

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----🙣🕮🙡----**



**BÁO CÁO PROJECT**

***Đề tài: Phân loại người vay có khả năng vỡ nợ bằng***

***Machine Learning.***

***Giảng viên hướng dẫn* : Hồ Đắc Quán**

***Sinh viên thực hiện* : Bùi Đình Hanh Du**

***Lớp* : DHKHMT16B**

**MSSV : 20115621**

***TP.HCM, ngày 15 tháng 11 năm 2023***

Nội dung

[1](#_Toc150977840)

[Phần 1: Giới thiệu tổng quan. 3](#_Toc150977841)

[1. Giới thiệu về đề tài. 4](#_Toc150977842)

[2. Phương pháp thực hiện đề tài: 4](#_Toc150977843)

[Phần 2: Cơ sở lí thuyết. 4](#_Toc150977844)

[1. Giới thiệu về các thuật toán. 4](#_Toc150977845)

[1.1 Thuật toán K-nearest neighbor. 4](#_Toc150977846)

[1.2 Thuật toán Random Forest Classifier 5](#_Toc150977847)

[1.3 Thuật toán Decision Tree 5](#_Toc150977848)

[1.4 Thuật toán Logistis Regression 6](#_Toc150977849)

[1.5 Thuật toán Gaussian Naive Bayes 7](#_Toc150977850)

[1.6 Neural Network 7](#_Toc150977851)

[2. Xử lí và chia tập dữ liệu. 8](#_Toc150977852)

[2.1 Mô tả dữ liệu: 8](#_Toc150977853)

[2.1 Xử lí dữ liệu. 9](#_Toc150977854)

[Phần 3: Đánh giá kết quả mô hình đạt được 11](#_Toc150977855)

[1. Thuật toán K-nearest neighbor. 11](#_Toc150977856)

[1.1 Cài đặt thuật toán 11](#_Toc150977857)

[1.2 Độ chính xác của thuật toán 12](#_Toc150977858)

[1.3 Confusion Matrix 12](#_Toc150977859)

[1.4 Receiver Operating Characteristic 13](#_Toc150977860)

[2. Thuật toán Random Forest Classifier 13](#_Toc150977861)

[2.1 Cài đặt thuật toán 13](#_Toc150977862)

[2.2 Độ chính xác của thuật toán 14](#_Toc150977863)

[2.3 Confusion Matrix 14](#_Toc150977864)

[2.4 Receiver Operating Charactristic 15](#_Toc150977865)

[3. Thuật toán Decision Tree 15](#_Toc150977866)

[3.1 Cài đặt thuật toán 15](#_Toc150977867)

[3.2 Độ chính xác của thuật toán 15](#_Toc150977868)

[3.3 Confusion Matrix 16](#_Toc150977869)

[3.4 Receiver Operating Characteristic 17](#_Toc150977870)

[4. Thuật toán Logistis Regression 17](#_Toc150977871)

[4.1 Cài đặt thuật toán 17](#_Toc150977872)

[4.2 Độ chính xác của thuật toán 18](#_Toc150977873)

[4.3 Confusion Matrix 18](#_Toc150977874)

[4.4 Receiver operating Charateristic 19](#_Toc150977875)

[5. Thuật toán Gaussian Naive Bayes 19](#_Toc150977876)

[5.1 Cài đặt thuật toán 19](#_Toc150977877)

[5.2 Độ chính xác của thuật toán 20](#_Toc150977878)

[5.3 Confusion Matrix 20](#_Toc150977879)

[5.4 Receiver Operating Characteristic 21](#_Toc150977880)

[6. Neural Network 21](#_Toc150977881)

[6.1 Cài đặt thuật toán 21](#_Toc150977882)

[6.2 Training and validation loss 22](#_Toc150977883)

[6.3 Training and validation Accyracy 23](#_Toc150977884)

[6.4 Độ chính xác của thuật toán 24](#_Toc150977885)

[6.5 Confusion Matrix 24](#_Toc150977886)

[6.6 Receiver Operating Characteristic 25](#_Toc150977887)

[Phần 4. Kết luận. 25](#_Toc150977888)

[1. Những điểm đạt được. 25](#_Toc150977889)

[2. Những hạn chế. 26](#_Toc150977890)

[3. Hướng phát triển. 26](#_Toc150977891)

[*Tài liệu tham khảo:* 27](#_Toc150977892)

# 

[*Hình 1: Confusion K – Nearest Neighbors……………………………………...12*](#_Toc150977902)

[*Hình 2: ROC K – Nearest Neighbors 13*](#_Toc150977903)

[*Hình 3: Confusion Matrix Random Forest 14*](#_Toc150977904)

[*Hình 4: ROC Random Forest 15*](#_Toc150977905)

[*Hình 5:Confusion Matrix Decision Tree 16*](#_Toc150977906)

[*Hình 6: ROC Decision Tree 17*](#_Toc150977907)

[*Hình 7:Confusion Matrix Logistic Regression 18*](#_Toc150977908)

[*Hình 8: ROC Logistic Regression 19*](#_Toc150977909)

[*Hình 9:Confusion Matrix Gaussian Naive Bayes 20*](#_Toc150977910)

[*Hình 10:ROC Gaussian Naive Bayes 21*](#_Toc150977911)

[*Hình 11: Training and Validation loss 22*](#_Toc150977912)

[*Hình 12: Training and Validation accuracy 23*](#_Toc150977913)

[*Hình 13: Confusion Matrix Neural Network 24*](#_Toc150977914)

[*Hình 14:ROC Neural Network 25*](#_Toc150977915)

# Phần 1: Giới thiệu tổng quan.

## 1. Giới thiệu về đề tài.

Ngành ngân hàng là một trong những ngành kinh tế quan trọng nhất trên thế giới. Hoạt động cho vay của các ngân hàng đóng vai trò quan trọng trong việc thúc đẩy kinh tế và phát triển xã hội. Tuy nhiên, hoạt động cho vay của các ngân hàng thường gắn liền với rủi ro vỡ nợ của người vay. Khi người vay không trả được khoản vay, ngân hàng sẽ phải chịu tổn thất tài chính.

Để giảm thiểu rủi ro vỡ nợ của người vay, các ngân hàng đã sử dụng nhiều phương pháp khác nhau, trong đó có Machine Learning. Machine Learning là một công nghệ hiện đại có khả năng học hỏi từ dữ liệu trong quá khứ để dự đoán kết quả trong tương lai. Trong trường hợp này, Machine Learning có thể được sử dụng để phân tích dữ liệu lịch sử về người vay vay tín dụng để dự đoán khả năng vỡ nợ của người vay mới.

Trong báo cáo này, tôi sử dụng tập dữ liệu Loan Default Dataset từ Kaggle. Tập dữ liệu này bao gồm dữ liệu các thông tin như thu nhập, lịch sử tín dụng, mục đích vay, v.v.

*Link dataset: https://www.kaggle.com/datasets/yasserh/loan-default-dataset*

## ****2. Phương pháp thực hiện đề tài:****

Sử dụng các thuật toán Machine Learning như Logistic Regression, K-nearest neighbor, Neural Network,… để phân tích dữ liệu về thu nhập, lịch sử tín dụng, và mục đích vay,... Tiền xử lý dữ liệu được thực hiện, và tập dữ liệu được chia thành tập huấn luyện và tập kiểm tra để đánh giá mô hình.

# Phần 2: Cơ sở lí thuyết.

## 1. Giới thiệu về các thuật toán.

#### 1.1 Thuật toán K-nearest neighbor.

K-nearest neighbor (KNN) là một thuật toán phân loại dựa trên sự tương tự. Thuật toán này hoạt động bằng cách tìm ra K điểm dữ liệu gần nhất với điểm dữ liệu mới cần phân loại. Điểm dữ liệu mới sẽ được phân loại theo nhãn của K điểm dữ liệu gần nhất.

**Ý tưởng của KNN**

Giả sử ta có tập dữ liệu huấn luyện gồm N điểm dữ liệu, mỗi điểm dữ liệu có một nhãn. Tập dữ liệu kiểm tra gồm M điểm dữ liệu, mỗi điểm dữ liệu cần được phân loại.

Để phân loại một điểm dữ liệu trong tập dữ liệu kiểm tra, thuật toán KNN sẽ tìm ra K điểm dữ liệu gần nhất với điểm dữ liệu đó trong tập dữ liệu huấn luyện. K thường được chọn là một số nguyên nhỏ, ví dụ như 3, 5, hoặc 10.

Các điểm dữ liệu gần nhất với điểm dữ liệu cần phân loại được gọi là các hàng xóm gần nhất (nearest neighbors).

Điểm dữ liệu cần phân loại sẽ được phân loại theo nhãn của các hàng xóm gần nhất. Ví dụ, nếu có 5 hàng xóm gần nhất và 4 trong số đó có nhãn là "chó", thì điểm dữ liệu cần phân loại sẽ được phân loại là "chó".

**Ưu điểm:**

* KNN là một thuật toán đơn giản và dễ hiểu.
* KNN có thể được sử dụng cho các bài toán phân loại đa nhãn (multi-class classification).
* KNN có thể xử lý các dữ liệu không tuyến tính (non-linear data).

**Nhược điểm:**

* KNN có thể kém hiệu quả khi tập dữ liệu lớn.
* KNN có thể dễ bị ảnh hưởng bởi các điểm dữ liệu ngoại lệ (outliers).

#### 1.2 Thuật toán Random Forest Classifier

RandomForestClassifier là một mô hình phân loại thuộc vào họ mô hình "Random Forest" trong machine learning. Nó là một phần quan trọng của ensemble learning, nơi nhiều mô hình con được kết hợp để tạo ra một mô hình mạnh mẽ hơn. RandomForest sử dụng nhiều cây quyết định (decision trees) và kết hợp kết quả của chúng để đưa ra dự đoán chính xác.

**Ưu điểm:**

* **Độ chính xác cao**: Rừng ngẫu nhiên được biết đến với độ chính xác cao trên nhiều nhiệm vụ học máy.
* **Khả năng chống quá khớp**: Rừng ngẫu nhiên ít bị quá khớp hơn các thuật toán khác, chẳng hạn như hồi quy logistic.
* **Có thể xử lý dữ liệu phức tạp**: Rừng ngẫu nhiên có thể xử lý dữ liệu phức tạp, bao gồm dữ liệu có giá trị thiếu và ngoại lệ.
* **Có thể được song song hóa**: Rừng ngẫu nhiên có thể được đào tạo song song, điều này có thể giảm đáng kể thời gian đào tạo.

**Nhược điểm:**

* **Có thể tốn kém về mặt tính toán**: Đào tạo một rừng ngẫu nhiên có thể tốn kém về mặt tính toán, đặc biệt là đối với các tập dữ liệu lớn.
* **Có thể khó giải thích**: Rừng ngẫu nhiên có thể khó giải thích, vì mỗi cây quyết định đóng góp một phần nhỏ vào dự đoán cuối cùng. Sự phức tạp của mô hình có thể làm cho việc hiểu rõ quá trình quyết định trở nên khó khăn.

#### 1.3 Thuật toán Decision Tree

Thuật toán Decision Tree là một thuật toán học máy có thể được sử dụng cho cả các bài toán phân loại và hồi quy. Nó hoạt động bằng cách phân chia tập dữ liệu thành các phần nhỏ hơn, mỗi phần có các điều kiện đặc trưng. Quá trình này được thực hiện đệ quy cho đến khi mỗi phần hoặc "nút lá" chỉ chứa một lớp hoặc giá trị.

**Ưu điểm:**

* **Đơn giản và dễ hiểu**: Decision Tree là một thuật toán đơn giản và dễ hiểu. Điều này khiến nó trở thành một lựa chọn tốt cho các nhà phát triển mới bắt đầu với học máy.
* **Thích ứng với dữ liệu phức tạp:** Decision Tree có thể thích ứng với dữ liệu phức tạp, bao gồm dữ liệu không tuyến tính.
* **Có thể được sử dụng cho cả phân loại và hồi quy**: Decision Tree có thể được sử dụng cho cả các bài toán phân loại và hồi quy.

**Nhược điểm:**

* **Có thể bị quá khớp**: Decision Tree có thể bị quá khớp với dữ liệu huấn luyện, điều này có thể dẫn đến dự đoán không chính xác trên dữ liệu mới.

#### 1.4 Thuật toán Logistis Regression

Thuật toán Logistic Regression là một thuật toán học máy được sử dụng để dự đoán xác suất của một kết quả nhị phân (ví dụ: có hoặc không, đậu hoặc trượt, spam hoặc không spam) dựa trên một tập hợp các biến độc lập (tính năng). Nó là một thuật toán được sử dụng rộng rãi trong học máy và đặc biệt phù hợp cho các nhiệm vụ như phân loại nhị phân,

**Ưu điểm:**

* **Tính giải thích**: Các trọng số được gán cho mỗi tính năng trong logistic regression có thể được giải thích là tầm quan trọng tương đối của mỗi tính năng trong việc dự đoán kết quả. Điều này làm cho logistic regression trở thành một thuật toán dễ giải thích hơn các phương pháp khác, chẳng hạn như mạng nơ-ron.
* **Khả năng chống lại các ngoại lệ**: Logistic regression tương đối chống lại các ngoại lệ, có nghĩa là nó không bị ảnh hưởng quá nhiều bởi các giá trị cực đoan trong dữ liệu.
* **Hiệu quả**: Logistic regression là một thuật toán tương đối hiệu quả để đào tạo và sử dụng, khiến nó phù hợp với các tập dữ liệu lớn.

**Nhược điểm:**

* **Overfitting:** Logistic regression dễ bị overfitting, xảy ra khi mô hình học dữ liệu huấn luyện quá tốt và không tổng quát hóa tốt sang dữ liệu mới. Điều này có thể được giảm thiểu bằng các kỹ thuật như regularizer và cross-validation.
* **Các giả định:** Logistic regression giả định rằng các tính năng độc lập tuyến tính và mối quan hệ giữa các tính năng và kết quả là tuyến tính. Nếu các giả định này không được đáp ứng, mô hình có thể không hoạt động tốt.

#### 1.5 Thuật toán Gaussian Naive Bayes

Thuật toán Gaussian Naive Bayes là một thuật toán học máy phân loại được sử dụng để phân loại các điểm dữ liệu dựa trên các thuộc tính của chúng. Thuật toán này giả định rằng các thuộc tính của các điểm dữ liệu phân bố theo phân phối Gaussian.

**Ưu điểm:**

* **Đơn giản và dễ hiểu**: Gaussian Naive Bayes là một thuật toán đơn giản và dễ hiểu. Điều này khiến nó trở thành một lựa chọn tốt cho các nhà phát triển mới bắt đầu với học máy.
* **Nhanh chóng**: Gaussian Naive Bayes là một thuật toán nhanh chóng để huấn luyện và sử dụng. Điều này khiến nó phù hợp với các tập dữ liệu lớn.
* **Ổn định**: Gaussian Naive Bayes là một thuật toán ổn định, có nghĩa là nó ít bị ảnh hưởng bởi nhiễu trong dữ liệu.

**Nhược điểm:**

* **Giả định không thực tế**: Gaussian Naive Bayes giả định rằng các thuộc tính của các điểm dữ liệu phân bố theo phân phối Gaussian. Điều này không phải lúc nào cũng đúng, đặc biệt là đối với các dữ liệu không tuyến tính.
* **Không thể giải thích**: Gaussian Naive Bayes không thể giải thích cách thuật toán đưa ra dự đoán của mình. Điều này có thể khiến nó khó sử dụng trong các ứng dụng mà tính giải thích là quan trọng.

#### 1.6 Neural Network

Mạng nơ ron nhân tạo (ANN) hay còn gọi là mạng nơ ron (neural network) là một mô hình máy học dựa trên cấu trúc và chức năng của mạng lưới thần kinh sinh học trong não người. ANN được tạo thành từ các lớp nơ ron được kết nối với nhau, trong đó mỗi nơ ron thực hiện một phép tính đơn giản và truyền kết quả của phép tính đó cho các nơ ron khác trong mạng.

Phân loại mạng nơ ron (neural network classifier) là một loại thuật toán phân loại sử dụng ANN để phân loại dữ liệu. Các phân loại mạng nơ ron có thể được sử dụng để phân loại dữ liệu tuyến tính và không tuyến tính, và chúng có thể đạt được độ chính xác cao trên nhiều loại vấn đề phân loại.

**Ưu điểm:**

* **Độ chính xác cao**: Phân loại mạng nơ ron có thể đạt được độ chính xác cao trên nhiều loại vấn đề phân loại.
* **Khả năng học các mô hình phức tạp**: Phân loại mạng nơ ron có thể học các mô hình phức tạp trong dữ liệu, điều này làm cho chúng phù hợp cho các vấn đề như nhận dạng hình ảnh và xử lý ngôn ngữ tự nhiên.
* **Khả năng tổng quát hóa tốt**: Phân loại mạng nơ ron có thể tổng quát hóa tốt sang dữ liệu mới, điều này có nghĩa là chúng có thể hoạt động tốt trên các nhiệm vụ mà chúng chưa được đào tạo rõ ràng.

**Nhược điểm:**

* **Độ phức tạp tính toán**: Phân loại mạng nơ ron có thể tính toán nhiều để huấn luyện, đặc biệt là cho các tập dữ liệu lớn.
* **Thiếu khả năng giải thích**: Phân loại mạng nơ ron có thể khó giải thích, điều này có nghĩa là có thể khó hiểu cách chúng đưa ra dự đoán của mình.

## 2. Xử lí và chia tập dữ liệu.

#### 2.1 Mô tả dữ liệu:

Bộ dữ liệu này gồm có 34 cột và 148.670 dòng, bao gồm thông tin về các khoản vay.

**Thông tin các cột của dữ liệu:**

* ***ID***: Mã số duy nhất cho mỗi khoản vay.
* ***year***: Năm mà khoản vay được thực hiện.
* ***loan\_limit***: Giới hạn vay, là số tiền tối đa mà người vay có thể vay.
* ***Gender***: Giới tính của người vay.
* ***approv\_in\_adv***: Phê duyệt trước đó của khoản vay.
* ***loan\_type***: Loại khoản vay, chẳng hạn như thế chấp nhà ở, thế chấp kinh doanh, v.v.
* ***loan\_purpose***: Mục đích sử dụng khoản vay, chẳng hạn như mua nhà, xây nhà, kinh doanh, v.v.
* ***Credit\_Worthiness***: Độ tin cậy tín dụng của người vay.
* ***open\_credit***: Số tín dụng mở của người vay, là tổng số thẻ tín dụng và khoản vay khác mà người vay đang có.
* ***business\_or\_commercial***: Loại hình kinh doanh hoặc thương mại của người vay.
* ***loan\_amount***: Số tiền vay, là số tiền mà người vay đã nhận được.
* ***rate\_of\_interest***: Lãi suất của khoản vay, là tỷ lệ phần trăm của số tiền vay mà người vay phải trả mỗi năm.
* ***Interest\_rate\_spread***: Chênh lệch lãi suất, là sự khác biệt giữa lãi suất mà người vay phải trả và lãi suất mà ngân hàng đang kiếm được.
* ***Upfront\_charges***: Phí trước, là các khoản phí mà người vay phải trả cho khoản vay, chẳng hạn như phí thẩm định, phí dịch vụ, v.v.
* ***term***: Kỳ hạn vay, là số năm mà người vay phải trả khoản vay.
* ***Neg\_ammortization***: Phương thức trả nợ tiêu chuẩn, là phương thức trả nợ mà người vay trả cả gốc và lãi mỗi tháng.
* ***interest\_only***: Trả lãi suất trước, là phương thức trả nợ mà người vay chỉ trả lãi suất mỗi tháng trong một số năm đầu tiên của khoản vay.
* ***lump\_sum\_payment***: Thanh toán một lần, là khoản tiền mà người vay trả cho khoản vay ngay lập tức.
* ***property\_value***: Giá trị bất động sản, là giá trị của tài sản được thế chấp cho khoản vay.
* ***construction\_type***: Loại công trình xây dựng, là loại tài sản được thế chấp cho khoản vay.
* ***occupancy\_type***: Loại cư trú, là mục đích sử dụng của tài sản được thế chấp cho khoản vay.
* ***Secured\_by***: Tài sản bảo đảm, là tài sản mà người vay sử dụng để thế chấp cho khoản vay.
* ***total\_units***: Tổng số đơn vị, là số lượng căn hộ hoặc phòng trong tài sản được thế chấp cho khoản vay.
* ***income***: Thu nhập hàng tháng, là tổng thu nhập của người vay.
* ***credit\_type***: Loại tín dụng, là loại tài sản được thế chấp cho khoản vay.
* ***Credit\_Score***: Điểm tín dụng, là một số điểm từ 300 đến 850 được sử dụng để đánh giá độ tin cậy tín dụng của một người.
* ***co-applicant\_credit\_type***: Loại tín dụng của đồng ứng viên, là loại tài sản được thế chấp cho khoản vay.
* ***age***: Tuổi của người vay.
* ***submission\_of\_application***: Thời gian nộp đơn, là ngày mà người vay nộp đơn xin vay.
* ***LTV***: Loan-to-Value tỷ lệ, là tỷ lệ giữa giá trị tài sản được thế chấp và số tiền vay.
* ***Region***: Khu vực địa lý, là khu vực địa lý mà tài sản được thế chấp được nằm.
* ***Security\_Type***: Loại tài sản đảm bảo, là loại tài sản được thế chấp cho khoản vay.
* ***Status***: Trạng thái của khoản vay, là trạng thái hiện tại của khoản vay, chẳng hạn như có nợ hay không.
* ***dtir1***: Tỷ lệ nợ thu nhập hàng tháng, là tỷ lệ giữa tổng số tiền nợ và thu nhập hàng tháng của người vay.

#### 2.1 Xử lí dữ liệu.

**Xóa các cột dữ liệu không ảnh hưởng đến mô hình:**

df.drop(['ID','year','term','Interest\_rate\_spread',

     'property\_value','submission\_of\_application'],axis**=**1,inplace**=True**)

* ***ID:***  Mã số duy nhất cho mỗi khoản vay
* ***year:*** Năm mà khoản vay được thực hiện.
* ***term:*** Kỳ hạn vay
* ***Interest\_rate\_spread:*** Chênh lệch lãi suất
* ***property\_value:*** Giá trị bất động sản thế chấp
* ***submission\_of\_application:*** Thời gian nộp đơn

**Phát hiện và loại bỏ các giá trị ngoại lệ:**

**def** detect\_outliers(data):

    threshold **=** 3

**for** i **in** data.columns:

**if** data[i].dtype**==**'float64' **or** data[i].dtype**==**'int64':

            mean **=** np.mean(data[i])

            std **=** np.std(data[i])

            z\_scores **=** [(y **-** mean) **/** std **for** y **in** data[column]]

            data **=** data[np.abs(z\_scores) **<=** threshold]

**return** data

df **=** detect\_outliers(df)

df

* + threshold = 3: Giá trị ngưỡng được đặt để xác định giá trị ngoại lệ. Các giá trị có khoảng cách với trung bình lớn hơn threshold lần độ lệch chuẩn được coi là ngoại lệ.
  + Vòng lặp for i in data.columns: Lặp qua từng cột trong dữ liệu.
  + if data[i].dtype=='float64' or data[i].dtype=='int64':: Kiểm tra xem kiểu dữ liệu của cột có phải là float64 hoặc int64 hay không. Chỉ xử lý các cột có kiểu dữ liệu số.
  + mean = np.mean(data[i]) và std = np.std(data[i]): Tính giá trị trung bình và độ lệch chuẩn của cột dữ liệu hiện tại.
  + z\_scores = [(y - mean) / std for y in data[column]]: Tính toán z-score cho từng giá trị trong cột dữ liệu. Z-score được tính bằng cách lấy hiệu giữa giá trị và giá trị trung bình, sau đó chia cho độ lệch chuẩn.
  + data = data[np.abs(z\_scores) <= threshold]: Lọc dữ liệu và giữ lại các hàng có giá trị z-score không vượt quá ngưỡng threshold. Các hàng này không được coi là ngoại lệ.
  + return data: Trả về dữ liệu đã lọc, chỉ chứa các hàng không chứa giá trị ngoại lệ.

**Thay đổi các giá trị NaN/Null.**

**for** i **in** df.columns:

**if** df[i].dtype**==**'float64' **or** df[i].dtype**==**'int64':

        df[i].fillna(df[i].mean(),inplace**=True**)

**else**:

        df[i].fillna(df[i].mode()[0],inplace**=True**)

df

* for i in df.columns:: Vòng lặp này duyệt qua tất cả các cột trong DataFrame df.
* if df[i].dtype=='float64' or df[i].dtype=='int64'::
* Điều kiện này kiểm tra xem cột hiện tại có kiểu dữ liệu là float64 hoặc int64 hay không. Chỉ các cột có kiểu dữ liệu số mới được xem xét để điền giá trị thiếu.
* df[i].fillna(df[i].mean(), inplace=True): Nếu cột hiện tại là kiểu dữ liệu số, thì giá trị thiếu trong cột đó sẽ được điền bằng giá trị trung bình của cột.
* Nếu cột hiện tại không phải là kiểu dữ liệu số, tức là kiểu dữ liệu khác, thì giá trị thiếu trong cột đó sẽ được điền bằng giá trị mode (giá trị xuất hiện nhiều nhất) của cột.
* df[i].mode()[0] .fillna(df[i].mode()[0]) được sử dụng để điền các giá trị thiếu bằng giá trị mode.

**Chuyển đổi các dữ liệu phân loại sang dạng số.**

**from** sklearn.preprocessing **import** LabelEncoder

categorical\_cols **=**df.select\_dtypes(include**=**['object']).columns

le **=** LabelEncoder()

df[categorical\_cols] **=** df[categorical\_cols].apply(**lambda** col: le.fit\_transform(col))

**Phân chia tập dữ liệu Train và Test.**

Phân chia tập dữ liệu thành tập huấn luyện (training set) và tập kiểm tra (test set) theo tỷ lệ 80-20

X**=**df.drop('Status',axis**=**'columns')

Y**=**df.Status

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X, Y, test\_size**=**0.2, random\_state**=True**)

# Phần 3: Đánh giá kết quả mô hình đạt được

## 1. Thuật toán K-nearest neighbor.

#### 1.1 Cài đặt thuật toán

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.model\_selection **import** GridSearchCV

param\_grid **=** {'n\_neighbors': [3, 5, 7, 9, 11]}

knn **=** KNeighborsClassifier()

grid\_search **=** GridSearchCV(knn, param\_grid, cv**=**5, scoring**=**'accuracy')

grid\_search.fit(X\_train, y\_train)

best\_k **=** grid\_search.best\_params\_['n\_neighbors']

print(**f**"Best value for k: {best\_k}")

y\_pred\_knn **=** grid\_search.predict(X\_test)

#### 1.2 Độ chính xác của thuật toán



**Accuracy (Độ chính xác):** Với một tỷ lệ chính xác là 92.87%, mô hình **K-Nearest Neighbors** đạt được mức độ chính xác tương đối cao trong việc dự đoán đúng nhãn cho các mẫu dữ liệu. Điều này cho thấy mô hình có khả năng phân loại đúng hơn 90% các điểm dữ liệu vào các nhãn tương ứng của chúng.

**F1-score**: Với một **f1-score** là 86.54%, mô hình **K-Nearest Neighbors** có khả năng phân loại dữ liệu tương đối tốt. **F1-score** là một phép đo kết hợp giữa độ chính xác (*precision*) và độ phủ (*recall*) và thường được sử dụng để đánh giá hiệu suất của mô hình phân loại. Một **f1-score** 86.54% cho thấy mô hình có khả năng đạt được cân bằng giữa độ chính xác và độ phủ trong quá trình phân loại.

#### 1.3 Confusion Matrix

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động

Hình 1: Confusion K – Nearest Neighbors

Nhìn chung, sơ đồ ma trận nhầm lẫn này cho thấy rằng mô hình dự đoán khá chính xác. Tuy nhiên, vẫn có một số trường hợp mà mô hình dự đoán sai.

#### 1.4 Receiver Operating Characteristic

Ảnh có chứa văn bản, hàng, ảnh chụp màn hình, Sơ đồ

Mô tả được tạo tự động

Hình 2: ROC K – Nearest Neighbors

Giá trị **AUC** (Area Under the Curve) là một thước đo hiệu suất của mô hình phân loại. Giá trị **AUC** càng cao, mô hình càng chính xác. Trong trường hợp này, giá trị **AUC** là 0,93. Đây là một giá trị **AUC** khá cao, cho thấy mô hình khá chính xác.

## 2. Thuật toán Random Forest Classifier

#### 2.1 Cài đặt thuật toán

**from** sklearn.ensemble **import** RandomForestClassifier

*# Xây dựng mô hình Random Forests*

rf\_model **=** RandomForestClassifier(n\_estimators**=**100, random\_state**=**42)

*# Huấn luyện mô hình trên tập huấn luyện*

rf\_model.fit(X\_train, y\_train)

*# Dự đoán nhãn cho tập kiểm tra*

y\_pred\_rf **=** rf\_model.predict(X\_test)

#### 2.2 Độ chính xác của thuật toán



Độ chính xác cao (99.99%) cho thấy mô hình có thể dự đoán chính xác 9999 mẫu trong số 10000 mẫu. Đây là một kết quả rất ấn tượng, cho thấy mô hình có khả năng phân loại tốt.

F1 score cao (99.99%) cho thấy mô hình có khả năng phân loại chính xác cả các mẫu dương tính và âm tính. Đây cũng là một kết quả rất ấn tượng, cho thấy mô hình có khả năng ứng dụng thực tế cao.

#### 2.3 Confusion Matrix

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động

Hình 3: Confusion Matrix Random Forest

Mô hình có khả năng dự đoán chính xác các mẫu dương tính, nhưng có sai sót trong dự đoán các mẫu âm tính.

#### 2.4 Receiver Operating Charactristic

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Sơ đồ

Mô tả được tạo tự động

Hình 4: ROC Random Forest

AUC = 1 là một kết quả rất tốt. Điều này cho thấy rằng mô hình có khả năng phân loại hoàn hảo và có thể được sử dụng để dự đoán các mẫu dương tính và âm tính với độ chính xác cao.

## 3. Thuật toán Decision Tree

#### 3.1 Cài đặt thuật toán

**from** sklearn.tree **import** DecisionTreeClassifier

model **=** DecisionTreeClassifier()

model.fit(X\_train, y\_train)

y\_pred\_dt **=** model.predict(X\_test)

dt\_accuracy **=** accuracy\_score(y\_test, y\_pred\_dt)

#### 3.2 Độ chính xác của thuật toán

Ảnh có chứa văn bản, Phông chữ, màu trắng, thuật in máy

Mô tả được tạo tự động

Độ chính xác cao (99.96%) cho thấy mô hình có thể dự đoán chính xác 9996 mẫu trong số 10000 mẫu. Đây là một kết quả rất ấn tượng, cho thấy mô hình có khả năng phân loại tốt.

F1 score cao (99.92%) cho thấy mô hình có khả năng phân loại chính xác cả các mẫu dương tính và âm tính. Đây cũng là một kết quả rất ấn tượng, cho thấy mô hình có khả năng ứng dụng thực tế cao.

#### 3.3 Confusion Matrix

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động

Hình 5:Confusion Matrix Decision Tree

Nhìn chung, sơ đồ này thể hiện rằng mô hình có thể dự đoán khá chính xác các mẫu dương tính và mô hình có thể dự đoán sai một số mẫu âm tính.

#### 3.4 Receiver Operating Characteristic

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Sơ đồ

Mô tả được tạo tự động

Hình 6: ROC Decision Tree

AUC = 1 là một kết quả rất tốt. Điều này cho thấy rằng mô hình có khả năng phân loại hoàn hảo và có thể được sử dụng để dự đoán các mẫu dương tính và âm tính với độ chính xác cao.

## 4. Thuật toán Logistis Regression

#### 4.1 Cài đặt thuật toán

**from** sklearn.linear\_model **import** LogisticRegression

*# Xây dựng mô hình Logistic Regression*

model **=** LogisticRegression()

*# Huấn luyện mô hình trên dữ liệu huấn luyện*

model.fit(X\_train, y\_train)

*# Dự đoán nhãn cho dữ liệu kiểm tra*

y\_pred\_lr **=** model.predict(X\_test)

#### 4.2 Độ chính xác của thuật toán

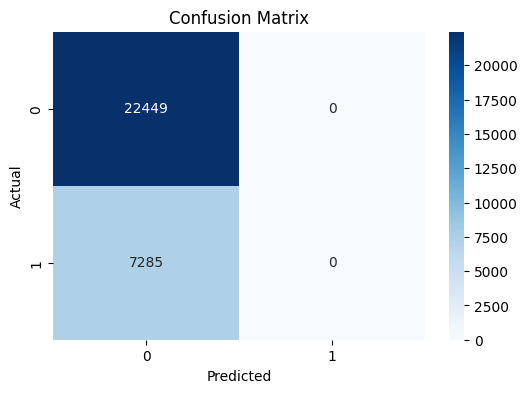
Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

Giá trị độ chính xác (accuracy) của mô hình Logistic Regression là 75.50%. Điều này cho thấy mô hình đã phân loại đúng khoảng 75.50% các điểm dữ liệu vào các nhãn tương ứng.

Tuy nhiên, giá trị f1-score của mô hình Logistic Regression là 0.00%. F1-score là một phép đo kết hợp giữa độ chính xác (precision) và độ phủ (recall). Một giá trị f1-score là 0.00% cho thấy mô hình không đạt được hiệu suất tốt trong việc phân loại dữ liệu. Có thể có một số vấn đề nghiêm trọng với mô hình, như việc dự đoán tất cả các mẫu đều thuộc vào một nhãn duy nhất hoặc mô hình không đủ mạnh để phân loại các mẫu đúng.

#### 4.3 Confusion Matrix



Hình 7:Confusion Matrix Logistic Regression

Tỷ lệ phân loại chính xác của mô hình là 75,7% là khá tốt. Tuy nhiên, nó có thể được cải thiện bằng cách giảm tỷ lệ dương tính giả.

Tỷ lệ dương tính giả là 41,4% là khá cao. Điều này có nghĩa là mô hình có thể dự đoán sai các mẫu dương tính là âm tính. Điều này có thể dẫn đến việc bỏ sót các trường hợp dương tính thực sự.

Tỷ lệ âm tính giả là 9,3% là khá thấp. Điều này có nghĩa là mô hình có thể dự đoán sai các mẫu âm tính là dương tính. Điều này có thể dẫn đến việc báo cáo sai các trường hợp dương tính.

Nhìn chung, ma trận nhầm lẫn này cho thấy rằng mô hình phân loại có thể được cải thiện bằng cách giảm tỷ lệ dương tính giả. Điều này có thể được thực hiện bằng cách điều chỉnh các tham số của mô hình hoặc bằng cách sử dụng dữ liệu đào tạo tốt hơn.

#### 4.4 Receiver operating Charateristic

Ảnh có chứa văn bản, hàng, ảnh chụp màn hình, Sơ đồ

Mô tả được tạo tự động

Hình 8: ROC Logistic Regression

Với giá trị **AUC** là 0.5, mô hình **Logistic Regression** không có khả năng phân loại tốt hơn một dự đoán ngẫu nhiên. Điều này cho thấy mô hình có thể không học được các mẫu và không có sự tương quan giữa dữ liệu đầu vào và đầu ra.

## 5. Thuật toán Gaussian Naive Bayes

#### 5.1 Cài đặt thuật toán

**from** sklearn.naive\_bayes **import** GaussianNB

gnb **=** GaussianNB()

gnb.fit(X\_train, y\_train)

y\_pred\_gnb**=**gnb.predict(X\_test)

#### 5.2 Độ chính xác của thuật toán



**Giá trị độ chính xác (accuracy)** của mô hình Gaussian Naive Bayes là 90.87%. Điều này cho thấy mô hình đã phân loại đúng khoảng 90.87% các điểm dữ liệu vào các nhãn tương ứng của chúng.

**Giá trị f1-score** của mô hình Gaussian Naive Bayes là 84.21%. F1-score là một phép đo kết hợp giữa độ chính xác (precision) và độ phủ (recall). Một giá trị f1-score là 84.21% cho thấy mô hình có khả năng phân loại tốt và đạt được mức độ cân bằng giữa độ chính xác và độ phủ trong quá trình phân loại.

Với giá trị **accuracy** 90.87% và **f1-score** 84.21%, mô hình **Gaussian Naive Bayes** cho thấy có hiệu suất tương đối tốt trong việc phân loại dữ liệu.

#### 5.3 Confusion Matrix

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động

Hình 9:Confusion Matrix Gaussian Naive Bayes

Nhìn chung, ma trận nhầm lẫn cho thấy mô hình đang hoạt động khá tốt. Phần lớn các mẫu được dự đoán chính xác. Tuy nhiên, vẫn có một số lỗi, đặc biệt là đối với các mẫu thuộc lớp 0.

#### 5.4 Receiver Operating Characteristic

Ảnh có chứa văn bản, hàng, ảnh chụp màn hình, Sơ đồ

Mô tả được tạo tự động

Hình 10:ROC Gaussian Naive Bayes

Giá trị AUC là 0,94. Đây là một giá trị AUC khá cao, cho thấy mô hình khá chính xác.

## 6. Neural Network

#### 6.1 Cài đặt thuật toán

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.callbacks **import** EarlyStopping

**from** keras.optimizers **import** RMSprop

**from** keras.optimizers **import** Adam

model **=** Sequential()

model.add(Dense(64, input\_shape**=**(X\_train.shape[1],), activation**=**'relu'))

model.add(Dense(64, activation**=**'relu'))

model.add(Dense(1, activation**=**'sigmoid'))

model.summary()

optimizers **=** Adam(learning\_rate**=**0.001)

model.compile(optimizer**=**optimizers,

              loss**=**'binary\_crossentropy',

              metrics**=**['accuracy'])

es **=** EarlyStopping(monitor**=**'val\_accuracy',

                                   mode**=**'max',

                                   patience**=**20,

                                   restore\_best\_weights**=True**)

history **=** model.fit(X\_train,

                    y\_train,

                    callbacks**=**[es],

                    epochs**=**100,

                    batch\_size**=**20,

                    validation\_split**=**0.3,

                    shuffle**=True**,

                    verbose**=**1)

#### 6.2 Training and validation loss

Ảnh có chứa văn bản, ảnh chụp màn hình, Sơ đồ, biểu đồ

Mô tả được tạo tự động

Hình 11: Training and Validation loss

Trong quá trình huấn luyện, loss của dữ liệu huấn luyện (training loss) giảm dần theo thời gian từ 233.8395 xuống 0.5584 sau epoch thứ 12. Điều này cho thấy mô hình đang học từ dữ liệu huấn luyện và cải thiện độ chính xác dự đoán trên dữ liệu huấn luyện.

Tuy nhiên, loss của dữ liệu đánh giá (validation loss) giảm mạnh từ 81.6800 ở epoch đầu tiên xuống 0.5597 sau epoch thứ 19, và sau đó ổn định ở mức này. Điều này cho thấy mô hình cũng có khả năng tổng quát hóa tốt trên dữ liệu đánh giá và không bị overfitting.

Giá trị training loss cuối cùng là 0.5584, và giá trị validation loss cuối cùng là 0.5597. Điều này cho thấy mô hình đã hội tụ tốt và không có sự chênh lệch lớn giữa training loss và validation loss, cho thấy mô hình không bị overfitting hoặc underfitting nghiêm trọng.

#### 6.3 Training and validation Accyracy

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Sơ đồ

Mô tả được tạo tự động

Hình 12: Training and Validation accuracy

**Training accuracy** tăng từ khoảng 0,6452 ở epoch đầu tiên lên đến khoảng 0,7536 và sau đó giữ ổn định. Điều này cho thấy mô hình đã học từ dữ liệu huấn luyện và có khả năng phân loại tốt hơn theo thời gian.

**Validation accuracy** ban đầu là khoảng 0,7484 ở epoch đầu tiên và tăng lên đến đỉnh ở khoảng 0,8056 sau epoch thứ 4. Điều này cho thấy mô hình có khả năng tổng quát hóa tốt trên dữ liệu đánh giá ban đầu.

Tuy nhiên, sau epoch thứ 4, độ chính xác đánh giá bắt đầu giảm dần và ổn định ở mức khoảng 0,7524 từ epoch thứ 10 trở đi. Điều này có thể cho thấy mô hình không có khả năng tổng quát hóa tốt hơn trên dữ liệu đánh giá sau một số epoch nhất định.

#### 6.4 Độ chính xác của thuật toán

Ảnh có chứa văn bản, Phông chữ, màu trắng, ảnh chụp màn hình

Mô tả được tạo tự động

**Accuracy** (Độ chính xác): Với một tỷ lệ chính xác 80.56%, mạng neural đang đạt được mức độ chính xác tương đối cao trong việc dự đoán đúng nhãn cho các mẫu dữ liệu. Tuy nhiên, không nói rõ được độ chính xác này áp dụng cho bài toán cụ thể nào, vì vậy không thể đánh giá được mức độ tốt hay kém của nó mà không có thông tin bổ sung.

**F1-score**: Với một f1-score là 62.04%, mạng neural có khả năng phân loại dữ liệu tương đối tốt. F1-score là một phép đo kết hợp giữa độ chính xác *(precision*) và độ phủ (*recall*) và nó thường được sử dụng để đánh giá hiệu suất của mô hình phân loại. Một f1-score 62.04% cho thấy mạng neural đạt được mức độ cân bằng giữa độ chính xác và độ phủ trong quá trình phân loại.

#### 6.5 Confusion Matrix

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động

Hình 13: Confusion Matrix Neural Network

Nhìn chung, ma trận nhầm lẫn cho thấy mô hình đang hoạt động khá tốt. Phần lớn các mẫu được dự đoán chính xác. Tuy nhiên, vẫn có một số lỗi, đặc biệt là đối với các mẫu thuộc lớp 0.

#### 6.6 Receiver Operating Characteristic

Ảnh có chứa văn bản, hàng, ảnh chụp màn hình, Sơ đồ

Mô tả được tạo tự động

Hình 14:ROC Neural Network

Với giá trị **AUC** là 0.87, mạng neural đạt được một hiệu suất tương đối tốt trong việc phân loại các mẫu dữ liệu. Điều này cho thấy mô hình có khả năng tốt hơn so với mô hình ngẫu nhiên trong việc phân loại các mẫu dữ liệu.

# Phần 4. Kết luận.

## 1. Những điểm đạt được.

Từ tập dữ liệu Loan Default Dataset trên Kaggle là một tập dữ liệu liên quan đến quá trình vay và những trường hợp không thể hoặc không muốn trả nợ. Tập dữ liệu đã được xử lí và chuẩn hoá để tạo ra một bộ dữ liệu chuẩn cho quá trình thực hiện các thuật toán Merchine learning, để có thể được sử dụng để phân tích dữ liệu lịch sử về người vay vay tín dụng để dự đoán khả năng vỡ nợ của người vay mới.

Tiếp theo, huấn luyện bộ dữ liệu theo mô hình theo 6 phương pháp: K – Nearest Neighbors, Random Forest, Decision Tree, Logistic Regression, Gaussian Naive Bayes, Neural Network. Và sử dụng giá trị *accuracy, f1-score, confusion matrix, receiver openrating charateristic* để đánh giá các mô hình. Kết quả cho được mô hình Random Forest cho kết quả tốt nhất với *accuracy = 99,99%, f1-score = 99,99%*.

## 2. Những hạn chế.

Hạn chế của mô hình Logistic Regression khi có giá trị F1-score bằng 0 là mô hình không thể phân loại đúng bất kỳ điểm dữ liệu nào. Điều này cho thấy mô hình không có khả năng phân loại chính xác các mẫu trong tập dữ liệu.

Đối với mô hình Neural Network, hạn chế khi mô hình chưa tối ưu là mô hình chưa học đủ từ dữ liệu và chưa tìm ra các trọng số tối ưu để phân loại chính xác. Mô hình Neural Network có nhiều lớp ẩn và số lượng các tham số, do đó quá trình tối ưu hóa mô hình có thể đòi hỏi thời gian và tài nguyên tính toán lớn.

Không thể thực hiện được SVM (Support Vector Machine) vì tốn nhiều thời gian với tập dữ liệu lớn và có số chiều nhiều.

## 3. Hướng phát triển.

Từ những mặt hạn chế ở trên chúng ta sẽ tiếp tục khắc phục những hạn chế vừa đề cập và nâng cao thêm độ chính xác của model. Ngoài ra sử dụng nhiều mô hình khác để tìm ra mô hình tối ưu nhất cho bộ dữ liệu trên.

# *Tài liệu tham khảo:*

<https://machinelearningcoban.com/2017/08/31/evaluation/#-accuracy>

<https://www.kaggle.com/datasets/yasserh/loan-default-dataset>

<https://www.analyticsvidhya.com/blog/2021/11/neural-network-for-classification-with-tensorflow/>

<https://phamdinhkhanh.github.io/2020/08/13/ModelMetric.html#6-trade-off-gi%E1%BB%AFa-precision-v%C3%A0-recall>