ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC VIỆT NHẬT

BÁO CÁO CUỐI KỲ

MÔN HỌC: HỌC MÁY

CHỦ ĐỀ: DỰ ĐOÁN PHÊ DUYỆT CHO VAY BẰNG MÔ HÌNH ENSEMBLE

Giảng viên hướng dẫn :	**TS.Lê Kim Quy**
Sinh viên thực hiện:	**Trần Tiến Toàn**
Mã sinh viên:	**21110107**
Chuyên ngành:	**Khoa học và kỹ thuật máy tính**
Khoá:	**2021**

Hà Nội, 2024

MỤC LỤC

TÓM TẮT

Trong ngành ngân hàng, việc đưa ra các quyết định phê duyệt cho vay một cách chính xác là rất quan trọng để tối ưu hóa hiệu quả tài chính và giảm thiểu rủi ro. Sự tiến bộ trong công nghệ học máy đã mở rộng khả năng áp dụng các mô hình tiên tiến vào quy trình này, nhằm cải thiện độ chính xác trong phân tích và xử lý dữ liệu phức tạp. Trong số đó, mô hình học tập tập hợp (Ensemble) - kết hợp các thuật toán như Random Forest, Boosting và Bagging - đã được chứng minh là vượt trội so với các phương pháp truyền thống, nhờ vào khả năng xử lý hiệu quả các tương tác phức tạp giữa các đặc trưng và giảm thiểu nguy cơ quá khớp. Báo cáo này tập trung vào việc khám phá và đánh giá hiệu quả của các mô hình Ensemble trong việc dự đoán phê duyệt cho vay. Bằng cách phân tích tầm quan trọng của các đặc trưng như điểm tín dụng và thu nhập hàng năm, nghiên cứu này nhằm mục đích xác định khả năng phê duyệt cho vay dựa trên dữ liệu lịch sử, qua đó cải thiện khả năng quản lý rủi ro và hiệu quả quyết định của các tổ chức tài chính.

I.GIỚI THIỆU

Trong lĩnh vực tài chính, việc đưa ra quyết định phê duyệt cho vay là một hoạt động cốt lõi, yêu cầu sự chính xác và hiệu quả cao để giảm thiểu rủi ro và tối ưu hóa lợi nhuận. Gần đây, việc áp dụng các kỹ thuật học máy vào dự đoán quyết định này đã thu hút sự quan tâm rộng rãi, mang lại hy vọng cải thiện đáng kể trong việc xử lý và phân tích dữ liệu phức tạp liên quan đến khoản vay. Các nghiên cứu về các thuật toán và phương pháp mới đang liên tục được khám phá để nâng cao độ chính xác và độ tin cậy của các mô hình dự đoán.

Phương pháp hồi quy logistic từ lâu đã được sử dụng như một công cụ chuẩn trong việc phân tích quyết định tài chính do tính đơn giản và dễ hiểu. Tuy nhiên, như Brown và Smith (2018) đã chỉ ra, kỹ thuật này thường không hiệu quả trong việc giải quyết các mối quan hệ phi tuyến tính phức tạp giữa các đặc trưng trong dữ liệu [5]. Đáp lại, các phương pháp dựa trên cây quyết định đã được phát triển để cung cấp cách tiếp cận linh hoạt hơn, có khả năng nắm bắt các tương tác phức tạp giữa các đặc trưng, nhưng chúng cũng dễ dàng dẫn đến tình trạng quá khớp khi đối mặt với dữ liệu lớn và nhiễu.

Với sự giới thiệu của các phương pháp học tổng hợp như RandomForest, một bước tiến mới đã được thực hiện trong lĩnh vực này. Breiman (2001) mô tả rằng RandomForest kết hợp nhiều cây quyết định để tạo ra một mô hình mạnh mẽ hơn bằng cách trung bình hóa các dự đoán, từ đó giảm thiểu nguy cơ quá khớp và cải thiện độ chính xác tổng thể của các dự đoán [4]. Nghiên cứu của Zhao và cộng sự (2020) cũng đã chỉ ra rằng các mô hình như RandomForest không chỉ cung cấp độ chính xác cao hơn so với các phương pháp truyền thống mà còn có khả năng xử lý tốt các tương tác đặc trưng phức tạp [12].

Điểm mấu chốt khác trong việc phát triển mô hình dự đoán là việc chọn lọc đặc trưng, điều mà Goyal và Kumar (2021) nhấn mạnh tầm quan trọng của việc xác định các đặc trưng quan trọng nhất trong việc dự đoán phê duyệt cho vay [7]. Sự hiểu biết này giúp tối ưu hóa các mô hình và tập trung vào các yếu tố có ảnh hưởng lớn nhất đến quyết định cuối cùng.

Trong báo cáo này, chúng tôi sẽ khám phá sâu hơn vào các mô hình học máy tiên tiến, đặc biệt là RandomForest và các kỹ thuật học Ensemble khác, để đánh giá khả năng và hiệu quả của chúng trong bối cảnh dự đoán phê duyệt cho vay. Bên cạnh đó, chúng tôi cũng sẽ so sánh chúng với các phương pháp truyền thống để xác định các ưu và nhược điểm, cũng như tiềm năng ứng dụng trong thực tiễn.

II.TỔNG QUAN LÝ THUYẾT

1. Decision Tree

Cây Quyết định (Decision Tree) cũng là thuật toán Học Máy có thể thực hiện cả hai tác vụ phân loại và hồi quy, thậm chí là cả các tác vụ đa đầu ra. Thuật toán này rất mạnh mẽ và có khả năng khớp những tập dữ liệu phức tạp. Cây Quyết định cũng là thành phần nền tảng của Rừng Ngẫu nhiên , một trong những thuật toán học máy mạnh mẽ nhất hiện nay.

Thuật toán Cây Phân loại và Hồi quy (Classification and Regression Tree – CART) được sử dụng để huấn luyện Cây Quyết định. Đầu tiên, thuật toán sẽ chia tập huấn luyện thành hai tập con theo đặc trưng k và ngưỡng tk. Thuật toán sẽ tìm kiếm cặp (k, tk) có thể tạo ra các tập con thuần khiết nhất (được đánh trọng số bởi kích thước tập). Với độ pha tạp Gini:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

Phương trình 1. Độ pha tạp Gini

Trong đó:

- $p_{i,k}$ là tỉ số giữa số mẫu thuộc lớp k và số mẫu huấn luyện trong nút thứ i.

Và phương trình hàm chi phí cần được cực tiểu hoá:

$$J(k, t_k) = \frac{m_{\text{trái}}}{m} G_{\text{trái}} + \frac{m_{\text{phải}}}{m} G_{\text{phải}}$$

$$\text{trong đó} \begin{cases} G_{\text{trái/phải}} \text{ đo độ pha tạp của tập con trái/phải,} \\ m_{\text{trái/phải}} \text{ là số lượng mẫu trong các tập con trái/phải.} \end{cases}$$

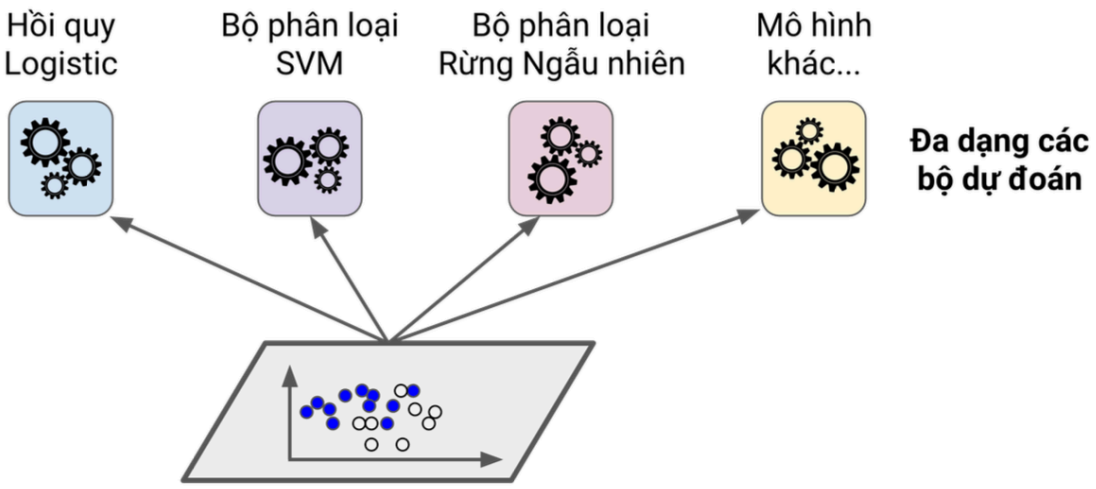
Phương trình 2. Hàm chi phí của CART cho bài toán phân loại

Một khi chia thành công tập huấn luyện thành hai tập con, thuật toán CART tiếp tục chia nhỏ các tập con với cùng logic, rồi đến các tập con nhỏ hơn, và cứ đệ quy như vậy. Thuật toán dừng lại khi đạt được chiều sâu tối đa (định nghĩa bởi siêu tham số max_depth), hoặc không tìm được cách chia để giảm độ pha tạp. Một vài siêu tham số khác để kiểm soát các điều kiện dừng phụ là

(min_samples_split, min_samples_leaf, max_leaf_nodes).

2. Học Ensemble

Một nhóm các bộ phân loại được gọi là ensemble. Học Ensemble là phương pháp kết hợp nhiều bộ phân loại yếu để tổng hợp các dự đoán của mỗi bộ phân loại, sau đó đưa ra kết quả. Những phân loại này có thể sử dụng các thuật toán khác nhau như một phân loại sử dụng thuật toán svm, một phân loại khác có thể sử dụng thuật toán quyết định, phân loại khác có thể sử dụng thuật toán k nearest neighbour, v.v.

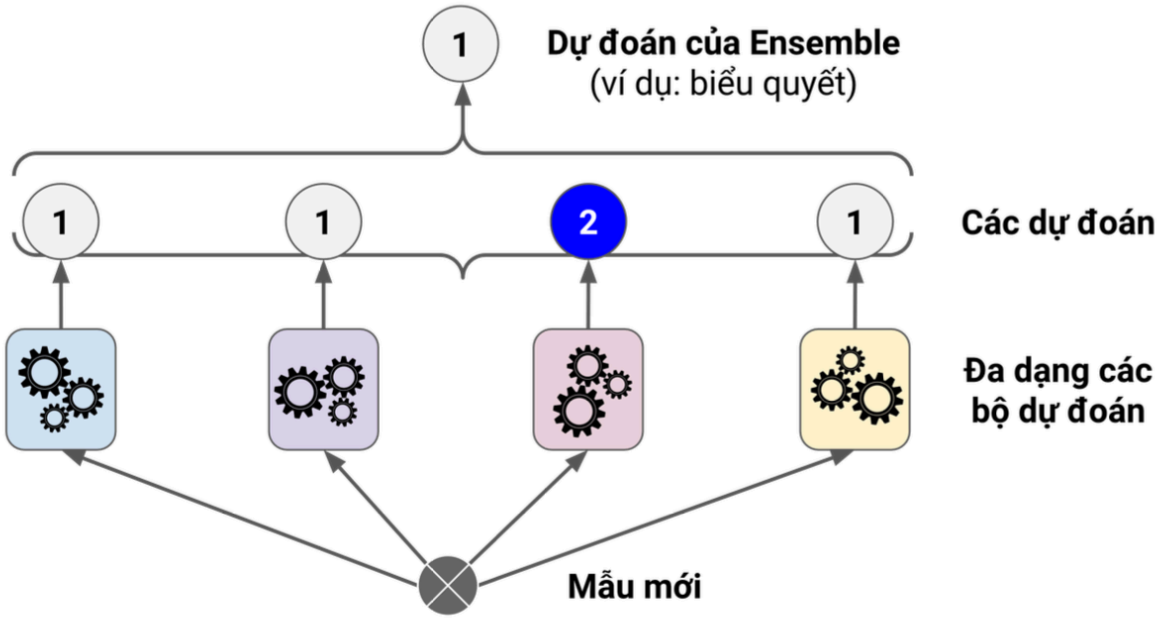


Nguồn: [2]

Hình 2 . Huấn luyện các bộ phân loại khác nhau

Một cách đơn giản để có được một bộ phân loại vượt trội hơn là tổng hợp các dự đoán của mỗi bộ phân loại riêng lẻ, sau đó đưa ra kết quả dựa trên lớp được biểu quyết

nhiều nhất. Bộ phân loại biểu quyết theo đa số này được gọi là bộ phân loại biểu quyết cứng (hard voting classifier)



Hình 3 . Dự đoán của bộ phân loại biểu quyết cứng

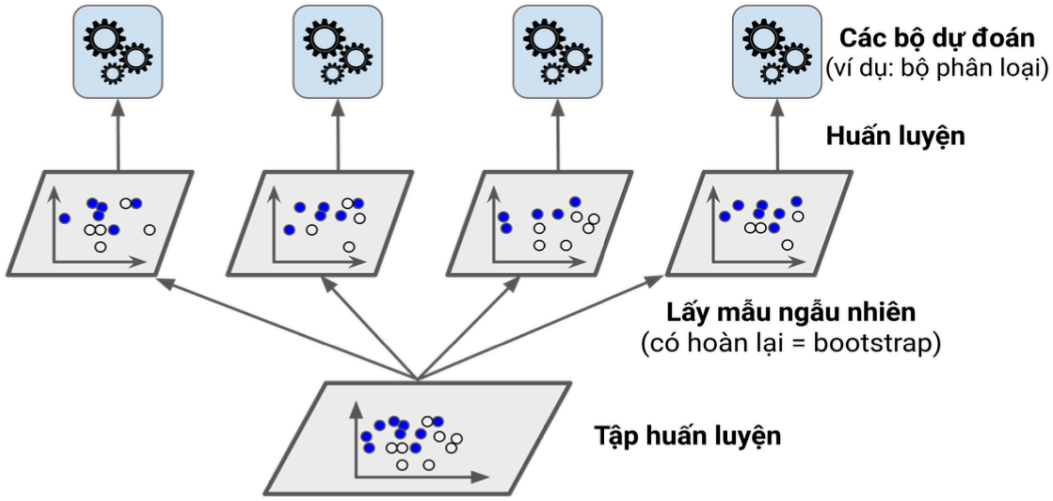
Nếu tất cả các bộ phân loại đều có thể ước tính xác suất của từng lớp , ta có thể lấy trung bình xác suất từng lớp trên tất cả các bộ phân loại riêng lẻ, rồi đưa ra dự đoán là lớp có xác suất trung bình cao nhất. Phương pháp này được gọi là biểu quyết mềm (soft voting).

Các phương pháp Ensemble phổ biến nhất, bao gồm: bagging, boosting, và stacking. Phương pháp Ensemble hoạt động tốt nhất khi các bộ dự đoán càng độc lập với nhau càng tốt.

Một ensemble gồm các Cây Quyết định được gọi là Rừng Ngẫu nhiên (Random Forest)

3. Bagging

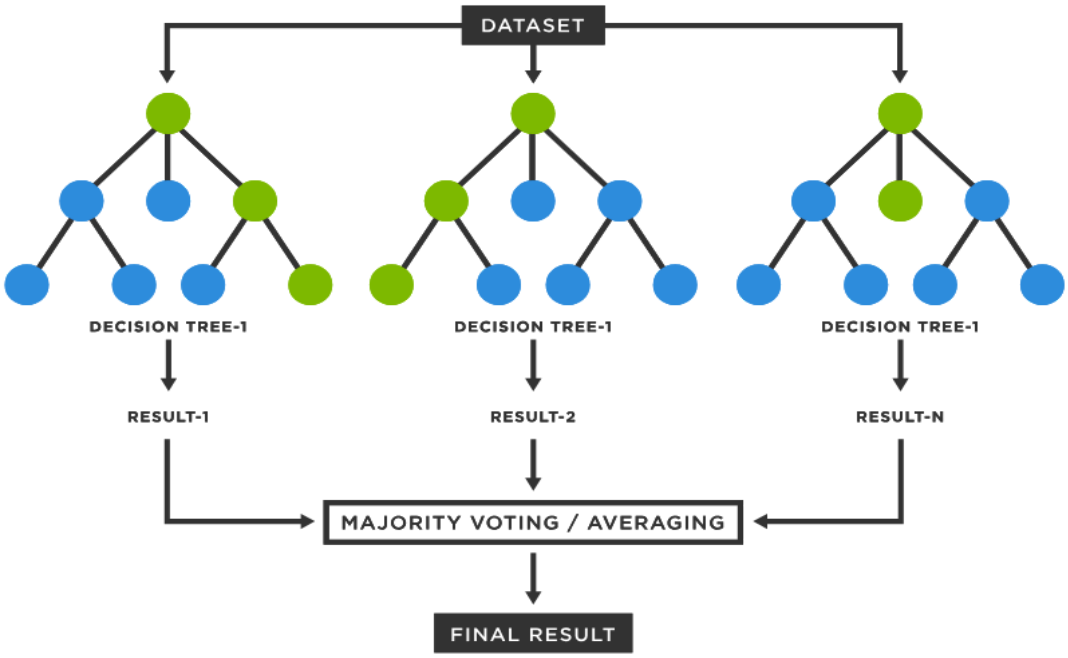
Là phương pháp cho phép lấy mẫu có hoàn lại, phương pháp này được gọi là bagging (viết tắt của bootstrap aggregating). Nó cho phép sử dụng cùng một thuật toán huấn luyện cho mỗi bộ dự đoán nhưng huấn luyện chúng trên các tập con ngẫu nhiên của tập huấn luyện



Hình 4 . Phương pháp bagging

4. Random Forest

Thực chất Random Forest chính là Ensemble của các Cây Quyết định được huấn luyện thông qua phương pháp bagging và đưa ra kết quả dựa trên biểu quyết cứng.



Nguồn ảnh: https://medium.com/@denizgunay/random-forest-af5bde5d7e1e

Hình 5. Rừng ngẫu nhiên

Rừng Ngẫu nhiên còn đưa thêm sự ngẫu nhiên vào quá trình xây dựng cây; thay vì tìm đặc trưng tốt nhất khi tách một nút, thuật toán này tìm đặc trưng tốt nhất trong một tập đặc trưng con ngẫu nhiên. Kết quả ta sẽ có các cây đa dạng hơn.

Rừng Ngẫu nhiên rất tiện lợi trong việc nắm bắt nhanh những đặc trưng thực sự quan trọng, đặc biệt khi ta cần trích chọn đặc trưng.

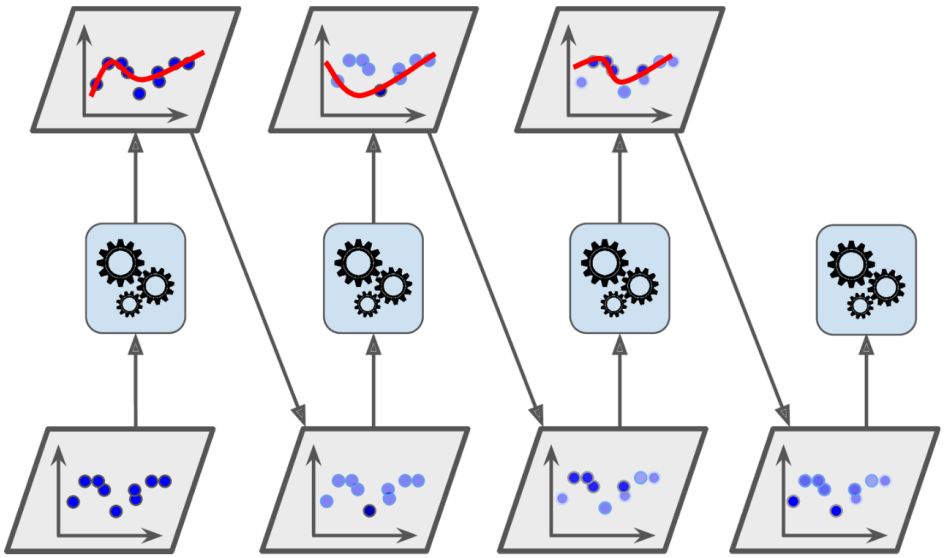
5. Boosting

Boosting (ban đầu được gọi là boosting giả thuyết – hypothesis boosting) là bất cứ phương pháp Ensemble nào có thể kết hợp một vài bộ học yếu thành một bộ học mạnh. Ý tưởng chung của hầu hết các phương pháp boosting là huấn luyện tuần tự các bộ dự đoán, bộ dự đoán sau cố gắng sửa lỗi cho bộ dự đoán trước [2]. Có nhiều phương pháp boosting nhưng hiện nay phổ biến nhất là AdaBoost13 (viết tắt của Adaptive Boosting – tạm dịch: Boosting Thích ứng) và Gradient Boosting.

AdaBoost

Một cách để bộ dự đoán sau có thể sửa lỗi cho bộ dự đoán trước là tập trung hơn vào các mẫu huấn luyện mà bộ dự đoán trước bị dưới khớp. Điều này giúp bộ dự đoán sau chú ý vào những trường hợp khó. Kỹ thuật này có tên gọi là AdaBoost.

Ví dụ, khi huấn luyện một bộ phân loại AdaBoost, đầu tiên thuật toán huấn luyện một bộ phân loại cơ sở (như Cây Quyết định) và sử dụng bộ phân loại này để dự đoán trên tập huấn luyện. Sau đó thuật toán sẽ tăng trọng số của các mẫu huấn luyện bị dự đoán sai. Tiếp theo thuật toán huấn luyện bộ phân loại thứ hai, sử dụng trọng số được cập nhật, sau đó lặp lại việc dự đoán trên tập huấn luyện, cập nhật trọng số cho mẫu dữ liệu và cứ tiếp tục như thế.



Hình . Huấn luyện tuần tự AdaBoost với trọng số mẫu được cập nhật

Một khi huấn luyện xong, cách mà ensemble đưa ra các dự đoán rất giống với phương pháp bagging hoặc pasting, ngoại trừ việc các bộ dự đoán có trọng số khác nhau tuỳ thuộc vào độ chính xác tổng thể trên tập huấn luyện đã đánh trọng số.

Một điểm trừ lớn của kỹ thuật học tuần tự này là không thể học song song (hoặc chỉ có thể song song hóa một phần). Ta chỉ có thể huấn luyện bộ dự đoán mới sau khi hoàn thành quá trình huấn luyện và đánh giá bộ dự đoán trước đó. Do đó, kỹ thuật này không mở rộng tốt bằng phương pháp bagging và pasting.

Gradient Boosting

Một thuật toán boosting phổ biến khác đó là Gradient Boosting.17 Cũng giống AdaBoost, Gradient Boosting hoạt động bằng cách lần lượt thêm các bộ dự đoán vào ensemble, mỗi bộ sửa lỗi cho bộ trước đó. Tuy nhiên, thay vì tác động vào trọng số của mẫu tại tất cả các vòng lặp như AdaBoost, phương pháp này cố gắng khớp bộ dự đoán mới vào sai số phần dư (residual error) của bộ dự đoán trước đó.

III.THÔNG SỐ KỸ THUẬT YÊU CẦU

Yêu cầu phần cứng

- Ổ đĩa cứng : 500 GB trở lên
- CPU : i3 trở lên
- Ram : 4GB trở lên

Yêu cầu phần mềm

- Hệ điều hành : Window 10 / MacOs
- Phần mềm : Python
- Công cụ : Anaconda (Jupyter Notebook IDE)

Công nghệ sử dụng

- Ngôn ngữ lập trình : Python
- Thư viện : pandas,numpy, matplotlib.pyplot, seaborn, plotly.express, plotly.graph_objects, plotly.subplots (make_subplots), colorama (Fore, Back, Style), sklearn.metrics (accuracy_score, confusion_matrix, ConfusionMatrixDisplay).

IV.PHƯƠNG PHÁP LUẬN

1. Phương pháp Thu thập Dữ liệu

Tập dữ liệu được sử dụng trong nghiên cứu này là dữ liệu thứ cấp thu thập từ nền tảng Kaggle, nơi các dữ liệu được tổng hợp và chuẩn bị cho các cuộc thi khoa học dữ liệu. Nguồn gốc của dữ liệu bao gồm các hồ sơ vay mượn được thu thập từ các ngân hàng và tổ chức tài chính, đã được xử lý và cung cấp dưới dạng tập dữ liệu sẵn sàng phân tích.

2. Phương pháp xây dựng và huấn luyện mô hình

Mô hình sẽ được triển khai trên các tham số mặc định sau đó sử dụng hai phương pháp chính là Grid Search hoặc Random Search để tìm kiếm siêu tham số tối ưu cho mô hình Random Forest. Sau đó sẽ tiếp tục huấn luyện với các mô hình Adaboost và Gradient Boosting.

Grid Search : một kỹ thuật tối ưu hóa siêu tham số thông dụng, nơi một lưới các giá trị siêu tham số được định nghĩa trước và mỗi kết hợp có thể của chúng được đánh giá. Phương pháp này thử tất cả các kết hợp siêu tham số có thể để tìm ra bộ tham số tốt nhất cho một mô hình. Điều này đảm bảo rằng mọi cài đặt được xem xét kỹ lưỡng, mang lại một giải pháp toàn diện và chính xác nhất trong không gian tham số đã xác định. Tuy nhiên, Grid Search có thể rất tốn kém về mặt tính toán, đặc biệt là khi không gian tham số lớn và phức tạp.

Random Search: một phương pháp thay thế cho Grid Search, trong đó các giá trị siêu tham số được chọn ngẫu nhiên từ một phân phối xác định cho mỗi tham số. Điều này không đảm bảo khám phá hết mọi kết hợp như Grid Search, nhưng lại có lợi thế về mặt hiệu quả tính toán. Random Search thường tìm được một giải pháp tốt trong thời gian ngắn hơn nhiều, đặc biệt là trong các trường hợp mà một số siêu tham số không ảnh hưởng đáng kể đến hiệu suất của mô hình. Phương pháp này rất hữu ích khi bạn cần một giải pháp tối ưu nhanh chóng và có thể chấp nhận độ chính xác kém hơn một chút so với giải pháp tối ưu toàn cục.

Các siêu tham số được sử dụng trong các mô hình:

Random Forest

n_estimators: Số lượng cây trong rừng. Mặc định là 100.

criterion: Chức năng đo lường chất lượng của một phân chia. "gini" cho độ tinh khiết Gini và "entropy" cho thông tin tăng.

max_depth: Độ sâu tối đa của cây. Nếu None, cây sẽ mở rộng cho đến khi tất cả lá là tinh khiết hoặc cho đến khi tất cả các lá chứa ít hơn min_samples_split mẫu.

min_samples_split: Số lượng mẫu tối thiểu cần thiết để phân chia một nút nội bộ.

min_samples_leaf: Số lượng mẫu tối thiểu được yêu cầu ở một nút lá.

min_weight_fraction_leaf: Phân số trọng số tối thiểu của tổng trọng số (của tất cả các mẫu đầu vào) cần thiết để ở một nút lá.

max_features: Số lượng đặc trưng cần xem xét khi tìm kiếm phân chia tốt nhất.

max_leaf_nodes: Số lượng nút lá tối đa.

min_impurity_decrease: Một nút sẽ được tách nếu phân chia này làm giảm độ tạp đi một lượng lớn hơn hoặc bằng giá trị này.

bootstrap: Có áp dụng lấy mẫu bootstrap cho cây hay không.

oob_score: Có sử dụng các mẫu out-of-bag để ước tính R^2 trên dữ liệu chưa thấy hay không.

random_state: Kiểm soát tính ngẫu nhiên của bộ ước lượng.

warm_start: Khi được đặt thành True, tái sử dụng giải pháp của lần gọi trước để phù hợp với bổ sung mới hoặc làm cho các ước lượng tốt hơn.

AdaBoost

base_estimator: Mô hình cơ sở để tăng cường (mặc định là cây quyết định).

n_estimators: Số lượng bộ ước lượng cần tăng cường.

learning_rate: Tốc độ học giảm trọng số của mỗi bộ ước lượng.

algorithm: Thuật toán tăng cường sử dụng, có thể là "SAMME" hoặc "SAMME.R".

random_state: Kiểm soát tính ngẫu nhiên của bộ ước lượng.

Gradient Boosting

loss: Chức năng mất mát để tối ưu hóa. Các tùy chọn bao gồm 'deviance' cho phân loại với hậu quả là logistic regression, và 'exponential' cho AdaBoost.

learning_rate: Tốc độ học, làm giảm đóng góp của mỗi cây.

n_estimators: Số lượng bước tăng cường để thực hiện.

subsample: Phân số mẫu để sử dụng để phù hợp với mỗi bộ ước lượng cơ sở. Mặc định là 1, có nghĩa là sử dụng lấy mẫu Stochastic Gradient Boosting.

criterion: Chức năng đo lường chất lượng của một phân chia.

min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_depth, min_impurity_decrease, min_impurity_split: Tương tự như trong Random Forest.

init: Một bộ ước lượng ban đầu để nâng cao năng lực của bộ ước lượng.

random_state, max_features, verbose, max_leaf_nodes, warm_start: tương tự như trong Random Forest.

3. **Đánh giá mô hình**

Accuracy (Độ chính xác):

- Đo lường tỷ lệ dự đoán đúng (cả dương tính và âm tính) trên tổng số trường hợp.
- Công thức:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Population}}$$

Precision (Độ chính xác):

- Đo lường tỷ lệ dự đoán dương tính chính xác so với tổng số dự đoán dương tính.
- Công thức:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall (Độ nhạy):

- Còn được gọi là Sensitivity hoặc True Positive Rate, đo lường tỷ lệ dự đoán dương tính chính xác so với tổng số trường hợp thực sự dương tính.
- Công thức:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score:

- Là trung bình điều hòa của Precision và Recall, giúp cân bằng giữa Precision và Recall.
- Công thức:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

V.TRIỂN KHAI

1. Thu thập và mô tả dữ liệu

Bộ dữ liệu được thu thập từ Kaggle[1]. Bộ dữ liệu đó bao gồm một số thông tin như giới tính, tình trạng hôn nhân, có tự kinh doanh hay không, thu nhập hàng tháng, v.v. Bộ dữ liệu chứa thông tin, việc khoản vay trước đó có được chấp thuận hay không phụ thuộc vào thông tin của khách hàng. Tập dữ liệu được sử dụng có định dạng csv.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	I
0	LP001002	Male	No	0	Graduate	No	5849	0.000000	nan	360.000000	1.000000	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.000000	128.000000	360.000000	1.000000	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.000000	66.000000	360.000000	1.000000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.000000	120.000000	360.000000	1.000000	
4	LP001008	Male	No	0	Graduate	No	6000	0.000000	141.000000	360.000000	1.000000	

Hình 6. 5 hàng đầu của tập dữ liệu

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             601 non-null    object
 1   Married            611 non-null    object
 2   Dependents         599 non-null    object
 3   Education          614 non-null    object
 4   Self_Employed      582 non-null    object
 5   ApplicantIncome    614 non-null    int64
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         592 non-null    float64
 8   Loan_Amount_Term   600 non-null    float64
 9   Credit_History     564 non-null    float64
 10  Property_Area      614 non-null    object
 11  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(7)
memory usage: 57.7+ KB
```

Hình 7. Mô tả tập dữ liệu

Trong tập dữ liệu này chúng ta có:

- 614 hàng hay 614 mẫu
- 12 thuộc tính
- 5 thuộc tính thuộc kiểu giá trị số (ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History)
- 7 thuộc tính thuộc kiểu dữ liệu object (Gender, Married, Dependents, Education, Self_Employed, Property_Area, Loan_Status)

Gender	Giới tính của người nộp đơn (ví dụ:Nam, Nữ).
Married	Tình trạng hôn nhân của người nộp đơn (Có hoặc Không).
Dependents	Số người phụ thuộc của người nộp đơn.
Education	Trình độ học vấn của người nộp đơn (Tốt nghiệp hoặc Chưa tốt nghiệp).
Self_Employed	Cho biết liệu người nộp đơn có tự kinh doanh không (Có hoặc Không).
ApplicantIncome	Thu nhập của người nộp đơn.
CoapplicantIncome	Thu nhập của người đồng nộp đơn.
LoanAmount	Số tiền vay được yêu cầu.
Loan_Amount_Term	Thời hạn của khoản vay tính theo tháng.
Credit_History	Hồ sơ lịch sử tín dụng của người nộp đơn (ví dụ, 1 cho lịch sử tốt).

Property_Area	Khu vực nơi tài sản được đặt (Đô thị, Nông thôn, Bán đô thị).
Loan_Status	Tình trạng của khoản vay (Y cho Đồng ý, N cho Không đồng ý).

Bảng 1: Các trường dữ liệu

2. Tiền xử lý và làm sạch dữ liệu

Thực hiện kiểm tra và loại bỏ các giá trị null và các giá trị trùng lặp

Hình 8 . Loại bỏ các giá trị null

3. Chuyển đổi - mã hoá dữ liệu:

Label Encoding đã được áp dụng cho các cột 'Married', 'Education', 'Self_Employed', 'Loan_Status', và 'Dependents' (do 'Dependents' có trật tự tự nhiên).

One-Hot Encoding đã được thực hiện trên các cột 'Gender' và 'Property_Area'. Điều này tạo ra các cột mới với giá trị 0 hoặc 1 để biểu diễn sự hiện diện của mỗi danh mục :

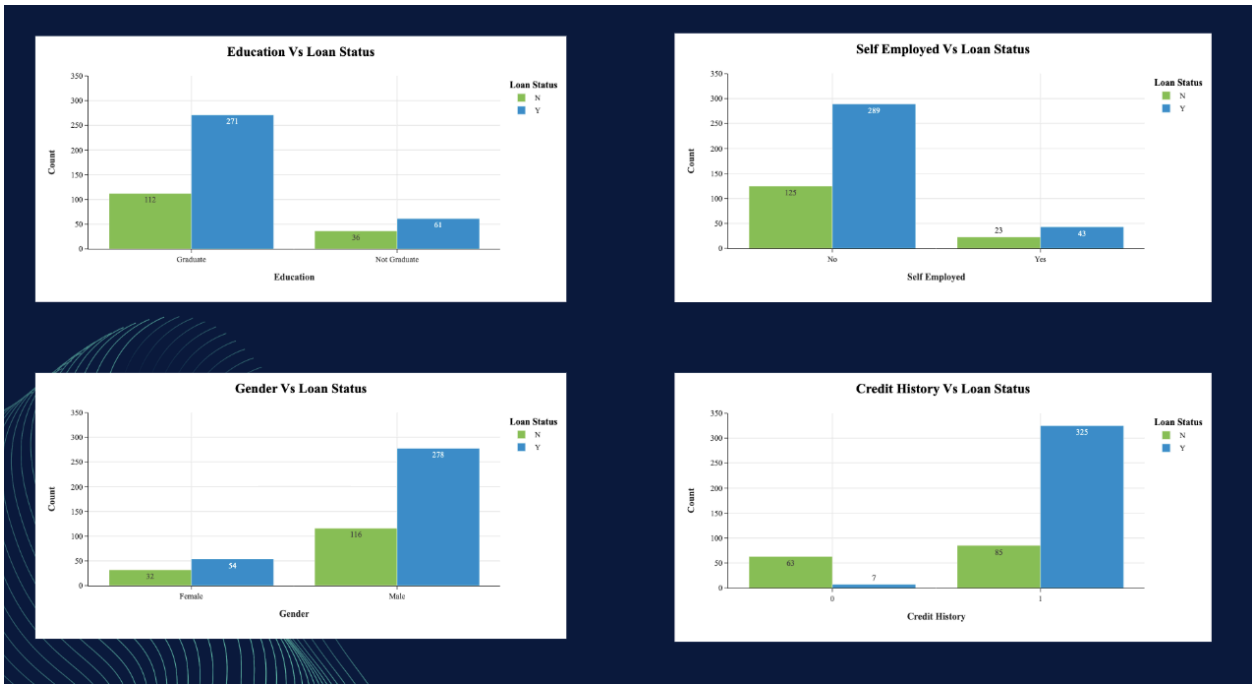
- Loại bỏ cột 'Gender', thêm cột Gender_Female, giả định rằng nếu 'Gender_Male' là 0 (False) thì giới tính là Nữ.
- Tương tự khi thực hiện One-Hot Encoding với cột Property_Area thì cột này bị loại bỏ và hai cột mới được tạo ra là 'PA_Semiurban' và 'PA_Urban'. danh mục 'Rural' không được trực tiếp biểu diễn trong dữ liệu sau khi mã hóa. Nó được biểu diễn gián tiếp: nếu cả 'PA_Semiurban' và 'PA_Urban' đều có giá trị 0, thì điều đó

ngụ ý rằng 'Property_Area' cho quan sát đó là 'Rural'. (Đây là một cách hiệu quả để mã hóa các biến phân loại có nhiều danh mục mà không làm tăng quá nhiều số lượng đặc trưng, đồng thời tránh vấn đề đa cộng tuyến trong mô hình học máy.)

Total_Income = 'ApplicantIncome + 'CoapplicantIncome' : xét thu nhập của người nộp đơn để đánh giá khả năng trả nợ, trước khi phê duyệt yêu cầu vay vốn. Nếu có nhiều hơn một người nộp đơn, tổng thu nhập của họ sẽ được xem xét để đánh giá khả năng trả nợ. Vì vậy, tôi sẽ tổng hợp thu nhập của Người nộp đơn và thu nhập của Người đồng nộp đơn để tạo ra một đặc trưng mới gọi là ' Total income - Tổng Thu Nhập.

4. **Phân tích khám phá dữ liệu**

- **Tác động của Giới tính, Giáo dục, Kinh doanh Tự do và Lịch sử Tín dụng đối với Tình trạng Khoản vay**

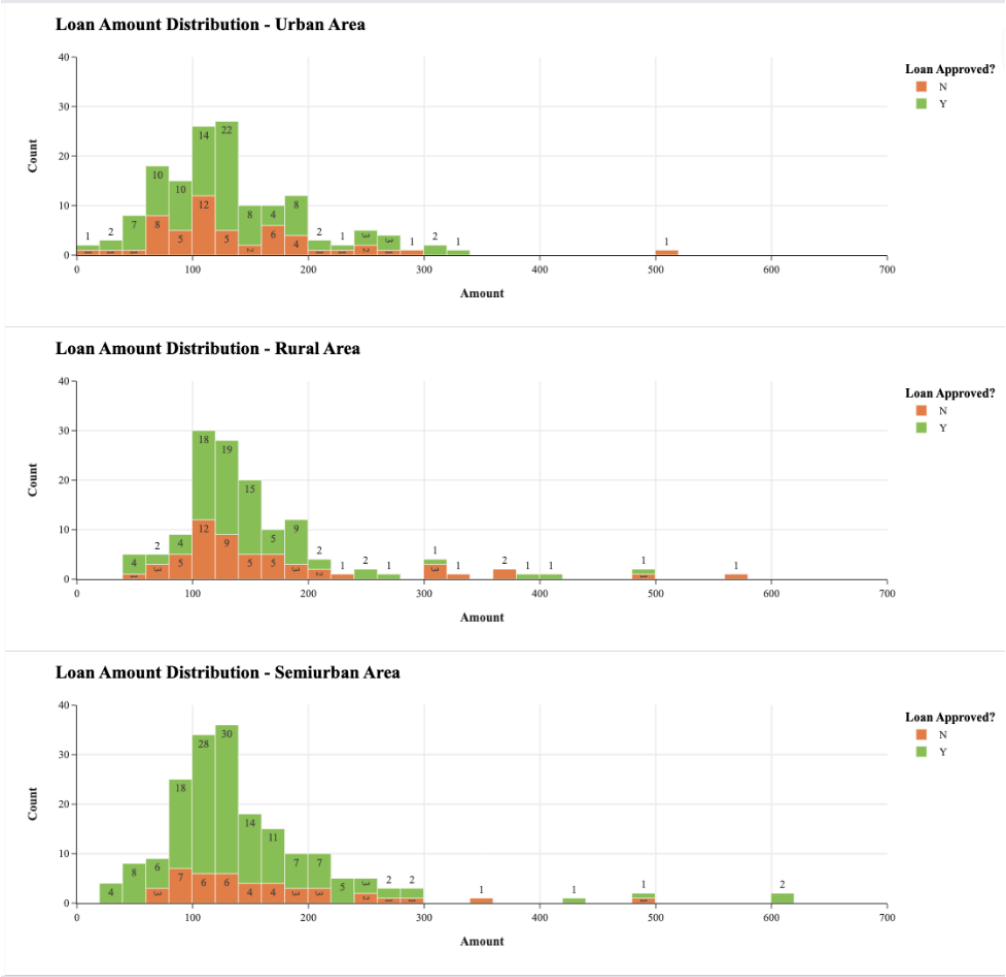


Hình 9. Biểu đồ Giới tính, Giáo dục, Kinh doanh Tự do và Lịch sử Tín dụng đối với Tình trạng Khoản vay

- Trong các biểu đồ trên, chúng ta thấy các trường hợp Phê duyệt và Từ chối Khoản vay gần như giống nhau trong trường hợp của cả Sinh viên tốt nghiệp và Không tốt nghiệp. Tương tự đối với Giới tính và Người tự kinh doanh. Vì vậy, các đặc trưng về Giới tính, Giáo dục và Tự kinh doanh không có tác động đáng kể đến Tình trạng khoản vay.

- Đối với Credit_History, chúng ta có thể thấy tỷ lệ phê duyệt Khoản vay nhiều hơn trong trường hợp Người đăng ký có Lịch sử Tín dụng. Vì vậy, đặc trưng này có tác động đáng kể đến (loan_status)Tình trạng khoản vay

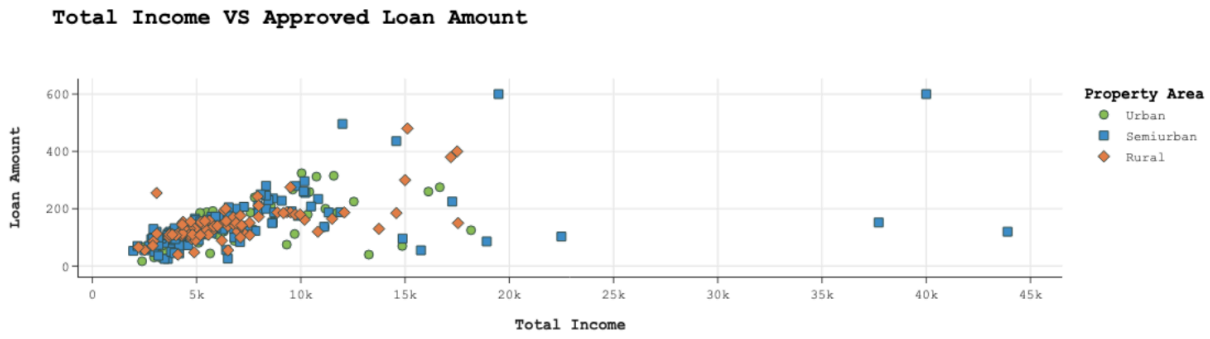
- **Tác động của vị trí bất động sản đến việc phê duyệt khoản vay**



Hình 10. vị trí bất động sản

- Số lượng khoản vay xuất hiện nhiều nhất ở khu vực Bán đô thị, tiếp theo là Khu vực Thành thị và Nông thôn.
- Phần phê duyệt hoặc từ chối khoản vay gần như giống nhau ở cả ba lĩnh vực tài sản.
- Khu vực bất động sản ít ảnh hưởng đến cơ hội phê duyệt khoản vay.

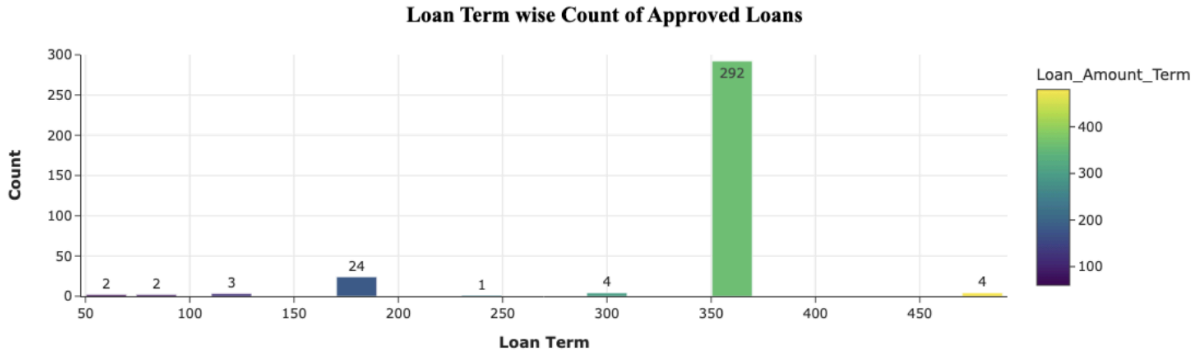
- **Tác động của thu nhập đến việc phê duyệt khoản vay**



Hình 11. Tổng thu nhập

- Có một xu hướng nhẹ rằng khi tổng thu nhập tăng lên, số tiền vay được phê duyệt cũng có xu hướng tăng lên, đặc biệt là trong khoảng thu nhập thấp đến trung bình. Tuy nhiên, ở mức thu nhập cao hơn, mối quan hệ này không còn rõ ràng nữa và có một số trường hợp có thu nhập rất cao nhưng số tiền vay được phê duyệt lại không lớn.
- Có một số điểm nằm rải rác và tách biệt khỏi cụm chính, đặc biệt là những khoản vay lớn không phù hợp với mức thu nhập tương ứng. Điều này cho thấy rằng tổng thu nhập không phải là yếu tố duy nhất quyết định số tiền vay được phê duyệt.

- **Tác động của thời hạn cho vay đến việc phê duyệt khoản vay**



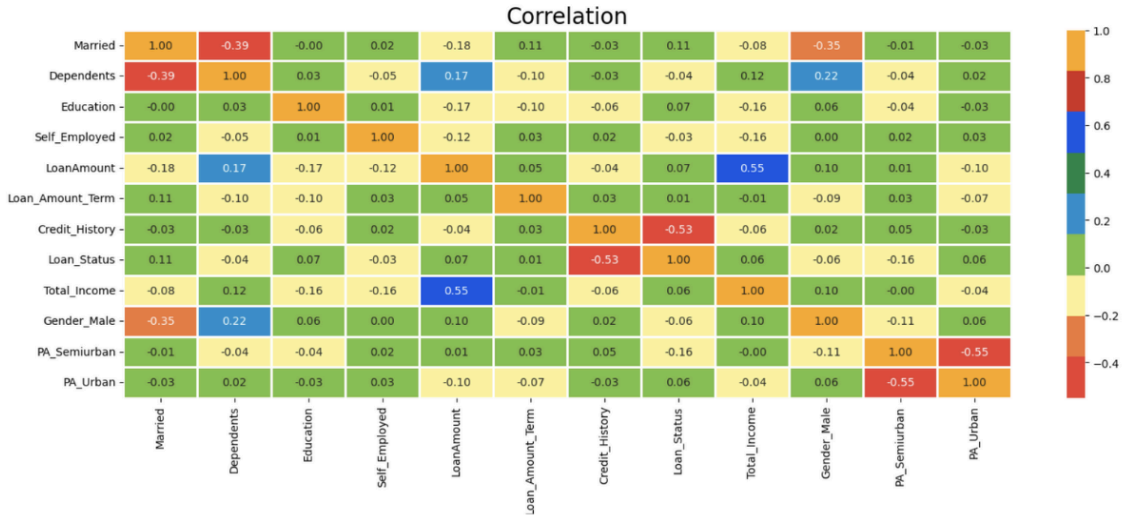
Hình 12. Kỳ hạn vay

Dựa vào biểu đồ, chúng ta có thể rút ra một số quan sát:

- Có một số lượng lớn khoản vay được phê duyệt với thời hạn khoảng 350 tháng (điểm cao nhất với 292 khoản vay).
- Số lượng khoản vay được phê duyệt với các thời hạn ngắn hơn (từ 50 đến 200 tháng) hoặc dài hơn (450 tháng) là ít hơn rất nhiều.
- Có một số ít khoản vay được phê duyệt với thời hạn 300 tháng.

Từ biểu đồ này, chúng ta có thể thấy rằng khách hàng có xu hướng được phê duyệt cho vay với thời hạn khoản vay cụ thể, có lẽ là 350 tháng, điều này có thể là do các điều kiện vay mặc định của ngân hàng hoặc sự ưa chuộng của khách hàng. Để dự đoán sự phê duyệt khoản vay, mô hình có thể cần xem xét thời hạn khoản vay là một đặc trưng quan trọng.

5. Phân tích và lựa chọn đặc trưng



Hình 13. Heatmap

- Credit History và Loan_Status: Có tương quan tiêu cực mạnh (khoảng -0.53), điều này có thể cho thấy rằng những người có lịch sử tín dụng không tốt có khả năng ít được phê duyệt khoản vay hơn.
- Total_Income và LoanAmount: Có một mức độ tương quan tích cực mạnh (0.55), cho thấy rằng tổng thu nhập cao hơn thường đi kèm với số tiền vay cao hơn.
- Married và Gender_Male: Có một mức độ tương quan tiêu cực (khoảng -0.35), có thể phản ánh một xu hướng trong dữ liệu mà đàn ông đã kết hôn có khả năng cao hơn trong việc nộp đơn vay vốn.
- PA_Semiurban và Loan_Status: Có một mức độ tương quan tiêu cực (khoảng -0.55), có thể ngụ ý rằng các đơn vay từ khu vực ngoại ô bán đô thị có khả năng được phê duyệt cao hơn so với các khu vực khác.

- Credit_History có tương quan tiêu cực mạnh (-0.53) với Loan_Status, điều này cho thấy rằng nó là một đặc trưng quan trọng có thể ảnh hưởng lớn đến quyết định phê duyệt khoản vay.

6. Triển khai mô hình

Random Forest

- Đầu tiên thực hiện triển khai mô hình Random Forest với các siêu tham số mặc định

```
#rfc = RandomForestClassifier(n_estimators=500, random_state=300)
#rfc = RandomForestClassifier(n_estimators=100,max_depth=3,min_samples_leaf = 10)

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Tạo mô hình

rfc = RandomForestClassifier(n_estimators=250, random_state=250)
rfc.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=250, random_state=250)
```

- Sử dụng Grid Search để tìm các siêu tham số tối ưu

```
from sklearn.model_selection import GridSearchCV

# Tạo một mô hình RandomForestClassifier
rf = RandomForestClassifier()

# Thiết lập lưới siêu tham số cho Grid Search
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'max_features': ['sqrt', 'log2']
}

# Tạo đối tượng GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy')

# Huấn luyện mô hình với Grid Search
grid_search.fit(X_train, y_train)

# In ra các siêu tham số tốt nhất và điểm số
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

Tham số tối ưu tìm được từ Grid Search:

```
Best parameters: {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 300}
Best score: 0.8359193438140806
```

- Sử dụng Random Search để tìm các tham số tối ưu

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint

# Tạo một mô hình RandomForestClassifier
rf = RandomForestClassifier()

# Thiết lập phạm vi siêu tham số cho Random Search
param_dist = {
    'n_estimators': sp_randint(100, 400),
    'max_depth': sp_randint(10, 50),
    'min_samples_split': sp_randint(2, 20),
    'max_features': ['sqrt', 'log2']
}

# Tạo đối tượng RandomizedSearchCV
random_search = RandomizedSearchCV(estimator=rf, param_distributions=param_dist, n_iter=100, cv=5, scoring='accuracy', random_state=42)

# Huấn luyện mô hình với Random Search
random_search.fit(X_train, y_train)

# In ra các siêu tham số tốt nhất và điểm số
print("Best parameters:", random_search.best_params_)
print("Best score:", random_search.best_score_)
```

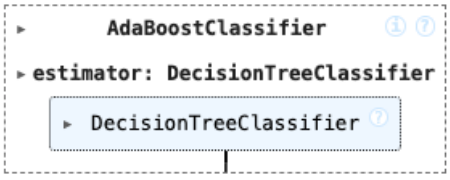
Tham số tối ưu tìm được từ Random Search:

```
Best parameters: {'max_depth': 44, 'max_features': 'log2', 'min_samples_split': 18, 'n_estimators': 149}
Best score: 0.8176691729323308
```

AdaBoost

Triển khai mô hình data boost

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
ada_clf = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1), n_estimators=200,
    algorithm="SAMME.R", learning_rate=0.5, random_state=42)
ada_clf.fit(X_train, y_train)
```



Base Estimator (DecisionTreeClassifier(max_depth=1)) với max_depth=1đồng nghĩa với sử dụng cây quyết định một cấp làm bộ ước lượng cơ bản để tránh overfitting và đảm bảo rằng từng bước boosting chỉ tập trung vào sửa lỗi của bước trước.

n_estimators=200: Đây là số lượng cây quyết định sẽ được tạo, cho phép mô hình có đủ 'ý kiến' khác nhau để học từ dữ liệu. Số lượng lớn cây giúp tăng cường độ chính xác nhưng cũng tăng yêu cầu về tài nguyên tính toán.

algorithm="SAMME.R": Là phiên bản nâng cấp của thuật toán SAMME, sử dụng xác suất dự đoán để cải thiện hiệu quả và tốc độ hội tụ của mô hình.

learning_rate=0.5: Tốc độ học 0.5 giúp cân bằng giữa tốc độ học và rủi ro overfitting, giúp mô hình đạt hiệu quả cao mà không bị ảnh hưởng quá nhiều bởi nhiễu hay các đặc điểm ngoại lai trong dữ liệu.

random_state=42: giá trị cố định cho random_state đảm bảo tính nhất quán và tái lập được của các kết quả khi thực hiện các thí nghiệm hoặc so sánh mô hình.

Gradient Boosting

Triển khai mô hình Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
gb_clf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

# Huấn luyện mô hình
gb_clf.fit(X_train, y_train)

# Dự đoán kết quả trên tập kiểm thử
y_pred = gb_clf.predict(X_test)

# Đánh giá mô hình
accuracy = accuracy_score(y_test, y_pred)
```

n_estimators=100: Đây là số lượng cây quyết định được tạo trong quá trình tăng cường. 100 cây đảm bảo đủ độ phức tạp để mô hình học các mẫu từ dữ liệu mà không quá nhiều dẫn đến rủi ro overfitting.

learning_rate=0.1: Tốc độ học này kiểm soát tốc độ mà mỗi cây ảnh hưởng đến mô hình cuối cùng. Một giá trị nhỏ như 0.1 giúp đảm bảo rằng mỗi cây chỉ đóng góp một cải tiến nhỏ, giúp mô hình ổn định và chống lại overfitting.

max_depth=3: Độ sâu tối đa của mỗi cây quyết định. Độ sâu 3 cho phép cây đủ sâu để nắm bắt các tương tác giữa các đặc trưng nhưng không quá sâu để gây overfitting.

random_state=42: Tham số này đảm bảo rằng kết quả của mô hình có thể tái tạo được, giúp nghiên cứu và so sánh các thử nghiệm khác nhau được nhất quán.

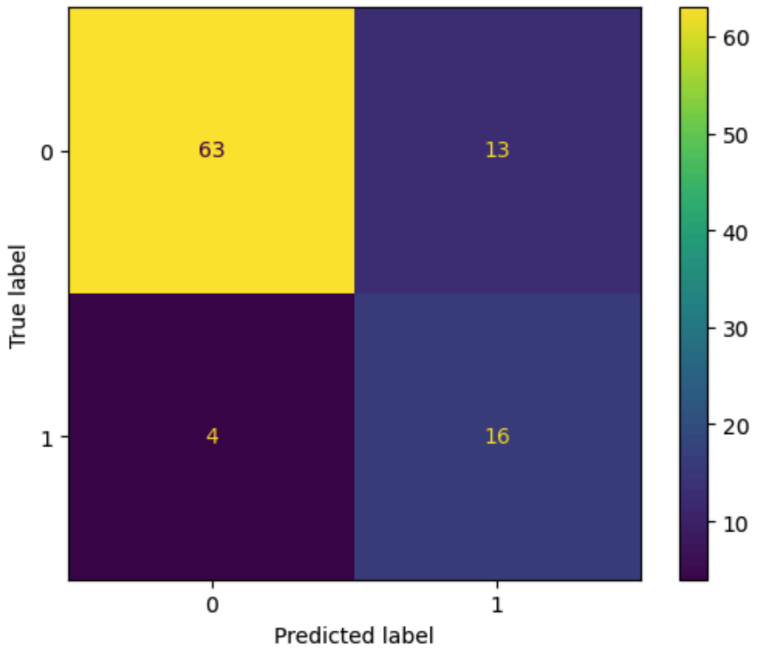
VI.KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH

1. **Random Forest**

Sử dụng tham số mặc định

```
Accuracy: 0.8229166666666666
F1 Score: 0.8811188811188811
Precision: 0.8289473684210527
Recall: 0.9402985074626866
```

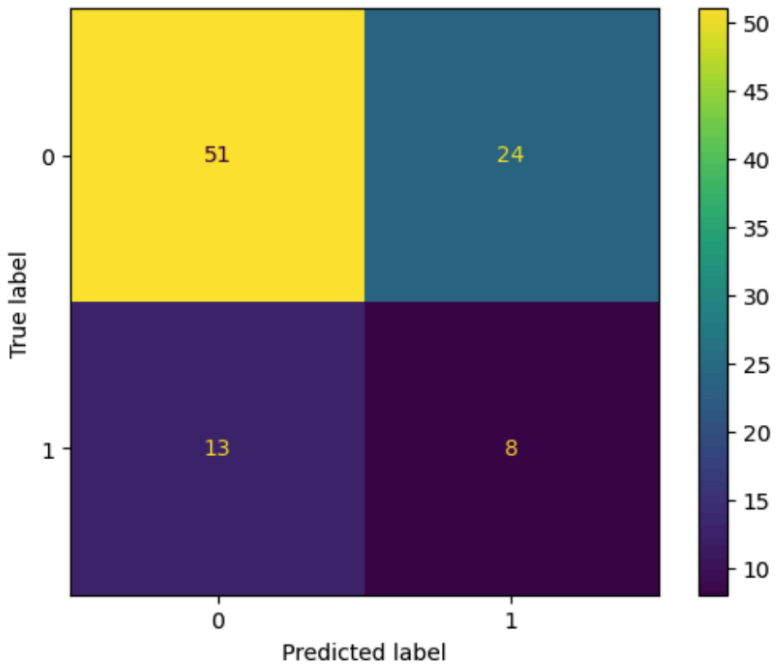
Confusion matrix:



Mô hình Random Forest với tham số mặc định đạt độ chính xác 82.29%, cho thấy một hiệu suất khá cao trong việc phân loại các mẫu. F1 Score ở mức 0.8811 thể hiện sự cân bằng giữa precision (0.8289) và recall (0.9403). Điều này cho thấy mô hình có khả năng phát hiện hầu hết các trường hợp phê duyệt cho vay (recall cao) trong khi vẫn duy trì một tỷ lệ chính xác tương đối cao (precision). Confusion matrix chỉ ra rằng mô hình có 63 dự đoán đúng về các trường hợp không phê duyệt và 16 dự đoán đúng về các trường hợp phê duyệt, với chỉ 4 sai sót trong việc dự đoán không phê duyệt thành phê duyệt.

Sử dụng tham số từ Grid Search

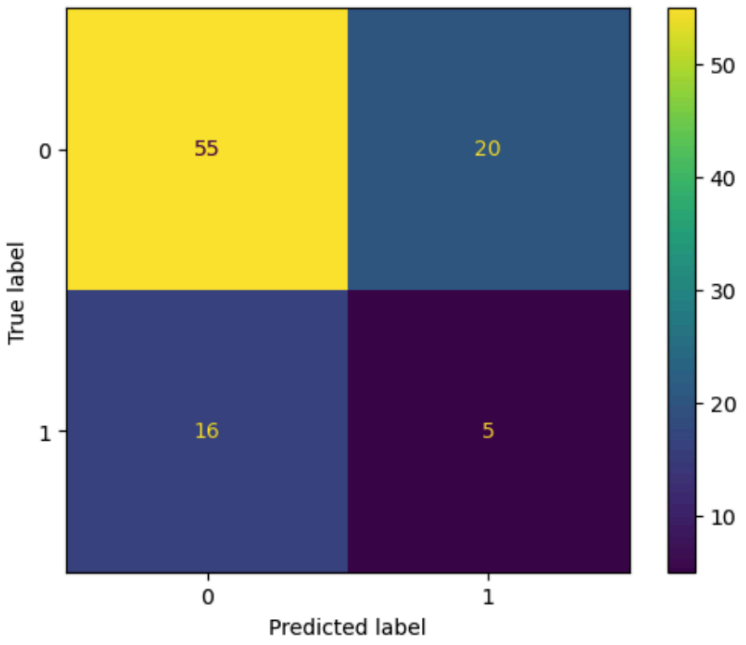
```
Best parameters: {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 300}
Best score: 0.8359193438140806
F1 Score: 0.7338129496402878
Precision: 0.68
Recall: 0.796875
```



Khi tối ưu hóa tham số bằng Grid Search, mô hình Random Forest đạt độ chính xác cao hơn một chút (83.59%) so với mô hình mặc định. Tuy nhiên, F1 Score giảm xuống còn 0.7338, cho thấy sự cân bằng giữa precision (0.6800) và recall (0.7969) không tốt bằng mô hình mặc định. Confusion matrix cho thấy mô hình có 51 dự đoán đúng về các trường hợp không phê duyệt và chỉ 8 dự đoán đúng về các trường hợp phê duyệt, đồng thời có 13 sai sót trong việc dự đoán không phê duyệt thành phê duyệt.

Sử dụng tham số từ Random Search

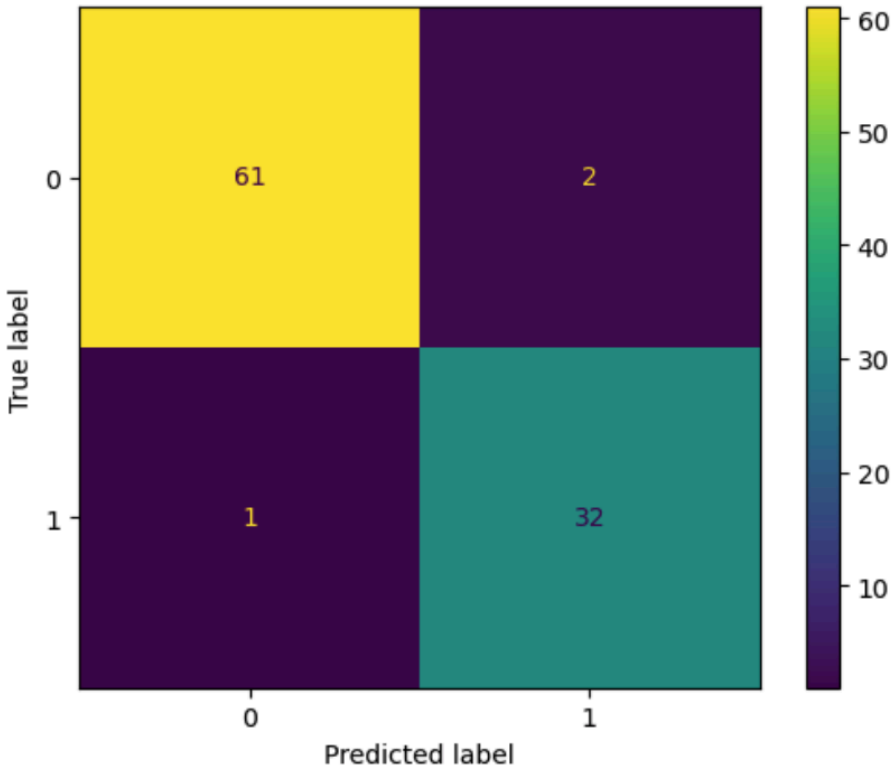
```
Best parameters: {'max_depth': 44, 'max_features': 'log2', 'min_samples_split': 18, 'n_estimators': 149}
Best score: 0.8176691729323308
F1 Score: 0.7534246575342466
Precision: 0.7333333333333333
Recall: 0.7746478873239436
```



Mô hình Random Forest với tham số từ Random Search có độ chính xác 81.77%, thấp hơn một chút so với hai mô hình trước. F1 Score đạt 0.7534, với precision (0.7333) và recall (0.7746) cho thấy sự cân bằng giữa độ chính xác và khả năng phát hiện các trường hợp phê duyệt là trung bình. Confusion matrix cho thấy mô hình có 55 dự đoán đúng về các trường hợp không phê duyệt và chỉ 5 dự đoán đúng về các trường hợp phê duyệt, với 16 sai sót trong việc dự đoán không phê duyệt thành phê duyệt.

2. AdaBoost

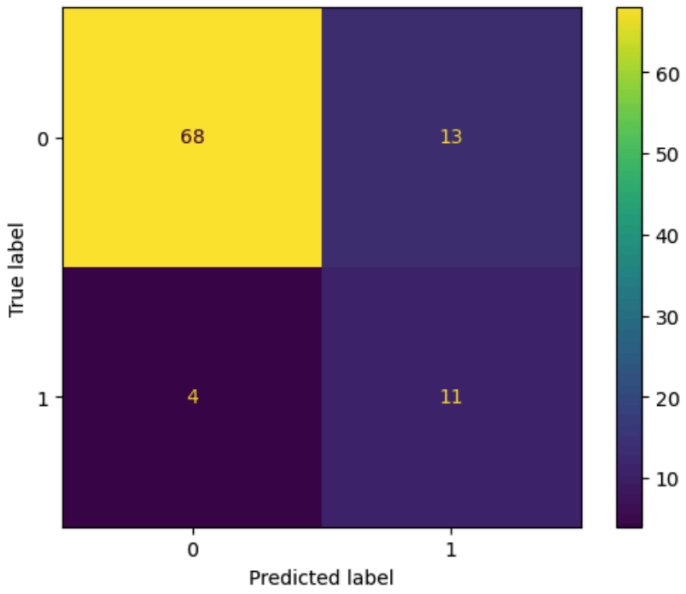
```
Accuracy: 0.96875
F1 Score: 0.976
Precision: 0.9682539682539683
Recall: 0.9838709677419355
```



AdaBoost đạt độ chính xác rất cao 96.88%, vượt trội so với tất cả các mô hình khác. F1 Score đạt 0.9760, với precision (0.9683) và recall (0.9839) cho thấy mô hình này không chỉ có khả năng phát hiện hầu hết các trường hợp phê duyệt (recall cao) mà còn duy trì một tỷ lệ chính xác rất cao (precision). Confusion matrix cho thấy mô hình chỉ có 2 sai sót trong việc dự đoán không phê duyệt thành phê duyệt và chỉ 1 sai sót trong việc dự đoán phê duyệt thành không phê duyệt, chứng tỏ hiệu suất vượt trội.

3. Gradient Boosting

```
Accuracy: 0.8229166666666666
F1 Score: 0.8888888888888888
Precision: 0.8395061728395061
Recall: 0.9444444444444444
```



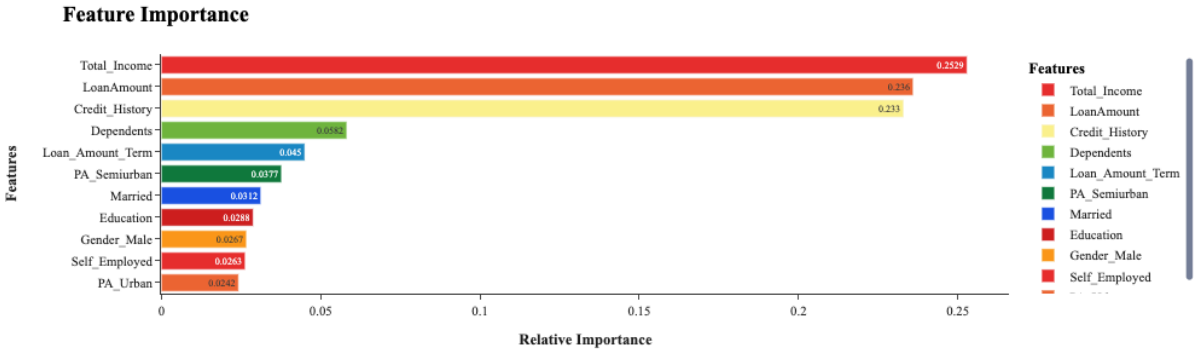
Gradient Boosting đạt độ chính xác 82.29%, tương đương với mô hình Random Forest mặc định. F1 Score cao (0.8889), với precision (0.8395) và recall (0.9444) cho thấy mô hình này có sự cân bằng tốt giữa độ chính xác và khả năng phát hiện các trường hợp phê duyệt. Confusion matrix chỉ ra rằng mô hình có 68 dự đoán đúng về các trường hợp không phê duyệt và 11 dự đoán đúng về các trường hợp phê duyệt, với chỉ 4 sai sót trong việc dự đoán không phê duyệt thành phê duyệt.

Khi so sánh các mô hình, rõ ràng AdaBoost thể hiện hiệu suất vượt trội so với các mô hình khác về mọi chỉ số (accuracy, F1 Score, precision, recall) và có số lượng sai sót thấp nhất trong confusion matrix. Gradient Boosting cũng có kết quả tốt, gần ngang ngửa với mô hình Random Forest mặc định nhưng với F1 Score cao hơn, cho thấy sự cân bằng tốt hơn giữa độ chính xác và khả năng phát hiện.

Random Forest với tham số mặc định và Gradient Boosting đều có hiệu suất tốt, tuy nhiên, sự cải thiện từ Grid Search và Random Search đối với Random Forest không đạt được kết quả mong đợi và thậm chí giảm hiệu suất của mô hình.

Vậy AdaBoost là mô hình xuất sắc nhất cho bài toán dự đoán phê duyệt cho vay trong nghiên cứu này, với hiệu suất cao nhất và số lượng sai sót tối thiểu, theo sau là Gradient Boosting và Random Forest với tham số mặc định. Các phương pháp tối ưu hóa tham số cho Random Forest (Grid Search và Random Search) không mang lại cải thiện đáng kể và thậm chí làm giảm hiệu suất của mô hình.

4. Độ quan trọng của các đặc trưng



Total_Income (0.2529), LoanAmount (0.236), và Credit_History (0.223) là các đặc trưng quan trọng nhất, đóng góp lớn nhất vào mô hình dự đoán phê duyệt cho vay.

Dependents (0.0582) và Loan_Amount_Term (0.0445) có tầm quan trọng trung bình, cho thấy chúng cũng có ảnh hưởng nhưng không đáng kể như ba đặc trưng đầu tiên.

Các đặc trưng như PA_Semiurban, Married, Education, Gender_Male, Self_Employed, và PA_Urban có tầm quan trọng thấp, chỉ đóng góp nhỏ vào dự đoán.

VII.KẾT LUẬN

Trong dự án này, chúng tôi đã triển khai và đánh giá hiệu quả của các mô hình học Ensemble, bao gồm Random Forest, AdaBoost, và Gradient Boosting, trong việc dự đoán phê duyệt cho vay. Kết quả cho thấy, mô hình AdaBoost thể hiện sự vượt trội với độ chính xác, F1 Score, precision, và recall cao nhất, cùng với số lượng sai sót tối thiểu trong confusion matrix. Điều này chứng tỏ AdaBoost là một công cụ mạnh mẽ và hiệu quả cho bài toán dự đoán phê duyệt cho vay.

Trong khi đó, Random Forest và Gradient Boosting cũng đạt được hiệu suất tốt, nhưng không vượt qua được AdaBoost. Việc tối ưu hóa tham số bằng Grid Search và Random Search cho Random Forest không mang lại cải thiện đáng kể và thậm chí làm giảm hiệu suất của mô hình.

Nhìn chung, dự án đã chứng minh rằng việc áp dụng các mô hình học Ensemble có thể cải thiện đáng kể khả năng dự đoán phê duyệt cho vay so với các phương pháp truyền thống. Tuy nhiên, cần tiếp tục nghiên cứu và thử nghiệm với các tập dữ liệu lớn hơn và đa dạng hơn để xác nhận tính tổng quát của các mô hình này trong thực tiễn. Sự hiểu biết sâu sắc về các đặc trưng quan trọng, như lịch sử tín dụng và thu nhập, cũng

đóng vai trò quan trọng trong việc tối ưu hóa các mô hình dự đoán, góp phần nâng cao hiệu quả quản lý rủi ro và quyết định tài chính của các tổ chức ngân hàng.

TÀI LIỆU THAM KHẢO

[1] Dataset: https://www.kaggle.com/datasets/rishikeshkonapure/home-loan-approval

[2] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[3]

TS.Thân Quang Khoát (2021), "Cây quyết định và rừng ngẫu nhiên":

L7-Random-forests.pdf

Source code:

[4] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

[5] Brown, J., & Smith, A. (2018). Logistic Regression in Financial Decision Making. Journal of Financial Analytics, 22(3), 45-56.

[6] Chen, Y., Liu, Z., & Huang, J. (2021). Balancing Accuracy and Interpretability in Predictive Models for Loan Approval. Journal of Computational Finance, 29(1), 67-85.

[7] Goyal, S., & Kumar, R. (2021). Feature Importance Analysis in Machine Learning Models for Loan Approval Prediction. International Journal of Data Science, 14(2), 112-129.

[8] Lee, H., & Lee, J. (2019). Decision Trees in Predictive Modeling for Loan Approval. Journal of Applied Machine Learning, 17(4), 89-103.

[9] Patel, M., & Desai, P. (2020). A Comparative Study of Machine Learning Algorithms for Loan Approval. Journal of Artificial Intelligence Research, 25(3), 145-158.

[10] Singh, R., Gupta, M., & Sharma, P. (2022). Key Predictors of Loan Approval: A Machine Learning Approach. International Journal of Financial Studies, 30(2), 98-115.

[11] Wang, Q., & Xu, L. (2019). Evaluating Machine Learning Models for Financial Decision Making. Journal of Financial Technology, 12(1), 34-50.

[12] Zhao, H., Zhang, L., & Li, M. (2020). Enhancing Loan Approval Prediction with RandomForest. Journal of Machine Learning Applications, 18(3), 205-218.