ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO

ỨNG DỤNG DỰ ĐOÁN XU HƯỚNG THỊ TRƯỜNG TIỀN MÃ HÓA SỬ DỤNG MÔ HÌNH HỌC MÁY.

Giảng viên: TS.Đặng Thanh Hải

Học phần: Chuyên đề công nghệ Nhật Bản

Mã LHP: INT3138 7

Nhóm sinh viên thực hiện:

Trần Mạnh Duy - 22026567

Phạm Khánh Linh - 21020080

Hà Nội, 2024

Mục lục

1 Giới thiệu

1.1 Mục đích của báo cáo

Báo cáo này nhằm trình bày việc sử dụng các mô hình Machine Learning để dự đoán xu hướng thị trường tài chính, tập trung vào việc phân tích các mô hình khác nhau và so sánh hiệu quả của chúng.

Thị trường tài chính là một hệ thống tổ chức và giao dịch các công cụ tài chính bao gồm:

- Forex (Thị trường Ngoại hối): Còn được gọi là thị trường FX, là nơi giao dịch các cặp tiền tệ.
- Thị trường vốn (Capital market): Bao gồm thị trường chứng khoán và trái phiếu.
- Thị trường tài chính phái sinh: Như CFDs (hợp đồng chênh lệch) hoặc hợp đồng tương lai. (ví dụ tiền ảo bitcoin, ETH,..)
- Thị trường hàng hóa: Bao gồm vàng, bạc và dầu.
- Thị trường tiền tệ: Như nợ ngắn hạn.

Trong báo cáo này chúng em sẽ áp dụng các mô hình để dự đoán xu hướng của thị trường tiền mã hóa. Đồng thời sử dụng dự đoán của mô hình để tự động giao dịch trên các sàn giao dịch tiền điện tử.

1.2 Tầm quan trọng của việc dự đoán thị trường tài chính

Dự đoán thị trường có ý nghĩa lớn đối với các nhà đầu tư và tổ chức tài chính, giúp họ đưa ra quyết định đầu tư chính xác, tối ưu hóa lợi nhuận và quản lý rủi ro. Dự đoán thị trường không chỉ là việc đoán xu hướng, mà còn là công cụ hỗ trợ quyết định đầu tư và quản lý rủi ro. Nắm vững thông tin và hiểu biết về thị trường là yếu tố quan trọng để đạt được thành công trong đầu tư.

2 Dữ liệu và Tiền xử lý

2.1 Nguồn dữ liệu

Dữ liệu bao gồm tập dữ liệu của các đồng tiền mã hóa trên thị trường, và cụ thể ở đây, nhóm em chọn nguồn dữ liệu từ sàn giao dịch Binance do có khối lượng giao dịch ổn định, điều này rất quan trọng trong việc tiền xử lý dữ liệu.

Các dữ liệu được lấy từ một đồng tiền mã hóa bao gồm, tất cả dữ liệu đều được lấy theo đồ thị khung 4h :

1. **Open:** Mức giá mở cửa của đồng tiền mã hóa trong một khoảng thời gian nhất định.

2. **High:** Mức giá cao nhất trong khoảng thời gian nhất định.
3. **Low:** Mức giá thấp nhất trong khoảng thời gian nhất định.
4. **Close:** Mức giá đóng cửa của đồng tiền mã hóa trong khoảng thời gian nhất định.
5. **Volume:** Khối lượng giao dịch của đồng tiền mã hóa trong một khoảng thời gian.
6. **Time:** Thời gian về ngày tháng và giờ của khoảng thời gian 4h.

2.2 Tiền xử lý dữ liệu

Dữ liệu được tiền xử lý bao gồm:

1. Chuẩn hóa dữ liệu.
2. Tạo các đặc trưng mới **(feature engineering)**.
3. Chia dữ liệu thành tập huấn luyện và tập kiểm tra.

2.2.1 Chuẩn hóa dữ liệu

Dữ liệu (Open, High, Low, Close, Volume) khi được thu thập về sẽ ở dạng `string` do đó cần chuyển về dạng `float` để có thể dễ dàng xử lý, và áp dụng các thư viện chỉ báo. Dữ liệu về time khi được thu thập về sẽ ở dạng `timestamp` do đó cũng cần định dạng lại về dạng : YY/MM/DD HH/MM/SS để có thể dễ dàng trong việc biểu diễn trên biểu đồ.

2.2.2 Tạo các đặc trưng mới

1. Chỉ báo kỹ thuật: là công cụ phân tích được sử dụng để dự đoán xu hướng giá của tài sản trong tương lai dựa trên dữ liệu giá và khối lượng giao dịch trong quá khứ. Các nhà giao dịch sử dụng chỉ báo kỹ thuật để xác định thời điểm mua hoặc bán tài sản, cũng như để quản lý rủi ro. Có rất nhiều loại chỉ báo kỹ thuật khác nhau, mỗi loại có những ưu và nhược điểm riêng. Một số loại chỉ báo kỹ thuật phổ biến bao gồm:
    - **Chỉ báo xu hướng**: Các chỉ báo này giúp xác định xu hướng chung của giá, chẳng hạn như đường trung bình động, MACD và RSI.
    - **Chỉ báo động lượng**: Các chỉ báo này đo lường tốc độ và mức độ thay đổi giá, chẳng hạn như Bollinger Bands, Stochastic và Momentum.
    - **Chỉ báo khối lượng**: Các chỉ báo này đo lường khối lượng giao dịch, chẳng hạn như Volume Oscillator và On-Balance Volume.
    - **Chỉ báo biến động**: Các chỉ báo này đo lường mức độ biến động giá, chẳng hạn như Average True Range (ATR) và Volatility Index.

Nhờ vào thế mạnh đó nên nhóm chúng em đã quyết định sử dụng chỉ báo kỹ thuật như là một công cụ hỗ trợ để có thể đưa ra quyết định xu hướng của thị trường. Với việc sử dụng thư viện `talib` : đây là một thư viện đã bao gồm hầu như tất cả các chỉ báo cần thiết để có thể sử dụng

2. Tạo các đặc trưng: Các chỉ báo cũng cần các tham số đầu vào phù hợp để có thể có đầu ra tối ưu. Ví dụ như việc sử dụng mô hình nến 4h thì chúng ta ưu tiên các chỉ báo có độ mạnh về khoảng thời gian trung bình, với mô hình nến 15 phút thì chúng ta ưu tiên các chỉ báo có độ mạnh thiên về ngắn hạn.

3. Chuẩn bị dữ liệu: Tệp dữ liệu sử dụng để train model ở đây nhóm em sử dụng thông tin về giá của Bitcoin trong khoảng thời gian từ tháng 6 năm 2020 đến tháng 10 năm 2022, vì

    - Đây là một tập dữ liệu đủ tốt có sự đa dạng về biến đổi giá cũng như về khối lượng. Và tổng khối lượng giao dịch luôn ở mức lớn do đó ít yếu tố chủ quan từ 1 người gây biến đổi giá mạnh (thao túng thị trường).
    - Bitcoin luôn là sự lựa chọn hàng đầu và có tác động mạnh mẽ đến thị trường.

    Tập dữ liệu được nhóm em lấy trên đồ thị 4h, vì đây là khoảng thời gian không quá dài (để model thực hiện được nhiều giao dịch và có đa dạng về mặt dữ liệu), và cũng không quá ngắn (những chỉ báo ngắn hạn thường có độ tin cậy thấp do đó dễ gây sai lệch dữ liệu). Các tùy chỉnh trên các thư viện chỉ báo được em tham khảo từ nhiều nguồn, cũng như là kinh nghiệm bản thân sao cho tối ưu với đồ thị 4h nhất.

    Về việc gán nhãn thì model sẽ dự đoán trong 3 trạng thái :

    - **Uptrend**: Khi dự đoán giá sẽ biến động tăng lên trong thời gian sắp tới.
    - **Unclear**: Khi dự đoán giá sẽ không có biến động và đi ngang trong khoảng thời gian tới.
    - **Downtrend**: Khi dự đoán giá sẽ biến động giảm trong thời gian sắp tới.

      3 trạng thái này được em gán nhãn tùy chỉnh sao cho phù hợp với chart 4h, và phù hợp với dữ liệu train hiện tại.

      Tập dữ liệu được chia theo tỉ lệ 80:20 : (train, test). Ngoài ra vẫn cần 1 khâu backtest lại xem model có thực sự hiệu quả. Lúc này sẽ cần 1 tập dữ liệu mới nằm ngoài khoảng dữ liệu đã được dùng trong khâu 1. Ở đây nhóm em dùng tập dữ liệu từ tháng 11 năm 2022 đến nay.

3 Các Mô hình Machine Learning

3.1 Hồi quy tuyến tính (Linear Regression)

Hồi quy tuyến tính nhằm mục đích tìm ra mối quan hệ tuyến tính giữa biến độc lập (input) và biến phụ thuộc (output). Công thức tổng quát của hồi quy tuyến tính đơn biến là:

$$y = \beta_0 + \beta_1 + \epsilon$$

- y là biến phụ thuộc (output).
- x là biến độc lập (input).
- β_0 là hệ số chặn (intercept).
- β_1 là hệ số gocs (slope), biểu thị độ dốc của đường thẳng hồi quy.
- ϵ là sai số (error term) giữa giá trị thực tế và giá trị dự đoán.

Đối với hồi quy tuyến tính đa biến, công thức mở rộng thành:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$$

Các hệ số hồi quy (β_0, β_1, ..., β_n) được ước lượng bằng phương pháp bình phương bé nhất (Ordinary Least Squares - OLS), phương pháp này tìm cách tối thiểu hóa tổng bình phương sai số (Residual Sum of Squares - RSS):

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_n x_{in}))^2$$

Để tìm các hệ số β, chúng ta cần giải hệ phương trình đạo hàm bậc nhất của hàm RSS đối với từng β_j(với $j = 0, 1, \ldots, n$) và đặt các đạo hàm này bằng 0. Kết quả là một hệ phương trình tuyến tính:

$$X^T X\beta = X^T y$$

Trong đó:

- X là ma trận thiết kế (design matrix), chứa các biến độc lập.
- y là vector chứa các giá trị của biến phụ thuộc.
- β là vector chứa các hệ số hồi quy cần tìm.

Để mô hình hồi quy tuyến tính hoạt động tốt và đưa ra kết quả chính xác, các giả định sau cần được đảm bảo:

- Linearity: Mối quan hệ giữa biến độc lập và biến phụ thuộc là tuyến tính.

- Independence: Các sai số (ϵ) là độc lập với nhau.
- Homoscedasticity: Phương sai của sai số là không đổi đối với mọi giá trị của biến độc lập.
- Normality: Sai số có phân phối chuẩn.

Để đánh giá chất lượng của mô hình hồi quy tuyến tính, ta có thể sử dụng các chỉ số sau:

- R-squared (R^2): Đo lường tỷ lệ phương sai của biến phụ thuộc được giải thích bởi các biến độc lập. Giá trị R^2 nằm trong khoảng từ 0 đến 1.
- Mean Squared Error (MSE): Trung bình bình phương sai số giữa giá trị thực tế và giá trị dự đoán.
- Root Mean Squared Error (RMSE): Căn bậc hai của MSE, cung cấp một đơn vị đo lường cùng đơn vị với biến phụ thuộc.

3.2 Mô hình Softmax (Softmax Regression)

Thuật toán Multinomial Logistic Regression (hoặc còn gọi là Softmax Regression) là một phương pháp học máy được sử dụng cho bài toán phân loại nhiều lớp. Nó là một biến thể của hồi quy logistic nhưng được mở rộng để áp dụng cho bài toán phân loại với ba hoặc nhiều hơn ba lớp.

Tương tự như hồi quy logistic nhị phân, mục tiêu của Multinomial Logistic Regression là dự đoán xác suất của mỗi lớp trong các lớp có thể có, và sau đó chọn lớp có xác suất cao nhất làm dự đoán cuối cùng.

Cách thức hoạt động của Multinomial Logistic Regression tương tự như hồi quy logistic nhị phân, nhưng với một số điểm khác biệt chính:

1. Hàm kích hoạt: Multinomial Logistic Regression sử dụng hàm kích hoạt Softmax thay vì hàm sigmoid. Hàm Softmax chuyển đổi đầu ra của mỗi lớp thành một phân phối xác suất, đảm bảo rằng tổng các xác suất của tất cả các lớp là bằng 1.
2. Hàm mất mát: Multinomial Logistic Regression sử dụng hàm mất mát Cross-Entropy Loss (hoặc còn gọi là Negative Log-Likelihood Loss) để đánh giá sự sai lệch giữa các xác suất dự đoán và các nhãn thực tế. Mục tiêu là tối thiểu hóa hàm mất mát này thông qua quá trình huấn luyện.
3. Thiết lập tham số: Multinomial Logistic Regression có một bộ tham số cho mỗi lớp, và quá trình huấn luyện là để điều chỉnh các tham số này sao cho mô hình có thể dự đoán đúng các xác suất cho mỗi lớp.

Ưu và nhược điểm của Softmax Regression:
Ưu điểm:

- Đơn giản và hiệu quả: Dễ hiểu và dễ triển khai.

- Xử lý tốt bài toán phân loại đa lớp: Hiệu quả trong việc phân loại nhiều lớp khác nhau.
- Tổng quát hóa tốt: Thường có khả năng tổng quát hóa tốt trên các tập dữ liệu mới.

Nhược điểm:

- Giả định tuyến tính: Giả định rằng các lớp có thể được phân tách tuyến tính trong không gian đặc trưng, điều này có thể không đúng trong nhiều trường hợp phức tạp.
- Nhạy cảm với outliers: Như các mô hình hồi quy khác, Softmax Regression có thể bị ảnh hưởng bởi các outliers trong dữ liệu.

3.3 Mô hình cây quyết định (Decision Tree)

Cây quyết định là một mô hình dự đoán sử dụng cấu trúc dạng cây để đưa ra quyết định dựa trên giá trị của các thuộc tính đầu vào. Mỗi nút trong cây đại diện cho một thuộc tính (feature) của dữ liệu, mỗi nhánh đại diện cho một giá trị hoặc một khoảng giá trị của thuộc tính đó, và mỗi lá (leaf) đại diện cho nhãn (label) hoặc giá trị dự đoán.

Cấu trúc của cây quyết định:

- Root Node (Nút gốc): Nút trên cùng của cây, đại diện cho toàn bộ tập dữ liệu và thuộc tính đầu tiên được phân chia.
- Internal Nodes (Nút nội bộ): Nút không phải nút gốc và không phải lá, đại diện cho các thuộc tính được phân chia ở các mức khác nhau.
- Leaf Nodes (Lá): Nút cuối cùng, đại diện cho quyết định hoặc dự đoán cuối cùng.

Quá trình xây dựng cây quyết định bao gồm các bước sau:

1. Chọn thuộc tính phân chia: Tại mỗi nút, chọn thuộc tính (feature) tốt nhất để phân chia dữ liệu. Tiêu chí chọn thuộc tính bao gồm:
    - Information Gain: Sử dụng trong thuật toán ID3, C4.5. Đo lường mức độ giảm thiểu entropy khi phân chia dữ liệu.
    - Gini Index: Sử dụng trong thuật toán CART (Classification and Regression Trees). Đo lường độ tinh khiết của tập dữ liệu sau khi phân chia.
2. Phân chia dữ liệu: Dữ liệu được phân chia thành các nhánh dựa trên giá trị của thuộc tính được chọn.
3. Đệ quy phân chia: Lặp lại quá trình chọn thuộc tính và phân chia dữ liệu cho mỗi nhánh con, cho đến khi đạt. Cây quyết định là một mô hình dự đoán sử dụng cấu trúc dạng cây để đưa ra quyết định dựa trên giá trị của các thuộc tính đầu vào. Mỗi nút trong cây đại diện cho một thuộc tính (feature) của dữ liệu, mỗi nhánh đại diện cho một giá trị hoặc một khoảng giá trị của thuộc tính đó, và mỗi lá (leaf) đại diện cho nhãn (label) hoặc giá trị dự đoán.

Điều kiện dừng và cắt tỉa cây (Pruning):

- Điều kiện dừng: Quá trình xây dựng cây có thể dừng khi:
  - Tất cả các mẫu trong nút đều thuộc về một lớp.
  - Không còn thuộc tính nào để phân chia.
  - Đạt đến độ sâu tối đa của cây.
- Cắt tỉa cây: Để tránh overfitting, có thể thực hiện cắt tỉa cây bằng hai phương pháp:
  - Pre-pruning: Dừng quá trình xây dựng cây sớm dựa trên một số điều kiện như độ sâu tối đa, số lượng mẫu tối thiểu trong một nút.
  - Post-pruning: Xóa bỏ các nhánh không cần thiết sau khi cây đã được xây dựng hoàn chỉnh.

Ưu và nhược điểm của cây quyết định:

- Ưu điểm:
  - Dễ hiểu và trực quan.
  - Có khả năng xử lý cả dữ liệu số và dữ liệu phân loại.
  - Không yêu cầu chuẩn hóa dữ liệu.
- Nhược điểm:
  - Dễ bị overfitting nếu không được cắt tỉa đúng cách.
  - Nhạy cảm với thay đổi nhỏ trong dữ liệu.
  - Hiệu quả không cao đối với các bài toán có nhiều thuộc tính phức tạp.

3.4 Mô hình Naive Bayes

Thuật toán Naive Bayes là một phương pháp học máy được sử dụng rộng rãi cho bài toán phân loại và dự đoán. Thuật toán này dựa trên nguyên lý của định lý Bayes, với giả định "ngây thơ" (naive) rằng các biến độc lập (features) là độc lập có điều kiện, nghĩa là mỗi biến độc lập đóng góp độc lập vào dự đoán kết quả.

Cơ sở của thuật toán Naive Bayes là công thức Bayes, một công thức toán học để tính xác suất có điều kiện của một sự kiện dựa trên xác suất của các sự kiện liên quan. Cụ thể, công thức Bayes được sử dụng để tính xác suất của một lớp (class) dựa trên các biến độc lập.

Bộ phân lớp Naive bayes hay bộ phân lớp Bayes (simple byes classifier) hoạt động như sau:

1. Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn bằng một vector chứa n giá trị thuộc tính A_1, A_2,...,A_n = $x_1, x_2, ..., x_n$.
   Giả sử có m lớp $C_1, C_2, .., C_m$. Cho một phần tử dữ liệu X, bộ phân lớp sẽ gán nhãn cho X

là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán X thuộc vào lớp C_i nếu và chỉ nếu: $P(C_i|X) > P(C_j|X)(1 <= i, j <= m, i \neq j)$. Giá trị này sẽ tính dựa trên định lý Bayes.

2. Để tìm xác suất lớn nhất, ta nhận thấy các giá trị $P(X)$ là giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của $P(X|C_i) * P(C_i)$. Chú ý rằng $P(C_i)$ được ước lượng bằng $|D_i|/|D|$, trong đó D_i là tập các phần tử dữ liệu thuộc lớp C_i. Nếu xác suất tiền nghiệm $P(C_i)$ cũng không xác định được thì ta coi chúng bằng nhau $P(C_1) = P(C_2) = ... = P(C_m)$, khi đó ta chỉ cần tìm giá trị $P(X|C_i)$ lớn nhất.

3. Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toàn $P(X|Ci)$ là rất lớn, dó đó có thể giảm độ phức tạp của thuật toán Naive Bayes giả thiết các thuộc tính độc lập nhau.Khi đó $P(X|C_i) = P(x_1|C_i)...P(x_n|C_i)$.

Ưu và nhược điểm của Naive Bayes:

- Ưu điểm:
  - Đơn giản và nhanh chóng: Dễ hiểu và dễ triển khai, huấn luyện nhanh.
  - Hiệu quả với dữ liệu lớn: Hoạt động tốt với các tập dữ liệu lớn và nhiều biến.
  - Hiệu quả với phân loại văn bản: Đặc biệt hiệu quả với các bài toán phân loại văn bản và lọc thư rác.
- Nhược điểm:
  - Giả định độc lập: Giả định các đặc trưng độc lập, không thực tế trong nhiều trường hợp.
  - Nhạy cảm với dữ liệu ít: Với các lớp có ít dữ liệu, Naive Bayes có thể không hoạt động tốt.
  - Không mô hình hóa tương tác: Không mô hình hóa được sự tương tác phức tạp giữa các đặc trưng.

3.5 Mô hình rừng ngẫu nhiên (Random Forest)

Random Forest là một tập hợp (ensemble) các cây quyết định, được tạo ra bằng cách sử dụng kỹ thuật bootstrap aggregating, hay còn gọi là bagging. Mỗi cây quyết định trong rừng được huấn luyện trên một mẫu ngẫu nhiên của tập dữ liệu huấn luyện, và khi dự đoán, các cây quyết định sẽ bỏ phiếu để đưa ra kết quả cuối cùng.

Bagging (Bootstrap Aggregating) là một phương pháp giảm phương sai của mô hình bằng cách tạo ra nhiều mẫu con từ tập dữ liệu gốc bằng cách lấy mẫu có hoàn lại (bootstrap sampling). Các mẫu này sau đó được sử dụng để huấn luyện các mô hình con (các cây quyết định trong trường hợp của Random Forest).Mỗi cây quyết định trong Random Forest được xây dựng từ một mẫu bootstrap của dữ liệu huấn luyện. Cây quyết định là một mô hình phân loại hoặc hồi quy dạng cây, trong đó mỗi nút lá (leaf node) đại diện cho một quyết định hoặc một dự đoán.

Tạo cây quyết định trong Random Forest:

- Chọn mẫu bootstrap: Lấy mẫu có hoàn lại từ tập dữ liệu huấn luyện để tạo ra một tập dữ liệu con.
- Xây dựng cây quyết định: Sử dụng tập dữ liệu con để xây dựng một cây quyết định. Mỗi nút trong cây quyết định được chia dựa trên một tập con ngẫu nhiên của các đặc trưng (features).

Quá trình xây dựng Random Forest:

1. Khởi tạo: Xác định số lượng cây quyết định ($n_{estimators}$) cần được xây dựng.
2. Tạo cây quyết định: Lặp lại cho đến khi đạt đủ số lượng cây:
    - Lấy mẫu bootstrap từ tập dữ liệu gốc.
    - Tạo một cây quyết định từ mẫu bootstrap.
3. Dự đoán:
    - Đối với phân loại: Mỗi cây đưa ra một dự đoán, và kết quả cuối cùng là lớp có nhiều phiếu nhất (majority vote).
    - Đối với hồi quy: Kết quả cuối cùng là trung bình của các dự đoán từ tất cả các cây.

Một trong những ưu điểm lớn nhất của Random Forest là khả năng giảm overfitting. Mặc dù các cây quyết định đơn lẻ có thể bị overfit, việc kết hợp nhiều cây với nhau giúp trung bình hóa các sai số và giảm phương sai, dẫn đến mô hình tổng thể ổn định và chính xác hơn.

Ưu và nhược điểm của Random Forest:

- Ưu điểm:
    - Hiệu suất cao: Random Forest thường có độ chính xác cao và hoạt động tốt trên nhiều loại dữ liệu.
    - Chống overfitting: Việc kết hợp nhiều cây quyết định giúp giảm overfitting so với việc sử dụng một cây duy nhất.
    - Làm việc tốt với dữ liệu lớn: Random Forest có thể xử lý các tập dữ liệu lớn với nhiều đặc trưng.
    - Đa năng: Random Forest có thể được sử dụng cho cả phân loại và hồi quy.
- Nhược điểm:
    - Tốn tài nguyên: Random Forest có thể tốn nhiều tài nguyên tính toán và bộ nhớ, đặc biệt với số lượng cây lớn.
    - Khó giải thích: Mặc dù mỗi cây quyết định có thể dễ dàng giải thích, việc kết hợp nhiều cây lại khiến mô hình tổng thể khó hiểu và giải thích hơn.

3.6 Mô hình XGBoost

XGBoost là một thuật toán học máy dựa trên phương pháp boosting, một kỹ thuật kết hợp nhiều mô hình yếu để tạo thành một mô hình mạnh. Thuật toán này được phát triển bởi Tianqi Chen và nổi bật nhờ hiệu suất cao và khả năng xử lý dữ liệu lớn, thiếu dữ liệu và tính tương tác phức tạp giữa các biến.

XGBoost xây dựng các cây quyết định theo cách thức tuần tự, mỗi cây mới cải thiện dự đoán của cây hiện tại bằng cách tập trung vào các lỗi của cây trước đó. Các thành phần chính của XGBoost bao gồm:

- Mục tiêu (Objective): Hàm mất mát cần tối thiểu hóa, bao gồm cả phần lỗi dự đoán và độ phức tạp của mô hình.
- Hàm mất mát (Loss Function): Đo lường sự khác biệt giữa dự đoán và giá trị thực tế, chẳng hạn như Mean Squared Error (MSE) cho bài toán hồi quy, hoặc Log Loss cho bài toán phân loại.
- Regularization: Phần điều chỉnh độ phức tạp của mô hình để tránh overfitting.

XGBoost xây dựng cây quyết định bằng cách tối thiểu hóa hàm mục tiêu thông qua việc thêm cây mới vào mô hình hiện tại. Quá trình này bao gồm:

- Khởi tạo mô hình: Bắt đầu với một mô hình đơn giản, thường là dự đoán trung bình.
- Xây dựng cây mới: Tại mỗi bước, xây dựng một cây mới để giảm thiểu sai số của mô hình hiện tại. Các bước bao gồm:
    1. Tính toán gradient và hessian: Gradient và hessian của hàm mất mát đối với dự đoán hiện tại được sử dụng để đánh giá mức độ cải thiện của việc chia nút.
    2. Chọn phân chia tối ưu: Chọn thuộc tính và điểm chia để tối ưu hóa hàm mục tiêu.
    3. Cập nhật mô hình: Thêm cây mới vào mô hình hiện tại với trọng số điều chỉnh.

Ưu và nhược điểm của XGBoost:

- Ưu điểm:
    - Hiệu suất cao: Tận dụng tối đa tài nguyên phần cứng, bao gồm CPU và GPU, để tăng tốc quá trình huấn luyện.
    - Khả năng xử lý dữ liệu lớn: Tối ưu hóa việc xử lý dữ liệu lớn và không đồng nhất.
    - Khả năng chống overfitting: Sử dụng các kỹ thuật regularization để kiểm soát độ phức tạp của mô hình.
    - Khả năng tự động xử lý giá trị thiếu: XGBoost tự động xác định và xử lý giá trị thiếu trong dữ liệu.
- Nhược điểm:

- Yêu cầu cấu hình cao: Cần điều chỉnh nhiều tham số để đạt hiệu suất tối ưu.
- Tốn tài nguyên tính toán: Có thể cần nhiều tài nguyên tính toán hơn so với các thuật toán khác, đặc biệt với dữ liệu lớn và phức tạp.
- Khó khăn trong việc giải thích: Mô hình ensemble phức tạp hơn và khó giải thích hơn so với các mô hình đơn giản như cây quyết định đơn.

4 Triển khai và Đánh giá Mô hình

```
Mean Squared Error: 0.7839931153184165
Accuracy: 0.24440619621342513
```

Hình 1: Kết quả huấn luyện mô hình Linear Regression.

```
Accuracy: 56.8847%
              precision    recall  f1-score   support

           0       0.86      0.60      0.71       930
           1       0.23      0.46      0.30       126
           2       0.16      0.39      0.23       106

    accuracy                           0.57      1162
   macro avg       0.42      0.48      0.41      1162
weighted avg       0.73      0.57      0.62      1162
```

Hình 2: Kết quả huấn luyện mô hình XGBoost.

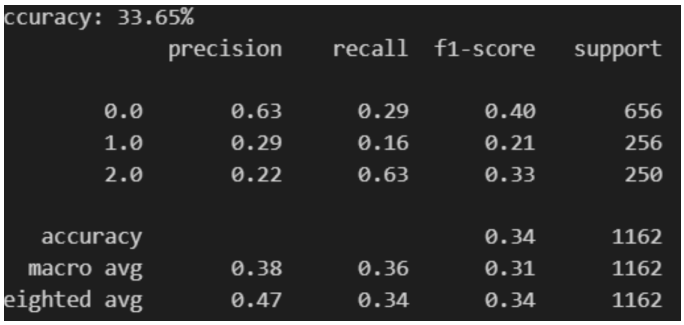
```
Accuracy: 56.37%
              precision    recall  f1-score   support

         0.0       0.57      1.00      0.72       656
         1.0       0.33      0.01      0.02       256
         2.0       0.00      0.00      0.00       250

    accuracy                           0.56      1162
   macro avg       0.30      0.33      0.25      1162
weighted avg       0.39      0.56      0.41      1162
```

Hình 3: Kết quả huấn luyện mô hình Softmax.

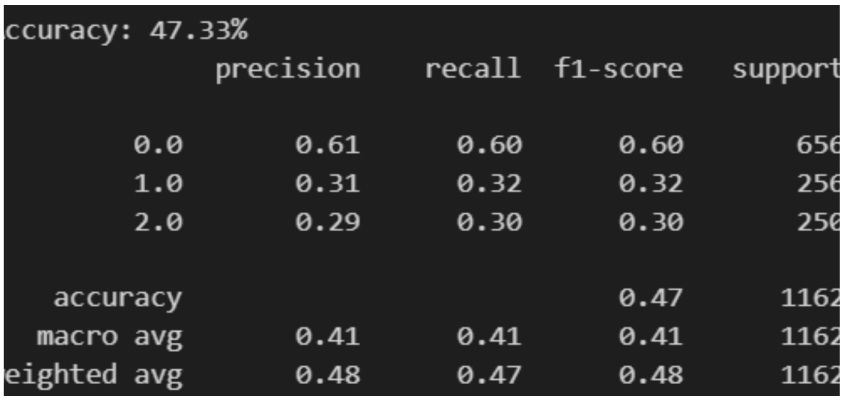
Dựa vào kết quả trên, ta có thể rút ra một số đánh giá chung như sau:



ccuracy: 33.65%

	precision	recall	f1-score	support
0.0	0.63	0.29	0.40	656
1.0	0.29	0.16	0.21	256
2.0	0.22	0.63	0.33	250
accuracy			0.34	1162
macro avg	0.38	0.36	0.31	1162
eighted avg	0.47	0.34	0.34	1162

Hình 4: Kết quả huấn luyện mô hình Navie Bayes.



ccuracy: 47.33%

	precision	recall	f1-score	support
0.0	0.61	0.60	0.60	656
1.0	0.31	0.32	0.32	256
2.0	0.29	0.30	0.30	250
accuracy			0.47	1162
macro avg	0.41	0.41	0.41	1162
eighted avg	0.48	0.47	0.48	1162

Hình 5: Kết quả huấn luyện mô hình Decision Tree.

1. Mô hình hiệu quả nhất: Random Forest cho thấy độ chính xác cao nhất (57.66%) và có hiệu suất tổng thể tốt nhất, đặc biệt ở lớp 0. Tuy nhiên, mô hình này vẫn gặp khó khăn với các lớp 1 và 2.

2. Mô hình không phù hợp: Linear Regression rõ ràng không phù hợp cho bài toán phân loại này do bản chất của nó là mô hình hồi quy, không phải phân loại.

3. XGBoost và Softmax Regression: Cả hai mô hình này có độ chính xác tương đương (khoảng 56.4-56.9%), với XGBoost có weighted avg F1-score cao hơn. Điều này cho thấy XGBoost có thể xử lý dữ liệu phức tạp hơn tốt hơn Softmax Regression trong trường hợp này.

4. Naive Bayes và Decision Tree: Naive Bayes có độ chính xác thấp nhất (33.65%) và Decision Tree có độ chính xác trung bình (47.33%). Cả hai mô hình này đều cho thấy sự khó khăn trong việc phân loại chính xác các lớp khác nhau.

5. Sự mất cân bằng lớp: Các kết quả chỉ ra rằng lớp 0 có số lượng mẫu nhiều hơn và hiệu suất cao hơn, trong khi các lớp 1 và 2 có ít mẫu hơn và hiệu suất thấp hơn. Điều này có thể chỉ ra vấn đề mất cân bằng lớp trong dữ liệu.

Accuracy: 57.66%

	precision	recall	f1-score	support
0.0	0.59	0.93	0.72	656
1.0	0.51	0.12	0.20	256
2.0	0.42	0.10	0.17	250
accuracy			0.58	1162
macro avg	0.51	0.39	0.36	1162
weighted avg	0.54	0.58	0.49	1162

Hình 6: Kết quả huấn luyện mô hình Random Forest.

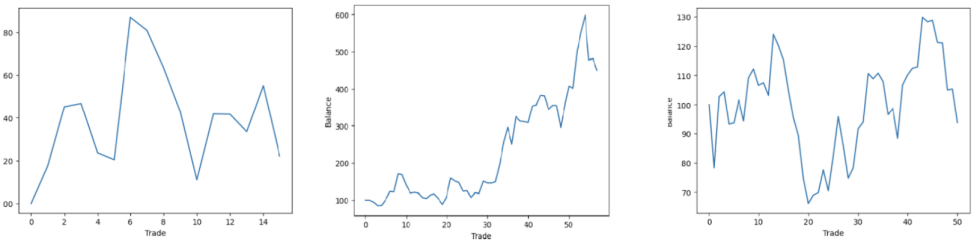
Ta nhận thấy 3 mô hình : Random Forest , XGBoost , Softmax Regression cho 3 kết quả có độ chính xác cao nhất xấp xỉ với nhau . Vì vậy chúng em sẽ chọn 3 mô hình này để đưa vào chạy trong thực tế và so sánh kết quả.

5 Kết luận

5.1 Tổng kết

Dựa trên nhiều lần chạy thử nghiệm với các loại tiền mã hóa với đặc trưng theo cảm quan bản thân em phân loại em chia ra làm các loại sau:

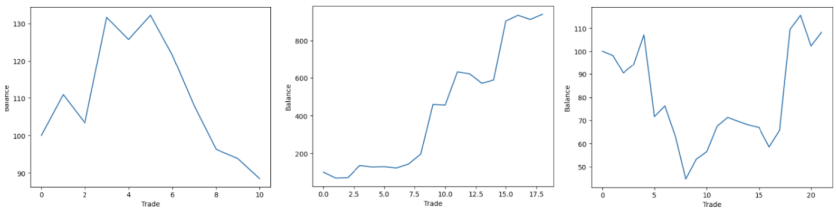
- Token với đường giá ổn định ít biến động nhưng khối lượng cao : XRP
- Token với đường giá chạy đột biến : SOL
- Token với khối lượng giao dịch thấp : C98



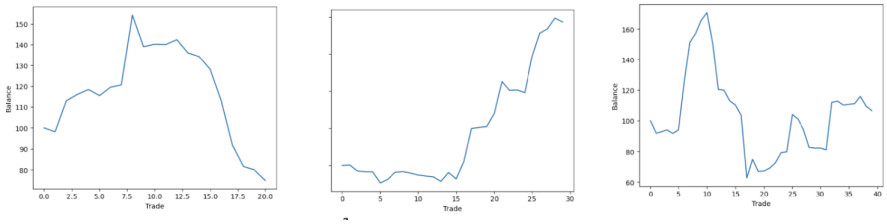
Hình 7: Kết quả giao dịch với mô hình Naive Bayes.

Về tổng quan thì ta thấy mô hình Random Forest luôn đạt mức tốt ở 3 trường hợp trên và các xu hướng nó phát hiện rõ ràng do đó nó có số lần giao dịch thấp nhất nhưng lại cho hiệu quả cao nhất.

Xgboost thì có hiệu suất khá ổn thiên về trạng thái cân bằng so với 2 mô hình khác. Với mô hình Naive Bayes, đây là mô hình khá nhạy cảm nên nó có số lần giao dịch cao gần gấp đôi so với Random forest. Tuy vậy nhờ đó mà nó giữ tài khoản ở mức không bị biến động quá nhiều kể cả khi thị trường đi xuống.



Hình 8: Kết quả giao dịch với mô hình Random Forest.



Hình 9: Kết quả giao dịch với mô hình XGBoost.

5.2 Hạn chế và hướng phát triển

- Hạn chế: Mô hình sẽ chỉ hoạt động cho là hiệu quả khi thị trường không bị làm giá quá đột biến.
- Hướng phát triển: Nhóm em sẽ cần nghiên cứu thêm một mô hình để phát hiện khối lượng giao dịch thay đổi đột ngột, từ đó có thể linh hoạt chuyển giao giữa 2 các loại mô hình sao cho phù hợp với xu thế thị trường.

6 Tài liệu tham khảo

1. Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. Mathematics for Machine Learning. Cambridge University Press, April 2020.
2. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc., 3rd edition, October 2022.
3. Sách Machine Learning Yearning (Khát khao Học Máy) (Andrew Ng).
4. Machine Learning mastery (Các thuật toán Machine Learning cơ bản).