Vehicle Insurance Interested Prediction

Nguyễn Trọng Thuận[1,2], Võ Thành Trung Dũng[1,2],
Nguyễn Thanh Tường Vi[1,2], Đỗ Trọng Hợp[1,2]

[1] Trường Đại học Công nghệ Thông tin
[2] Đại học Quốc gia Thành phố Hồ Chí Minh
{18521471,18520641,18521636}@gm.uit.edu.vn,
{hopdt}@uit.edu.vn,

Tóm tắt nội dung Để chiếm được thị phần, giữ vững được khách hàng truyền thống và phát triển hợp đồng mới thì việc tìm kiếm nguồn khách hàng tiềm năng càng trở nên cần thiết đối với các công ty Bảo hiểm. Về mặt tài chính, các hợp đồng Bảo hiểm Xe sẽ giúp khách hàng chi trả quyền lợi bồi thường cho những thiệt hại về người và tài sản do lỗi của chủ phương tiện gây ra khi tham gia giao thông theo đúng quy định của pháp luật. Vì vậy, các doanh nghiệp cần thực hiện nhiều hình thức kinh doanh từ việc quảng cáo đến phân tích nhu cầu khách hàng để thu được lợi nhuận tối đa. Ở đây, chúng tôi giúp doanh nghiệp phân tích nguồn khách hàng tiềm năng từ dữ liệu do công ty cung cấp bằng cách sử dụng các công cụ phân tích dữ liệu, huấn luyện các mô hình học máy để phân khúc khách hàng. Qua đó các công ty có thể đưa ra các chiến lược phù hợp.

Keywords: Apache Spark · Binary Classification · Vehicle Insurance Interested Prediction

1 Giới thiệu

Hiện nay, thực trạng giao thông đường bộ ở nhiều nơi trên thế giới đang ngày càng phức tạp, đặc biệt là các nước đang phát triển. Khi lưu thông trên đường ta sẽ không thể tránh khỏi những lần va quẹt hoặc gặp tai nạn do ý thức người tham gia giao thông, hệ thống cầu đường không chắc chắn, do thời tiết,... Và mức phí thanh toán cho những lần gặp rủi ro đó không hề nhỏ. Bảo hiểm Xe có ý nghĩa rất quan trọng, vì lợi ích cho chính người tham gia giao thông cũng như xã hội. Khi tai nạn xảy ra, trong phạm vi mức trách nhiệm Bảo hiểm, doanh nghiệp phải bồi thường cho chủ xe số tiền mà chủ xe đã hoặc sẽ phải bồi thường cho người bị thiệt hại. Trường hợp chủ xe (người mua Bảo hiểm) chết hoặc bị thương tật vĩnh viễn, doanh nghiệp sẽ bồi thường trực tiếp cho người bị thiệt hại.

Trong bài báo này, chúng tôi xây dựng các mô hình dự đoán liệu những khách hàng (chủ hợp đồng) trong năm qua có quan tâm đến Bảo hiểm Xe do công ty cung cấp hay không và từ đó, công ty có thể lập kế hoạch chiến lược truyền thông để tiếp cận những khách hàng đó và tối ưu hóa mô hình kinh doanh cũng như doanh thu của mình.

Phần còn lại của bài báo được tóm tắt như sau. Đầu tiên chúng tôi xem xét các nghiên cứu liên quan cho bài toán phân loại và các phương pháp học máy truyền thống. Phần 3 sẽ trình bày tổng quát về bộ dữ liệu dùng cho thực nghiệm. Phần 4 trình bày các phương pháp chúng tôi áp dụng để giải quyết bài toán. Phần 5 trình bày toàn bộ quá trình thực nghiệm, đánh giá và cuối cùng là phần 6 sẽ đưa ra lời nhận xét và tổng kết cho bài báo.

2 Nghiên cứu liên quan

Trong phần này, chúng tôi trình bày về bài toán phân loại nhị phân, các phương pháp học máy sử dụng cho bài toán cần giải quyết.

2.1 Bài toán phân loại nhị phân

Phân loại nhị phân đề cập đến các nhiệm vụ phân loại có hai nhãn, trong bài toán này, nhãn chính là sự quan tâm của khách hàng đến dịch vụ Bảo Hiểm Xe: quan tâm (nhãn 1) và không quan tâm (nhãn 0). Đầu vào của bài toán là đặc trưng của mỗi điểm dữ liệu (mỗi khách hàng), đầu ra là nhãn của điểm dữ liệu đó.

2.2 Các phương pháp phân loại

Logistic Regression [7] Hồi quy Logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu ra rời rạc. Nguyên lý hoạt động của hồi quy logistic là đưa đầu ra của mô hình hồi quy tuyến tính đi qua một hàm kích hoạt có tên là sigmoid để tất cả giá trị output của hàm giả định sẽ nằm trong khoảng [0,1]. Hàm sigmoid được định nghĩa như sau:

$$f(s) = \frac{1}{1+e^{-s}} \triangleq \sigma(s) \tag{1}$$

Support Vector Machine [6] Đầu tiên, trong thuật toán SVM có một khái niệm là margin – thuật ngữ chỉ khoảng cách gần nhất từ một điểm dữ liệu tới mặt phân cách giữa các lớp. Bài toán SVM là bài toán đi tìm một siêu phẳng tối ưu mà tại đó margin của các lớp dữ liệu là lớn nhất và bằng nhau.

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \tag{2}$$

Margin được tính theo công thức 2; Sau đó tối ưu hóa bằng cách cập nhật lại w,b sao cho margin là lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n+b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\} \tag{3}$$

Naive Bayes [3] là một thuật toán phân lớp được mô hình hoá dựa trên định lý Bayes trong xác suất thống kê. Với $P(y|X)$ là xác suất của mục tiêu y với điều kiện có đặc trưng X (posterior probability), $P(X|y)$ là xác suất của đặc trưng X khi đã biết mục tiêu y (likelihood), $P(y)$ là prior probability của mục tiêu y, $P(X)$ là prior probability của vector đặc trưng $X(x_1, x_2, ..., x_n)$. Khi đó,

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{4}$$

Trong mô hình Naive Bayes, có hai giả thiết được đặt ra:

- Các đặc trưng đưa vào mô hình là độc lập với nhau. Tức là sự thay đổi giá trị của một đặc trưng không ảnh hưởng đến các đặc trưng còn lại.
- Các đặc trưng đưa vào mô hình có ảnh hưởng ngang nhau đối với đầu ra mục tiêu. Khi đó, kết quả mục tiêu y để $P(y|X)$ đạt cực đại trở thành:

$$y = argmax_y P(y) \prod_{i=1}^{N} P(x_i|y) \tag{5}$$

Chính vì hai giả thiết gần như không tồn tại trong thực tế trên, mô hình này mới được gọi là naive (ngây thơ). Tuy nhiên, chính sự đơn giản của nó với việc dự đoán rất nhanh kết quả đầu ra khiến nó được sử dụng rất nhiều trong thực tế trên những bộ dữ liệu lớn, đem lại kết quả khả quan.

Decision Trees [1] Trong khoa học máy tính, cây quyết định được sử dụng như một mô hình dự đoán để đi từ các quan sát (thuộc tính) - đại diện là các nhánh để kết luận về giá trị mục tiêu (được thể hiện trong các lá). Các mục tiêu có thể có một tập hợp các giá trị được gọi là cây phân loại. Đây là một trong những cách tiếp cận mô hình dự đoán được sử dụng trong thống kê, khai thác dữ liệu và học máy. Trong các cấu trúc cây này, lá đại diện cho các nhãn và các nhánh đại diện cho các liên kết của thuộc tính dẫn đến những nhãn lớp. Cây quyết định là một bộ phân loại ở dạng cấu trúc cây, trong đó bao gồm các thành phần chính:

- Node quyết định (Decision node): chỉ định kiểm tra trên một thuộc tính duy nhất.
- Nút lá (Leaf node): cho biết giá trị của thuộc tính mục tiêu.
- Arc / edge: tách một thuộc tính.
- Path: một sự kết hợp để tạo ra quyết định cuối cùng.

Cây quyết định phân loại các trường hợp hoặc ví dụ bằng cách bắt đầu từ rễ của cây và di chuyển đến một nút lá. Có nhiều thuật toán được sử dụng để xây dựng cây quyết định có thể kể đến như: ID3, C4.5, CART, CHAID, MARS, Conditional Inference Trees.

Random Forest [5] Với thuật toán Random Forest, trong mỗi tập dữ liệu, ta có thể xây dựng được nhiều cây quyết định khác nhau. Random Forest sẽ kết hợp các cây quyết định khác nhau đó để tạo ra một mô hình mới. Kết quả đầu ra của mô hình được tổng hợp từ kết quả của các cây quyết định mà nó tạo ra. Tương ứng với mỗi điểm dữ liệu, toàn bộ quá trình dự đoán sẽ được thực hiện trên tất cả cây quyết định. Kết quả đầu ra của điểm dữ liệu này có thể được lấy là trung bình cộng dự đoán của tất cả các cây quyết định.

Trên đây chính là những thuật toán phổ biến thường được dùng để giải quyết bài toán phân loại nhị phân. Ngoài ra, chúng tôi còn xây dựng mạng học sâu cùng các phương pháp xử lý dữ liệu để tìm ra mô hình hoạt động tốt nhất. Trong dự án lần này, chúng tôi sẽ sử dụng Apache Spark vì tốc độ xử lý nhanh chóng của nó phù hợp với các thuật toán học máy khi giải quyết các vấn đề liên quan đến dữ liệu có kích thước lớn.

3 Bộ dữ liệu

Trong bài báo này, chúng tôi sử dụng bộ dữ liệu từ Kaggle [4] với nhiệm vụ dự đoán khách hàng có quan tâm đến Bảo hiểm Xe hay không dựa vào các thông tin được thu thập trước đó. Các thông tin này được lấy từ một công ty Bảo hiểm đã cung cấp Bảo hiểm Y tế cho khách hàng của mình. Hiện tại, họ cần xây dựng một mô hình để dự đoán liệu các chủ hợp đồng (khách hàng) từ năm trước có quan tâm đến Bảo hiểm Xe do công ty cung cấp hay không.

Hợp đồng bảo hiểm: công ty cam kết cung cấp bảo đảm bồi thường cho những tổn thất, thiệt hại, bệnh tật hoặc tử vong cụ thể để đổi lại việc thanh toán một khoản phí bảo hiểm cụ thể. Phí bảo hiểm là một khoản tiền mà khách hàng cần phải trả thường xuyên cho công ty bảo hiểm để được bảo lãnh này. Cũng giống như Bảo hiểm Y tế, có Bảo hiểm Xe mà hàng năm khách hàng cần phải đóng phí bảo hiểm một số tiền nhất định cho công ty cung cấp bảo hiểm để trong trường hợp xe bị tai nạn không may, công ty bảo hiểm sẽ bồi thường cho khách hàng.

Dữ liệu huấn luyện được cung cấp theo định dạng CSV chứa 381,109 dòng và 12 cột. Ý nghĩa từng thuộc tính được thể hiện trong bảng 1. Trong bảng 1 bên dưới, thuộc tính *id* là mã dùng để phân biệt mỗi khách hàng của công ty. Thuộc tính *Response* phân loại phản hồi của khách hàng, xem khách hàng đó có quan tâm đến Bảo hiểm Xe do công ty cung cấp hay không. 10 thuộc tính còn lại chứa thông tin của từng khách hàng và từ những thông tin đó có thể rút trích được một tập đặc trưng có thể xác định được những đối tượng có đặc điểm như thế nào sẽ quan tâm đến Bảo hiểm Xe. Từ đó, công ty sẽ đưa ra các chính sách phù hợp cho từng loại khách hàng để tối đa hóa lợi nhuận của mình.

STT	Tên thuộc tính	Giải thích
1	id	Mã khách hàng
2	Gender	Giới tính
3	Age	Tuổi
4	Driving_License	0: Khách hàng không có giấy phép lái xe 1: Khách hàng đã có giấy phép lái xe
5	Region_Code	Mã khu vực của khách hàng
6	Previously_Insured	1: Khách hàng đã có Bảo hiểm Xe 0: Khách hàng không có Bảo hiểm Xe
7	Vehicle_Age	Tuổi của xe
8	Vehicle_Damage	1: Xe của khách hàng đã từng bị hỏng 0: Xe của khách hàng chưa từng bị hỏng
9	Annual_Premium	Phí bảo hiểm khách hàng cần trả trong năm
10	PolicySalesChannel	Mã ẩn danh cho kênh liên lạc với khách hàng. VD: qua thư, điện thoại, gặp trực tiếp,...
11	Vintage	Số ngày khách hàng tham gia bảo hiểm
12	Response	Phản hồi của khách hàng: 1: Quan tâm 0: Không quan tâm

Bảng 1: Ý nghĩa từng thuộc tính trong dữ liệu huấn luyện.

4 Phương pháp

4.1 Tiền xử lý

StringIndexer Dữ liệu phân loại là các biến chứa giá trị nhãn chứ không phải giá trị số. Số lượng các giá trị có thể có thường được giới hạn trong một tập hợp cố định. Các biến phân loại thường được gọi là danh nghĩa. Nhiều thuật toán học máy không thể hoạt động trực tiếp trên dữ liệu nhãn. Chúng yêu cầu tất cả các biến đầu vào và biến đầu ra phải là số. Điều này có nghĩa là dữ liệu phân loại phải được chuyển đổi sang dạng số. StringIndexer là một trong những kỹ thuật Feature Engineering được sử dụng trong Spark. StringIndexer mã hóa một cột biến phân loại thành dạng chỉ số. Các chỉ số nằm trong *[0, n)* với n là số lượng nhãn của cột đó. Thứ tự của các chỉ mục là tần suất tăng dần của các nhãn, vì vậy chỉ số của các nhãn xảy ra nhiều nhất là 0. Nếu cột đầu vào là kiểu số, chúng tôi sẽ chuyển nó thành chuỗi và sử dụng chuỗi để dịch sang chỉ mục.

OneHotEncoder Đối với các biến phân loại không tồn tại mối quan hệ thứ tự như vậy, mã hóa số nguyên là không đủ. Trên thực tế, việc sử dụng mã hóa này và cho phép mô hình giả định thứ tự tự nhiên giữa các danh mục có thể dẫn đến hiệu suất kém hoặc kết quả không mong muốn (dự đoán giữa các danh mục) khi các giá trị số có thể bị giải thích sai bởi các thuật toán vì có một số loại phân cấp / thứ tự trong chúng. Trong trường hợp này, **OneHotEncoder** có thể được áp dụng cho biểu diễn số nguyên. Trong thuật toán này, mỗi giá trị

danh mục được chuyển đổi thành một cột mới và được gán giá trị 1 hoặc 0 (ký hiệu cho đúng / sai) cho cột và số lượng cột mới được thêm vào sẽ bằng với số lượng nhãn. Mặc dù cách tiếp cận này giúp loại bỏ các vấn đề về thứ bậc / thứ tự nhưng có nhược điểm là thêm nhiều cột vào tập dữ liệu. Điều này dẫn đến số lượng cột mở rộng rất nhiều nếu có nhiều nhãn trong một cột danh mục.

Standard Scaler Mô hình học máy học cách ánh xạ từ biến đầu vào đến biến đầu ra. Do đó, quy mô và sự phân phối của dữ liệu được lấy từ các miền có thể khác nhau đối với mỗi biến. Sự khác biệt trong thang đo giữa các biến đầu vào có thể làm tăng độ khó của vấn đề được mô hình hóa. Một ví dụ về điều này là các giá trị đầu vào lớn (ví dụ: khoảng cách hàng trăm hoặc hàng nghìn đơn vị) có thể dẫn đến một mô hình học các giá trị trọng số lớn. Một mô hình có giá trị trọng số lớn thường không ổn định, có nghĩa là nó có thể bị kém hiệu suất trong quá trình học và nhạy cảm với các giá trị đầu vào dẫn đến sai số tổng quát hóa cao hơn. Các thuật toán phù hợp với mô hình sử dụng tổng trọng số của các biến đầu vào sẽ bị ảnh hưởng, chẳng hạn như hồi quy tuyến tính, hồi quy logistic và mạng nơ-ron nhân tạo (học sâu). Bên cạnh đó đối với các bài toán hồi quy, một biến mục tiêu có sự chênh lệch giá trị lớn có thể dẫn đến giá trị gradient sai số lớn khiến giá trị trọng số thay đổi đột ngột, làm cho quá trình học không ổn định. Chia tỷ lệ dữ liệu (Data scaling) là bước xử lý trước được khuyến nghị khi làm việc với nhiều thuật toán học máy.

Ý tưởng đằng sau StandardScaler là nó sẽ biến đổi dữ liệu của bạn sao cho phân phối của nó sẽ có giá trị trung bình là 0 và độ lệch chuẩn là 1. Trong trường hợp dữ liệu đa biến, điều này được thực hiện theo tính năng (nói cách khác là độc lập cho từng cột dữ liệu).

Chuẩn hóa z:

$$z = \frac{x - \mu}{\sigma} \tag{6}$$

Với giá trị trung bình μ:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i) \tag{7}$$

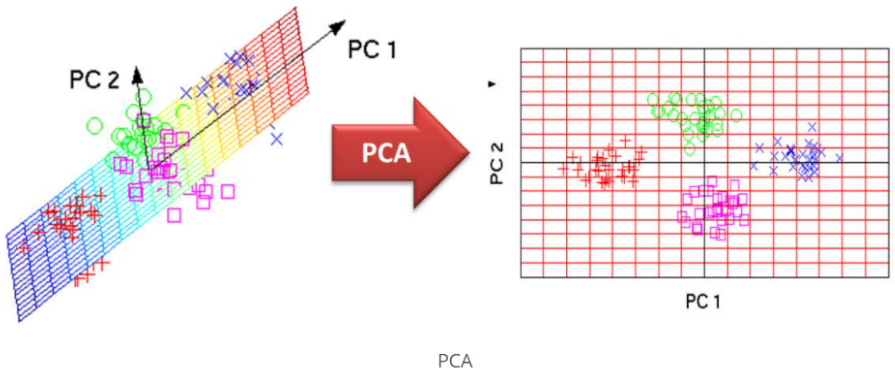
và độ lệch chuẩn σ

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{8}$$

VectorAssembler được sử dụng trong spark như là một transformer kết hợp một danh sách các cột nhất định thành một cột vectơ duy nhất (hay còn gọi là vector đặc trưng - feature vector). Đây là một cách tiếp cận rất hữu ích để kết hợp các đặc trưng thô và các đặc trưng được tạo bởi các phép biến đổi đặc trưng khác nhau thành một vector đặc trưng duy nhất để tiến hành huấn luyện các mô hình máy học. VectorAssembler chấp nhận các kiểu cột đầu vào sau: tất

cả các kiểu số, kiểu boolean và kiểu vectơ. Trong mỗi hàng, giá trị của các cột đầu vào sẽ được nối thành một vectơ theo thứ tự được chỉ định.

Principal Components Analysis - PCA [2] được Karl Pearson đề xuất vào năm 1901. Hiện nay, PCA được sử dụng như công cụ để phân tích dữ liệu nghiên cứu và thực hiện các mô hình dự đoán. PCA là một thuật toán thống kê sử dụng phép biến đổi trực giao để biến đổi một tập hợp dữ liệu từ một không gian nhiều chiều sang một không gian mới ít chiều hơn nhằm tối ưu hóa việc thể hiện sự biến thiên của dữ liệu. Mục tiêu của PCA là đưa không gian đặc trưng có số



Hình 1: *Minh họa cho ý tưởng của thuật toán giảm chiều dữ liệu PCA* [3]

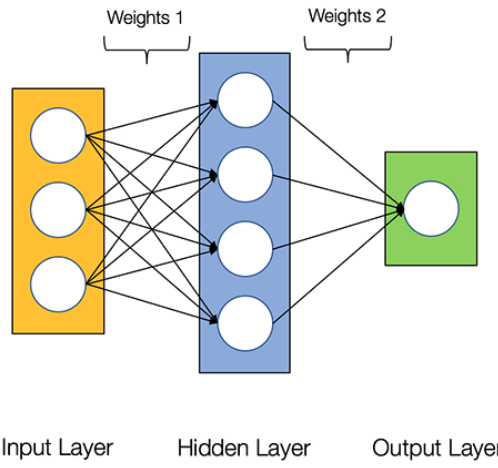
chiều lớn về không gian đặc trưng có số chiều nhỏ hơn (Hình 3) sao cho vẫn đảm bảo được "tối đa thông tin quan trọng nhất", với mục đích giảm thời gian tính toán và tìm kiếm. Các trục tọa độ trong không gian mới được xây dựng sao cho trên mỗi trục, độ biến thiên của dữ liệu trên đó là lớn nhất có thể.

4.2 Mô hình phân lớp

Deep Learning Một mạng nơ-ron có thể có ba loại lớp: lớp đầu vào nhận đầu vào thô từ miền, lớp ẩn nhận đầu vào từ lớp khác và chuyển đầu ra cho lớp khác và lớp đầu ra đưa ra dự đoán. Mạng học sâu được phân biệt với mạng nơ-ron một lớp ẩn phổ biến hơn bởi độ sâu của chúng, tức là số lớp nút mà dữ liệu phải đi qua trong một quá trình nhận dạng mẫu nhiều bước. Các phiên bản trước của mạng nơ-ron như các perceptron đầu tiên rất nông, bao gồm một lớp đầu vào và một lớp đầu ra, và nhiều nhất là một lớp ẩn ở giữa. Hơn ba lớp (bao gồm đầu vào và đầu ra) đủ điều kiện là học "sâu". Trong mạng học sâu, mỗi node

[3] https://kindsonthegenius.com/blog/what-is-principal-component-analysis-pca-a-simple-tutorial/

huấn luyện trên một tập hợp các đặc trưng riêng biệt dựa trên kết quả đầu ra của lớp trước. Mạng càng sâu, các đặc trưng của các node ra càng phức tạp, vì chúng tổng hợp và kết hợp lại các tính năng từ lớp trước đó. Vì vậy, đặc trưng cuối cùng sẽ được phân tách một cách tuyến tính trước khi cho qua lớp phân lớp với một hàm kích hoạt biến đổi phi tuyến (thường là softmax).



Hình 2: *Minh họa cho một mạng học sâu đơn giản với một lớp ẩn* [4]

Một hàm kích hoạt trong mạng nơron xác định cách tổng trọng số của đầu vào được chuyển thành đầu ra từ một node hoặc các node trong một lớp của mạng. Nếu phạm vi đầu ra của hàm kích hoạt bị giới hạn. Trong phạm vi, bài toán này chúng tôi chỉ sử dụng hai hàm kích hoạt là Sigmoid đã được giới thiệu ở 2.2 và hàm ReLU.

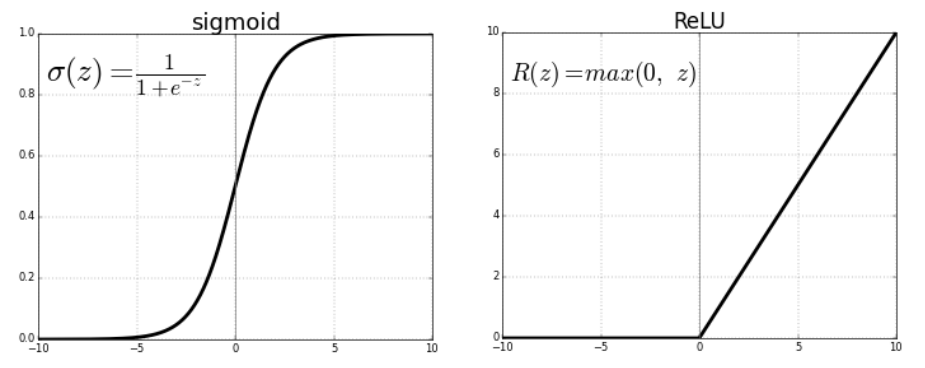
Hàm kích hoạt ReLU là hàm phổ biến nhất được sử dụng cho các lớp ẩn. Nó phổ biến vì nó vừa đơn giản để thực hiện vừa có hiệu quả khắc phục những hạn chế của các hàm kích hoạt phổ biến khác trước đây, chẳng hạn như Sigmoid và Tanh. Cụ thể, nó ít bị ảnh hưởng bởi vanishing gradients ngăn cản việc huấn luyện các mô hình sâu, mặc dù nó có thể gặp phải các vấn đề khác như các đơn vị bão hòa hoặc "chết". Hàm ReLU được tính như sau:

$$max(0, x) \tag{9}$$

Vì lý do hạn chế về mặt tài nguyên cũng như nhân thấy mô hình đã hoạt động tốt với yêu cầu ban đầu đặt ra. Để giải quyết bài toán đã đặt ra, chúng tôi đã

[4] https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6

[5] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6



Hình 3: *Hàm kích hoạt Sigmoid (bển trái) và ReLu (bên phải)* [5]

xây dựng một mạng học sâu đơn giản với 2 lớp ẩn và số node tại từng lớp tưng ứng là 256 và 128 với hàm kích hoạt ReLU. Bên cạnh đó tại mỗi lớp, để ngăn cho mô hình không bị overfitting, chúng tôi sử dụng thêm dropout với rate là 0.3. Vì là bài toán phân loại nhị phân nên đầu ra của chúng tôi sẽ được cho qua hàm kích hoạt sigmoid.

4.3 Độ đo

Để đánh giá hiệu suất phân loại cho các mô hình đã huấn luyện, chúng tôi sử dụng độ đo ROC-AUC score. Đây là độ đo thường được dùng trong các bài toán với thách thức lớn nhất đến từ việc mất cân bằng dữ liệu. Trong dữ liệu huấn luyện, số lượng khách hàng quan tâm đến Bảo hiểm Xe do công ty cung cấp chỉ chiếm khoảng 12% trong tổng số khách hàng công ty đang có. Các chỉ số được sử dụng trong ROC-AUC score:

TPR (True Positive Rate/Sentivity/Recall). Tỉ lệ số điểm dự đoán là Positive trong số những điểm thực sự là Positive.

$$TPR = \frac{TP}{TP+FN} \tag{10}$$

Specificity Tỉ lệ số điểm dự đoán là Negative trong số những điểm thực sự là Negative.

$$Specificity = \frac{TN}{TN+FP} \tag{11}$$

FPR (False Positive Rate/Fall-out). Tỉ lệ gắn nhãn sai các mẫu Negative thành Positive trên tất cả các mẫu thực sự là Negative.

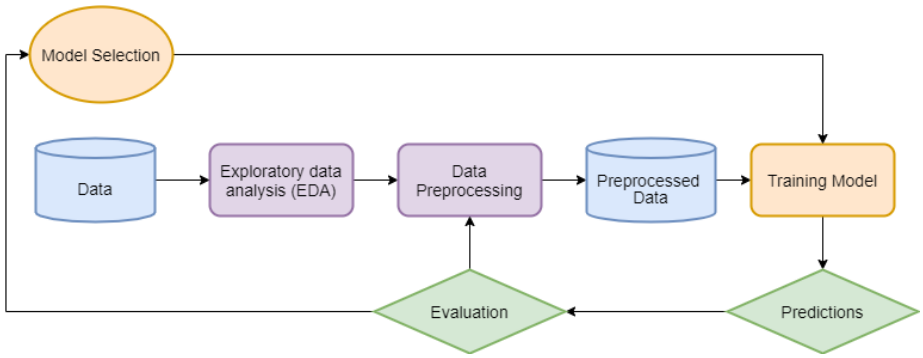
$$FPR = 1 - Specificity = \frac{FP}{TN+FP} \tag{12}$$

Ta sử dụng đường cong ROC để hiển thị từng cặp (TPR, FPR) cho các ngưỡng khác nhau. Sau đó tính chỉ số AUC cho đường cong này. Chỉ số AUC chính là con số thể hiện hiệu suất phân loại của mô hình.

5 Thực nghiệm

5.1 Quy trình thực hiện

Hình 4 đã minh họa toàn bộ quy trình thực hiện dự án này. Từ dữ liệu gốc, việc đầu tiên của chúng tôi là tiến hành phân tích thăm dò dữ liệu (EDA). Quá trình này giúp chúng tôi xác định các lỗi cụ thể, cũng như hiểu rõ hơn về các mẫu dữ liệu, phát hiện các điểm bất thường và tìm ra mối quan hệ giữa các thuộc tính với nhau. Dựa vào kết quả EDA, chúng tôi sẽ đề ra những kế hoạch để tiếp cận bài toán một cách hợp lý. Tiếp theo, chúng tôi sẽ thực hiện quá trình tiền xử lý dữ liệu (Data Preprocessing). Trong giai đoạn này, chúng tôi sẽ tách dữ liệu ban đầu thành tập huấn luyện và tập thử nghiệm, dùng các thuật toán được đề cập trong phần 4.1 để chuẩn hóa dữ liệu cho phù hợp với các mô hình học máy, điều này giúp làm tăng độ chính xác và hiệu quả của các mô hình. Dữ liệu sau khi được xử lý (Preprocessed Data) sẽ được huấn luyện (Training Model) với các mô hình học máy và sau đó dự đoán (Predict) trên tập kiểm tra. Cuối cùng, chúng tôi tiến hành đánh giá (Evaluate) kết quả dự đoán và lựa chọn các mô hình (Model Selection). Quy trình này sẽ hoạt động như một vòng lặp đến khi nào tìm ra được mô hình đạt hiệu suất tốt nhất.

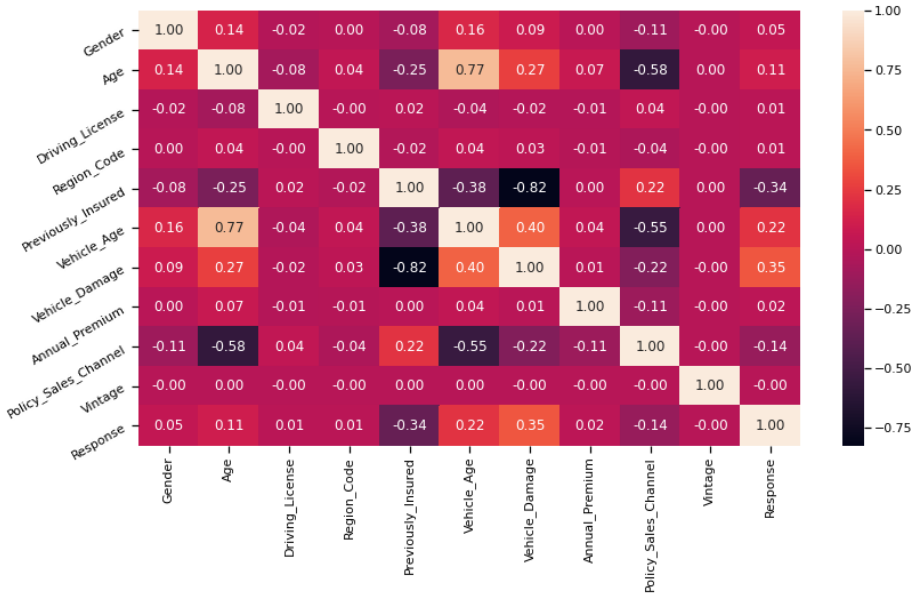


Hình 4: Quy trình thực hiện công việc

5.2 Kết quả EDA

Sau khi thực hiện quá trình phân tích thăm dò dữ liệu, chúng tôi thu được các đặc điểm và thông tin quan trọng. Bộ dữ liệu gồm 381, 109 điểm dữ liệu và

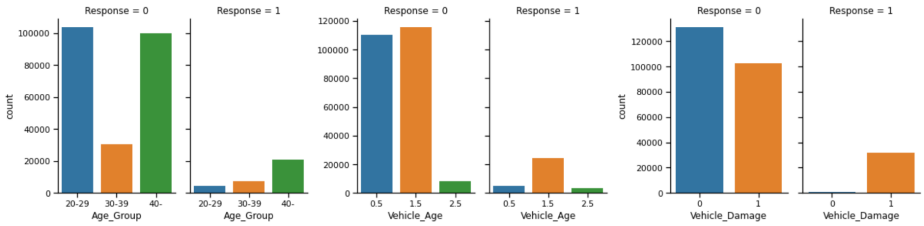
12 thuộc tính, trong đó thuộc tính *Response* là thuộc tính dự đoán, thuộc tính *id* là mã khách hàng tương ứng với một khách hàng duy nhất và 10 thuộc tính còn lại dùng để phân tích và xây dựng mô hình. Các thuộc tính đều không bị khuyết giá trị và kiểu dữ liệu đúng với giá trị thuộc tính.



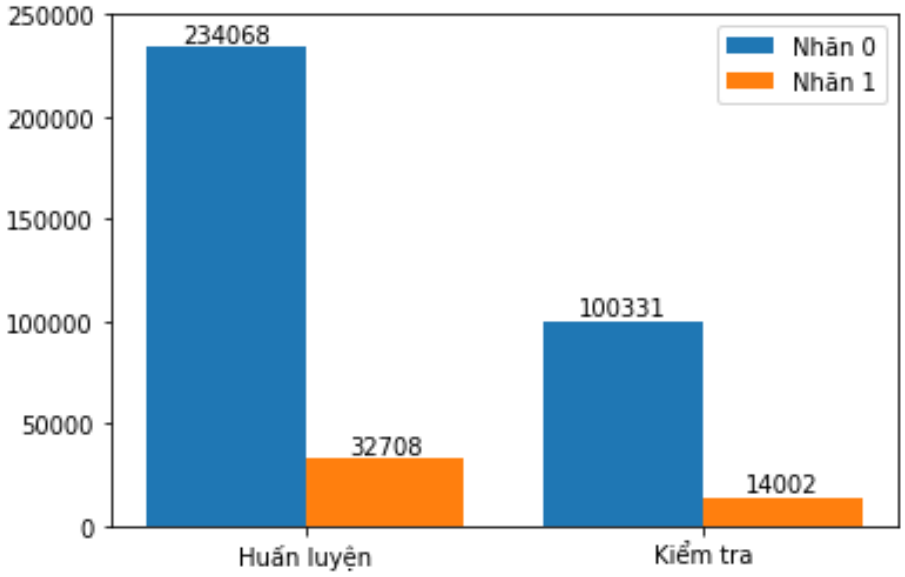
Hình 5: Độ tương quan giữa các thuộc tính trong tập huấn luyện

Dựa vào mối tương quan giữa các thuộc tính được minh họa trong hình 5, chúng tôi nhận thấy rằng *Age, Vehicle_ Age, Vehicle_ Damage* là ba thuộc tính có mối tương quan tích cực cao nhất với nhau và với thuộc tính dự đoán *Response*, điều này có ích cho việc lựa chọn thuộc tính khi huấn luyện, kết quả thực nghiệm được so sánh trong bảng 3. Cụ thể trong hình 6, sau khi chia các độ tuổi của khách hàng thành ba nhóm $20-29, 30-39, \geq 40$ chúng tôi thấy rằng số lượng khách hàng ở độ tuổi $30-39$ ít hơn nhiều so với hai nhóm còn lại. Và trong số những khách hàng quan tâm đến dịch vụ bảo hiểm xe thì nhóm ≥ 40 chiếm phần lớn. Đối với thuộc tính *Vehicle_ Age* cho biết tuổi xe của các khách hàng phần lớn đều dưới 2 năm và trong số những khách hàng quan tâm đến dịch vụ bảo hiểm xe thì khách hàng có tuổi xe trong khoảng $1-2$ năm là chủ yếu. Cuối cùng, *Vehicle_ Damage* là thuộc tính có mức tương quan cao nhất với thuộc tính dự đoán. Trong số những khách hàng quan tâm đến dịch vụ bảo hiểm xe thì hầu hết xe của họ đều đã từng bị hỏng. Ngược lại, những khách hàng mà xe của họ chưa từng hỏng thì họ không quan tâm.

Một điều quan trọng mà chúng tôi phát hiện đó chính là sự chênh lệch giữa số lượng hai nhãn quá lớn, được minh họa trong hình 7, tỉ lệ chênh lệch là



Hình 6: Sự phân bố dữ liệu của thuộc tính *Age (bên trái), Vehicle_ Age (ở giữa), Vehicle_ Damage (bên phải)* trong tập huấn luyện



Hình 7: Thống kê số lượng nhãn ở mỗi tập

$(334,399 : 46,710) \approx (7 : 1)$. Vì thế chúng tôi đã thử nghiệm áp dụng phương pháp tăng cường dữ liệu ở nhãn thiểu số đến tỉ lệ $\approx (1 : 1)$ để cân bằng hóa dữ liệu ở tập huấn luyện, kết quả được thể hiện trong bảng 2.

5.3 Kết quả thực nghiệm và đánh giá

Sau quá trinh phân tích và xử lý dữ liệu chúng tôi có được tập huấn luyện bao gồm $266,776$ dòng và tập kiểm tra bao gồm $114,333$ dòng. Đối với tất cả các thử nghiệm, chúng tôi dùng độ đo ROC-AUC score để đánh giá, từ đây chọn ra mô hình đạt hiệu suất tốt nhất.

Kết quả các phương pháp học máy được báo cáo trong bảng 2. Cụ thể, chúng tôi sẽ thử nghiệm các phương pháp này trong hai trường hợp trước và sau khi

Thuộc tính	Cân bằng dữ liệu	Phương pháp	ROC-AUC Score
Tất cả	Không	Logistic Regression	50.0
		Naive Bayes	**58.8**
		LinearSVM	50.0
		Decision Tree	50.0
		Random Forest	50.0
	Có	Logistic Regression	80.7
		Naive Bayes	57.3
		LinearSVM	79.1
		Decision Tree	**87.9**
		Random Forest	84.1

Bảng 2: Kết quả thực nghiệm trước và sau khi cân bằng dữ liệu được đánh giá bằng độ đo ROC-AUC (%)

cân bằng dữ liệu. Chúng ta dễ dàng nhận thấy kết quả các mô hình sau khi cân bằng dữ liệu đạt hiệu suất cao hơn rõ rệt so với khi dữ liệu không cân bằng. Trước khi dữ liệu được cân bằng, kết quả độ đo ROC-AUC ở cả năm phương pháp đều ở khoảng 50%, đều này cho thấy rằng việc dữ liệu mất cân bằng đã ảnh hưởng rất tiêu cực đến việc dự đoán. Khi AUC $\approx$ 50%, nghĩa là các mô hình phân loại đang không thể phân biệt giữa hai nhãn 0 và 1. Ngược lại, sau khi tăng số lượng nhãn 1 đến tỉ lệ cân bằng, kết quả cao nhất chúng tôi thu được đã tăng lên $\approx$ 38% (khi dùng mô hình Decision Tree đạt 87.9%).

Thuộc tính	PCA	ROC-AUC Score
Tất cả	dim = 23	50.0
'Age', 'Vehicle_Age', 'Vehicle_Damage'	dim = 10	98.7
'Age', 'Vehicle_Age', 'Vehicle_Damage'	k = 3	92.9
'Age', 'Vehicle_Age', 'Vehicle_Damage'	k = 5	98.6
'Age', 'Vehicle_Age', 'Vehicle_Damage'	k = 7	98.3
'Age', 'Vehicle_Age', 'Vehicle_Damage'	**k = 9**	**99.1**

Bảng 3: Kết quả thực nghiệm của mô hình học sâu kết hợp thuật toán PCA được đánh giá bằng độ đo ROC-AUC (%)

Mặc dù kết quả đạt được không quá tệ nhưng chúng tôi đã tiếp tục tìm cách cải tiến hiệu suất phân loại bằng phương pháp học sâu kết hợp thuật toán PCA. Nhìn vào số liệu bảng 3, chúng ta có thể thấy được việc huấn luyện với ba thuộc tính có mức độ tương quan cao *Age, Vehicle_Age, Vehicle_Damage* không những không làm giảm hiệu suất ngược lại còn đạt kết quả thật đáng mong đợi. Cụ thể, kết quả các thử nghiệm với ba thuộc tính này đều cao hơn 98% và cao nhất là trường hợp số chiều sau khi trích xuất đặc trưng là 9 (k = 9) đạt 99.1%.

6 Kết luận

Trong báo cáo này, chúng tôi đã giải quyết bài toán phân loại xem liệu một khách hàng có hứng thú với bảo hiểm xe hay không qua nhiều bước xử lý và cách tiếp cận hiệu quả. Chúng tôi đã phân tích, xử lý, chuẩn hóa dữ liệu trước khi đưa vào huấn luyện. Dựa vào đó, chúng tôi đã đề xuất các các phương pháp giải quyết phù hợp và đạt kết quả cao, kết quả được thể hiện rõ trong phần 5.3. Hiệu suất cao nhất mà phương pháp chúng tôi đạt được là 99.1%, điều này thể hiện hiệu quả trong cách cải tiến của chúng tôi.

Lời cảm ơn

Nhóm chúng tôi xin chân thành gửi lời cảm ơn đến TS. Đỗ Trọng Hợp – giảng viên phụ trách môn Phân tích Dữ liệu lớn, đã tận tình giúp đỡ, đưa ra những góp ý để chúng tôi hoàn thiện hơn bài báo cáo.

Tài liệu

[1] Robin LP Chang and Theodosios Pavlidis. "Fuzzy decision tree algorithms". In: *IEEE Transactions on systems, Man, and cybernetics* 7.1 (1977), pp. 28–35.

[2] Xingdong Chen, Chao Chen, Li Jin, et al. "Principal component analyses in anthropological genetics". In: *Advances in Anthropology* 1.02 (2011), p. 9.

[3] Ron Kohavi et al. "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." In: *Kdd*. Vol. 96. 1996, pp. 202–207.

[4] Anmol Kumar. *Health Insurance Cross Sell Prediction*. Sept. 2020. URL: `https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction`.

[5] Mark R Segal. "Machine learning benchmarks and random forest regression". In: (2004).

[6] Lipo Wang. *Support vector machines: theory and applications*. Vol. 177. Springer Science & Business Media, 2005.

[7] Raymond E Wright. "Logistic regression." In: (1995).