Bài 22 - Scorecard model

17 Jan 2020 - phamdinhkhanh

Menu

1. ScoreCard model là gì?

Scorecard model là một lớp mô hình được ứng dụng trong nhiều lĩnh vực như tài chính, kinh doanh, quản lý xã hội. Mô hình Scorecard có tác dụng lượng hóa một hồ sơ cá nhân hoặc tổ chức thành một điểm tín nhiệm dựa trên khả năng xảy ra của một sự kiện nào đó như vỡ nợ, vi phạm luật. Dựa trên điểm tín nhiệm, các tổ chức tài chính hoặc chính phủ có thể cung cấp các sản phẩm, dịch vụ tốt hơn nếu chủ thể có điểm tín nhiệm cao và thấp hơn đối với chủ thể có điểm tín nhiệm thấp.

Ứng dụng trong quản trị rủi ro ngân hàng:

Trong hoạt động tín dụng, mục tiêu của mô hình scorecard là đánh giá năng lực trả nợ của người giữ vị thế vay trong tương lai. Đầu vào của mô hình Scorecard gồm các thông tin nằm trong hồ sơ cá nhân của khách hàng. Chúng được thu thập chủ yếu từ cục tín dụng hoặc data warehouse của ngân hàng. Các thông tin được chia thành các nhóm cơ bản:

- Nhân khẩu học (demographic): Là những thông tin liên quan đến đặc điểm cá nhân như trình độ học vấn, thu nhập, giới tính, độ tuổi, nghề nghiệp, trạng thái hôn nhân, qui mô gia đình, số người phụ thuộc,….
- Lịch sử tín dụng (credit history): Đây là những thông tin được quản lý tập trung tại cục tín dụng (bureau credit). Dữ liệu lịch sử vay của khách hàng được tổng hợp từ toàn bộ các ngân hàng hoạt động trên lãnh thổ của một quốc gia vào một data center. Như vậy ngân hàng có thể kiểm tra chéo thông tin tín dụng của khách hàng từ những ngân hàng khác.
- Thông tin giao dịch: Lịch sử giao dịch trên các thẻ tín dụng hoặc thẻ ATM sẽ đánh giá một phần nào đó về năng lực tài chính của khách hàng. Do đó thông tin này rất hữu ích đối với dự báo khả năng trả nợ.
- Thông tin tài sản đảm bảo: Đây là một thông tin đi kèm với các khoản vay thế chấp. Gía trị của tài sản đảm bảo sẽ là phương tiện cover rủi ro trong trường hợp khách hàng vỡ nợ.

Ứng dụng trong xếp hạng tín nhiệm xã hội:

Ngoài ứng dụng trong rủi ro tín dụng, scorecard còn được một số quốc gia ứng dụng vào phát triển hệ thống xếp hạng tín nhiệm xã hội (social credit system) (https://en.wikipedia.org/wiki/Social_Credit_System). Để xây dựng những hệ thống này đòi hỏi phải có dữ liệu lớn tích hợp cùng các giải pháp công nghệ.

Tại Trung Quốc, chính phủ và 9 tập đoàn công nghệ hàng đầu như `Alibaba, Tencent, Didi Chuxing, Baihe, ...` đã lên kế hoạch tổ chức dữ liệu và phát triển mô hình đánh giá tín nhiệm công dân. Các dữ liệu được thu thập từ rất nhiều nguồn khác nhau như: camera giám sát, dữ liệu tài chính, dữ liệu nhân khẩu học, dữ liệu mạng xã hội,… khiến cho điểm tín nhiệm xã hội trở thành một thước đo toàn diện.

Nhờ sự những lợi thế được tạo ra khi sở hữu một điểm tín nhiệm xã hội cao như: Được hưởng dịch vụ công và phúc lợi xã hội tốt hơn, xin việc dễ hơn mà mọi người dân Trung Quốc đều có ý thức tuân thủ luật pháp. Do đó tỷ lệ tội phạm giảm xuống trên cả nước. Tuy nhiên điểm tín nhiệm xã hội cũng vấp phải sự phản đối vì nó vi phạm quyền tự do cá nhân và một số phiền hà mà nó gây ra đối với người có điểm số thấp. Nhưng dù sao thì đây cũng là một mô hình tiến bộ và đang được áp dụng rất thành công tại Trung Quốc.

Như vậy chúng ta đã hình dung được ứng dụng của mô hình score card trong rất nhiều các lĩnh vực. Vậy làm thể nào để xây dựng một mô hình score card? Tôi sẽ trình bày ở phần tiếp theo.

2. Xây dựng model Scorecard trong credit risk

Top

Ở các quốc gia đang phát triển như Việt Nam thì market trading còn ở qui mô nhỏ. Chính vì thế tín dụng là hoạt động mang lại lợi nhuận chủ yếu cho ngân hàng. Tuy nhiên mọi khoản vay đều tiềm ẩn các nguy cơ rủi ro như vỡ nợ, trả nợ chậm, người vay phá sản,…. Để giám sát, giảm thiểu, phòng ngừa và kiểm soát rủi ro thì các ngân hàng tìm cách lượng hóa rủi ro bằng các công cụ mô hình theo khuyến nghị của của hội nghị basel. Kết quả từ các mô hình rủi ro tỏ ra rất hiệu quả trong hỗ trợ quyết định cho vay của ngân hàng. Căn cứ trên điểm tín nhiệm, ngân hàng có thể chấp nhận hoặc loại bỏ hồ sơ vay hoặc đưa ra thêm các điều kiện kèm theo như hạn mức, lãi suất, kì hạn, tài sản đảm bảo,….

Vai trò của mô hình scorecard đối với tín dụng ngân hàng

Credit scorecard là phương pháp dựa trên các mô hình học máy nhằm mục đích ước lượng giá trị xác suất xảy ra một sự kiện đối với một khách hàng như vỡ nợ, phá sản, hoặc trả nợ chậm liên quan đến vị thế tín dụng của người vay. Đây là những mô hình khá đơn giản với số lượng biến đầu vào ít, chỉ khoảng từ 10 đến 20 biến. Dữ liệu được sử dụng cho các mô hình score card phần lớn đến từ cục tín dụng (credit bureaus) kết hợp với dữ liệu nội bộ ngân hàng. Hội nghị basel cũng đưa ra những qui định chặt chẽ đối với đầu vào của mô hình credit scorecard. Các thông tin thu thập phải đảm bảo quyền cá nhân và không vi phạm tới các nội dung như phân biệt tôn giáo, chủng tộc, vùng miền, quốc gia,….

Bộ dữ liệu xây dựng mô hình credit scorecard sẽ được tổng hợp từ những hồ sơ vỡ nợ và những hồ sơ tốt. Sau đó, thông qua các kĩ thuật tiền xử lý dữ liệu để biến đổi các biến đầu vào thành những features có sức mạnh dự báo nguy cơ xảy ra vỡ nợ. Lớp thuật toán được sử dụng chủ yếu trong các mô hình credit scorecard đó là hồi qui logistic, probit hoặc linear. Để đơn giản, trong bài này tôi chỉ sử dụng hồi qui logistic. Đây là những phương pháp đơn giản thuộc lớp mô hình phân loại trong học có giám sát. Chúng là những phương pháp có khả năng giải thích cao thông qua phương trình hồi qui.

Chính nhờ khả năng giải thích nên các mô hình credit scorecard có thể đánh giá được điểm tín nhiệm của từng biến độc lập (hoặc biến đầu vào). Tổng hợp các điểm tín nhiệm của biến ta thu được điểm tín nhiệm (`credit score`) cuối cùng. Một số thuật toán khác như neural network, random forest, SVM tuy có kết quả phân loại tốt hơn nhưng khả năng giải thích kết quả không tốt nên không thường được sử dụng để xây dựng các mô hình credit scorecard trên thực tế.

2.1. Phương pháp chuyên gia và mô hình

Để đánh giá rủi ro tín dụng chúng ta có thể áp dụng phương pháp chuyên gia hoặc sử dụng mô hình thống kê. Mỗi phương pháp đều có các ưu nhược điểm khác nhau.

Phương pháp chuyên gia: Như tên gọi của nó, phương pháp chuyên gia sẽ dựa trên ý kiến thẩm định của các chuyên gia về rủi ro đối với một khoản tín dụng. Rủi ro sẽ được căn cứ trên các thông tin chủ yếu đó là:

- **Đặc điểm của chủ thể vay (character):** Thẩm định danh tiếng, tính trung thực của người vay vốn.
- **Vốn (capital):** Thẩm định sự chênh lệch giữa tài sản và nguồn vốn của người cho vay. Tài sản chính là những giá trị mà ngân hàng có thể thu hồi khi người vay không trả được nợ. Nguồn vốn có thể là các chi phí mà người vay đang phải chi trả như chi tiêu gia đình, chi phí thuê nhà,…. Sau khi trừ đi các chi phí chúng ta sẽ biết được người vay sẽ tiết kiệm được bao nhiêu và chi phí đó có đủ để trang trải lãi vay hay không?
- **Tài sản đảm bảo (collateral):** Sẽ có 2 loại hình thức cho vay được phân chia dựa trên tài sản đảm bảo đó là vay thế chấp (có tài sản đảm bảo) và vay tín chấp (không có tài sản đảm bảo). Rủi ro của 2 hình thức cho vay này là khác biệt nhau nên lãi suất và hạn mức giữa chúng cũng sẽ khác biệt để đảm bảo dung hòa giữa lợi nhuận và rủi ro đối với ngân hàng. Đối với vay thế chấp ngân hàng sẽ phải định giá chính xác giá trị của các tài sản thế chấp. Gía trị các tài sản này sẽ quyết định hạn mức tín dụng mà ngân hàng sẽ cấp cho người vay. Rủi ro đối với các khoản vay thế chấp là thấp hơn tín chấp vì trong trường hợp khách hàng không có khả năng thanh toán, ngân hàng được quyền thu hồi tài sản đảm bảo.
- **Khả năng trả nợ (capacity):** Là các thông tin liên quan trực tiếp đến khả năng tài chính của người vay đó là: nghề nghiệp, mức thu nhập, trạng thái hôn nhân, số người phụ thuộc,….
- **Điều kiện (condition):** Đánh giá sơ bộ trạng thái của người vay có tham chiếu tới điều kiện thị trường, bối cảnh tài chính, áp lực cạnh tranh, mục đích sử dụng vốn,…. Chẳng hạn người vay là hộ dân trồng cafe nhưng năm vừa qua thị trường cafe giảm giá mạnh. Do đó sẽ khiến lợi nhuận và khản năng thanh toán của người vay xuống thấp hơn dự kiến.

Phương pháp chuyên gia là phương pháp thủ công vì nó dựa trên kinh nghiệm của con người. Do đó quá trình thẩm định sẽ tốn kém về thời gian. Đồng thời ý kiến đánh giá cũng không nhất quán giữa các chuyên gia. Do đó một phương pháp khác được khuyến nghị phát triển ở hội nghị basel nhằm đưa ra các đánh giá nhanh chóng và nhất quán hơn. Đó chính là phương pháp mô hình.

Phương pháp mô hình: Phương pháp mô hình sẽ dựa trên điểm số được lượng hóa từ mô hình học máy. Phương pháp này có nhiều điểm tối ưu hơn so với phương pháp chuyên gia:

- Những mô hình đưa ra kết quả dường như là ngay lập tức. Do đó thời gian thẩm định hồ sơ nhanh chóng và rất phù hợp với các nền tảng cho vay online.
- Năng suất thẩm định từ mô hình cao hơn rất nhiều so với các chuyên gia. Một môt hình có thể giải quyết số lượng hồ sơ bằng khối lượng công việc của hàng trăm chuyên gia.
- Giảm thiểu chi phí lao động khi không phải chi trả lương cho các chuyên gia thẩm định.
- Kết quả đánh giá hồ sơ là rất nhất quán dựa trên điểm số tín nhiệm là duy nhất, trong khi đó các chuyên gia có thể đưa ra kết quả đánh giá khác nhau dựa trên cảm quan của họ về rủi ro. Khi xảy ra bất đồng ý kiến, sẽ cần hội đồng chuyên gia đánh giá lại hồ sơ và khá tốn thời gian để hoàn thành thẩm định.

Top

- Mô hình sẽ xem xét toàn diện các biến số đầu vào và thậm chí có thể gia tăng số lượng biến tùy ý mà không ảnh hưởng tới thời gian dự báo. Trong khi phương pháp chuyên gia sẽ chịu hạn chế bởi khả năng của con người là có hạn. Việc đánh giá hồ sơ đôi khi chỉ được nhận định trên một số biến chính.

Chính vì những lợi thế đó, phương pháp mô hình đang dần thay thế phương pháp chuyên gia và trở thành phương pháp thẩm định chủ yếu tại các ngân hàng.

2.2. Xây dựng mô hình credit scorecard

Có nhiều thuật toán khác nhau được áp dụng để xây dựng mô hình credit scorecard. Trong bài này tôi sẽ chỉ giới thiệu phương pháp thông dụng nhất đó là hồi qui logistic. Về hồi qui logistic các bạn xem thêm tại Logistic Regression - Machine learning cơ bản (https://machinelearningcoban.com/2017/01/27/logisticregression/), rất chi tiết và đầy đủ. Quá trình hồi qui sẽ tiếp nhận các features đầu vào đã được tiền xử lý theo phương pháp trọng số dấu hiệu (WOE - weight of evidence), cụ thể sẽ được giới thiệu bên dưới. Cuối cùng đầu ra của mô hình là xác suất vỡ nợ (default probability) của một hồ sơ vay vốn. Xác suất càng cao là dấu hiệu cho thấy khả năng xảy ra rủi ro càng lớn. Từ xác suất, thông qua các phép scale ta sẽ biến đổi sang điểm số tín nhiệm (credit score) đại diện cho mức độ uy tín của khách hàng. Điểm số này bằng tổng các điểm số tương ứng với mỗi một đặc trưng của người dùng được tạo ra từ WOE.

2.2.1. Weight of Evidence - WOE

WOE (weight of evidence) là một trong những kĩ thuật feature engineering và feature selection thường được áp dụng trong mô hình scorecard. Phương pháp này sẽ xếp hạng các biến thành mạnh, trung bình, yếu, không tác động,… dựa trên khả năng, sức mạnh dự báo nợ xấu. Tiêu chuẩn xếp hạng sẽ là chỉ số giá trị thông tin IV (information value) được tính toán từ phương pháp WOE. Đồng thời mô hình cũng tạo ra các giá trị features cho mỗi biến. Giá trị này sẽ đo lường sự khác biệt trong phân phối giữa good và bad. Cụ thể như sau:

Phương pháp WOE sẽ có các kĩ thuật xử lý khác biệt đối với biến liên tục và biến phân loại:

- Trường hợp biến liên tục, WOE sẽ gán nhãn cho mỗi một quan sát theo nhãn giá trị bins mà nó thuộc về. Các bins sẽ là các khoảng liên tiếp được xác định từ biến liên tục sao cho số lượng quan sát ở mỗi bin là bằng nhau. Để xác định các bins thì ta cần xác định số lượng bins. Chúng ta có thể hình dung đầu mút của các khoảng bins chính là các quantile (https://en.wikipedia.org/wiki/Quantile). Định nghĩa quantile tôi sẽ không giải thích thêm vì đây là kiến thức cơ bản của thống kê.
- Trường hợp biến phân loại, WOE có thể cân nhắc mỗi một class là một bin hoặc có thể nhóm vài nhóm có số lượng quan sát ít vào một bin. Ngoài ra mức độ chênh lệch giữa phân phối good/bad được đo lường thông qua chỉ số WOE cũng có thể được sử dụng để nhận diện các nhóm có cùng tính chất phân loại. Nếu giá trị WOE của chúng càng gần nhau thì có thể chúng sẽ được nhóm vào một nhóm. Ngoài ra, trường hợp Null cũng có thể được coi là một nhóm riêng biệt nếu số lượng của nó là đáng kể hoặc nhóm vào các nhóm khác nếu nó là thiểu số.

Để hình dung cách tính WOE, tôi lấy ví dụ:

Độ tuổi của một khách hàng vay vốn rơi vào khoảng từ 20-60 tuổi. Ta sẽ phân chia độ tuổi này thành các bins sao cho số lượng quan sát của chúng là gần bằng nhau và thống kê số lượng hồ sơ good và bad trên từng bin đó. Sau cùng ta thu được một bảng như bên dưới:

Bins	No observation	No Good	No Bad	Good/Bad	%Good	%Bad	WOE	IV
20-30	1000	105	895	0,117	0,313	0,192	0,491	0,060
30-35	1000	90	910	0,099	0,269	0,195	0,320	0,024
35-40	1000	80	920	0,087	0,239	0,197	0,191	0,008
40-50	1000	50	950	0,053	0,149	0,204	-0,311	0,017
50+	1000	10	990	0,010	0,030	0,212	-1,961	0,358
								0,466

Bảng 1: Bảng tính toán hệ số bằng chứng WOE và giá trị thông tin IV.

Các cột có ý nghĩa như sau:

- **No observation**: Số lượng các quan sát trong từng bins. Thường sẽ được chia bằng nhau giữa các bins để không có sự thiên lệch.
- **No Good**: Số lượng hồ sơ là nợ xấu ở mỗi bins. Chúng ta coi những hồ sơ vỡ nợ là Good vì Good không phải đại diện cho chất lượng của hồ sơ mà chỉ đơn thuần đánh dấu các hồ sơ nhãn là 1.
- **No Bad**: Số lượng hồ sơ không là nợ xấu ở mỗi bins. Nhãn của hồ sơ là 0.
- **Good/Bad**: Tỷ lệ hồ sơ Good/Bad ở mỗi bins.
- **%Good**: Phân phối của các hồ sơ good trên toàn bộ các bins. Tổng cột bằng 1.
- **%Bad**: Phân phối của các hồ sơ bad trên toàn bộ các bins. Tương tự như `%Good` cũng có tổng bằng 1.
- **WOE (Weight of Evidence):** Trọng số bằng chứng được sử dụng để đo lường sự khác biệt giữa tỷ lệ `%Good` và `%Bad` trên từng bin. Trọng số bằng chứng được tính toán bằng logarit của `%Good/%Bad`. Chẳng hạn tại bin `20-30` chúng ta biết được `%Good = 0.313` và `%Bad = 0.192`, khi đó giá trị `WOE` tại bin này như sau:

Top

$$\text{WOE}_{20-30} = \ln(\frac{\%\text{Good}}{\%\text{Bad}}) = \ln(\frac{0.313}{0.192}) = 0.491$$

Tính chất của WOE: Giá trị `WOE` tại một bin càng lớn là dấu hiệu chứng tỏ đặc trưng rất tốt trong việc nhận diện hồ sơ `Good` và trái lại nếu giá trị `WOE` càng nhỏ thì đặc trưng bin sẽ rất tốt trong việc nhận diện hồ sơ `Bad` . `WOE` > 1 thì phân phối của hồ sơ `Good` đang chiếm ưu thế hơn `Bad` và trái lại.

- **IV (Information Value):** Chỉ số giá trị thông tin, có tác dụng đánh giá một biến có sức mạnh trong việc phân loại nợ xấu hay không. Công thức tính IV:

$$\sum_{i=1}^{n}(\ \%\text{Good}_i - \ \%\text{Bad}_i).\text{WOE}_i$$

Ta nhận thấy `IV` luôn nhận giá trị dương vì WOE_i và $(\ \%\text{Good}_i - \ \%\text{Bad}_i)$ đồng biến. Gía trị `IV` sẽ cho ta biết mức độ chênh lệch của `%Good` và `%Bad` ở mỗi bin là nhiều hay ít. Nếu `IV` cao thì sự khác biệt trong phân phối giữa `%Good` và `%Bad` sẽ lớn và biến hữu ích hơn trong việc phân loại hồ sơ và trái lại `IV` nhỏ thì biến ít hữu ích trong việc phân loại hồ sơ. Một số tài liệu cũng đưa ra tiêu chuẩn phân loại sức mạnh của biến theo giá trị `IV` như bên dưới:

- <= 0.02: Biến không có tác dụng trong việc phân loại hồ sơ `Good/Bad`
- 0.02 - 0.1: yếu
- 0.1 - 0.3: trung bình
- 0.3 - 0.5: mạnh
- => 0.5: Biến rất mạnh, tuy nhiên trường hợp này cần được điều tra lại để tránh trường hợp biến có mối quan hệ trực tiếp để định nghĩa hồ sơ `good/bad` .

2.2.2. Ưu và nhược điểm của phương pháp WOE

Sở dĩ các mô hình scorecard lại ưa chuộng `WOE` bởi vì đây là phương pháp biến đổi biến có nhiều ảnh hưởng tích cực tới quá trình hồi qui logistic. Các lợi thế đó là:

- `WOE` giúp biến đổi các biến độc lập liên tục thành những biến có mối quan hệ đơn điệu (đồng biến hoặc nghịch biến) đối với biến phụ thuộc. Thật vậy, giả sử thống kê cho thấy đối với độ tuổi của khách hàng trong các khoảng `20-30, 30-40, 40-50, 50+` thì tỷ lệ `%GOOD/%BAD` là `[0.1, 0.5, 0.2, 0.3]` . Như vậy `AGE` sẽ không có mối quan hệ đồng biến với tỷ lệ `%GOOD/%BAD` , tức là nó không có mối quan hệ đồng biến với biến phụ thuộc. Trong khi nếu hồi qui giá trị gốc của `AGE` theo logistic thì mối quan hệ với biến phụ thuộc của `AGE` sẽ là đơn điệu (đơn điệu tăng hay giảm phụ thuộc vào dấu của hệ số hồi qui). Điều này là trái với thực tế. Do đó, phương pháp `WOE` giúp ta chia nhỏ các biến liên tục thành các khoảng biến mà giá trị của nó là đơn điệu với biến phụ thuộc dựa trên `WOE` tương ứng với mỗi khoảng. Do đó các hệ số trong phương trình hồi qui logistic sẽ giải thích được đúng thực tế mối quan hệ giữa biến độc lập với biến phụ thuộc.
- Phương pháp `WOE` giúp loại bỏ các outliers vì các biến có khoảng biến thiên lớn sẽ được group về các nhóm bins. Gía trị của các quan sát outliers sẽ không còn khác biệt so với các những quan sát khác thuộc cùng nhóm vì chúng cùng được gán giá trị bằng trọng số WOE.
- Giá trị `WOE` phản ánh được ảnh hưởng của từng category lên biến phụ thuộc. Vì giá trị `WOE` thể hiện chênh lệch về tỷ lệ giữa `%GOOD/%BAD` , đây là chỉ số ảnh hưởng trực tiếp đến xác suất vỡ nợ của khách hàng.
- Đối với các biến quá phân tán thì `WOE` sẽ nhóm thành những nhóm thành các category và hệ số WOE thể hiện thông tin cho toàn bộ nhóm.

Chính nhờ những lợi thế trên mà WOE đã được sử dụng phổ biến trong các mô hình credit scorecard.

Tuy nhiên phương pháp WOE cũng có những hạn chế nhất định đó là:

- Khi tính toán WOE, rất khó để biết phân chia bao nhiêu bins là phù hợp đối với biến liên tục hoặc khi nào thì nên nhóm các nhóm với nhau hoặc tách nhóm.
- Do các biến WOE là luôn đơn điệu với biến phụ thuộc nên giữa các biến độc lập luôn có sự tương quan (do cùng tương quan với biến phụ thuộc). Điều này có thể dẫn đến nguy cơ đa cộng tuyến cao ảnh hưởng tới khả năng giải thích của hệ số hồi qui.
- Dễ dàng xảy ra overfitting do có thể hiệu chỉnh ảnh hưởng của biến bằng cách nhóm các categories.

2.3. Tiêu chuẩn mô hình scorecard

Vì là một mô hình liên quan đến định lượng điểm tín nhiệm khách hàng và ảnh hưởng đến quyết định cấp vốn nên khi phát triển một mô hình scorecard và cân nhắc ứng dụng mô hình đó vào thực tiễn, ngân hàng cần phải xem xét đến các khía cạnh rủi ro có thể phát sinh để đảm bảo việc áp dụng mang lại hiệu quả và hạn chế các rủi ro tiềm ẩn. Các tiêu chuẩn cần đạt được của một mô hình credit scorecard đó là:

- **Khả năng diễn giải (Interpretability):** Một mô hình có khả năng diễn giải tốt sẽ dễ dàng áp dụng vào thực tiễn hơn so với các mô hình có khả năng diễn giải kém. Chẳng hạn khi áp dụng 2 thuật toán khác nhau là hồi qui `logistic` và thuật toán `neural network` vào xây dựng một mô hình phân loại nợ xấu thì mô hình `neural network` có thể mặc dù mang lại kết quả khả quan hơn nhưng mô hình `logistic` vẫn được ưa chuộng hơn vì nó có khả năng giải thích được tác động của các đặc trưng lên điểm số cuối cùng. Trong kinh nghiệm làm việc của tôi có một dự án liên quan đến phân loại `đơn hàng đầu cơ` . Giữa rất nhiều các lớp mô hình độ chính xác cao hơn hồi qui `logistic` nhưng khi áp dụng thì khả năng giải

Top

thích tường minh của chúng gặp hạn chế. Trong khi nghiệp vụ yêu cầu mô hình có thể tự lý giải kết quả như con người, đây là một điều rất khó đạt được đối với các mô hình black box phức tạp. Chính vì thế tôi đã cân nhắc lựa chọn mô hình `logistic`.

- **Độ chính xác kì vọng (Expectation Accuracy):** Một mô hình trước khi đưa vào áp dụng cần phải đạt được một ngưỡng chính xác kì vọng. Ngưỡng chính xác này nên được thiết lập theo một metric phù hợp, chúng có thể là: Accuracy, Precision, Recall, F1, Kappa, AUC, Gini,…. là những thước đo đánh giá khả năng phân loại của mô hình. Thời điểm thiết lập kì vọng nên được tiến hành ở đầu dự án để tạo ra mục tiêu nhất quán trong suốt quá trình phát triển mô hình. Ngoài ra giá trị thiết lập nên được tham chiếu đến một số cơ sở thực tiễn chẳng hạn như: Khuyến nghị của các tổ chức tư vấn tài chính, độ chính xác mà một số ngân hàng khác đã đạt được trong cùng một thị trường, độ chính xác của chuyên gia,….
- **Chi phí xây dựng (Cost):** Để xây dựng một mô hình scorecard sẽ khá tốn kém về mặt chi phí và nguồn lực. Đội ngũ phát triển sẽ phải bỏ công sức thu thập dữ liệu, gán nhãn, xử lý dữ liệu, huấn luyện mô hình. Ngoài ra việc sử dụng một số nguồn dữ liệu ngoài như dữ liệu lịch sử tín dụng từ cục tín dụng sẽ phải mất phí. Quá trình xây dựng mô hình cần phải tuân theo chuẩn basel nên có thể sẽ phải tốn thêm chi phí tư vấn tài chính. Sau khi xây dựng mô hình sẽ phát sinh thêm các chi phí về hậu kiểm, vận hành và đánh giá lại mô hình định kì. Tổng hợp các chi phí trên sẽ là một khoản chi phí rất lớn đối với ngân hàng. Do đó cần ước tính chúng trước khi xây dựng mô hình để dự án đạt hiệu quả lớn nhất.

Như vậy việc xây dựng mô hình credit scorecard không chỉ đơn thuần là đưa dữ liệu vào và chạy. Sẽ cần rất nhiều các tiêu chuẩn khắt khe cần đạt được khi triển khai xây dựng mô hình và áp dụng mô hình vào thực tiễn. Tuy nhiên đó là những gì vượt xa ngoài mục đích giới thiệu phương pháp scorecard. Tiếp theo chúng ta sẽ cùng thử nghiệm xây dựng một mô hình scorecard.

2.4. Xây dựng model ScoreCard

Sau đây chúng ta sẽ cùng xây dựng một model scorecard từ bộ dữ liệu hmeq (http://www.creditriskanalytics.net/uploads/1/9/5/1/19511601/hmeq.csv). Thông tin cụ thể về các biến trong bộ dữ liệu này như sau:

Bộ dữ liệu HMEQ bao gồm các đặc trưng thông tin nợ quá hạn của 5960 khoản vay mua nhà. Đây là những khoản vay mà người vay sử dụng vốn chủ sở hữu làm tài sản thế chấp cơ sở. Tập dữ liệu bao gồm những trường sau:

- BAD: 1 = Hồ sơ vay là vi phạm hoặc mất khả năng trả nợ; 0 = hồ sơ vay đã và đang trả nợ.
- LOAN: Số tiền yêu cầu cho vay.
- MORTDUE: Số tiền đến hạn của khoản thế chấp hiện có.
- VALUE: Giá trị tài sản hiện tại.
- REASON: DebtCon = nợ hợp nhất; HomeImp = cải thiện nhà.
- JOB: Thể loại nghề nghiệp.
- YOJ: Số năm kinh nghiệm trong nghề nghiệp hiện tại.
- DEROG: Số lượng báo cáo không tín nhiệm.
- DELINQ: Số hạn mức tín dụng quá hạn.
- CLAGE: Tuổi của hạn mức tín dụng cũ nhất tính theo tháng.
- NINQ: Số câu hỏi tín dụng gần đây.
- CLNO: Số lượng hạn mức tín dụng.
- DEBTINC: Tỷ lệ nợ trên thu nhập.

Có tổng cộng 12 biến đầu vào bao gồm cả biến numeric và biến category. Về cơ bản số lượng quan sát là đủ lớn để xây dựng mô hình credit scorecard. Tiếp theo chúng ta sẽ khảo sát dữ liệu.

2.4.1. Khảo sát dữ liệu.

```
1  import pandas as pd
2
3  data = pd.read_csv('http://www.creditriskanalytics.net/uploads/1/9/5/1/19511601/hmeq.csv', head
```

Khảo sát dữ liệu

```
1  data.describe()
```

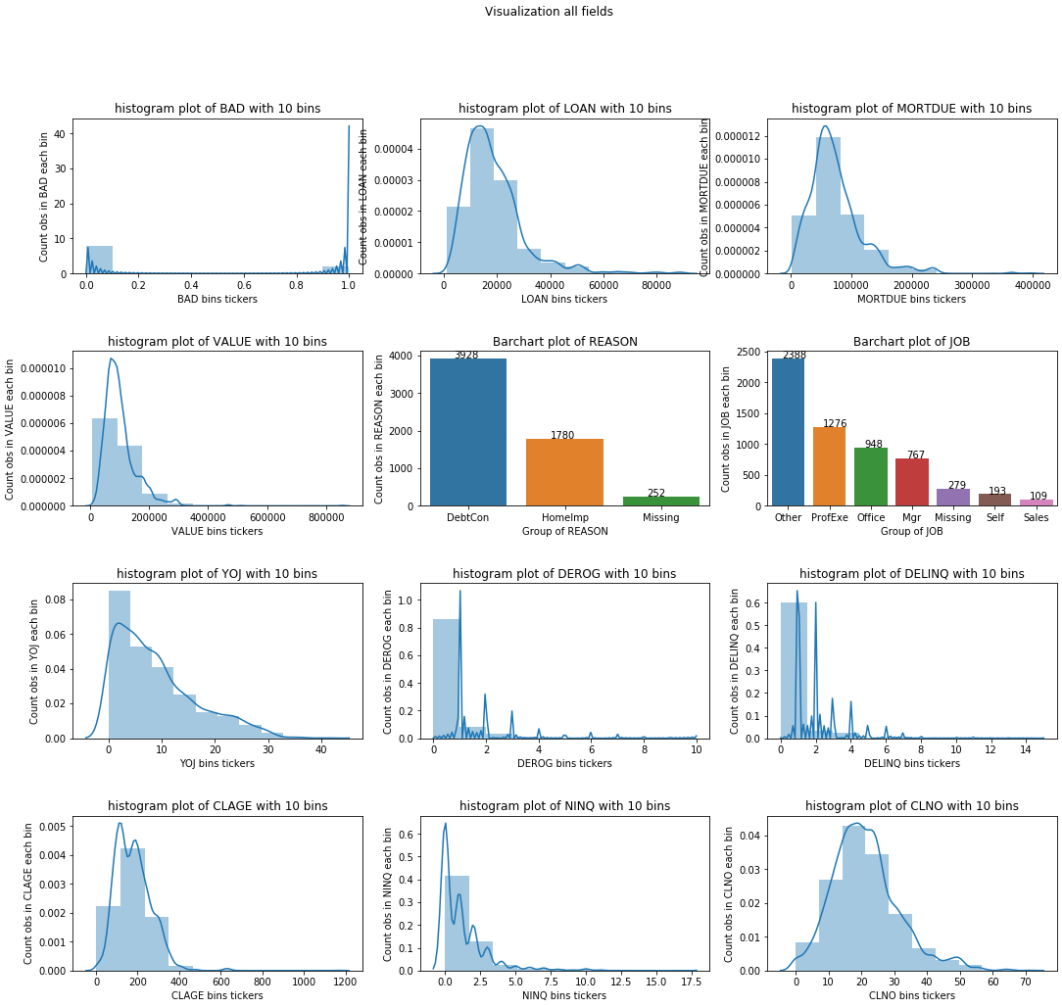
	BAD	LOAN	MORTDUE	VALUE	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
count	5960.000000	5960.000000	5442.000000	5848.000000	5445.000000	5252.000000	5380.000000	5652.000000	5450.000000	5738.000000	4693.000000
mean	0.199497	18607.969799	73760.817200	101776.048741	8.922268	0.254570	0.449442	179.766275	1.186055	21.296096	33.779915
std	0.399656	11207.480417	44457.609458	57385.775334	7.573982	0.846047	1.127266	85.810092	1.728675	10.138933	8.601746
min	0.000000	1100.000000	2063.000000	8000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.524499
25%	0.000000	11100.000000	46276.000000	66075.500000	3.000000	0.000000	0.000000	115.116702	0.000000	15.000000	29.140031
50%	0.000000	16300.000000	65019.000000	89235.500000	7.000000	0.000000	0.000000	173.466667	1.000000	20.000000	34.818262
75%	0.000000	23300.000000	91488.000000	119824.250000	13.000000	0.000000	0.000000	231.562278	2.000000	26.000000	39.003141
max	1.000000	89900.000000	399550.000000	855909.000000	41.000000	10.000000	15.000000	1168.233561	17.000000	71.000000	203.312149

mean của BAD chính là tỷ lệ số hợp đồng nợ xấu và chiếm 19.95%. Ta sẽ visualize phân phối của các biến để tìm hiểu phân phối của chúng.

Top

```
import seaborn as sns
import matplotlib.pyplot as plt

# Biểu đồ histogram
def _plot_hist_subplot(x, fieldname, bins = 10, use_kde = True):
  x = x.dropna()
  xlabel = '{} bins tickers'.format(fieldname)
  ylabel = 'Count obs in {} each bin'.format(fieldname)
  title = 'histogram plot of {} with {} bins'.format(fieldname, bins)
  ax = sns.distplot(x, bins = bins, kde = use_kde)
  ax.set_xlabel(xlabel)
  ax.set_ylabel(ylabel)
  ax.set_title(title)
  return ax

# Biểu đồ barchart
def _plot_barchart_subplot(x, fieldname):
  xlabel = 'Group of {}'.format(fieldname)
  ylabel = 'Count obs in {} each bin'.format(fieldname)
  title = 'Barchart plot of {}'.format(fieldname)
  x = x.fillna('Missing')
  df_summary = x.value_counts(dropna = False)
  y_values = df_summary.values
  x_index = df_summary.index
  ax = sns.barplot(x = x_index, y = y_values, order = x_index)
  # Tạo vòng for lấy tọa độ đỉnh trên cùng của biểu đồ và thêm label thông qua annotate.
  labels = list(set(x))
  for label, p in zip(y_values, ax.patches):
    ax.annotate(label, (p.get_x()+0.25, p.get_height()+0.15))
  plt.xlabel(xlabel)
  plt.ylabel(ylabel)
  plt.title(title)
  return ax

# Khởi tạo figure cho đồ thị (Kích thước W*H = 16x12) và cách nhau là 0.2 giữa các đồ thị
fig = plt.figure(figsize=(18, 16))
fig.subplots_adjust(hspace=0.5, wspace=0.2)
# Tạo vòng for check định dạng của biến và visualize
for i, (fieldname, dtype) in enumerate(zip(data.columns, data.dtypes.values)):
  if i <= 11:
    ax_i = fig.add_subplot(4, 3, i+1)
    if dtype in ['float64', 'int64']:
      ax_i = _plot_hist_subplot(data[fieldname], fieldname=fieldname)
    else:
      ax_i = _plot_barchart_subplot(data[fieldname], fieldname=fieldname)

fig.suptitle('Visualization all fields')
plt.show()
```

Top



Nhận xét:

- Từ biểu đồ phân phối cho ta biết hình dạng phân phối của các biến.
- Các outlier của các biến là những điểm nào?
- Hình dạng phân phối cũng giúp ta phát hiện các dị thường trong dữ liệu. Chắc hẳn các bạn còn nhớ gian lận thi cử năm 2018 tại Sơn La và Hà Giang đã được phát hiện như thế nào?

2.4.2. Áp dụng phương pháp WOE

Tiếp theo ta sẽ áp dụng phương pháp WOE vào việc xếp hạng sức mạnh dự báo của biến và tạo features cho mô hình. Nhưng trước đó sẽ cần phải xử lý dữ liệu missing data bằng giá trị trung bình.

Đối với biến numeric:

```
1     columns_num = data.select_dtypes(['float', 'int']).columns
2     data[columns_num] = data[columns_num].apply(lambda x: x.fillna(x.mean()), axis=0)
```

Đối với biến category:

```
1     columns_obj = data.select_dtypes(['object']).columns
2     data[columns_obj] = data[columns_obj].apply(lambda x: x.fillna('Missing'), axis=0)
```

Tính toán WOE cho từng biến

Hàm `_bin_table()` bên dưới sẽ có tác dụng phân chia các nhóm bins và thống kê số lượng các quan sát, số lượng good và bad ở mỗi nhóm.

Top

```
import numpy as np
import matplotlib.pyplot as plt

MAX_VAL = 999999999
MIN_VAL = -MAX_VAL

def _bin_table(data, colname, n_bins = 10, qcut = None):
  X = data[[colname, 'BAD']]
  X = X.sort_values(colname)
  coltype = X[colname].dtype

  if coltype in ['float', 'int']:
    if qcut is None:
      try:
        bins, thres = pd.qcut(X[colname], q = n_bins, retbins=True)
        # Thay thế threshold đầu và cuối của thres
        thres[0] = MIN_VAL
        thres[-1] = MAX_VAL
        bins, thres = pd.cut(X[colname], bins=thres, retbins=True)
        X['bins'] = bins
      except:
        print('n_bins must be lower to bin interval is valid!')
    else:
      bins, thres = pd.cut(X[colname], bins=qcut, retbins=True)
      X['bins'] = bins
  elif coltype == 'object':
    X['bins'] = X[colname]

  df_GB = pd.pivot_table(X,
                index = ['bins'],
                values = ['BAD'],
                columns = ['BAD'],
                aggfunc = {
                    'BAD':np.size
                })

  df_Count = pd.pivot_table(X,
                index = ['bins'],
                values = ['BAD'],
                aggfunc = {
                    'BAD': np.size
                })

  if coltype in ['float', 'int']:
    df_Thres = pd.DataFrame({'Thres':thres[1:]}, index=df_GB.index)
  elif coltype == 'object':
    df_Thres = pd.DataFrame(index=df_GB.index)
    thres = None
  df_Count.columns = ['No_Obs']
  df_GB.columns = ['#BAD', '#GOOD']
  df_summary = df_Thres.join(df_Count).join(df_GB)
  return df_summary, thres
```

Phân chia các bins theo ngưỡng cutpoints. Phù hợp với những biến thứ bậc.

```
df_summary, thres = _bin_table(data, 'DELINQ', qcut=[MIN_VAL, 2, MAX_VAL])
df_summary
```

bins	Thres	No_Obs	#BAD	#GOOD
(-999999999, 2]	2	5663	4674	989
(2, 999999999]	999999999	297	97	200

Phân chia các bins theo khai báo số lượng bins. Phù hợp với các biến liên tục.

```
df_summary, thres = _bin_table(data, 'DEBTINC', n_bins=5)
df_summary
```

Top

bins	Thres	No_Obs	#BAD	#GOOD
(-999999999.0, 29.214]	2.921447e+01	1192	1127	65
(29.214, 33.78]	3.377992e+01	2156	1316	840
(33.78, 34.688]	3.468780e+01	228	216	12
(34.688, 38.956]	3.895595e+01	1192	1102	90
(38.956, 999999999.0]	1.000000e+09	1192	1010	182

Tiếp theo hàm `_WOE()` sẽ tính toán các trọng số `WOE` ở mỗi bins và chỉ số `IV` cho từng biến. Trong trường hợp một bin có số lượng quan sát nhỏ hơn một ngưỡng tối thiểu ta sẽ ghép nó vào bin liền trước.

```
def _WOE(data, colname, n_bins = None, min_obs = 100, qcut = None):
  # Thống kê bins và lấy ra thres hold ban đầu
  df_summary, thres = _bin_table(data, colname, n_bins = n_bins, qcut = qcut)
  # Thay thế giá trị 0 của #BAD trong df_summary bằng 1 để không bị lỗi chia cho 0
  df_summary['#BAD'] = df_summary['#BAD'].replace({0:1})

  if qcut is not None:
    # Lọc bỏ threshold để tạo thành threshold mới mà thỏa mãn số lượng quan sát >= min_obs
    exclude_ind = np.where(df_summary['No_Obs'] <= min_obs)[0]
    if exclude_ind.shape[0] > 0:
      new_thres = np.delete(thres, exclude_ind)
      print('Auto combine {} bins into {} bins'.format(n_bins, new_thres.shape[0]-1))
      # Tính toán lại bảng summary
      df_summary, thres = _bin_table(data, colname, qcut=new_thres)

  new_thres = thres
  df_summary['GOOD/BAD'] = df_summary['#GOOD']/df_summary['#BAD']
  df_summary['%BAD'] = df_summary['#BAD']/df_summary['#BAD'].sum()
  df_summary['%GOOD'] = df_summary['#GOOD']/df_summary['#GOOD'].sum()
  df_summary['WOE'] = np.log(df_summary['%GOOD']/df_summary['%BAD'])
  df_summary['IV'] = (df_summary['%GOOD']-df_summary['%BAD'])*df_summary['WOE']
  df_summary['COLUMN'] = colname
  IV = df_summary['IV'].sum()
  print('Information Value of {} column: {}'.format(colname, IV))
  return df_summary, IV, new_thres

df_summary, IV, thres = _WOE(data, 'DEBTINC', n_bins = 7, min_obs= 100)
df_summary
```

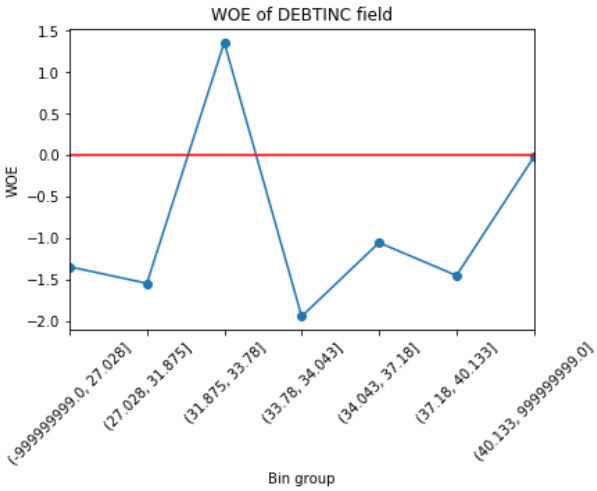
Information Value of DEBTINC column: 1.3795573580411762

bins	Thres	No_Obs	#BAD	#GOOD	GOOD/BAD	%BAD	%GOOD	WOE	IV	COLUMN
(-999999999.0, 27.028]	2.702826e+01	852	800	52	0.065000	0.167680	0.043734	-1.343925	0.166573	DEBTINC
(27.028, 31.875]	3.187511e+01	851	808	43	0.053218	0.169357	0.036165	-1.543919	0.205637	DEBTINC
(31.875, 33.78]	3.377992e+01	1645	835	810	0.970060	0.175016	0.681245	1.359046	0.687988	DEBTINC
(33.78, 34.043]	3.404341e+01	58	56	2	0.035714	0.011738	0.001682	-1.942761	0.019535	DEBTINC
(34.043, 37.18]	3.717985e+01	851	783	68	0.086845	0.164117	0.057191	-1.054182	0.112719	DEBTINC
(37.18, 40.133]	4.013282e+01	851	804	47	0.058458	0.168518	0.039529	-1.450008	0.187035	DEBTINC
(40.133, 999999999.0]	1.000000e+09	852	685	167	0.243796	0.143576	0.140454	-0.021982	0.000069	DEBTINC

Vẽ biểu đồ giá trị WOE của các bins.

```
def _plot(df_summary):
  colname = list(df_summary['COLUMN'].unique())[0]
  df_summary['WOE'].plot(linestyle='-', marker='o')
  plt.title('WOE of {} field'.format(colname))
  plt.axhline(y=0, color = 'red')
  plt.xticks(rotation=45)
  plt.ylabel('WOE')
  plt.xlabel('Bin group')

_plot(df_summary)
```

Top



Tiếp theo ta sẽ tính toán giá trị WOE cho toàn bộ các biến.

```
# Đối với các biến numeric
WOE_dict=dict()
nbins = {'LOAN': 10, 'MORTDUE': 10, 'VALUE': 10, 'YOJ':10, 'CLAGE':10, 'NINQ': 2, 'CLNO':10, 'D
for (col, bins) in nbins.items():
  df_summary, IV, thres = _WOE(data, colname=col, n_bins=bins)
  WOE_dict[col] = {'table':df_summary, 'IV':IV}
```

```
Information Value of LOAN column: 0.1601563338988017
Information Value of MORTDUE column: 0.05131351983314017
Information Value of VALUE column: 0.14188912125986042
Information Value of YOJ column: 0.06714693781095009
Information Value of CLAGE column: 0.22171042878294653
Information Value of NINQ column: 0.06965935231976197
Information Value of CLNO column: 0.06043698467606807
Information Value of DEBTINC column: 1.3795573580411762
```

Do các biến `DEROG, DELINQ` có xu hướng là biến thứ bậc hơn là biến liên tục nên áp dụng cách phân chia theo quantile sẽ tạo ra những khoảng bins có độ dài bằng 0. Do đó chúng ta sẽ phân chia theo ngưỡng cutpoint.

```
for col in ['DEROG', 'DELINQ']:
  df_summary, IV, thres = _WOE(data, colname=col, n_bins=5, qcut=[MIN_VAL, 2, MAX_VAL])
  WOE_dict[col] = {'table':df_summary, 'IV':IV}
```

```
Information Value of DEROG column: 0.19008112833205368
Information Value of DELINQ column: 0.3366699247263777
```

Tiếp theo ta sẽ tính toán IV cho các biến category là `REASON` và `JOB` .

```
for col in ['REASON', 'JOB']:
  df_summary, IV, thres = _WOE(data, colname=col)
  WOE_dict[col] = {'table':df_summary, 'IV':IV}
```

```
Information Value of REASON column: 0.008618460238864025
Information Value of JOB column: 0.1237305657142077
```

2.4.3. Xếp hạng các biến theo sức mạnh dự báo

Dựa trên giá trị IV đã tính toán ở bước trước, ta sẽ xếp hạng các biến này như bên dưới.

Top

```
columns = []
IVs = []
for col in data.columns:
  if col != 'BAD':
    columns.append(col)
    IVs.append(WOE_dict[col]['IV'])
df_WOE = pd.DataFrame({'column': columns, 'IV': IVs})

def _rank_IV(iv):
  if iv <= 0.02:
    return 'Useless'
  elif iv <= 0.1:
    return 'Weak'
  elif iv <= 0.3:
    return 'Medium'
  elif iv <= 0.5:
    return 'Strong'
  else:
    return 'suspicious'

df_WOE['rank']=df_WOE['IV'].apply(lambda x: _rank_IV(x))
df_WOE.sort_values('IV', ascending=False)
```

	column	IV	rank
11	DEBTINC	1.379557	suspicious
7	DELINQ	0.336670	Strong
8	CLAGE	0.221710	Medium
6	DEROG	0.190081	Medium
0	LOAN	0.160156	Medium
2	VALUE	0.141889	Medium
4	JOB	0.123731	Medium
9	NINQ	0.069659	Weak
5	YOJ	0.067147	Weak
10	CLNO	0.060437	Weak
1	MORTDUE	0.051314	Weak
3	REASON	0.008618	Useless

Như vậy trong các biến trên, biến `REASON` không có tác dụng trong việc phân loại hồ sơ nợ xấu. Các biến còn lại đều có tác dụng hỗ trợ một phần phân loại hồ sơ. Trong đó các biến có sức mạnh nhất là `DELINQ, DEBTINC` . Tiếp theo `CLAGE, DEROG, LOAN, VALUE, JOB` là các biến có sức mạnh trung bình. Các biến còn lại bao gồm `NINQ, YOJ, CLNO` và `MORTDUE` cũng có sức mạnh phân loại nhưng yếu hơn. `DELINQ` là biến có tương quan rất lớn đến việc phân loại nên chúng ta cần phải review lại giá trị của biến.

2.4.4. Hồi qui logistic

Phương trình hồi qui logistic trong credit scorecard sẽ không hồi qui trực tiếp trên các biến gốc mà thay vào đó giá trị WOE ở từng biến sẽ được sử dụng thay thế để làm đầu vào. Ta sẽ tính toán các biến WOE bằng cách map mỗi khoảng bin tương ứng với giá trị WOE của nó như sau:

```
for col in WOE_dict.keys():
  try:
    key = list(WOE_dict[col]['table']['WOE'].index)
    woe = list(WOE_dict[col]['table']['WOE'])
    d = dict(zip(key, woe))
    col_woe = col+'_WOE'
    data[col_woe] = data[col].map(d)
  except:
    print(col)
```

Gán giá trị input là các biến WOE và biến mục tiêu là data[‘BAD’].

```
X = data.filter(like='_WOE', axis = 1)
y = data['BAD']
```

Phân chia tập train/test có tỷ lệ kích thước mẫu là 80:20. Tỷ lệ của GOOD/BAD là cân bằng trên cả train và test.

Top

```
from sklearn.model_selection import train_test_split

ids = np.arange(X.shape[0])
X_train, X_test, y_train, y_test, id_train, id_test = train_test_split(X, y, ids, test_size = 0

print('X_train shape: ', X_train.shape)
print('X_test shape: ', X_test.shape)
print('y_train shape: ', y_train.shape)
print('y_test shape: ', y_test.shape)
```

```
X_train shape:  (4768, 12)
X_test shape:  (1192, 12)
y_train shape:  (4768,)
y_test shape:  (1192,)
```

Xây dựng phương trình hồi qui logistic các biến đầu vào `WOE` .

```
from sklearn.linear_model import LogisticRegression

logit_model = LogisticRegression(solver = 'lbfgs', max_iter=1000, fit_intercept=True, tol=0.000
logit_model.fit(X_train, y_train)
```

```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Dự báo và kiểm tra accuracy trên tập train/test

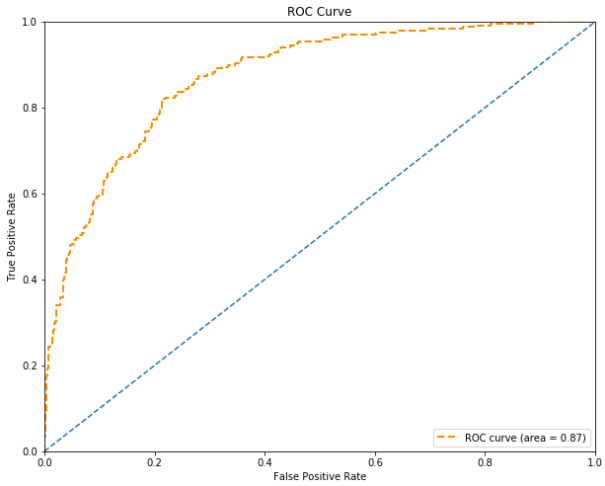
```
from sklearn.metrics import accuracy_score

y_pred_train = logit_model.predict(X_train)
acc_train = accuracy_score(y_pred_train, y_train)
y_pred_test = logit_model.predict(X_test)
acc_test = accuracy_score(y_pred_test, y_test)

print('accuracy on train: ', acc_train)
print('accuracy on test: ', acc_test)
```

```
accuracy on train:  0.8655620805369127
accuracy on test:  0.8557046979865772
```

Đường cong ROC trên tập test

```
 from sklearn.metrics import roc_curve, auc

 y_pred_prob_test = logit_model.predict_proba(X_test)[:, 1]
 fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
 roc_auc = auc(fpr, tpr)

 def _plot_roc_curve(fpr, tpr, thres, auc):
     plt.figure(figsize = (10, 8))
     plt.plot(fpr, tpr, 'b-', color='darkorange', lw=2, linestyle='--', label='ROC curve (area
     plt.plot([0, 1], [0, 1], '--')
     plt.axis([0, 1, 0, 1])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.legend(loc='lower right')
     plt.title('ROC Curve')

 _plot_roc_curve(fpr, tpr, thres, roc_auc)
```

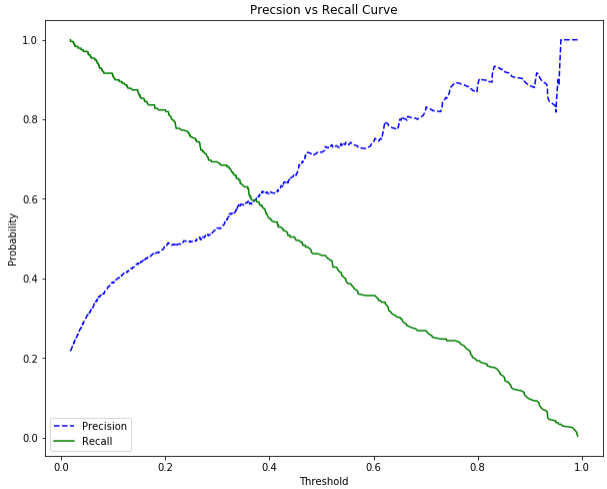
Top



Chỉ số AUC (area under curve) đo lường phần diện tích nằm dưới đường cong ROC cho biết khả năng phân loại của các hợp đồng GOOD/BAD của mô hình hồi qui logistic là mạnh hay yếu. AUC $\in [0, 1]$, giá trị của nó càng lớn thì mô hình càng tốt. Đối với mô hình hồi qui logistic này, AUC = 0.87 là khá cao, cho thấy khả năng dự báo của mô hình tốt và có thể áp dụng mô hình vào thực tiễn.

Đường cong precision và recall trên tập test

```
from sklearn.metrics import precision_recall_curve
precision, recall, thres = precision_recall_curve(y_test, y_pred_prob_test)

def _plot_prec_rec_curve(prec, rec, thres):
    plt.figure(figsize = (10, 8))
    plt.plot(thres, prec[:-1], 'b--', label = 'Precision')
    plt.plot(thres, rec[:-1], 'g-', label = 'Recall')
    plt.xlabel('Threshold')
    plt.ylabel('Probability')
    plt.title('Precsion vs Recall Curve')
    plt.legend()

_plot_prec_rec_curve(precision, recall, thres)
```



Đường cong precision, recall giúp lựa chọn ngưỡng xác suất phù hợp để mang lại độ chính xác cao hơn trong precision hoặc recall. Precision cho ta biết tỷ lệ dự báo chính xác trong số các hồ sơ được dự báo là GOOD (tức nhãn là 1). Recall đo lường tỷ lệ dự báo chính xác các hồ sơ GOOD trên thực tế. Luôn có sự đánh đổi giữa 2 tỷ lệ này, nên ta cần phải dựa vào biểu đồ của 2 đường `precision vs recall` để tìm ra ngưỡng tối ưu. Thông thường sẽ dựa trên kì vọng về precision hoặc recall từ trước để lựa chọn ngưỡng threshold. Chẳng hạn trước khi bước vào dự án ta kì vọng tỷ lệ dự báo đúng hồ sơ GOOD là 70% thì cần thiết lập các threshold để recall >= 70%. Trường hợp khác, ta kì vọng tỷ lệ dự báo đúng trong số các hồ sơ được dự báo là GOOD là 70% thì cần lựa chọn các threshold để precision >= 70%. Rất khó để nói ngưỡng threshold nào nên được lựa chọn là tốt nhất. Điều này phụ thuộc vào mục tiêu của mô hình là ưu tiên phân loại đúng hồ sơ GOOD hay hồ sơ BAD hơn.

Kiểm định Kolmogorov-Smirnov:

Đây là kiểm định đo lường sự khác biệt trong phân phối giữa GOOD và BAD theo các tỷ lệ ngưỡng threshold. Nếu mô hình có khả năng phân loại GOOD và BAD tốt thì đường cong phân phối xác suất tích lũy (cumulative distribution function - cdf) giữa GOOD và BAD phải có sự tách biệt lớn. Trái lại, nếu mô hình rất yếu và kết quả dự báo của nó chỉ ngang bằng một phép lựa

Top

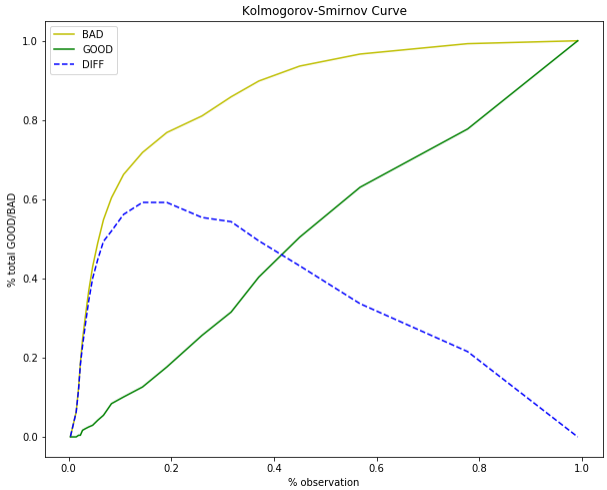
chọn ngẫu nhiên. Khi đó đường phân phối xác suất tích lũy của GOOD và BAD sẽ nằm sát nhau và tiệm cận đường chéo 45 độ. Kiểm định Kolmogorov-Smirnov sẽ kiểm tra giả thuyết Ho là hai phân phối xác suất GOOD và BAD không có sự khác biệt. Khi P-value < 0.05 bác bỏ giả thuyết Ho.

Tính toán phân phối xác suất tích lũy của GOOD và BAD

```
def _KM(y_pred, n_bins):
  _, thresholds = pd.qcut(y_pred, q=n_bins, retbins=True)
  cmd_BAD = []
  cmd_GOOD = []
  BAD_id = set(np.where(y_test == 0)[0])
  GOOD_id = set(np.where(y_test == 1)[0])
  total_BAD = len(BAD_id)
  total_GOOD = len(GOOD_id)
  for thres in thresholds:
    pred_id = set(np.where(y_pred <= thres)[0])
    # Đếm % số lượng hồ sơ BAD có xác suất dự báo nhỏ hơn hoặc bằng thres
    per_BAD = len(pred_id.intersection(BAD_id))/total_BAD
    cmd_BAD.append(per_BAD)
    # Đếm % số lượng hồ sơ GOOD có xác suất dự báo nhỏ hơn hoặc bằng thres
    per_GOOD = len(pred_id.intersection(GOOD_id))/total_GOOD
    cmd_GOOD.append(per_GOOD)
  cmd_BAD = np.array(cmd_BAD)
  cmd_GOOD = np.array(cmd_GOOD)
  return cmd_BAD, cmd_GOOD, thresholds

cmd_BAD, cmd_GOOD, thresholds = _KM(y_pred_prob_test, n_bins=20)
```

Biểu đồ phân phối xác suất tích lũy của GOOD và BAD

```
def _plot_KM(cmd_BAD, cmd_GOOD, thresholds):
  plt.figure(figsize = (10, 8))
  plt.plot(thresholds, cmd_BAD, 'y-', label = 'BAD')
  plt.plot(thresholds, cmd_GOOD, 'g-', label = 'GOOD')
  plt.plot(thresholds, cmd_BAD-cmd_GOOD, 'b--', label = 'DIFF')
  plt.xlabel('% observation')
  plt.ylabel('% total GOOD/BAD')
  plt.title('Kolmogorov-Smirnov Curve')
  plt.legend()

_plot_KM(cmd_BAD, cmd_GOOD, thresholds)
```



Kiểm định Kolmogorov-Smirnov test:

```
from scipy import stats

stats.ks_2samp(cmd_BAD, cmd_GOOD)
```

```
Ks_2sampResult(statistic=0.5238095238095238, pvalue=0.005467427576534314)
```

p-value < 0.05 cho thấy phân phối tích lũy giữa tỷ lệ BAD và GOOD là khác biệt nhau. Do đó mô hình có ý nghĩa trong phân loại hồ sơ.

Top

2.4.5. Tính điểm credit score cho mỗi feature

Bước cuối cùng là tính ra điểm tín nhiệm (credit scorecard) của mỗi khách hàng bằng cách tính điểm số cho mỗi feature (feature ở đây là một khoảng bin của biến liên tục hoặc một class của biến category). Điểm sẽ được scale theo công thức sau:

$$\text{Score} = (\beta . \text{WOE} + \frac{\alpha}{n}) . \text{Factor} + \frac{\text{Offset}}{n} \tag{1}$$

Trong đó:

- β: Hệ số của biến trong phương trình hồi qui logistic.
- α: Hệ số chặn của phương trình hồi qui logistic.
- WOE: Hệ số trọng số bằng chứng của mỗi feature.
- n: Số lượng các biến của mô hình.
- $\text{Factor}, \text{Offset}$: Là các tham số được thiết lập để tính Score.

4 tham số đầu tiên là những tham số đã biết. Hai tham số $\text{Factor}, \text{Offset}$ được tính từ các tham số pdo và Odds:

- $\text{Factor} = \frac{\text{pdo}}{\ln(2)}$
- $\text{Offset} = \text{Base\_Score} - (\text{Factor} . \ln(\text{Odds}))$

Như chúng ta đã biết, Odds chính là tỷ lệ giữa xác suất GOOD/BAD.

Gỉa sử xác suất để hợp đồng là GOOD bằng p thì tỷ lệ:

$$\text{Odds} = \frac{p}{1-p}$$

p được tính theo hàm sigmoid. Nên giá trị:

$$\ln(\text{Odds}) = \ln\left(\frac{p}{1-p}\right) = \ln\left(e^{\beta \mathbf{X}}\right) = \beta \mathbf{X}$$

Nếu coi $\frac{\text{Offset}}{n} = C$ là một hằng số. Như vậy phương trình (1) ta có thể viết thành:

$$\text{Score} = (\ln(\text{Odds}) * \text{Factor}) + C$$

Lấy đạo hàm:

$$\text{Factor} = \frac{\delta \text{Score}}{\delta \ln(\text{Odds})}$$

Ý nghĩa của Factor:

Gỉa sử với mức điểm cơ sở 600 thì tỷ lệ odds ratio là $1:30$. Điểm càng cao thì hồ sơ càng tốt, do đó khi mức điểm giảm xuống còn 580 thì tỷ lệ hồ sơ xấu (nhãn `GOOD`) tăng lên và khiến cho tỷ lệ odds ratio tăng gấp đôi thành $1:15$. Chúng ta có thể hiểu $\text{pdo} = -20$ (point double odds ratio) chính là điểm thay đổi để gấp đôi odds ratio. Do đó:

$$\text{Factor} = \frac{\delta \text{Score}}{\delta \ln(\text{Odds})} = \frac{\text{pdo}}{\ln(2)} \tag{2}$$

Như vậy Factor chính là tác động biên của việc gia tăng điểm số theo odds ratio.

Offset có thể được xem như phần bù của điểm tín nhiệm cơ sở Base_Score sau khi trừ đi tổ hợp tuyến tính đầu vào với nhân tố Factor.

Để hình dung rõ hơn quá trình tính điểm score chúng ta lấy ví dụ:

Một mô hình credit scorecard có các tham số thiết lập gồm tỷ lệ $\text{Odds} = 1:50$ tại $\text{Base\_Score} = 600$ điểm và $\text{pdo} = 20$, hai tham số $\text{Factor}, \text{Offset}$ được tính như sau:

- $\text{Factor} = \frac{20}{\ln(2)} = 28.85$
- $\text{Offset} = 600 - 28.85 \times \ln(1/50) = 712.86$

Phương trình hồi qui có hệ số đối với 1 biến bất kì là $\beta = 0.5$, hệ số tự do $\alpha = 1$, giá trị $\text{WOE} = 0.15$ và số lượng các biến $n = 12$. Khi đó áp dụng phương trình (1) tính điểm tín nhiệm Score được đóng góp bởi biến đó sẽ là:

$$\text{Score} = (0.5 \times 0.15 + \frac{1}{12}) . 28.85 + \frac{712.85}{12} = 63.97$$

Ta có thể tạo ra hàm số tính điểm cho mỗi feature như sau:

Top

```
import numpy as np

def _CreditScore(beta, alpha, woe, n = 12, odds = 1/4, pdo = -50, thres_score = 600):
  factor = pdo/np.log(2)
  offset = thres_score - factor*np.log(odds)
  score = (beta*woe+alpha/n)*factor+offset/n
  return score

_CreditScore(beta = 0.5, alpha = -1, woe = 0.15, n = 12)
```

```
42.2677896003704
```

Gán các giá trị β và α vào dictionary.

```
betas_dict = dict(zip(list(X_train.columns), logit_model.coef_[0]))
alpha = logit_model.intercept_[0]
betas_dict
```

```
{'CLAGE_WOE': 0.8622728924386177,
 'CLNO_WOE': 0.8543750687914803,
 'DEBTINC_WOE': 0.9440898350669948,
 'DELINQ_WOE': 1.0549527325475119,
 'DEROG_WOE': 0.9016268860831601,
 'JOB_WOE': 0.8871024491457594,
 'LOAN_WOE': 0.47103241070043256,
 'MORTDUE_WOE': 0.5825609027920614,
 'NINQ_WOE': 0.5431039926781271,
 'REASON_WOE': 0.6474770785463775,
 'VALUE_WOE': 0.7543949126408607,
 'YOJ_WOE': 0.8837887914052942}
```

Tính `WOE` cho từng features.

```
cols = []
features = []
woes = []
betas = []
scores = []

for col in columns:
  for feature, woe in WOE_dict[col]['table']['WOE'].to_frame().iterrows():
      cols.append(col)
      # Add feature
      feature = str(feature)
      features.append(feature)
      # Add woe
      woe = woe.values[0]
      woes.append(woe)
      # Add beta
      col_woe = col+'_WOE'
      beta = betas_dict[col_woe]
      betas.append(beta)
      # Add score
      score = _CreditScore(beta = beta, alpha = alpha, woe = woe, n = 12)
      scores.append(score)

df_WOE = pd.DataFrame({'Columns': cols, 'Features': features, 'WOE': woes, 'Betas':betas, 'Sco
df_WOE.head()
```

	Columns	Features	WOE	Betas	Scores
0	LOAN	(-999999999.0, 7600.0]	0.898910	0.471032	19.446581
1	LOAN	(7600.0, 10000.0]	0.154566	0.471032	44.737736
2	LOAN	(10000.0, 12100.0]	-0.112535	0.471032	53.813262
3	LOAN	(12100.0, 14400.0]	-0.158318	0.471032	55.368853
4	LOAN	(14400.0, 16300.0]	0.162082	0.471032	44.482362

Top

Như vậy ta đã hoàn thiện bảng tính điểm số cho mỗi features. Từ điểm số này ta có thể suy ra điểm tín nhiệm của mỗi một hồ sơ bằng cách tính tổng điểm số của toàn bộ các features của hồ sơ đó. Bên dưới ta sẽ thực hành tính điểm tín nhiệm cho một hồ sơ ngẫu nhiên:

```
# Giả sử một hồ sơ ngẫu nhiên có các thông số như sau
test_obs = data[columns].iloc[0:1, :]
test_obs
```

	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
0	1100	25860.0	39025.0	HomeImp	Other	10.5	0.0	0.0	94.366667	1.0	9.0	33.779915

Viết hàm tính toán điểm số cho mỗi trường của một bộ hồ sơ:

```
def _search_score(obs, col):
  feature = [str(inter) for inter in list(WOE_dict[col]['table'].index) if obs[col].values[0] i
  score = df_WOE[(df_WOE['Columns'] == col) & (df_WOE['Features'] == feature)]['Scores'].values
  return score

# Tính điểm cho trường 'LOAN' của bộ hồ sơ test
score = _search_score(test_obs, 'LOAN')
score
```

```
19.44658078049054
```

Điểm chi tiết của từng trường trong bộ hồ sơ và điểm của toàn bộ bộ hồ sơ sẽ là:

```
 def _total_score(obs, columns = columns):
   scores = dict()
   for col in columns:
     scores[col] = _search_score(obs, col)
   total_score = sum(scores.values())
   return scores, total_score

 scores, total_score = _total_score(test_obs)
 print('score for each fields: \n', scores)
 print('final total score: ', total_score)
```

```
score for each fields:
 {'LOAN': 19.44658078049054, 'MORTDUE': 33.012513466194235, 'VALUE': 15.69994163833087, 'REASON
final total score:  372.6213833053248
```

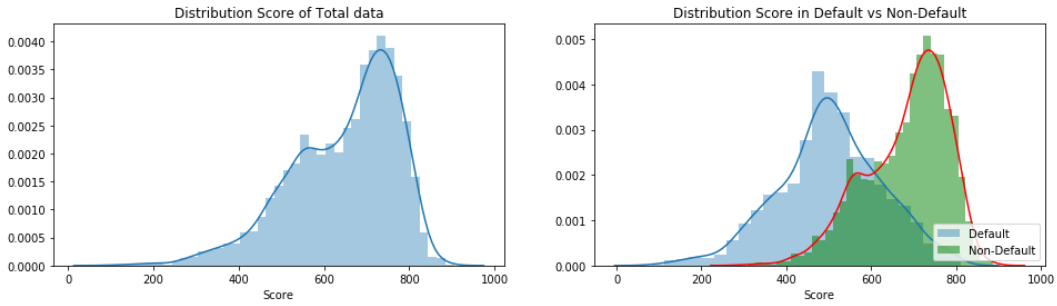
Ta có thể tính toán điểm tín nhiệm cho toàn bộ các hồ sơ trên tập data như sau:

```
total_scores = []
for i in np.arange(data[columns].shape[0]):
  obs = data[columns].iloc[i:(i+1), :]
  _, score = _total_score(obs)
  total_scores.append(score)
data['Score'] = total_scores
```

Biểu đồ phân phối của điểm số theo GOOD và BAD

```
 plt.figure(figsize=(16, 4))
 plt.subplot(121)
 sns.distplot(data['Score'])
 plt.title('Distribution Score of Total data')
 plt.subplot(122)
 sns.distplot(data[data['BAD']==1]['Score'], label='Default')
 sns.distplot(data[data['BAD']==0]['Score'], label='Non-Default',
              kde_kws={"color": "r"},
              hist_kws={"color": "g", "alpha":0.5})
 plt.legend(loc = 'lower right')
 plt.title('Distribution Score in Default vs Non-Default')
```

Top



Như vậy ta có thể nhận thấy phân phối điểm số của hồ sơ Default và Non-Default là khác biệt nhau. Điểm số tín nhiệm đánh giá mức độ tin cậy để khách hàng có hành vi cam kết trả nợ cho ngân hàng. Một khách hàng có điểm số tín nhiệm cao sẽ gia tăng mức độ tin tưởng của Ngân hàng dành cho họ. Ngoài ra Ngân hàng cũng có thể dựa trên điểm tín nhiệm để phân loại khách hàng thành những nhóm tiềm năng khác nhau giống như chỉ số FICO score (https://www.experian.com/blogs/ask-experian/credit-education/score-basics/what-is-a-good-credit-score/) đã phân loại:

Hình 1: Tỷ lệ phân chia các nhóm khách hàng theo ngưỡng điểm tín nhiệm.

Bên dưới là bảng thống kê đối với danh mục khách hàng vay vốn theo điểm scorecard.

Credit Score	Rating	% of People	Impact
300-579	Very Poor	16%	Credit applicants may be required to pay a fee or deposit, and applicants with this rating may not be approved for credit at all.
580-669	Fair	17%	Applicants with scores in this range are considered to be subprime borrowers.
670-739	Good	21%	Only 8% of applicants in this score range are likely to become seriously delinquent in the future.
740-799	Very Good	25%	Applicants with scores here are likely to receive better than average rates from lenders.
800-850	Exceptional	21%	Applicants with scores in this range are at the top of the list for the best rates from lenders.

Bảng 2: Bảng ảnh hưởng của các nhóm khách hàng theo điểm tín nhiệm.

Để giảm thiểu rủi ro thì Ngân hàng có thể đưa ra các điều kiện ràng buộc khi cho vay. Chẳng hạn như đối với khách hàng rơi vào nhóm `Very Poor` , Ngân hàng có thể đề nghị khách trả phí vay và tiền đặt cọc hoặc thậm chí là không cho vay với nhóm khách hàng này.

3. Kết luận

Như vậy tôi đã giới thiệu tới các bạn các bước để xây dựng một mô hình credit scorecard từ các bước như thu thập dữ liệu, xử lý biến, hồi qui và hậu kiểm mô hình, dự báo xác suất, tính toán điểm tín nhiệm cho một hồ sơ. Về cơ bản lý thuyết về mô hình credit scorecard là khá đơn giản vì nó chỉ dựa trên phương trình hồi qui logistic. Tuy nhiên quá trình xây dựng mô hình đòi hỏi phải đáp ứng được rất nhiều tiêu chí như khả năng giải thích, độ chính xác, chi phí xây dựng, khả năng áp dụng vào thực tiễn. Điều này đòi hỏi chúng ta phải bỏ ra nhiều công sức hơn để hoàn thiện mô hình credit scorecard chứ không chỉ đơn thuần là đưa dữ liệu hồi qui mô hình là xong.

Lớp mô hình credit scorecard ngoài áp dụng trong lĩnh vực quản trị rủi ro tín dụng còn có thể mở rộng và áp dụng sang rất nhiều các lĩnh vực khác như xếp hạng khách hàng trong marketing, xếp hạng doanh nghiệp, xếp hạng quốc gia, tổ chức tài chính,… Thậm chí mô hình còn rất tiềm năng trong quản lý xã hội. Đã được áp dụng tại một số quốc gia và mang lại hiệu quả tích cực.

Để hiểu hơn về mô hình credit scorecard các bạn có thể tham khảo thêm những nguồn tài liệu rất chất lượng bên dưới.

4. Tài liệu tham khảo

Top

1. FICO score (https://www.experian.com/blogs/ask-experian/credit-education/score-basics/what-is-a-good-credit-score/)
2. WOE - Weight of Evidence (https://multithreaded.stitchfix.com/blog/2015/08/13/weight-of-evidence/)

3. Scorecard model - Nguyen Chi Dung VietNam (https://rpubs.com/chidungkt/442168)
4. Introduction to credit scoring and scorecard approach - Jenny Nguyen (https://rstudio-pubs-static.s3.amazonaws.com/441239_e4a46ef90bd5421bb810e95c06f22df8.html#introduction_to_credit_scoring_and_scorecard_approac
5. Introduction to credit scoring and scorecard approach- Nguyen Chi Dung VietNam (https://rpubs.com/chidungkt/442168)
6. Attribute relevance analysis in python IV and WOE - R language (https://towardsdatascience.com/attribute-relevance-analysis-in-python-iv-and-woe-b5651443fc04)
7. Scorecard package in R language (https://cran.r-project.org/web/packages/scorecard/scorecard.pdf)
8. Scorecard package in python (https://pypi.org/project/scorecardpy/)
9. Creditworthiness - investopedia (https://www.investopedia.com/terms/c/credit-worthiness.asp)
10. Credit Risk Scorecards Implementing Intelligent - Naeem Siddiqui (https://www.amazon.com/Credit-Risk-Scorecards-Implementing-Intelligent/dp/047175451X)
11. Credit Scoring Application at Banks: Mapping to Basel II - Do Hoai Linh, Luong Thi Thu Hang, Nguyen Xuan Thang - NEU University VietNam (https://www.researchgate.net/publication/332098964_Credit_Scoring_Application_at_Banks_Mapping_to_Basel_II)
12. Credit Risk Factor Modeling and the Basel II IRB Approach (https://epub.uni-regensburg.de/8241/1/hamerle1.pdf)

Top