

**ĐẠI HỌC UEH  
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ  
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH  
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ  
XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**ĐỀ TÀI: XÂY DỰNG MÔ HÌNH DỰ BÁO KHẢ NĂNG  
VỖ NỢ CÁC KHOẢN VAY CỦA KHÁCH HÀNG TẠI  
LENDING CLUB**

**Thành viên**

Nguyễn Hoàng Nhật Hồng Nguyên - 31201024493

Võ Ngọc Dung - 31201020182

Đoàn Anh Thư - 31201024519

Lý Trần Thiên Kim - 31201024493

**GVHD: ĐẶNG NGỌC HOÀNG THÀNH**

*TP Hồ Chí Minh, tháng 12 năm 2022*

## MỤC LỤC

<b>LỜI MỞ ĐẦU</b>	<b>4</b>
<b>CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI</b>	<b>5</b>
1. Giới thiệu bài toán	5
2. Lý do chọn bài toán	5
3. Giới hạn phạm vi đề tài	5
4. Phương pháp nghiên cứu	6
5. Ngôn ngữ sử dụng	6
<b>CHƯƠNG 2. TỔNG QUAN BỘ DỮ LIỆU NGHIÊN CỨU</b>	<b>7</b>
1. Tổng quan bộ dữ liệu thu thập	7
1.1. Giới thiệu bộ dữ liệu	7
1.2. Các thuộc tính của bộ dữ liệu	7
2. Các phương pháp tiền xử lý dữ liệu	8
2.1. Kiểm tra tình trạng dữ liệu gốc	8
2.2 Xử lý dữ liệu đầu vào	9
a. Làm sạch dữ liệu 1	9
b. Làm sạch dữ liệu 2	13
c. Xác định Missing Values	18
d. Xác định biến Target	22
e. Trực quan hóa kết quả dữ liệu với biến Target và kiểm tra mô hình tương quan	23
f. Xử lý Missing value	28
3. Xử lý các biến phân loại Categorical Variables	29
4. Xử lý dữ liệu với NLP	34
a. Tiền xử lý	34
b. Train test split	36
c. Vecto hóa	36
<b>CHƯƠNG 3. PHÂN LỚP</b>	<b>37</b>
1. Định nghĩa phân lớp dữ liệu	37

<b>2. Xây dựng tập huấn luyện 20% bộ dữ liệu</b>	<b>37</b>
<b>3. Xây dựng mô hình</b>	<b>39</b>
3.1. Mô hình Naive Bayes	39
3.2. Mô hình Decision Tree	39
<b>4. Sử dụng ma trận nhầm lẫn đánh giá mô hình xây dựng</b>	<b>39</b>
4.1. Mô hình Naive Bayes	39
4.2. Mô hình Decision Tree	42
<b>5. Lựa chọn mô hình sử dụng</b>	<b>45</b>
<b>CHƯƠNG 4. TỔNG KẾT</b>	<b>46</b>
1. Kết quả đạt được	46
2. Tóm tắt vấn đề nghiên cứu	46
3. Cải tiến trong tương lai	46
<b><i>Tài liệu tham khảo</i></b>	<b>47</b>

## LỜI MỞ ĐẦU

Xuất phát từ thực trạng hiện nay đó là tín dụng ngân hàng - hình thức tín dụng được đánh giá là có nhiều ưu điểm nhất, do nhiều nguyên nhân vẫn chưa phát huy được hết vai trò của mình trong hoạt động cấp vốn cho một số chủ thể quan trọng trong nền kinh tế, một hình thức tín dụng mới đã ra đời nhằm đáp ứng nhu cầu vay vốn đa dạng của các cá nhân và các doanh nghiệp vừa và nhỏ, đồng thời đem đến một hình thức tín dụng mới đơn giản, hiệu quả. Đó là cho vay ngang hàng (P2P Lending).

Theo đó, doanh nghiệp cho vay ngang hàng cung cấp nền tảng giao dịch trực tuyến để người vay kết nối trực tiếp vay mượn với người cho vay. Toàn bộ hoạt động vay, trả nợ (gốc, lãi) giữa người vay và người cho vay được nền tảng giao dịch trực tuyến ghi nhận và lưu trữ bằng các bảng điện tử, số hóa. Cho vay ngang hàng được thực hiện dựa trên nền tảng công nghệ thông tin, những doanh nghiệp hoạt động theo loại hình này sử dụng công nghệ Big Data để thu thập tất cả dữ liệu của cả hai phía người cho vay và cho vay. Cách một nền tảng cho vay ngang hàng kiếm được doanh thu phụ thuộc phần lớn vào mô hình kinh doanh của nền tảng và cách cấu trúc khoản đầu tư, điều này sẽ có tác động lớn đến cấu trúc rủi ro mà các nhà đầu tư phải đối mặt.

Bài báo cáo ***“Xây dựng mô hình dự báo khả năng vỡ nợ các khoản vay của khách hàng tại LendingClub”*** trình bày kết quả nghiên cứu hai mô hình Naive Bayes và Decision Tree trên cơ sở dữ liệu khách hàng vay vốn tại Lendingclub (LC), một tổ chức cho vay ngang hàng lớn nhất trên thế giới hiện nay, nhằm mục đích xây dựng một mô hình đo lường rủi ro vỡ nợ các khoản vay tiềm ẩn của khách hàng.

Do điều kiện về mặt thời gian và nhận thức còn nhiều hạn chế nên đồ án không khỏi có phần sai sót, kính mong quý giảng viên nhiệt tình góp ý để nhóm chúng tôi có những tiếp thu tốt hơn trong môn học này. Qua bài đồ án, nhóm chúng tôi xin cảm ơn giảng viên phụ trách môn học Xử lý ngôn ngữ tự nhiên, thầy Đặng Ngọc Hoàng Thành đã truyền đạt những kiến thức quý báu cho học viên bằng cả tấm lòng nhiệt tình và sự tận tâm của thầy.

## CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

### 1. Giới thiệu bài toán

Trên LendingClub người đi vay chỉ cần điền thông tin vào đơn và nộp trực tuyến. Hệ thống của LC sẽ tự động phân tích dữ liệu, đánh giá rủi ro, chấm điểm tín dụng và đưa ra mức lãi suất phù hợp. Sau đó, công ty kết nối người đi vay với nhà đầu tư để nhà đầu tư lựa chọn khoản đầu tư thích hợp dựa trên những thông tin về người đi vay như chấm điểm và xếp hạng tín dụng, mục đích vay tiền, lịch sử tín dụng,... Đây là hình thức cho vay mà người có tiền và người cần tiền được kết nối trực tiếp với nhau mà không cần thông qua ngân hàng.

Bài báo cáo sử dụng bộ dữ liệu khách hàng cá nhân trong hai năm 2014 và 2015 được đăng tải trên website [www.lendingclub.com](http://www.lendingclub.com) để mô tả các bước xây dựng mô hình đánh giá khả năng vỡ nợ các khoản vay của khách hàng.

### 2. Lý do chọn bài toán

Việc đánh giá rủi ro tín dụng, phân loại và xếp hạng tín dụng các khách hàng là khâu quan trọng và tất yếu trước khi tiến hành hoạt động cho vay. Vấn đề đặt ra là cần có một mô hình đo lường rủi ro vỡ nợ của khách hàng có đủ tin cậy hay không và khả năng dự báo chính xác, nghĩa là cần có một mô hình cần được kiểm chứng thực nghiệm trên bộ dữ liệu tại thị trường cụ thể (Lendingclub).

Với sự phát triển của trí tuệ nhân tạo và máy móc đã có nhiều nghiên cứu sử dụng các phương pháp ước lượng để lựa chọn mô hình phù hợp như Logistic, SVM, Decision Tree, Random Forest,.. Để phát triển mô hình dự đoán rủi ro. Những mô hình này không chỉ mang lại lợi ích cho các tổ chức tài chính mà còn cả khách hàng vì nó sẽ giúp họ nhận thức được rõ hơn về khả năng vỡ nợ tiềm ẩn của mình. Với sự tò mò và mong muốn hiểu sâu hơn về mô hình cũng như cách thức hoạt động của nó, nhóm chúng em quyết định chọn bộ dữ liệu “Lending Club” để làm báo cáo với đề tài “***Xây dựng mô hình dự báo khả năng vỡ nợ các khoản vay của khách hàng tại LendingClub***” làm đồ án cuối kỳ môn học Xử lý ngôn ngữ tự nhiên

### 3. Giới hạn phạm vi đề tài

Trong bài báo cáo này chủ yếu là chạy phân tích dữ liệu để tìm hiểu thông tin chi tiết về khách hàng từ dữ liệu khoản vay, sử dụng hai mô hình máy học là Decision Tree và Naive Bayes để tìm hiểu và giúp xác định các khoản vay hay liệu người đi vay này có khả năng vỡ nợ hay không vỡ nợ. Theo các nghiên cứu gần đây, hằng năm có 3-4% các khoản vay bị vỡ nợ. Đây là một rủi ro rất lớn đối với các nhà đầu tư tài trợ cho các khoản vay. Từ đó, các nhà đầu tư yêu cầu

phải đánh giá toàn diện hơn về những người đi vay này so với những dữ liệu thu thập được từ Lending Club để đưa ra quyết định kinh doanh tốt hơn. Các kỹ thuật khai phá, phân tích dữ liệu và mô hình máy học có thể giúp họ dự đoán khả năng vỡ nợ khoản vay, điều này cho phép các nhà đầu tư tránh vỡ nợ, do đó hạn chế rủi ro cho các khoản đầu tư của họ.

#### **4. Phương pháp nghiên cứu**

Sử dụng phương pháp phân lớp áp dụng 2 mô hình Naive Bayes và Decision Tree để xác định các khoản vay có khả năng vỡ nợ hay không vỡ nợ.

#### **5. Ngôn ngữ sử dụng**

Ngôn ngữ lập trình Python

## CHƯƠNG 2. TỔNG QUAN BỘ DỮ LIỆU NGHIÊN CỨU

### 1. Tổng quan bộ dữ liệu thu thập

#### 1.1. Giới thiệu bộ dữ liệu

Bộ dữ liệu Loan.csv được thu thập và phát hành trong giai đoạn 2007 - 2015, bao gồm tình trạng thanh toán hiện tại (Current, Late, Fully Paid,...) và thông tin thanh toán mới nhất. Các biến bổ sung bao gồm credit scores (điểm tín dụng), địa chỉ bao gồm zip codes, tiểu bang... Có 111 cột với thông tin đại diện cho các khoản vay cá nhân với 39.717 quan sát.

#### 1.2. Các thuộc tính của bộ dữ liệu

STT	Tên thuộc tính	Mô tả
1	ID	Mỗi id được chỉ định cho mỗi khách hàng
2	Member	Id mà LC chỉ định cho mỗi thành viên cho vay
3	Loan_amnt	Số tiền đi vay
4	Funded_amnt	Số tiền được tài trợ
5	Term	Thời hạn vay của khoản vay
6	Int_rate	Lãi suất cho vay
7	Installment	Khoản trả góp - khoản thanh toán hàng tháng của người đi vay nếu khoản vay bắt đầu
8	Grade	LC đánh giá về mức độ rủi ro đối với mỗi bộ hồ sơ vay dựa trên thông tin cá nhân và thông tin hồ sơ tín dụng của người vay, grade có các giá trị A, B, C, D,...
9	Sub_grade	Tương tự như biến grade, sub_grade có các giá trị A1, A2, ..., B1, B2,... càng gần A1 càng được đánh giá là tốt và lãi suất vay càng thấp.
10	Emp_title	Chức danh/công việc
11	Emp_length	1 chuỗi từ 0 đến 10 trong đó 0 có nghĩa là dưới 1 năm và 10 có nghĩa là từ mười năm trở lên
12	Home_ownership	Tình trạng sở hữu nhà được người đi vay cung cấp trong quá trình đăng ký (RENT: cho thuê, OWN: sở hữu, MORTGAGE: thế chấp, OTHER: khác)
13	Annual_inc	Thu nhập hàng năm của người vay

14	Verification_status	Trạng thái xác minh thu nhập của khách hàng
15	Loan_status	Trạng thái thanh toán (current: các khoản vay còn dư nợ nhưng thanh toán đầy đủ các khoản nợ hàng tháng, fully paid: khách hàng đã thanh toán đầy đủ khoản vay, charged off: các khoản vay quá hạn, nợ không thể hoàn trả (chậm quá 150 ngày)
16	Revol_until	Các khoản tín dụng của người vay được sử dụng liên quan đến tất cả các tín dụng quay vòng có sẵn
17	Desc	Mô tả khoản vay được cung cấp bởi người đi vay (string)
18	Issue_d	Tháng mà khoản vay được tài trợ
19	Pymnt_plan	Một kế hoạch thanh toán đã được đưa ra cho khoản vay
20	Addr_state	Tiểu bang được người đi vay cung cấp
21	Dti	% số thu nhập dành cho việc trả nợ hàng tháng
22	Earlist_cr-line	Số tháng kể từ khi mở tài khoản đầu tiên

2. Các phương pháp tiền xử lý dữ liệu

2.1. Kiểm tra tình trạng dữ liệu gốc

Dữ liệu được dùng cho báo cáo lần này là dữ liệu có cấu trúc với một biến có giá trị bị thiếu. Dữ liệu này bao gồm hơn 80 biến thuộc ba loại riêng biệt: liên tục, phân loại và thứ tự. Áp dụng một số kiến thức liên quan đến kinh doanh để xử lý việc làm sạch dữ liệu và xử lý dữ liệu bị thiếu.

Dữ liệu đầu vào quan sát từ Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	id	member_id	loan_amnt	funded_amnt	funded_amnt_1term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status	issue_d	loan_status	pymnt_plan	url	desc	
2	1077501	1296596	5000	5000	4975	36 months	10.65%	162.87	B2		10+ years	RENT	24000	Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
3	1077430	1314167	2500	2500	2500	60 months	15.27%	59.83	C4	Ryder	< 1 year	RENT	30000	Source Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
4	1077175	1313524	2400	2400	2400	36 months	15.96%	84.33	C5		10+ years	RENT	12252	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
5	1076863	1277178	10000	10000	10000	36 months	13.49%	339.31	C1	AIR RESOURCE	10+ years	RENT	48200	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
6	1075358	1311748	3000	3000	3000	60 months	12.69%	67.79	B	US University Med	1 year	RENT	80000	Source Verified	Dec-11	Current	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
7	1075269	1311441	5000	5000	5000	36 months	7.90%	156.46	A4	Veolia Transport	3 years	RENT	36000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
8	1069639	1304742	7000	7000	7000	60 months	15.96%	170.08	C5	Southern Star PI	8 years	RENT	47004	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
9	1072053	1286686	3000	3000	3000	36 months	18.64%	109.43	E1	MKC Accounting	9 years	RENT	48000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
10	1071795	1306957	5600	5600	5600	60 months	21.28%	152.39	F	F2	4 years	OWN	40000	Source Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
11	1071570	1306721	5375	5375	5350	60 months	12.69%	121.45	B	Starbucks	< 1 year	RENT	15000	Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
12	1070078	1305201	6500	6500	6500	60 months	14.65%	153.45	C	Southwest Rural	5 years	OWN	72000	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
13	1069908	1305008	12000	12000	12000	36 months	12.69%	402.54	B5	UCLA	10+ years	OWN	75000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
14	1064687	1298717	9000	9000	9000	36 months	13.49%	305.38	C1	Va. Dept of Core	< 1 year	RENT	30000	Source Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
15	1068666	1304956	3000	3000	3000	36 months	9.91%	96.68	B		3 years	RENT	15000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
16	1069057	1302503	10000	10000	10000	36 months	10.65%	325.74	B2	SFRATA	3 years	RENT	110E+06	Source Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
17	1069759	1304871	1000	1000	1000	36 months	16.29%	35.31	D	Internal revenue	< 1 year	RENT	28000	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
18	1065775	1299699	10000	10000	10000	36 months	15.27%	347.98	C	Chi's Restaurant	4 years	RENT	42000	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
19	1060971	1304884	3600	3600	3600	36 months	6.03%	109.57	A1	DuraCell	10+ years	MORTGAGE	110000	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
20	1062474	1294539	6000	6000	6000	36 months	11.71%	198.46	B	Connexion Insp	1 year	MORTGAGE	84000	Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
21	1069742	1304656	6200	6200	6200	36 months	6.03%	280.01	A1	Network Interop	8 years	RENT	77385	19 Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
22	1069740	1284848	20250	20250	19142	16108	60 months	15.27%	484.63	C	Archdiocese of	13 years	RENT	43370	Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added
23	1039153	1268093	21000	21000	21000	36 months	12.42%	701.73	B4	Oranm Sylvania	10+ years	RENT	105000	Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
24	1069710	1304821	10000	10000	10000	36 months	11.71%	330.76	B	Value Air	10+ years	OWN	50000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
25	1069700	1304810	10000	10000	10000	36 months	11.71%	330.76	B	Wells Fargo Ban	5 years	RENT	50000	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
26	1069559	1304634	6000	6000	6000	36 months	11.71%	198.46	B	hmg-educational	1 year	RENT	78000	Not Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
27	1069697	1273773	15000	15000	15000	36 months	9.91%	483.38	B	Winfield Patholo	2 years	MORTGAGE	92000	Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
28	1069800	1304679	15000	15000	8725	36 months	14.27%	514.64	C2	nyc transit	9 years	RENT	60000	Not Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
29	1069657	1304764	5000	5000	5000	60 months	16.77%	123.65	D	Frito Lay	2 years	RENT	50004	Not Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
30	1069799	1304678	4000	4000	4000	36 months	11.71%	132.31	B	Shands Hospital	10+ years	MORTGAGE	106000	Not Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
31	1047794	1278906	8500	8500	8500	36 months	11.71%	281.15	B	Oakridge homes	< 1 year	RENT	29000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
32	1032111	1261745	4375	4375	4375	36 months	7.51%	136.11	A	7 years	MORTGAGE	17108	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added		
33	1069539	1304608	31825	31825	31825	36 months	7.90%	995.82	A	Audubon Mutual	5 years	MORTGAGE	75000	Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
34	1065420	1299514	10000	10000	9975	60 months	15.96%	242.97	C	US Legal Suppo	2 years	RENT	29120	Verified	Dec-11	Current	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
35	1069991	1304299	5000	5000	5000	36 months	8.90%	158.77	A5	Good Samaritan	2 years	RENT	24644	Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
36	1069530	1291385	7000	7000	7000	36 months	15.96%	245.97	C	GREG BARRETT	7 years	RENT	34000	Source Verified	Dec-11	Fully Paid	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	
37	1068579	1304630	15000	15000	15000	36 months	10.65%	409.04	B	Chesam 1 man for	4th unknown	OWN	41000	Not Verified	Dec-11	Charged Off	n	<a href="https://lendingclub.com/browse/sale">https://lendingclub.com/browse/sale</a>	Borrower added	



## Nội dung bộ dữ liệu đầu vào

**TRUYỀN DỮ LIỆU**

```
start_df = pd.read_csv("/content/loan.csv", low_memory=False)
```

```
df = start_df.copy(deep=True)
df.head()
```

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_dpd_24m	num_tl_op_past_12m	pct_
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	NaN	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	NaN	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	NaN	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	NaN	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	NaN	NaN	

5 rows × 111 columns

```
[ ] print('Có {} hàng và {} cột trong tập dữ liệu.'.format(df.shape[0], df.shape[1]))
```

Có 39717 hàng và 111 cột trong tập dữ liệu.

## 2.2 Xử lý dữ liệu đầu vào

### a. Làm sạch dữ liệu 1

Các cột có hơn 40% missing values sẽ bị xóa, bước này để giải phóng dung lượng và giúp quá trình xử lý nhanh hơn. Các biến như "recoveries" và "collection\_recovery\_fee" là dữ liệu về tương lai của khoản vay. Các biến như "last\_pymnt\_d", "last\_pmynt\_amnt" mô tả ngày kết thúc trả nợ, không biết trước do thực tế khách hàng có thể trả hết khoản vay sớm hơn ban đầu.

**Loại bỏ các cột 'funded\_amnt', 'funded\_amnt\_inv', 'last\_pymnt\_amnt', 'delinq\_amnt'** vì "Biến last\_pymnt\_amnt mô tả ngày kết thúc trả nợ không thực tế do không thể biết được chính xác ngày khách hàng có thể hoàn trả hết khoản vay. Biến funded\_amnt và biến funded\_amnt\_inv đều là mối lo ngại về tương lai, cho dù khoản vay đã được phê duyệt tại thời điểm đó, do đó không được xem xét trong tập dữ liệu mà nhóm sẽ sử dụng để xây dựng mô hình. Tương tự với biến delinq\_amnt.

```
# Xóa các cột có dữ liệu không liên quan khoản vay để xây dựng mô hình dự đoán
df.drop(['funded_amnt', 'funded_amnt_inv', 'last_pymnt_amnt', 'delinq_amnt'],
axis=1, inplace=True)
```

**Loại bỏ các cột “grade”, “sub\_grade”, “int\_rate”** vì grade và sub\_grade đã bị xóa vì chúng truyền tải cùng ý nghĩa với biến int\_rate

```
# Xóa grade, sub_grade và int_rate
df.drop(['grade', 'sub_grade', 'int_rate'], axis=1, inplace=True)
```

**Tương tự với các cột như 'id', 'member\_id', 'issue\_d** vì Biến issue\_d là dữ liệu về tháng khi khoản vay được tài trợ, nó đại diện cho thông tin ở tương lai, do đó nó đã bị xóa vì thông tin đó sẽ không có sẵn. Biến issue\_d và member\_id đã bị xóa vì chúng không cung cấp bất kỳ thông tin nào hữu ích về khách hàng. Đây là những biến ngẫu nhiên được cung cấp bởi Lending Club.

```
drop_list = ['id', 'member_id', 'issue_d']
df = df.drop(drop_list, axis=1)
```

**Loại bỏ cột “emp\_title”** với lí do tương tự là không xây dựng mô hình nghiên cứu vì emp\_title là biến đã được cải tiến, tuy nhiên dữ liệu sẽ rất khó để đánh giá dựa trên biến này.

```
drop_list2 = ['emp_title']
df = df.drop(drop_list2, axis=1)
```

**Đối với cột “Title”, cột này chỉ dùng để mô tả chi tiết hơn việc vay mượn nợ của khách hàng, tức bổ sung chi tiết cho cột Purpose nên ta sẽ loại bỏ đi vì không cần sử dụng đến.**

```
drop_list5 = ['title']
df = df.drop(drop_list5, axis=1)
```

**Loại bỏ cột “Url”** vì không sử dụng đến

```
# Bỏ cột URL
drop_list3 = ['url']
df = df.drop(drop_list3, axis=1)
```

**Đối với biến “Purpose”** có tồn tại nhiều giá trị không đồng nhất, vì vậy ta cần chuẩn hoá để hợp nhất dữ liệu các biến có giá trị tương tự nhau. Sử dụng hàm df.loc để truy cập vào các dòng và cột của biến “Purpose” theo các nhãn cho trước.

- Tạo thêm 1 cột “Purpose\_n” dành cho các biến “Purpose” bằng nan
- Đối với các biến Purpose có giá trị bằng ‘debt\_consolidation’, ‘credit\_card’, ‘purpose\_n’, ta đồng bộ giá trị về “debt”

- Đối với các biến Purpose có giá trị bằng “home\_improvement”, “major\_purchase”, “car”, “house”, “vacation”, “renewable\_energy”, “purpose\_n”, ta đồng bộ giá trị về “major\_purchases”
- Đối với các biến Purpose có giá trị bằng “small\_business”, “medical”, “moving”, “wedding”, “educational”, “purpose\_n”, ta đồng bộ giá trị về “life\_event”

```
df['purpose_n'] = np.nan # tạo cột mới với các giá trị Nan
df.loc[(df['purpose'] == 'debt_consolidation') | (df['purpose'] == "credit_card"),
       'purpose_n'] = 'debt'
df.loc[(df['purpose'] == 'home_improvement') | (df['purpose']
== "major_purchase") |
       (df['purpose'] == 'car') | (df['purpose'] == "house") |
       (df['purpose'] == 'vacation') | (df['purpose']
== "renewable_energy"),
       'purpose_n'] = 'major_purchases'
df.loc[(df['purpose'] == 'small_business') | (df['purpose'] == "medical") |
       (df['purpose'] == 'moving') | (df['purpose'] == "wedding") |
       (df['purpose'] == 'educational'),
       'purpose_n'] = 'life_events'
df.loc[(df['purpose'] == 'other'), 'purpose_n'] = 'other'
```

### Quan sát cột “zip\_code”

```
df['zip_code'].value_counts()
# df['addr_state'].value_counts()
```

100xx	597
945xx	545
112xx	516
606xx	503
070xx	473
...	
381xx	1
378xx	1
739xx	1
396xx	1
469xx	1

Name: zip\_code, Length: 823, dtype: int64

### Quan sát cột “add\_state”

```
# df['zip_code'].value_counts()
df['addr_state'].value_counts()
```

CA	7099
NY	3812
FL	2866
TX	2727
NJ	1850
IL	1525

Cả hai cột đều cung cấp thông tin về địa chỉ của khách hàng, tuy nhiên cột `zip_code` không thêm bất kỳ giá trị nào vì nó đã tồn tại trong biến `addr_state`, nên nhóm sẽ chỉ giữ lại cột `addr_state`.

```
drop_list6 = ['zip_code']
df = df.drop(drop_list6,axis=1)
```

Cột `'earliest_cr_line'` mô tả ngày mức tín dụng đầu tiên được mở. Thông thường, một người giữ hạn mức tín dụng càng lâu càng mong đợi trở thành người cho vay. Biến này sẽ hữu ích hơn nếu thành thước đo thời gian của một người đã giữ hạn mức tín dụng

```
import datetime

# Tính toán thời gian từ khi mức tín dụng đầu tiên được mở
now = datetime.datetime.today()
def credit_age(x):
    if x != 'nan':
        c1 = datetime.datetime.strptime(x, '%b-%y')

        return (now-c1).days/365.25

    else:
        return None

df['earliest_cr_line_n'] = df['earliest_cr_line'].astype(str)
df['earliest_cr_line_n'] =
df['earliest_cr_line_n'].apply(credit_age)
```

Sau khi đã biến đổi cột này thành thước đo thời gian giữ hạn mức tín dụng, ta sẽ loại bỏ cột dữ liệu cũ

```
drop_list7 = ['earliest_cr_line']
df = df.drop(drop_list7,axis=1)
```

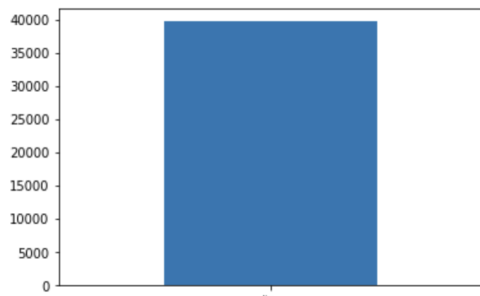
**b. Làm sạch dữ liệu 2**

*Tiếp tục quan sát các cột sau và xác định sẽ loại bỏ những cột nào không sử dụng trong mô hình*

STT	Tên cột	Ý nghĩa
1	open_acc	Số hạn mức tín dụng ghi chép trong hồ sơ người đi vay
2	pub_rec	Số hồ sơ vi phạm
3	revol_bal	Tổng số dư vòng quay tín dụng
4	mths_since_last_delinq	Số tháng kể từ lần cuối cùng người vay bị quá hạn
5	initial_list_status	Tình trạng niêm yết ban đầu của khoản vay
6	policy_code	Mã code chính sách
7	collections_12_mths_ex_med	Số lần thu nợ trong 12 tháng không bao gồm các khoản thu y tế
8	acc_now_delinq	Số lượng giao dịch được mở trong 24 tháng qua
9	chargeoff_within_12_mths	Số lần giảm trong 12 tháng
10	tax_clients	Thuế

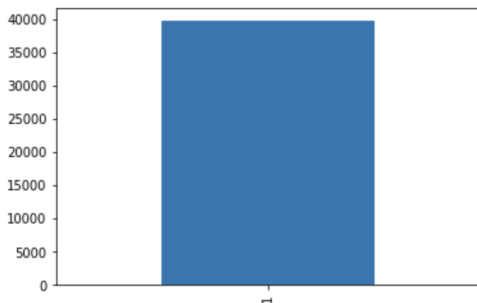
*Quan sát biểu đồ thấy biến “initial\_list\_status” chỉ nhận giá trị là f ở hầu hết các dòng*

```
df['initial_list_status'].value_counts().plot.bar()
plt.show()
```



### ***Tương ứng quan sát đối với cột “policy\_code” cũng chỉ nhận giá trị là 1***

```
df['policy_code'].value_counts().plot.bar()
plt.show()
```



Dùng hàm value\_counts() để xem số lượng giá trị đặc biệt của 2 biến trên, kết quả trả về cho thấy cả 2 đều chỉ tồn tại 1 giá trị.

```
print(df.policy_code.value_counts())
print(df.initial_list_status.value_counts())
```

```
1    39717
Name: policy_code, dtype: int64
1    39717
Name: initial_list_status, dtype: int64
```

***Các biến chỉ có một giá trị riêng biệt đã bị loại bỏ vì chúng không hữu ích cho việc xây dựng mô hình***

```
drop_list10=['policy_code', 'initial_list_status']
df = df.drop(drop_list10,axis=1)
```

### ***Đối với cột “collections\_12\_mths\_ex\_med”***

```
df['collections_12_mths_ex_med'].value_counts()
```

```
0.0    39661
Name: collections_12_mths_ex_med, dtype: int64
```

Ta sẽ loại bỏ cột “collections\_12\_mths\_ex\_med” với lí do tương tự là chỉ nhận giá trị 0

```
df = df.drop(['collections_12_mths_ex_med'],axis=1)
```

### ***Đối với cột “acc\_now\_delinq”***

```
[ ] df['acc_now_delinq'].value_counts()
```

```
0    39717
Name: acc_now_delinq, dtype: int64
```

Dùng `value_counts()` để xem các giá trị đặc biệt của cột dữ liệu “acc\_now\_delinq”, ta thấy chỉ có kết quả là giá trị 0, vậy có thể cột này chỉ mang duy nhất 1 giá

```
df = df.drop(['acc_now_delinq'],axis=1)
```

Cột “acc\_now\_delinq” không chứa missing values nhưng phần lớn dữ liệu chỉ mang giá trị 0, không phục vụ cho việc xây dựng mô hình nên ta sẽ loại bỏ cột này

```
# Không có missing value và phần lớn dữ liệu của cột là giá trị 0.
df['acc_now_delinq'] = df['acc_now_delinq'].fillna(0)
```

### ***Bộ dữ liệu sau khi loại bỏ các cột trên***

```
[ ] df.head(1)
```

	loan_amnt	term	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	pymnt_plan	desc	...	open_acc	pub_rec	revol_b
0	5000	36 months	162.87	10+ years	RENT	24000.0	Verified	Fully Paid	n	Borrower added on 12/22/11 >I need to upgra...	...	3	0	136

1 rows x 24 columns

```
[ ] df.columns
```

```
Index(['loan_amnt', 'term', 'installment', 'emp_length', 'home_ownership',
       'annual_inc', 'verification_status', 'loan_status', 'pymnt_plan',
       'desc', 'addr_state', 'dti', 'delinq_2yrs', 'inq_last_6mths',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'chargeoff_within_12_mths', 'pub_rec_bankruptcies', 'tax_liens',
       'purpose_n', 'earliest_cr_line_n'],
      dtype='object')
```

### ***Kiểm tra lại thông tin của bộ dữ liệu sau khi xử lý loại bỏ các cột không sử dụng đến***

```
def check_stats(col):
    print(df[col].head())
    print(df[col].describe())
    print(df[col].value_counts())
```

### ***Đối với cột “chargeoff\_within\_12\_mths”***

```
[ ] check_stats('chargeoff_within_12_mths')
```

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: chargeoff_within_12_mths, dtype: float64
count    39661.0
mean         0.0
std         0.0
min         0.0
25%         0.0
50%         0.0
75%         0.0
max         0.0
Name: chargeoff_within_12_mths, dtype: float64
0.0    39661
Name: chargeoff_within_12_mths, dtype: int64
```

Tương tự với những biến trước, 'chargeoff\_within\_12\_mths' chứa hầu hết giá trị là 0 và sẽ không có ích cho mô hình nên ta sẽ loại bỏ nó

```
df = df.drop(['chargeoff_within_12_mths'], axis=1)
```

### Đối với cột “pub\_rec\_bankruptcies”, “tax\_liens”

Tạo một dataframe mới bao gồm 2 cột 'pub\_rec\_bankruptcies', 'tax\_liens' và đặt tên là list\_next\_10\_till\_debt\_settlement và kiểm tra

```
list_debt_settlement = [ 'pub_rec_bankruptcies', 'tax_liens']
for col in list_debt_settlement:
    print(check_stats(col))
```

### Kết quả kiểm tra

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: pub_rec_bankruptcies, dtype: float64
count    39020.000000
mean      0.043260
std       0.204324
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       2.000000
Name: pub_rec_bankruptcies, dtype: float64
0.0    37339
1.0    1674
2.0      7
Name: pub_rec_bankruptcies, dtype: int64
None
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: tax_liens, dtype: float64
count    39678.0
mean      0.0
std       0.0
min       0.0
25%       0.0
50%       0.0
75%       0.0
max       0.0
Name: tax_liens, dtype: float64
0.0    39678
Name: tax_liens, dtype: int64
None
```

Ta thấy cột 'tax\_liens' chứa hầu hết các giá trị là 0 nên sẽ không sử dụng được cho việc xây dựng mô hình nên sẽ tiến hành loại bỏ

```
# Cột 'tax_liens' chứa hầu hết các giá trị là 0 nên sẽ không sử dụng
được cho modeling -> drop
df = df.drop(['tax_liens'], axis=1)
```



### ***Đối với cột “pymnt\_plan”***

```
[ ] df['pymnt_plan'].value_counts()
```

```
n      39717  
Name: pymnt_plan, dtype: int64
```

Quan sát thấy cột “pymnt\_plan” có 1 giá trị duy nhất và không ảnh hưởng đến bộ dữ liệu nên ta sẽ loại bỏ thuộc tính này đi

```
complete_df = complete_df.drop(['pymnt_plan'],axis=1)
```

### ***Đánh giá chung cả hai quá trình làm sạch dữ liệu 1 và quá trình làm sạch dữ liệu 2***

*Quá trình làm sạch dữ liệu gồm*

- Các biến chỉ có một giá trị riêng biệt đã bị loại bỏ vì chúng không hữu ích cho nhiệm vụ.
- Các biến có ít hơn 5% dữ liệu hay chứa nhiều dữ liệu rỗng đã bị xóa vì trong không hữu ích trong việc tạo ra một mô hình tốt
- Các biến không liên quan đến mục đích của báo cáo bị loại bỏ.
- Các biến có thể gây rò rỉ dữ liệu đã bị xóa, ví dụ 'last\_pymnt\_d', 'total\_pymnt', 'total\_rec\_int', 'funded\_amnt',...
- Các biến có cùng ý nghĩa đã bị xóa.

### c. Xác định Missing Values

Đầu tiên ta tiến hành kiểm tra sơ bộ các giá trị missing values tồn tại ở cột nào và chiếm bao nhiêu phần trăm (%) trong các biến này. Tạo một hàm `null_values` với tham số đầu vào là bộ dữ liệu ở dạng `DataFrame`, hàm sẽ bao gồm tính tổng các giá trị null ở từng cột và đem chia cho tổng số dòng ở mỗi cột để ra được % giá trị bị thiếu của mỗi cột

```
def null_values(df):
    mis_val = df.isnull().sum()
    mis_val_percent = 100 * df.isnull().sum() / len(df)
    mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
    mis_val_table_rename_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})
    mis_val_table_rename_columns = mis_val_table_rename_columns[
        mis_val_table_rename_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
    print ("Tập dữ liệu " + str(df.shape[1]) + " cột.\n"
          "Có " + str(mis_val_table_rename_columns.shape[0]) +
          " cột có missing values.")
    return mis_val_table_rename_columns
```

Kết quả khi chạy hàm `null_values` cho thấy tập dữ liệu hiện có 97 cột nhưng tồn tại 66 cột có missing values và hầu hết các cột đều rỗng hay có ít dữ liệu so sánh với các biến được dùng, nó không có ích cho bộ dữ liệu, có thể bỏ đi. Vì vậy, nhóm dự đoán có khoảng 70% missing values không hữu ích và phải xóa nó.

```
# Thống kê missing value
miss_values = null_values(df)
miss_values.head(20)
```

Tập dữ liệu 97 cột.  
Có 66 cột có missing values.

	Missing Values	% of Total Values
bc_util	39717	100.0
num_bc_tl	39717	100.0
mo_sin_old_il_acct	39717	100.0
mo_sin_old_rev_tl_op	39717	100.0
mo_sin_rcnt_rev_tl_op	39717	100.0
mo_sin_rcnt_tl	39717	100.0
mort_acc	39717	100.0
mths_since_recent_bc	39717	100.0
mths_since_recent_bc_dlq	39717	100.0
mths_since_recent_inq	39717	100.0
mths_since_recent_revol_delinq	39717	100.0
num_accts_ever_120_pd	39717	100.0
num_actv_bc_tl	39717	100.0
num_actv_rev_tl	39717	100.0
num_bc_sats	39717	100.0
num_il_tl	39717	100.0

Tiến hành xóa toàn bộ các cột không dùng được bằng cách lọc ra các cột có số lượng missing values lớn hơn 70%. Sau đó hiển thị ra các cột đã bị xóa cũng như số lượng cột còn lại và tên những cột đó

```
mis_val_percent = 100 * df.isnull().sum() / len(df)
missing_features_list = []
missing_features = mis_val_percent[mis_val_percent > 70].index
missing_features_list.append(missing_features)
print(missing_features_list)
df.drop(missing_features, axis=1, inplace=True)
print('Bây giờ tập dữ liệu còn {} cột'.format(len(df.columns)))
print('Các cột còn lại bao gồm: ')
print(df.columns)
```

Bây giờ tập dữ liệu còn lại 41 cột được hiển thị kết quả như sau

```
[Index(['mths_since_last_record', 'next_pymnt_d', 'mths_since_last_major_derog',
       'annual_inc_joint', 'dti_joint', 'verification_status_joint',
       'tot_coll_amt', 'tot_cur_bal', 'open_acc_6m', 'open_il_6m',
       'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il',
       'il_util', 'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util',
       'total_rev_hi_lim', 'inq_fi', 'total_cu_tl', 'inq_last_12m',
       'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
       'mo_sin_old_il_acct', 'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
       'mo_sin_rcnt_tl', 'mort_acc', 'mths_since_recent_bc',
       'mths_since_recent_bc_dlt', 'mths_since_recent_inq',
       'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd',
       'num_actv_bc_tl', 'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl',
       'num_il_tl', 'num_op_rev_tl', 'num_rev_accts', 'num_rev_tl_bal_gt_0',
       'num_sats', 'num_tl_120dpd_2m', 'num_tl_30dpd', 'num_tl_90g_dpd_24m',
       'num_tl_op_past_12m', 'pct_tl_nvr_dlq', 'percent_bc_gt_75',
       'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit'],
      dtype='object')]
```

Bây giờ tập dữ liệu còn 41 cột

Các cột còn lại bao gồm:

```
Index(['loan_amnt', 'term', 'installment', 'emp_length', 'home_ownership',
       'annual_inc', 'verification_status', 'loan_status', 'pymnt_plan',
       'desc', 'addr_state', 'dti', 'delinq_2yrs', 'inq_last_6mths',
       'mths_since_last_delinq', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status', 'out_prncp',
       'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
       'total_rec_int', 'total_rec_late_fee', 'recoveries',
       'collection_recovery_fee', 'last_pymnt_d', 'last_credit_pull_d',
       'collections_12_mths_ex_med', 'policy_code', 'application_type',
       'acc_now_delinq', 'chargeoff_within_12_mths', 'pub_rec_bankruptcies',
       'tax_liens', 'purpose_n', 'earliest_cr_line_n'],
      dtype='object')
```

### ***Đối với cột “mths\_since\_last\_delinq”***

Thực hiện tính thống kê trung bình với biến số tháng kể từ lần cuối người vay bị quá hạn

```
df['mths_since_last_delinq'].count()
print(100*df['mths_since_last_delinq'].isna().sum()/len(df))
```

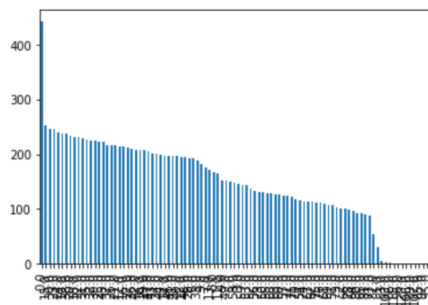
### **Kết quả thống kê thu được**

```
df['mths_since_last_delinq'].describe()

count    14035.000000
mean      35.900962
std       22.020060
min        0.000000
25%       18.000000
50%       34.000000
75%       52.000000
max      120.000000
Name: mths_since_last_delinq, dtype: float64
```

### ***Biểu diễn trực quan biến số tháng kể từ lần cuối người vay bị quá hạn***

```
df['mths_since_last_delinq'].value_counts().plot.bar()
plt.show()
```



Theo thống kê Missing Value quan sát cho thấy biến mths\_since\_last\_delinq có hơn 50% dữ liệu bị thiếu nên ta sẽ tiến hành loại bỏ nó

```
df = df.drop(['mths_since_last_delinq'],axis=1)
```

***Tiếp đến ta loại bỏ các cột tiếp theo không phục vụ cho việc xây dựng mô hình, thực hiện như sau***

```
drop_list9 =
['out_prncp','out_prncp_inv','total_pymnt','total_pymnt_inv','total_rec_prncp',
'total_rec_int',
'total_rec_late_fee','recoveries','collection_recovery_fee','last_pymnt_d',
'last_credit_pull_d']
df = df.drop(drop_list9,axis=1)
```

Bộ dữ liệu sau khi loại bỏ các cột trên

```
[ ] df.head(1)
```

	loan_amnt	term	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	pymnt_plan	desc	...	initial_list_status	colle
0	5000	36 months	162.87	10+ years	RENT	24000.0	Verified	Fully Paid	n	Borrower added on 12/22/11 > I need to upgra...	...	f	

1 rows x 29 columns

**Đối với cột “initial\_list\_status”**

```
df['initial_list_status'].head()
df['initial_list_status'].unique()
df['initial_list_status'].count()
```

Sau đó ta tạo một DataFrame mới là một phiên bản sao chép của dataframe gốc sau khi đã được tiền xử lý

```
# Copy Dataframe
complete_df = df.copy()
```

Kiểm tra số lượng cột bộ dữ liệu “complete\_df” và xem có tồn tại missing values không

```
# Missing values statistics
miss_values = null_values(complete_df)
miss_values.head(20)
```

Kết quả thu được

Tập dữ liệu có 22 cột.  
Có 4 cột có missing values.

	Missing Values	% of Total Values
desc	12940	32.6
emp_length	1075	2.7
pub_rec_bankruptcies	697	1.8
revol_util	50	0.1

**Nhận xét:** Sau các bước xử lý ở trên, nhóm đã giảm số cột từ 111 thành 22 cột nhưng vẫn bảo đảm không làm mất ý nghĩa của bộ dữ liệu

#### d. Xác định biến Target

Mục tiêu là dự đoán liệu một người có thể trả hết khoản vay hay người đó sẽ vỡ nợ. Chúng ta có thể thấy từ tập dữ liệu, **biến 'loan\_status' là biến duy nhất mô tả trạng thái thanh khoản cho nên biến này sẽ là biến target.**

```
complete_df['loan_status'].value_counts()
```

```
Fully Paid      32950
Charged Off     5627
Current         1140
Name: loan_status, dtype: int64
```

Tuy nhiên, biến “loan\_status” đang chứa các giá trị văn bản cần được chuyển đổi thành giá trị số để đủ điều kiện xây dựng mô hình

```
type(complete_df['loan_status'][0])
```

```
str
```

***Vậy ta sẽ đặt ra các quy tắc sau đây***

- Biến phụ thuộc = Loan\_Status
- Charged Off = 1
- Default = 1
- Current = 0
- Fully Paid = 0

***Đối với cột “Loan\_status”***

Những dòng nào có giá trị bằng “Charged Off” thì nhóm sẽ đổi thành “Default”

```
complete_df['loan_status'].value_counts()
```

Xét cột loan\_status, những dòng nào có giá trị là “Default” thì sẽ được chuyển đổi sang giá trị 1, còn lại sẽ mang giá trị 0. Sau khi chuyển đổi toàn bộ kiểu dữ liệu của cột "loan\_status" từ chuỗi sang số, ta sẽ đưa toàn bộ các giá trị này vào 1 list và đưa list này vào lại dataframe complete\_df để chuẩn hoá lại giá trị của cột target. Cuối cùng dùng hàm value\_counts() để đếm số lượng của hai giá trị 0 và 1

```
target_list = [1 if i=='Default' else 0 for i in complete_df['loan_status']]
```

```
complete_df['TARGET'] = target_list
complete_df['TARGET'].value_counts()
```

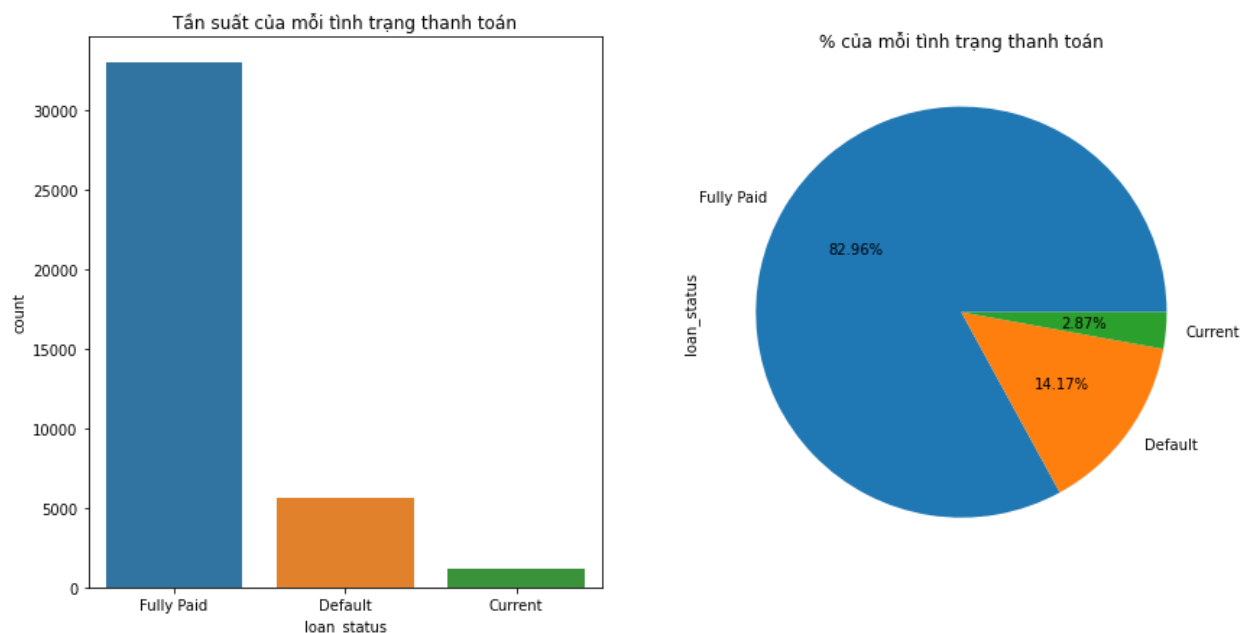
```
0      34090
1       5627
Name: TARGET, dtype: int64
```

### e. Trực quan hóa kết quả dữ liệu với biến Target và kiểm tra mô hình tương quan

Ta chọn “loan\_status” làm biến mục tiêu hướng đến nên ta sẽ sử dụng biểu đồ countplot nhằm xây dựng *tần suất của mỗi tình trạng thanh toán của khách hàng* và biểu đồ Pie để xây dựng trực quan *% phần trăm của tần suất thanh toán đó*

```
fig, axs = plt.subplots(1,2,figsize=(14,7))
sns.countplot(x='loan_status',data=complete_df,ax=axs[0])
axs[0].set_title("Tần suất của mỗi tình trạng thanh toán")
complete_df.loan_status.value_counts().plot(x=None,y=None,kind='pie',
ax=axs[1],autopct='%1.2f%%')
axs[1].set_title("% của mỗi tình trạng thanh toán")
plt.show()
```

#### *Biểu đồ thu được*



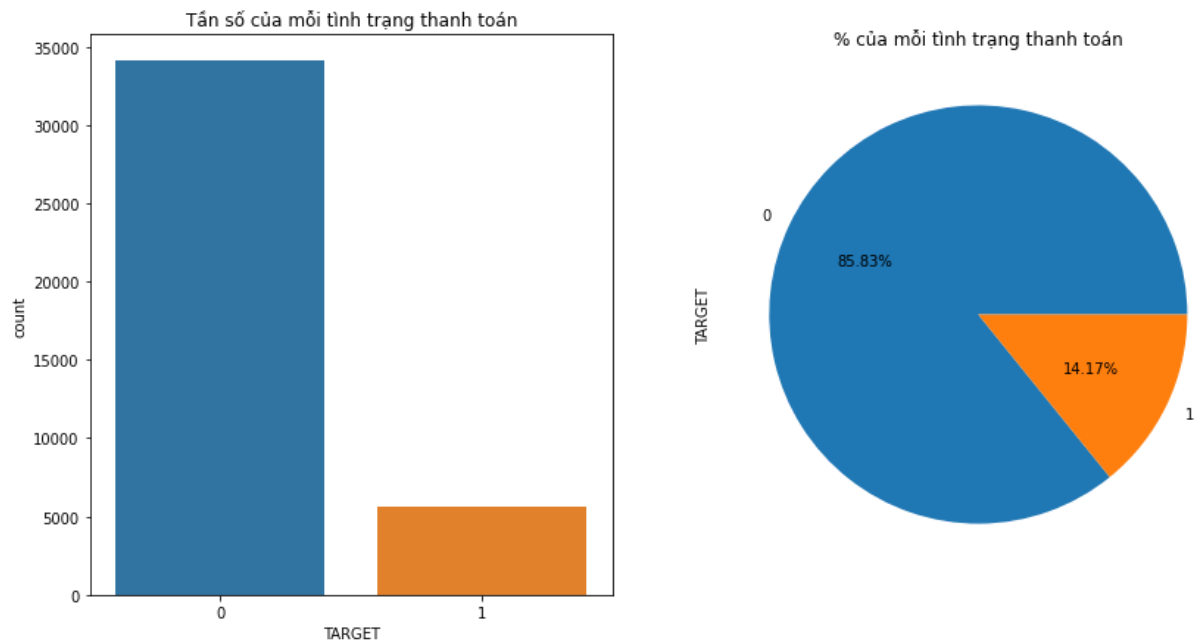
#### *Đánh giá*

Tần suất cao nhất thuộc về Fully Paid với hơn 30 000 lượt thanh toán ứng với 82,96%, Default\_loan\_status xếp sau Fully Paid với dưới 10 000 lượt thanh toán ứng với 14,17% và xếp cuối cùng là current với tổng tần suất thanh toán là dưới 5000 lượt ứng với 2,87%. Từ biểu đồ trên chúng ta có thể thấy rằng chúng ta đang xử lý một tập dữ liệu không cân bằng. Trong trường hợp này, chúng ta có Fully paid chiếm nhiều còn 2 giá trị còn lại chiếm rất ít tần suất xuất hiện.

Tương ứng như trên, ta tiếp tục xây dựng 2 biểu đồ counplot và biểu đồ pie xây dựng với biến x là mục tiêu Target

```
fig, axs = plt.subplots(1,2,figsize=(14,7))
sns.countplot(x='TARGET',data=complete_df,ax=axs[0])
axs[0].set_title("Tần số của mỗi tình trạng thanh toán")
complete_df.TARGET.value_counts().plot(x=None,y=None,kind='pie',
ax=axs[1],autopct='%1.2f%%')
axs[1].set_title("% của mỗi tình trạng thanh toán")
plt.show()
```

### Biểu đồ thu được



### Đánh giá

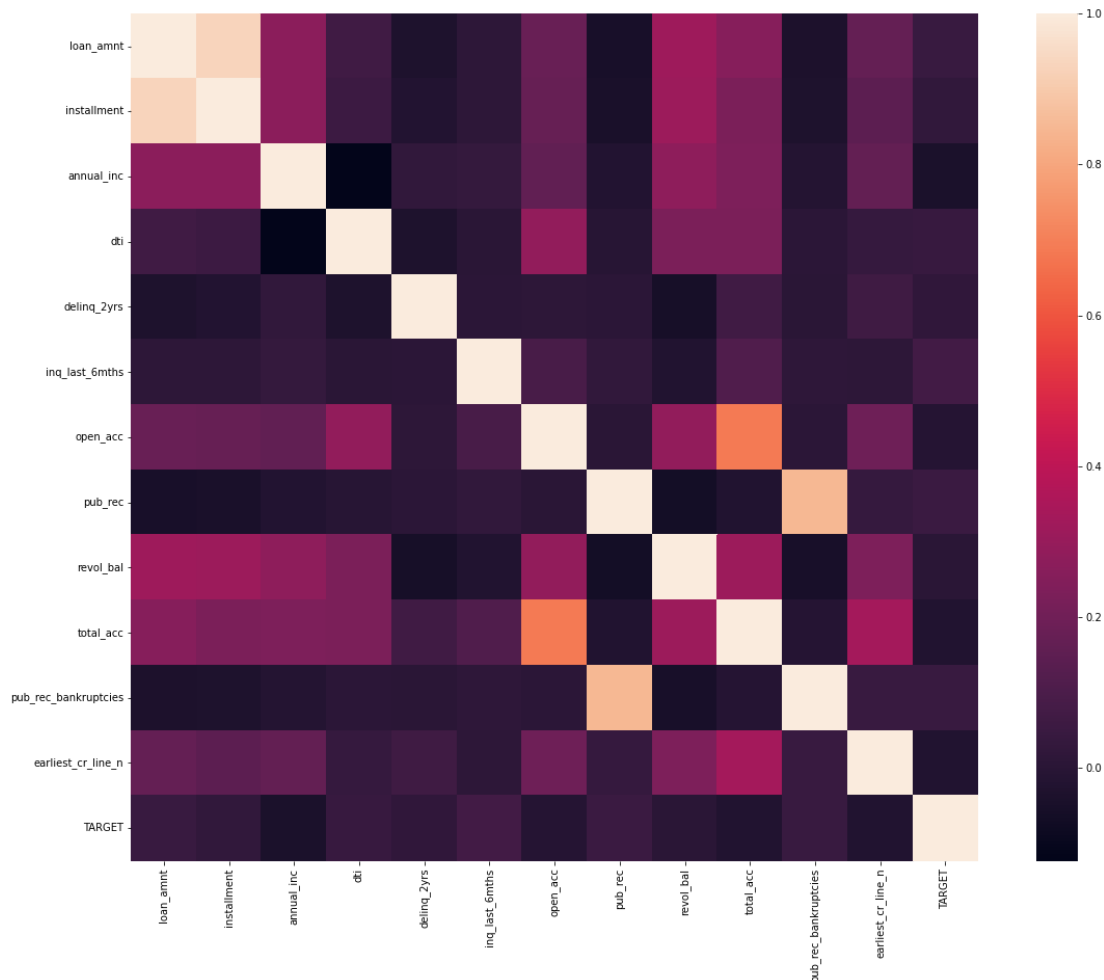
Như đã đề cập ở *Chương 2 phần 2 đề mục 2.2.2.c.Xác định biến Target* thì những dòng nào có giá trị là “Default” thì sẽ được chuyển đổi sang giá trị 1, còn lại sẽ mang giá trị 0. Qua biểu đồ trên ta thấy Target giá trị 0 chiếm hơn 30 000 lượt thanh toán ứng với 85,83% và Target giá trị 1 chiếm dưới 10 000 lượt thanh toán ứng với 14,17%.



### Kiểm tra mô hình tương quan giữa các thuộc tính

```
import seaborn as sns
from matplotlib import pyplot as plt
cor = complete_df.corr()
plt.subplots(figsize=(20,15))
sns.heatmap(cor, square = True)
```

Mô hình kết quả tương quan thu được giữa các thuộc tính với nhau



### Kiểm tra mô hình tương quan giữa số khoản vay phân phối và số năm làm việc

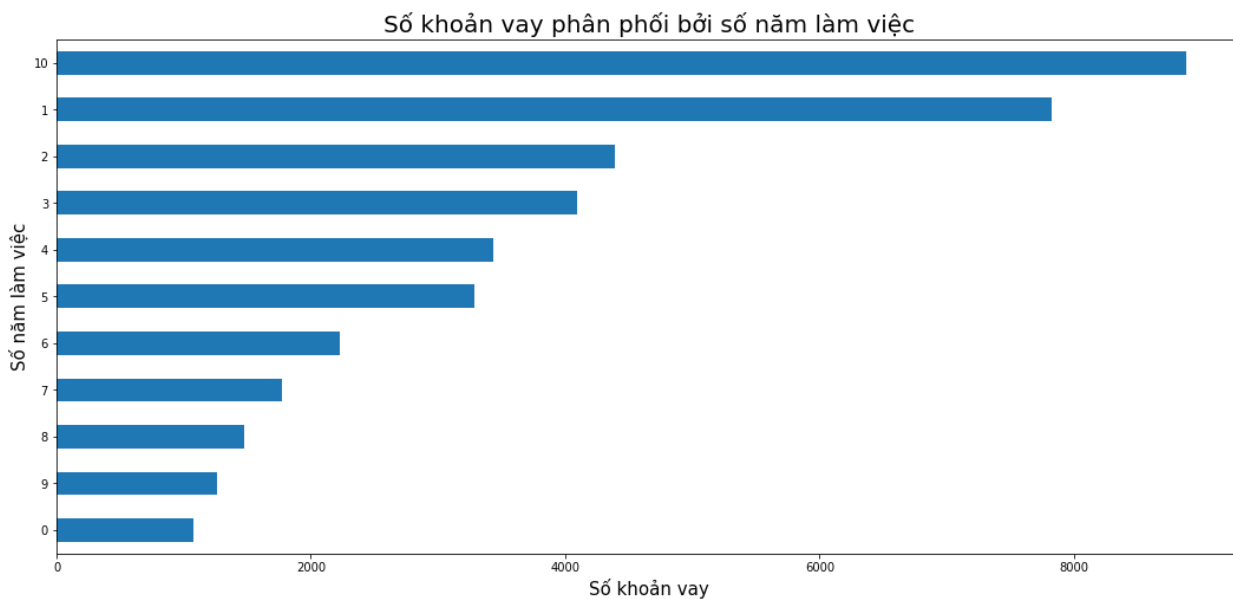
Nhóm sẽ thay các giá trị null bằng 0 với giả định rằng người đi vay đã không làm việc nhiều năm để thông tin của khách hàng được ghi lại

```
[ ] complete_df['emp_length'].head(3)

0    10+ years
1    < 1 year
2    10+ years
Name: emp_length, dtype: object
```

```
complete_df['emp_length'].fillna(value=0,inplace=True)
complete_df['emp_length'].replace(to_replace='^[^0-9]+', value='', inplace=True,
regex=True )
complete_df['emp_length'].value_counts().sort_values().plot(kind='barh',figsize
=(18,8))
plt.title('Số khoản vay phân phối bởi số năm làm việc',fontsize=20)
plt.xlabel('Số khoản vay',fontsize=15)
plt.ylabel('Số năm làm việc',fontsize=15);
```

### ***Mô hình kết quả thu được giữa biến số khoản vay và số năm làm việc***



### ***Đánh giá***

Qua biểu đồ trên ta quan sát thấy những người có số khoản vay trên 8000 là những người thuộc nhóm có số năm làm việc là 10 năm và đứng thứ 2 với số khoản vay gần 8000 là những người thuộc nhóm có số năm làm việc là 1 năm. Các số năm làm việc còn lại ứng với 2,3,4,5,6,7,8,9,0 năm làm việc sẽ có số khoản vay theo thứ tự liên tục giảm dần với nhóm có số năm làm việc bằng 0 đứng cuối cùng. Có thể nói, những người đã làm việc từ 10 năm trở lên có nhiều khả năng vay hơn các nhóm với số năm làm việc còn lại.

**Tiếp đến ta cần tạo một danh sách các miền chứa các bang để xem xét khả năng ảnh hưởng của thuộc tính này lên biến dữ liệu nghiên cứu**

```
west = ['CA', 'OR', 'UT', 'WA', 'CO', 'NV', 'AK', 'MT', 'HI', 'WY', 'ID'] #  
western  
south_west = ['AZ', 'TX', 'NM', 'OK'] # south western states  
south_east = ['GA', 'NC', 'VA', 'FL', 'KY', 'SC', 'LA', 'AL', 'WV', 'DC', 'AR',  
'DE', 'MS', 'TN'] #all south eastern states  
mid_west = ['IL', 'MO', 'MN', 'OH', 'WI', 'KS', 'MI', 'SD', 'IA', 'NE', 'IN',  
'ND'] #all mid western states  
north_east = ['CT', 'NY', 'PA', 'NJ', 'RI', 'MA', 'MD', 'VT', 'NH', 'ME'] #all  
north eastern states
```

**Tạo cột mới với tất cả giá trị nan null**

```
complete_df['region'] = np.nan # tạo cột mới với tất cả giá trị nan null  
def finding_regions(state):  
    if state in west:  
        return 'West'  
    elif state in south_west:  
        return 'SouthWest'  
    elif state in south_east:  
        return 'SouthEast'  
    elif state in mid_west:  
        return 'MidWest'  
    elif state in north_east:  
        return 'NorthEast'  
  
complete_df['region'] = complete_df['addr_state'].apply(finding_regions)
```

## f. Xử lý Missing value

Ở đây missing value có số lượng lớn trên tổng số quan sát nhất định trong bộ dữ liệu nghiên cứu nhóm sử dụng nên nhóm chọn phương pháp Data Impute tức là thay thế missing value bằng một giá trị khác thay vì loại bỏ hết missing value. Việc loại bỏ missing value sẽ tác động đến các biến phụ thuộc khác và làm thay đổi sự ảnh hưởng của chúng lên nhau và rất có thể sẽ ảnh hưởng đến mức độ tương quan giữa các dữ liệu với nhau.

Để xử lý các giá trị bị thiếu trong mỗi cột, nhóm sẽ thực hiện gán nhãn dữ liệu cho từng loại dtype khác nhau. Đối với dtype = object, nhóm sẽ sử dụng các giá trị thường xuyên nhất trong khi đối với các dtype số, nhóm sẽ dùng trung vị để impute các giá trị còn lại :

```
obj_cols = complete_df.columns[complete_df.dtypes==object]

# Hàm của imputer
imputer = lambda x:x.fillna(x.value_counts().index[0])

#Impute dtype=object với giá trị xuất hiện thường xuyên
complete_df[obj_cols] = complete_df[obj_cols].apply(imputer)

#Impute giá trị còn lại bằng trung vị
complete_df = complete_df.fillna(df.median(axis=0))
```

Kiểm tra bộ dữ liệu sau khi tiền xử lý

```
# Missing values statistics
miss_values = null_values(complete_df)
miss_values.head(5)
```

### *Kết quả đạt được*

```
Tập dữ liệu có 22 cột.
Có 0 cột có missing values.

Missing Values % of Total Values
```

### *Nhận xét*

Sau khi thực hiện các phân tích xác định Missing Value ở phần b và xử lý Missing Value ở phần e cũng như thực hiện Data Impute thay thế missing value bằng giá trị trung vị thì bộ dữ liệu hiện tại có 22 cột và đã không còn Missing Value.

### 3. Xử lý các biến phân loại Categorical Variables

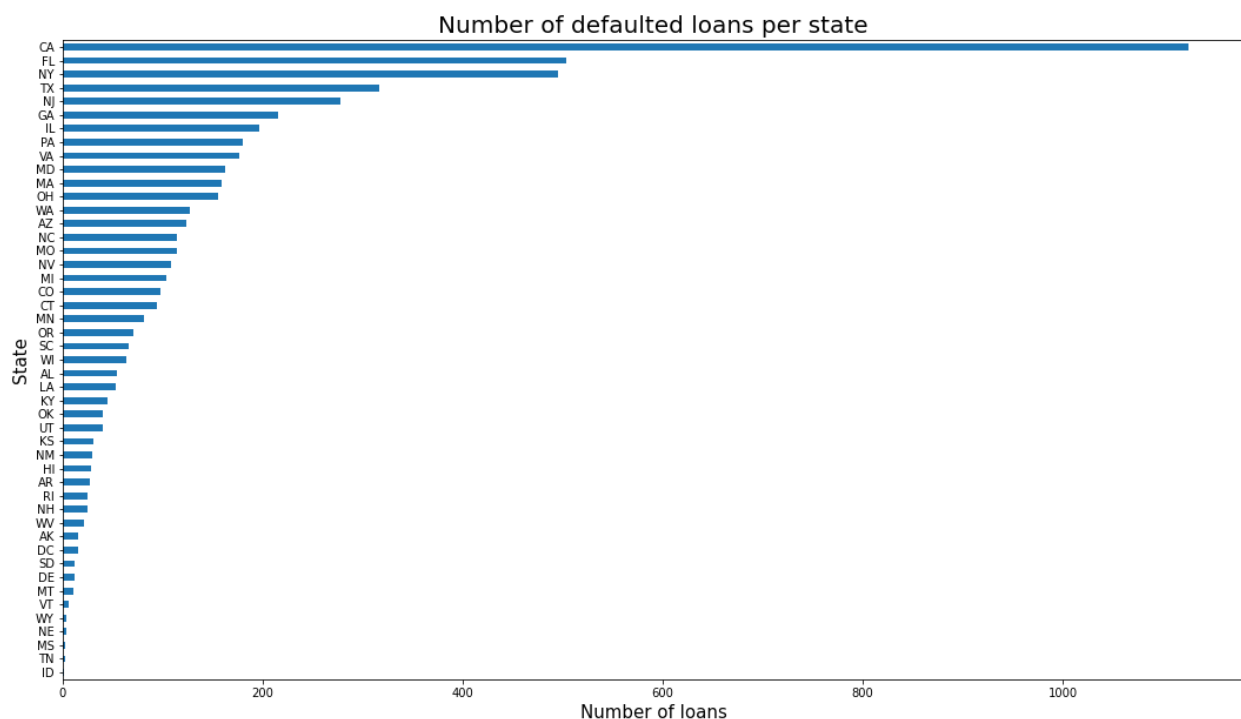
*Phần xử lý biến phân loại Categorical Variables nhóm sẽ thực hiện các mục tiêu sau:*

- Nghiên cứu phân loại cột
- Chuyển dữ liệu phân loại thành dữ liệu số
- Ánh xạ các giá trị thứ tự nguyên
- Mã hóa các giá trị danh nghĩa

*Trực quan các giữa biến mục tiêu Target và các bang “addr\_state”*

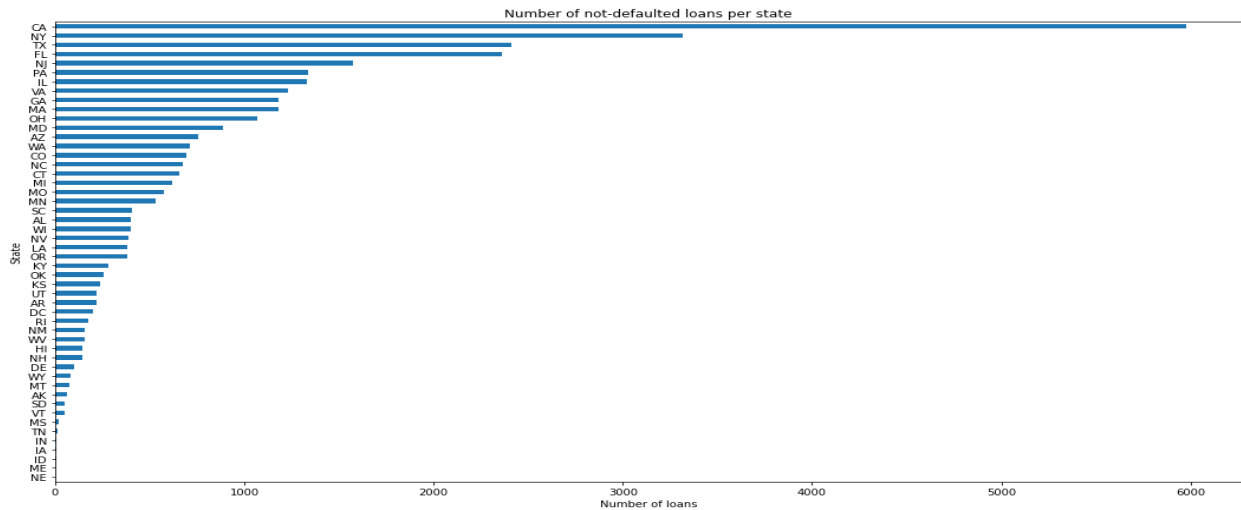
```
fig = plt.figure(figsize=(18,10))
complete_df[complete_df['TARGET']==1].groupby('addr_state')['TARGET'].count().sort_values().plot(kind='barh')
plt.ylabel('State',fontsize=15)
plt.xlabel('Number of loans',fontsize=15)
plt.title('Number of defaulted loans per state',fontsize=20);
```

Biểu đồ trực quan thu được



```
fig = plt.figure(figsize=(18,10))
complete_df[complete_df['TARGET']==0].groupby('addr_state')['TARGET'].count().sort_values().plot(kind='barh')
plt.ylabel('State')
plt.xlabel('Number of loans')
plt.title('Number of not-defaulted loans per state');
```

Biểu đồ trực quan thu được



### Đánh giá chung

Sử dụng biểu đồ Countplot biểu thị các khoản vay có khả năng vỡ nợ hoặc không vỡ nợ giữa các bang thông qua hai biểu đồ. Ta thấy Có nhiều khoản vay được thực hiện từ một tiểu bang có nhiều rủi ro vỡ nợ hơn. Đây là lý do tại sao không thể coi trạng thái là yếu tố tác động đến khoản vay liệu có bị vỡ nợ hay không.

**Do đó, ta lựa chọn việc loại bỏ luôn cột liên quan đến các ban trong bộ dữ liệu**

```
complete_df = complete_df.drop(['addr_state'],axis=1)
```

Bộ dữ liệu sau khi loại bỏ “Add\_state”

```
[ ] print(complete_df.shape)
    complete_df.head(1)

(39717, 23)
   loan_amnt  term  installment  emp_length  home_ownership  annual_inc  verification_status
0         5000   36 months      162.87         10           RENT      24000.0           Verified

1 rows x 23 columns
```

## Thực hiện kiểm tra chi tiết bộ dữ liệu

### Kiểm tra dữ liệu rỗng

```
[ ] null_counts = complete_df.isnull().sum()
    print("Number of null values in each column:\n{}".format(null_counts))
```

```
Number of null values in each column:
loan_amnt      0
term           0
installment    0
emp_length     0
home_ownership 0
annual_inc     0
verification_status 0
desc          0
dti            0
delinq_2yrs    0
inq_last_6mths 0
open_acc       0
pub_rec        0
revol_bal      0
revol_util     0
total_acc      0
acc_now_delinq 0
chargeoff_within_12_mths 0
pub_rec_bankruptcies 0
purpose_n      0
earliest_cr_line_n 0
TARGET         0
region         0
dtype: int64
```

### Kiểm tra các kiểu dữ liệu hiện có và tần suất của các biến quan sát

```
[ ] print("Data types and their frequency\n{}".format(complete_df.dtypes.value_counts()))
```

```
Data types and their frequency
int64      10
object      7
float64     6
dtype: int64
```

```
[ ] object_columns_df = complete_df.select_dtypes(include=['object'])
    print(object_columns_df.iloc[0])
```


```
term           36 months
home_ownership RENT
verification_status Verified
desc          Borrower added on 12/22/11 > I need to upgra...
revol_util      83.70%
purpose_n      debt
region         SouthWest
Name: 0, dtype: object
```


Chúng ta có thể thấy các biến 'term', 'emp\_length', 'revol\_util' chứa các dữ liệu số, nhưng được định dạng là một object

Tiếp đến, ta sẽ thực hiện chuyển đổi các tính năng danh nghĩa thành các tính năng số yêu cầu mã hóa chúng dưới dạng các biến giả cũng như loại bỏ nominal\_columns để tránh sự phụ thuộc tuyến tính giữa các kết quả đạt được

```
complete_df['emp_length'] = complete_df['emp_length'].astype('int64')
#converting the type of emp_length column to int64 from string
#Converting nominal features into numerical features requires encoding them as
dummy variables.
nominal_columns = ["home_ownership", "verification_status", "purpose_n",
"term"]
dummy_df = pd.get_dummies(complete_df[nominal_columns], drop_first=True)
#creating dummies for the above nominal columns and removing first dummy
variable to
#drop the first one to avoid linear dependency between the resulted features
since some algorithms may struggle with this issue.
complete_df = pd.concat([complete_df, dummy_df], axis=1) #merging the newly
created dummy columns with the working dataset
complete_df = complete_df.drop(nominal_columns, axis=1) #dropping the original
nominal columns as they are not required anymore
#df = pd.get_dummies(df, columns=["purpose"], drop_first=True)
```

Quan sát bộ dữ liệu thu được

 complete\_df.info()

 <class 'pandas.core.frame.DataFrame'>  
RangeIndex: 39717 entries, 0 to 39716  
Data columns (total 29 columns):

#	Column	Non-Null Count	Dtype
0	loan_amnt	39717 non-null	int64
1	installment	39717 non-null	float64
2	emp_length	39717 non-null	int64
3	annual_inc	39717 non-null	float64
4	desc	39717 non-null	object
5	dti	39717 non-null	float64
6	delinq_2yrs	39717 non-null	int64
7	inq_last_6mths	39717 non-null	int64
8	open_acc	39717 non-null	int64
9	pub_rec	39717 non-null	int64
10	revol_bal	39717 non-null	int64
11	revol_util	39717 non-null	float64
12	total_acc	39717 non-null	int64
13	acc_now_delinq	39717 non-null	int64
14	chargeoff_within_12_mths	39717 non-null	float64
15	pub_rec_bankruptcies	39717 non-null	float64
16	earliest_cr_line_n	39717 non-null	float64
17	TARGET	39717 non-null	int64
18	region	39717 non-null	object
19	home_ownership_NONE	39717 non-null	uint8
20	home_ownership_OTHER	39717 non-null	uint8
21	home_ownership_OWN	39717 non-null	uint8
22	home_ownership_RENT	39717 non-null	uint8



### *Xử lý thuộc tính Region*

```
#Converting region into dummy variable too.
nominal_columns2 = ["region"]
dummy_df2 = pd.get_dummies(complete_df[nominal_columns2], drop_first=True)
#creating dummies for the above nominal columns and removing first dummy
variable to
#avoid linear dependency between the resulted features since some
algorithms may struggle with this issue.
complete_df = pd.concat([complete_df, dummy_df2], axis=1) #merging the
newly created dummy columns with the working dataset
complete_df = complete_df.drop(nominal_columns2, axis=1) #dropping the
original nominal columns as they are not required anymore
#df = pd.get_dummies(df, columns=["purpose"], drop_first=True)
```

### *Bộ dữ liệu thu được sau khi đã hoàn thành xử lý các cột và thuộc tính*

```
[ ] complete_df.head()
```

	loan_amnt	installment	emp_length	annual_inc	desc	dti	delinq_2yrs	inq_last_6mths	open_acc	pub_rec	...	verification_status	Source	Verified	verification_
0	5000	162.87	10	24000.0	Borrower added on 12/22/11 > I need to upgra...	27.65	0	1	3	0	...			0	
1	2500	59.83	1	30000.0	Borrower added on 12/22/11 > I plan to use t...	1.00	0	5	3	0	...			1	
2	2400	84.33	10	12252.0		8.72	0	2	2	0	...			0	
3	10000	339.31	10	49200.0	Borrower added on 12/21/11 > to pay for prop...	20.00	0	1	10	0	...			1	
4	3000	67.79	1	80000.0	Borrower added on 12/21/11 > I plan on combi...	17.94	0	0	15	0	...			1	

5 rows x 32 columns

#### 4. Xử lý dữ liệu với NLP

Natural Language Processing (NLP) là một nhánh của ngôn ngữ học, khoa học máy tính và trí tuệ nhân tạo liên quan đến sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người, giọng nói hoặc văn bản. Những tiến bộ về mặt kỹ thuật trong lĩnh vực NLP có thể được chia thành: Mô hình hệ thống dựa trên quy tắc (rule-based), mô hình máy học cổ điển và Deep Learning. Đối với bài báo cáo này, sẽ chủ yếu quan tâm về mô hình máy học cổ điển.

Máy học cổ điển có thể giải nhiều bài toán thách thức hơn (VD: Phát hiện spam,...) thông qua việc trích lọc các Features (các thuộc tính, VD: tên, họ, năm sinh, doanh thu,...) bằng cách sử dụng những phương pháp như Bag of Words, Part of Speech, sau đó xây dựng các mô hình máy học (Machine Learning Models) như Support Vector Machine, Naive Bayes,... Các mô hình này sẽ khai thác những mẫu câu có ngữ nghĩa (semantics patterns) trong dữ liệu train (huấn luyện) để đưa ra các dự đoán tương lai.

##### *Áp dụng với bài toán của chúng ta*

Gắn thẻ POS: Gắn thẻ POS là từ viết tắt của Part of Speech, một quá trình gắn mỗi từ trong một câu với một thẻ phù hợp từ một tập hợp thẻ nhất định (ví dụ: N đại diện cho danh từ, J đại diện cho tính từ, V đại diện cho động từ, R đại diện cho trạng từ) theo định nghĩa, ngữ cảnh.

```
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

##### a. Tiền xử lý

Bước đầu tiên trong NLP là đọc dữ liệu để phân tích sâu hơn. Để đảm bảo mô hình hoạt động hiệu quả, tiền hàng làm sạch trên từng câu:

- Chỉ chứa các ký tự chữ và số (gợi ý: sử dụng cụm từ thông dụng)
- Hoàn toàn viết thường

- Không có dấu cách hoặc dấu cách hàng đầu
- Không phải chuỗi rỗng (gợi ý sử dụng bộ lọc)

```
def process_text(texts):
    final_text_list=[]
    for sent in texts:
        filtered_sentence=[]

        sent = sent.lower()
        sent = sent.strip()
        sent = re.sub('\s+', ' ', sent)
        sent = re.compile('<.*?>').sub('', sent)
        sent = re.sub(r"\b[A-Z]{2,}\b", "", sent)
        sent = re.sub(r"\d+", "", sent)
        sent = re.sub(r'http\S+', '', sent)
```

Tonize và Stem Words: Để đơn giản và giảm thời gian xử lý, sử dụng Snowballstemmer để tách từng từ. Như vậy, chỉ cần rút gọn từng mô tả về dạng gốc của chúng.

- Với thư viện nltk, sử dụng word\_tokenize để mã hóa từng mô tả (danh sách các từ)
- Sử dụng Snowballstemmer để ngắt từng từ (sử dụng vòng lặp for)
- Tham gia các từ bắt nguồn bằng chuỗi

Tất cả các bước ngoại trừ bước 1, phải được thực hiện trong cùng một chức năng. Hàm sẽ trả về một chuỗi.

***Sử dụng phương thức append(), tạo một cột mới gọi là final\_text\_list cho final\_string.***

```
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize
snow = SnowballStemmer('english')
for w in word_tokenize(sent):

    if(not w.isnumeric()) and (len(w)>2) and (w not in stop) and
(w not in string.punctuation) and (w in words or not w.isalpha()):

        filtered_sentence.append(w1.lemmatize(w))
    final_string = " ".join(filtered_sentence)

    final_text_list.append(final_string)

return final_text_list
```

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

- **Stop words:** Để chỉ giữ lại các từ có liên quan, xóa tất cả các từ dừng khỏi mỗi mô tả sử dụng kho từ để xóa chung. Tạo danh sách từ khóa bằng tiếng anh

### b. Train test split

Chia dữ liệu thành các tập huấn luyện và kiểm tra bằng cách sử dụng sklearn.model\_selection với train\_test\_split thành 4 biến khác nhau: X\_train, X\_test, y\_train, y\_test trong đó:

- Đối số đầu tiên là cột desc1 nhưng dưới dạng DataFrame (2 dấu ngoặc vuông)
- Đối số thứ hai chỉ nên là biến mục tiêu

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df2[['desc1']],
df2['TARGET'].tolist(), test_size=0.2, shuffle=True)
```

### c. Vecto hóa

Vì mô hình lựa chọn là mô hình phân loại của Naive Bayes, cần chuyển đổi mô tả của desc thành ma trận 1 và 0. Sử dụng CountVectorizer để chuyển đổi desc1 thành một ma trận, áp dụng cho hai biến X\_train và X\_test

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True, max_features=50)

# Fit and transform
X_train_text_vectors = vectorizer.fit_transform(X_train["desc1"].tolist())

# Only transform
X_test_text_vectors = vectorizer.transform(X_test["desc1"].tolist())
```

## CHƯƠNG 3. PHÂN LỚP

### 1. Định nghĩa phân lớp dữ liệu

Phân lớp dữ liệu là kỹ thuật dựa trên tập huấn luyện và những giá trị hay là nhãn của lớp trong một thuộc tính phân lớp và sử dụng nó trong việc phân lớp dữ liệu mới. Nghĩa là xếp các mẫu dữ liệu hay các đối tượng vào một trong các lớp đã được định nghĩa trước. Kỹ thuật phân lớp được tiến hành bao gồm 2 bước:

- **Bước 1:** Học, mục đích của bước này là xây dựng mô hình xác định tập các lớp dữ liệu.
- **Bước 2:** Kiểm tra và đánh giá, bước này sử dụng mô hình phân lớp đã được xây dựng ở bước 1 vào việc phân lớp

### 2. Xây dựng tập huấn luyện 20% bộ dữ liệu

Việc tạo các mẫu khác nhau để đào tạo và thử nghiệm giúp chúng ta đánh giá hiệu suất dự đoán của mô hình và xác nhận mô hình. Trước đó chúng ta cần xây dựng một mô hình máy học, đầu tiên nên cung cấp tập dữ liệu vào thuật toán máy học. Tập dữ liệu ban đầu để giúp thuật toán “học” gọi là tập dữ liệu huấn luyện (training set). Và để “kiểm tra” xem mô hình có hoạt động khi cần thiết hay không, ta dùng đến tập dữ liệu thử nghiệm (test set) cho mô hình sau khi mô hình được xây dựng

*Nhóm sẽ chia tập dữ liệu của mình thành các tập con train (80%) và test (20%).*

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(df2[['desc1']],df2['TARGET'].tolist(),test_size=0.2,shuffle=True)
```

Sử dụng hàm `train_test_split` của thư viện Scikit-learn (sklearn) để tách dữ liệu thành các tập con train và test.

```
x_train.head()
```

	desc1
4914	
4447	borrower added job home made lot debt wanting ...
6206	
3803	borrower added look forward expanding business...
1196	

### ***Dữ liệu sau khi được vector hóa như sau:***

```
print(vectorizer.vocabulary_)  
{'borrower': 4, 'added': 0, 'loan': 25, 'pay': 36, 'credit': 10, 'card': 7, 'time': 43, 'job': 22, 'need': 32, 'consolidate': 9, 'used': 45,
```

### ***Phân chia dữ liệu cho mô hình Decision Tree***

Tạo một dataframe df3 với các cột dữ liệu bị bỏ đi là “desc”, “desc1” và “desc2” vì các biến này ở dạng dữ liệu string, mô hình Decision Tree không xử lý data có dạng này nên ta sẽ loại bỏ

```
df3 = df2.drop(["desc", 'desc1', 'desc2'], axis='columns')
```

Sử dụng hàm train\_test\_split của thư viện Scikit-learn (sklearn) để tách dữ liệu thành các tập con train và test với một lệnh gọi hàm duy nhất trên đây. Ta đã tích hợp việc tách các thuộc tính (features) cho mô hình khỏi biến mục tiêu (target). Mảng đầu vào chứa dữ liệu tất cả các cột ngoại trừ ‘target’ là thuộc tính mà ta muốn sử dụng để xây dựng mô hình, tiếp theo là mảng chứa các giá trị đích là “target”, vì mục tiêu của ta muốn nhãn của dữ liệu thể hiện giá trị 0 hoặc 1. Ta đặt cho tỷ lệ 0.2 (20%) các quan sát sẽ có trong bộ kiểm tra, và 0.8 (80%) quan sát được chỉ định cho đào tạo

```
x_train_dtree, x_test_dtree, y_train_dtree, y_test_dtree =  
train_test_split(df3.drop(['TARGET'], axis=1), df3['TARGET'], random_state =  
0, test_size = 0.2)
```

Từ đó ta phân vùng khung dữ liệu ban đầu thành 4 tập dữ liệu khác nhau cho mô hình Decision Tree, với

```
[ ] print (x_train_dtree.shape, y_train_dtree.shape)  
    print (x_test_dtree.shape, y_test_dtree.shape)  
  
(54544, 28) (54544,)  
(13636, 28) (13636,)
```

### ***Quan sát thấy***

- Tập x\_train\_dtree có 54544 dòng dữ liệu đào tạo, 28 cột ứng với 28 thuộc tính
- Tập y\_train\_dtree có 54544 dòng dữ liệu đào tạo, 1 cột ứng với 1 biến mục tiêu
- Tập x\_test\_dtree có 13636 dòng dữ liệu kiểm thử, 28 cột ứng với 28 thuộc tính
- Tập y\_test\_dtree có 13636 dòng dữ liệu kiểm thử, 1 cột ứng với 1 biến mục tiêu

### 3. Xây dựng mô hình

Mô hình được xây dựng với biến 'loan\_status' là biến phụ thuộc, các biến còn lại sẽ là biến độc lập. Các mô hình được sử dụng để phân tích bao gồm: Decision Tree và Naive Bayes. Mô hình được xây dựng dựa trên ngôn ngữ lập trình Python và thư viện Scikit-learning.

Vì bộ dữ liệu không cân bằng, sau khi sử dụng phương thức sklearn's resample để tăng kích thước mẫu, tiến hành xây dựng các mô hình dự đoán khả năng vỡ nợ.

#### 3.1. Mô hình Naive Bayes

Khởi tạo mô hình phân lớp bằng phương pháp Naive Bayes với phân phối Multinomial NB. Chọn phân phối Multinomial vì Multinomial Naive Bayes là một trong những thuật toán phân lớp có giám sát phổ biến nhất được sử dụng để phân tích dữ liệu văn bản, đồng thời phù hợp với bộ dữ liệu đã xử lý và mang lại hiệu suất cao hơn các phân phối còn lại Naive Bayes.

```
model = MultinomialNB()
model.fit(X_train_text_vectors, y_train)
```

#### 3.2. Mô hình Decision Tree

Khởi tạo mô hình phân lớp bằng phương pháp cây quyết định DecisionTreeClassifier

```
dec = DecisionTreeClassifier()
dec.fit(x_train_dtree, y_train_dtree)
```

### 4. Sử dụng ma trận nhầm lẫn đánh giá mô hình xây dựng

#### 4.1. Mô hình Naive Bayes

Tạo mảng 1 chiều y\_pred để chứa các giá trị được dự báo của tập huấn luyện X\_test đã chuyển sang dạng vector

```
y_pred = model.predict(X_test_text_vectors)
y_pred
```

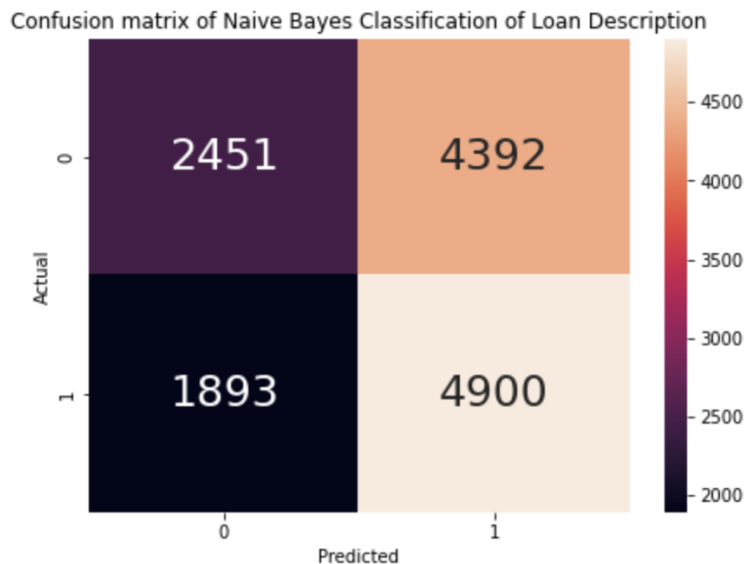
```
array([1, 0, 1, ..., 1, 1, 1])
```

#### *Sử dụng Ma trận nhầm lẫn (Confusion Matrix) để đánh giá mô hình*

```
confu_matr = confusion_matrix(y_test, y_pred)
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(1,1,1)
options = {
    'annot': True,
```

```
'fmt': '.0f',
    'annot_kws': {"size": 25}}
sns.heatmap(confu_matr, ax=ax, **options)
ax.set_title('Confusion matrix of Naive Bayes Classification of Loan
Description')
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
ax.set_ylim(2,0)
ax.set_xlim(0,2)
plt.show()
```

### ***Mã trận nhầm lẫn thu được từ Naive Bayes***



### **Nhận xét ma trận nhầm lẫn**

- True positive (TP): Số trường hợp dự đoán trạng thái thanh toán “1” là đúng: 4900
- True negative (TN): Số trường hợp dự đoán trạng thái thanh toán “0” là đúng: 2451
- False positive (FP): Số trường hợp dự đoán trạng thái thanh toán “1” là sai: 4392
- False negative (FN): Số trường hợp dự đoán trạng thái thanh toán “0” là sai: 1893
- Có 4900 trong số 9292 khoản vay chiếm 52,73% khoản vay được hoàn trả
- Có 2451 trong số 6843 khoản vay chiếm 35,8% khoản vay chưa được được hoàn trả hay thanh toán.



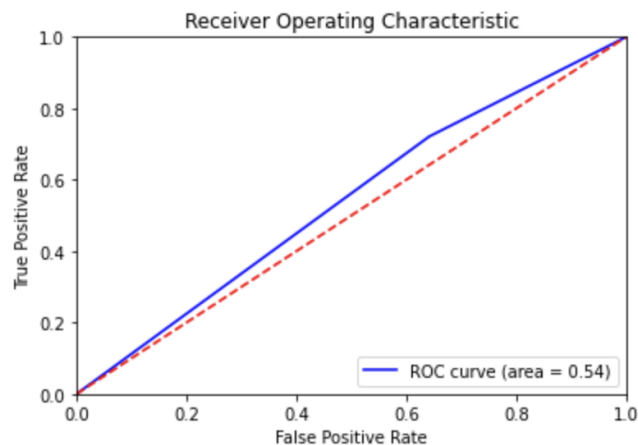
## Sử dụng ROC Curve

Đồng thời ta cũng sử dụng đến ROC Curve (một đường cong biểu diễn hiệu suất phân loại của một mô hình phân loại tại các ngưỡng threshold) để đánh giá mô hình. Về cơ bản, nó hiển thị True Positive Rate (TPR) so với False Positive Rate (FPR) đối với các giá trị ngưỡng khác nhau

```
#ROC curve
fpr, tpr, threshold = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'ROC curve (area = %0.2f)' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

ROC tìm ra TPR và FPR ứng với các giá trị ngưỡng khác nhau và vẽ biểu đồ để dễ dàng quan sát TPR so với FPR



Ta thu được diện tích nằm dưới đường cong ROC và trên trục hoành chính là AUC (chỉ số được tính toán dựa trên đường cong ROC nhằm đánh giá khả năng phân loại của mô hình tốt như thế nào), có giá trị là 0.6.

**Các giá trị ma trận nhầm lẫn thu được**

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(classification_report(y_test_dtree, y_hat))
print("Accuracy:", accuracy_score(y_test_dtree, y_hat))
```

	precision	recall	f1-score	support
0	0.55	0.34	0.42	6827
1	0.52	0.72	0.60	6809
accuracy			0.53	13636
macro avg	0.53	0.53	0.51	13636
weighted avg	0.53	0.53	0.51	13636

Accuracy: 0.5283807568201818

➤ **Kết quả thu được độ chính xác là 0.52**

**4.2. Mô hình Decision Tree**

Giống như mô hình phân lớp Naive Bayes, ta cũng tạo mảng 1 chiều y\_pred để chứa các giá trị được dự báo của tập huấn luyện x\_test\_dtree đã chuyển sang dạng vector

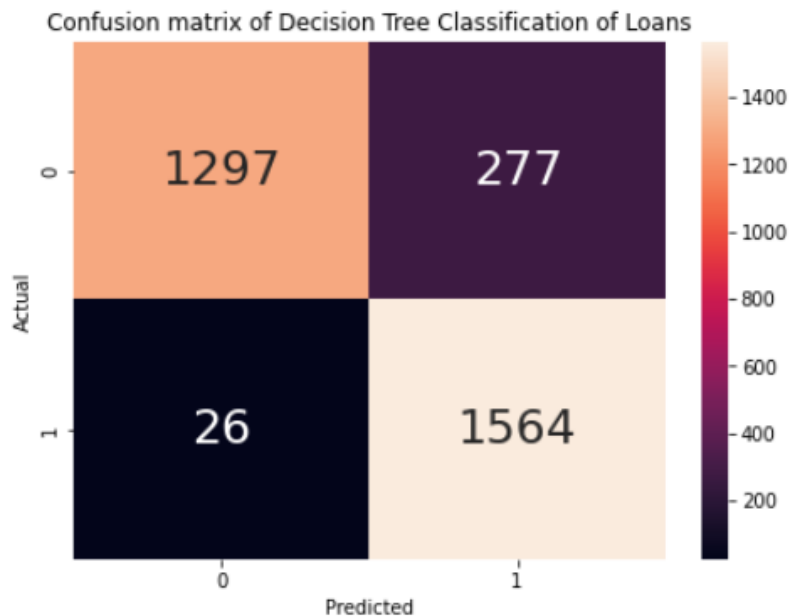
```
y_hat=dec.predict(x_test_dtree)
```

```
array([1, 0, 1, ..., 1, 1, 1])
```

**Sử dụng Ma trận nhầm lẫn (Confusion Matrix) để đánh giá mô hình**

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test_dtree, yhad))
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(1,1,1)
options = {
    'annot': True,
    'fmt': '.0f',
    'annot_kws' : {"size": 25}}
sns.heatmap(cm_tree, ax=ax, **options)
ax.set_title('Confusion matrix of Decision Tree Classification of Loans')
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
ax.set_ylim(2,0)
ax.set_xlim(0,2)
plt.show()
```

### ***Mã trận nhầm lẫn thu được từ Decision Tree***



### ***Nhận xét mã trận nhầm lẫn***

- True positive (TP): Số trường hợp dự đoán trạng thái thanh toán “1” là đúng: 1564
- True negative (TN): Số trường hợp dự đoán trạng thái thanh toán “0” là đúng: 1297
- False positive (FP): Số trường hợp dự đoán trạng thái thanh toán “1” là sai: 277
- False negative (FN): Số trường hợp dự đoán trạng thái thanh toán “0” là sai: 26

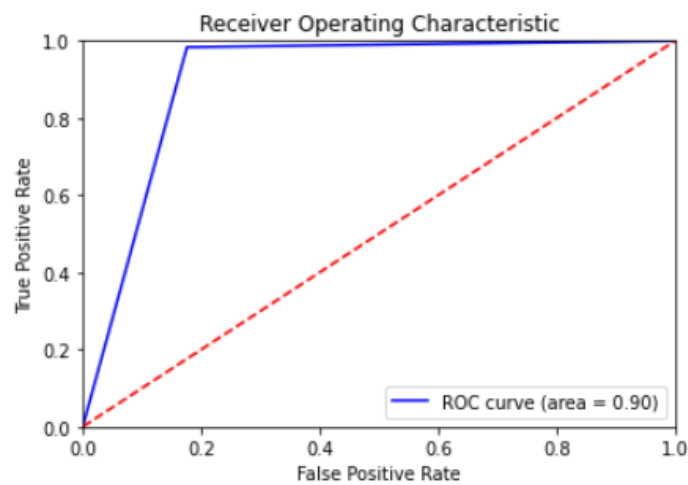
### **Sử dụng ROC Curve**

Giống mô hình Naive Bayes, ta cũng sử dụng đường cong ROC để đánh giá mô hình Decision Tree. ROC tìm ra TPR và FPR ứng với các giá trị ngưỡng khác nhau và vẽ biểu đồ để dễ dàng quan sát TPR so với FPR

```
#ROC curve
# calculate the fpr and tpr for all thresholds of the classification
preds = y_hat[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test_dtree, preds)
roc_auc = metrics.auc(fpr, tpr)
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'ROC curve (area = %0.2f)' % roc_auc)
plt.legend(loc = 'lower right')
```

```
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

ROC tìm ra TPR và FPR ứng với các giá trị ngưỡng khác nhau và vẽ biểu đồ để dễ dàng quan sát TPR so với FPR. Ta thu được chỉ số AUC có giá trị là 0.90



### *Các giá trị ma trận nhầm lẫn thu được*

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(classification_report(y_test_dtree, y_hat))
print("Accuracy:", accuracy_score(y_test_dtree, y_hat))
```

	precision	recall	f1-score	support
0	0.98	0.82	0.90	1574
1	0.85	0.98	0.91	1590
accuracy			0.90	3164
macro avg	0.91	0.90	0.90	3164
weighted avg	0.91	0.90	0.90	3164

Accuracy: 0.9042351453855879

Kết quả thu được độ chính xác là 0.9

### 5. Lựa chọn mô hình sử dụng

Với phương pháp Naive Bayes, chỉ số trong một lần tính toán được

- Accuracy = 0.528
- AUC = 0.54

Với phương pháp Decision Tree, chỉ số trong một lần tính toán được :

- Accuracy = 0.9042
- AUC = 0.90

**Kết luận:** Vì phương pháp Decision Tree có tất cả các điểm số đều cao vượt trội hơn mô hình Naive Bayes, nên ta sẽ sử dụng phương pháp Decision Tree để áp dụng mô hình

## **CHƯƠNG 4. TỔNG KẾT**

### **1. Kết quả đạt được**

Thông qua các thử nghiệm, mô hình Decision Tree được xem là phù hợp nhất với tập dữ liệu và phục vụ mục đích cung cấp cho nhà đầu tư một mô hình giúp tăng cơ hội kiếm lợi nhuận của họ.

Nó có mức độ chính xác accuracy và precision xấp xỉ 0.9. Nhà đầu tư có thể có nhiều cơ hội cho vay, nhưng rất ít có hội mất tiền.

### **2. Tóm tắt vấn đề nghiên cứu**

Khi bắt đầu, tập dữ liệu đã được làm sạch. Sau đó, phân tích dữ liệu và áp dụng các kỹ thuật đã học, một mô hình phù hợp đã được xây dựng để dự đoán liệu khách hàng sẽ trả khoản vay hay không.

### **3. Cải tiến trong tương lai**

Các mô hình học máy khác nhau (Random Forest, SVM, GA,...) có thể được thực hiện và so sánh để có kết quả tốt hơn.

## Tài liệu tham khảo

### Link Colab nhóm:

<https://colab.research.google.com/drive/1wEl1wjwrhMblW7ighkpga9a-3R3aiTXn?usp=sharing>

[1] Bài giảng học phần Xử Lý Ngôn Ngữ Tự Nhiên, khoa Công Nghệ Thông Tin Kinh Doanh, Trường Đại học Kinh tế Tp.HCM, 2022.

[2] Bài giảng học phần Lập trình phân tích dữ liệu, khoa Công Nghệ Thông Tin Kinh Doanh, Trường Đại học Kinh tế Tp.HCM, 2022.

[3] Scikit Learn, “*sklearn.Naïves.Bayes.MultinomialNB*”, [Trực tuyến]. Địa chỉ:  
[https://scikit-learn.org/stable/modules/generated/sklearn.naive\\_bayes.MultinomialNB.html?fbclid=IwAR2gHQCdHdEUJG0Vx05de92C1fuFiLbmHUacEvGpRv9z8\\_SW9hTPc\\_jOfhus](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html?fbclid=IwAR2gHQCdHdEUJG0Vx05de92C1fuFiLbmHUacEvGpRv9z8_SW9hTPc_jOfhus)

[3] Scikit Learn, “*sklearn.tree.DecisionTreeClassifier*”, [Trực tuyến]. Địa chỉ:  
[https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?fbclid=IwAR26S0wkLzUuedH5FSXr6UeC5-R1O\\_\\_50j4rUcZRdTCEy0jJwY6g6tIpNY4](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?fbclid=IwAR26S0wkLzUuedH5FSXr6UeC5-R1O__50j4rUcZRdTCEy0jJwY6g6tIpNY4)

[4] [3] Scikit Learn, “*sklearn.model\_selection.train\_test\_split*”, [Trực tuyến]. Địa chỉ:  
[https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html?fbclid=IwAR3HeGtTp7dVfQTbTHK\\_qnxz5-zAdQ1YPtqhtYkanhoTrYKpcbI3K6Ke6Qg](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?fbclid=IwAR3HeGtTp7dVfQTbTHK_qnxz5-zAdQ1YPtqhtYkanhoTrYKpcbI3K6Ke6Qg)

[5] Sarang Narkhede, “*Understanding Confusion Matrix*”, 2018. [Trực tuyến]. Địa chỉ:  
<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>. [Truy cập 9/5/2018]