hình dù đã sử dụng nhiều phương pháp để cải thiện. Các biến đang được lấy dựa trên các nghiên cứu từ trước có thể còn chưa xét hết đến các yếu tố tác động lên kết quả và chưa có sự đánh giá độ tin cậy của mỗi biến dẫn đến hiệu quả mô hình chưa cao.

Chương 4

Kết luận và hướng phát triển đề tài

4.1 Kết luận

- Bước đầu xây dựng nền tảng một mô hình phát hiện gian lận trên thị trường chứng khoán và chạy thử thành công bằng dữ liệu thực tế thu thập được.
- Thuật toán Staking Ensemble Learning kết hợp tương đối hiệu quả các thuật toán học máy phân loại khác để đưa ra kết quả cuối cùng.
- Tuy dữ liệu còn hạn chế nhưng nhờ việc sử dụng nhiều phương pháp như chia dữ liệu bằng cross-validation, lấy mẫu quá mức lớp thiểu số bằng Borderline Smote nên mô hình vẫn cho ra kết quả khả quan.
- Các biến được sử dụng dựa trên các nghiên cứu từ trước sẽ cần phải được đánh giá cẩn thận lại để mô hình có thể hoạt động hiệu quả hơn.
- Mặc dù hiệu quả mô hình chưa cao nhưng sẽ tạo ra cơ sở tiền đề để nghiên cứu và hoàn thiện trong tương lai.

4.2 Hướng phát triển của đề tài

- Thu thập thêm dữ liệu và cập nhật những thay đổi trong thao túng chứng khoán để có được một bộ dữ liệu tốt hơn và không bị lỗi thời.

- Tìm hiểu, nghiên cứu nhằm đánh giá lại hoặc cập nhật thêm các biến và mức độ ảnh hưởng của mỗi biến lên kết quả.
- Tối ưu mô hình sao cho giảm độ phức tạp và đạt được hiệu quả cao hơn.
- Thử ứng dụng và chờ đợi kết quả trong thực tế.

Tài liệu tham khảo

Tiếng Anh

[1] Qingbai Liu, Chuanjie Wang, Ping Zhang, Kaixin Zheng. *"Detecting stock market manipulation via machine learning: Evidence from China Securities Regulatory Commission punishment cases"*, International Review of Financial Analysis (2021).

[2] Samira Khodabandehlou , Seyyed Alireza Hashemi Golpayegani. *"Market manipulation detection: A systematic literature review"*, Expert Systems With Applications (2022).

[3] Stéphane Daul, Thibault Jaisson, Alexandra Nagy. *"Performance attribution of machine learning methods for stock returns prediction"*, The Journal of Finance and Data Science (2022).

[4] Ammar Mohammed, Rania Kora. *"A comprehensive review on ensemble deep learning: Opportunities and challenges"*, Journal of King Saud University - Computer and Information Sciences (2023).

PHỤ LỤC

CODE PYTHON MÔ PHỎNG SỐ LIỆU

```
# Vòng lặp chính
import numpy as np
val_preds = np.zeros((len(X_train), 4))
for i, (train_index, val_index) in enumerate(kf.split(X_train)):
    X_train_fold, y_train_fold = X_train[train_index], y_train[train_index]
    X_val_fold, y_val_fold = X_train[val_index], y_train[val_index]

    # Training các model trên fold hiện tại
    svm_clf.fit(X_train_fold, y_train_fold)
    ann_clf.fit(X_train_fold, y_train_fold)
    rf_clf.fit(X_train_fold, y_train_fold)
    lr_clf.fit(X_train_fold, y_train_fold)

    # Lưu trữ output của các model trên fold hiện tại
    val_preds[val_index, :] = np.column_stack((svm_clf.predict(X_val_fold),
                                               ann_clf.predict(X_val_fold),
                                               rf_clf.predict(X_val_fold),
                                               lr_clf.predict(X_val_fold)))
```

```
test_preds_l1= np.column_stack((svm_clf.predict(X_test),
                                ann_clf.predict(X_test),
                                rf_clf.predict(X_test),
                                lr_clf.predict(X_test)))
```

```
ann_clf.fit(X_new, y_train)
```

```
MLPClassifier(learning_rate_init=0.01, max_iter=1000, random_state=42)
```

```
[25] y_pred = ann_clf.predict(test_preds_l1)
```

```
[26] from sklearn.metrics import confusion_matrix, f1_score
     print(f1_score(y_pred,y_test))
     print(confusion_matrix(y_pred, y_test))
```

```
0.6666666666666666
[[24  1]
 [ 1  2]]
```