TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

PHÂN TÍCH THIẾT KẾ HỆ THỐNG
HỆ HỖ TRỢ DỰ BÁO KHẢ NĂNG THÀNH CÔNG CỦA CHIẾN DỊCH TIẾP THỊ BẢO HIỂM Ô TÔ

GV hướng dẫn: TS. Lê Hải Hà

Sinh viên thực hiện:

Họ và tên	*MSSV*	*Lớp*
Bùi Doãn Đang	**20185437**	**HTTTQL01-K63**

Hà Nội, 2021

Mục lục

Lời mở đầu

Ngày nay, với sự bùng nổ và phát triển của thời đại 4.0, Công nghệ đang đóng một vai trò quan trọng và vươn lên một tầm cao mới. Đặc biệt với việc công nghệ luôn thay đổi hàng ngày, đòi hỏi những người "kiếm tiền" của một tổ chức hay một doanh nghiệp hay còn gọi là nhân viên tiếp thị phải thay đổi theo.

Thật vậy, tiếp thị là một bộ phận đóng vai trò rất quan trọng trong kinh doanh, có thể nói một công ty thành công hay không phụ thuộc rất nhiều vào bộ phận tiếp thị. Vì vậy, có một bộ phận tiếp thị chuyên nghiệp là một thế mạnh rất lớn cho các tổ chức hay công ty hiện nay. Với sự phát triển vượt bậc của công nghệ, tiếp thị hiện nay cũng là một lĩnh vực được quan tâm, chú trọng áp dụng những công nghệ tiên tiến để tăng hiệu quả công việc và phù hợp với sự phát triển của thời đại 4.0 hiện nay.

Trong lĩnh vực bảo hiểm ô tô ngoài các hình thức tiếp thị qua email, quảng cáo, kỹ thuật số,.. thì hình thức phổ biến nhất vẫn là tiếp thị qua điện thoại. Hình thức này đem lại nhiều lợi ích và hiệu quả trong việc quảng bá thương hiệu, duy trì và cải thiện mối quan hệ với khách hàng, đem lại lợi nhuận cho công ty. Để giúp chiến chiến dịch này đạt được hiệu quả cao hơn trong quá trình tiếp thị thì trong môn học này em xin xây dựng một hệ thống dự đoán khả năngthành công của chiến dịch tiếp thị đến từng khách hàng một từ đó giúp nhà quản lý đưa ra những chính sách cụ thể cho từng nhóm khách hàng.

Báo cáo của em gồm 4 phần:

Phần I: Xác định và phát biểu vấn đề

Phần II: Thu thập dữ liệu

Phần III: Lập mô hình

Phần IV: Triển khai phần mềm

Em xin chân thành cảm ơn thầy Lê Hải Hà – Viện Toán ứng dụng và Tin học, Trường Đại học Bách khoa Hà Nội đã tận tình chỉ dạy, hướng dẫn, truyền đạt kiến thức hữu ích trong suốt quá trình em thực hiện.

Do còn nhiều hạn chế về kiến thức nên báo cáo không thể tránh khỏi có một số sai sót. Vì vậy, em rất mong nhận được góp ý, đánh giá của thầy để báo cáo của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hà Nội, tháng 6 năm 2021

Tác giả báo cáo

Bùi Doãn Đang

1 Xác định và phát biểu vấn đề

1.1 Xác định vấn đề

Từ xưa đến nay, bộ phân tiếp thị trong mỗi công ty luôn đóng vai trò quan trọng trong sự phát triển của công ty đó. Có thể nói tiếp thị là cầu nối chung gian giữa doanh nghiệp và thị trường, giúp doanh nghiệp bán sản phẩm và đảm bảo doanh thu. Tiếp thị mang lại cho công ty nhiều lợi ích to lớn có thể kể đến như là: duy trì và phát triển doanh nghiệp, đảm bảo doanh thu và lợi nhuận, thị trường và thị phần, duy trì và cải thiện mối quan hệ với khách hàng.

Vì vậy, bài toán đặt ra là làm sao để cải thiện chất lượng trong hoạt động tiếp thị luôn là một vấn đề đau đầu với các nhà quản lý. Một giải pháp được đề ra đó chính là dự đoán trước nhưng khách hàng nào có khả năng mua bảo hiểm hay không, từ đó hỗ trợ cho bộ phận tiếp thị tập trung vào những nhóm khách hàng tiềm năng, có thể mua bảo hiểm từ công ty. Bên cạnh đó cũng giúp nhà quản lý có thể đưa ra các chiến lược khac nhau, tiếp cận đến từng đối tượng khách hàng và đạt được hiệu quả cao trong chiến dịch tiếp thị, từ đó giúp công ty phát triển.

1.2 Phát biểu vấn đề

Khi công ty mở một chiến dịch tiếp thị để quảng bá thương hiệu và bán sản phẩm, bộ phận tiếp thị nhận được một danh sách về khách hàng và có nhiệp vụ triển khai chiến dịch đến từng khách hàng. Trong quá trình triển khai chiến dịch nhân viên tiếp thị có nhiệm vụ thu thập các thông tin về cuộc gọi tiếp thị như: số lần tiếp thị trong chiến dịch này, kết quả của chiến dịch tiếp thị lần trước, thời gian bắt đầu gọi và thời gian kết thúc,.. kết hợp với các thông tin về khách hàng đã có sẵn ta sẽ dự đoán những khách hàng nào có khả năng mua bảo hiểm hay không mua bảo hiểm và phân tích những yếu tố dẫn đến các khả năng đó. Từ những kết quả phân tích được nhà quản lý sẽ vạch ra những chính sách tiếp thị trong tương lai.

2 Thu thập dữ liệu

2.1 Tìm kiếm dữ liệu

Trong quá trình tìm kiếm trên các trang web dataset, em có tìm được một bộ dữ liệu trên kaggle.com. Cụ thể dữ liệu mô tả về một ngân hàng ở hoa kỳ cung cấp các dịch vụ thông thường, bên cạnh đó ngân hàng này cũng cung cấp dịch vụ bảo hiểm xe hơi, họ có thông tin của các khách hàng tiềm năng, trong các chiến dịch tiếp thị nhân viên sẽ gọi cho khách hàng để tiếp thị về các gói bảo hiểm oto của ngân hàng và thu thập thông tin của chiến dịch hiện tại như: số lần đã gọi trong chiến dịch này, ngày-tháng liên lạc cuối cùng, thời gian bắt đầu và kết thúc của cuộc gọi gần nhất.

Với thông tin của khách hàng, thông tin về chiến dịch tiếp thị hiện tại kết hợp với thông tin của các chiến dịch tiếp thị trước đó (số lần liền hệ trước chiến dịch này,số ngày đã trôi qua kể từ lần cuối đã liên hệ từ chiến dịch trước đó,kết quả của chiến dịch trước) nhà quản lý sẽ dự đoán những khách hàng nào có khả năng mua bảo hiểm và không mua bảo hiểm, thông qua mô hình dự đoán có thể trích xuất ra những đặc trưng dẫn đến khả năng khách hàng mua hay không mua bảo hiểm, từ đó giúp nhà quản lý vạch ra những chiến lược cụ thể để tiếp cận với từng nhóm khách hàng và đạt được hiệu quả cao nhất, tiếp kiệm nguồn lực và chi phí trong chiến dịch tiếp thị.

2.2 Mô tả dữ liệu

Dữ liệu chia làm 2 file, file carInsurance_train.csv bao gồm 4000 bản ghi thông tin về khách hàng, với 18 trường thuộc tính. Mô hình bài toán sẽ dự đoán liệu khách hàng có mua bảo hiểm xe hay không.

File thứ 2 carInsuarance_test.csv bao gồm 1000 bản ghi với đầy đủ thông tin đầu vào nhưng thiếu thông tin đầu ra là khách hàng đó có mua bảo hiểm hay không. Tập dữ liệu này dùng để test xem mô hình hoạt động ổn không.

Nguồn: https://www.kaggle.com/kondla/carinsurance

Bộ dữ liệu tập train và tập test đều tương đối đầy đủ, nhưng trong đó vẫn có những dữ liệu bị thiếu (NaN) cụ thể ở các trường như communication(hình thức liên lạc), outcome (kết quả của chiến dịch trước đó).

18 trường bao gồm : age, job, Marital, education, default(có khoản nợ xấu hay không), Balance (số dư bình quân trong tài khoản), HH Insurance(có bảo hiểm gia đình hay không), car loan(có khoản vay mua oto hay không), communication(hình thức liên lạc), last contact day (ngày liên lạc cuối cùng), last contact month(tháng liên hệ cuối cùng), NoOfContacts(số lần liên hệ được thực hiện trong chiến dịch này), DaysPassed (số ngày đã qua kể từ lần cuối cùng liên hệ của chiến dịch tiếp thị trước đó), PrevAttempts (số lần liên hệ được thực hiện trước chiến dịch này), outcome (kết quả của chiến dịch trước đó), CallStart (thời gian bắt đầu gọi), callend (thời gian kết thúc cuộc gọi), car Insuracne (khách hàng có mua bảo hiểm hay không).

Feature	Description	Example
Id	Unique ID number. Predictions file should contain this feature.	“1” ... “5000”
Age	Age of the client	
Job	Job of the client.	"admin.", "blue-collar", etc.
Marital	Marital status of the client	"divorced", "married", "single"
Education	Education level of the client	"primary", "secondary", etc.
Default	Has credit in default?	"yes" - 1,"no" - 0
Balance	Average yearly balance, in USD	
HHInsurance	Is household insured	"yes" - 1,"no" - 0
CarLoan	Has the client a car loan	"yes" - 1,"no" - 0
Communication	Contact communication type	"cellular", "telephone", “NA”
LastContactMonth	Month of the last contact	"jan", "feb", etc.
LastContactDay	Day of the last contact	
CallStart	Start time of the last call (HH:MM:SS)	12:43:15
CallEnd	End time of the last call (HH:MM:SS)	12:43:15
NoOfContacts	Number of contacts performed during this campaign for this client	
DaysPassed	Number of days that passed by after the client was last contacted from a previous campaign (numeric; -1 means client was not previously contacted)	
PrevAttempts	Number of contacts performed before this campaign and for this client	
Outcome	Outcome of the previous marketing campaign	"failure", "other", "success", “NA”
CarInsurance	Has the client subscribed a CarInsurance?	"yes" - 1,"no" - 0

Hình 1: Giải thích các trường

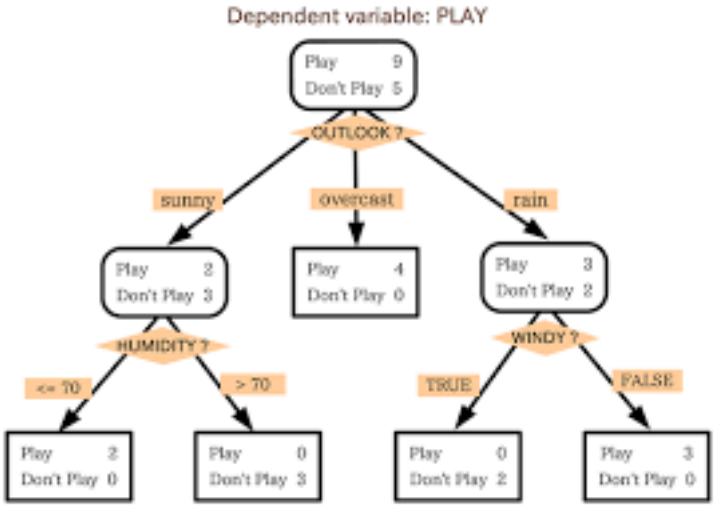
3 Lập mô hình

3.1 Cây quyết định (Decision Tree)

3.1.1 Tổng quan về cây quyết định

Thuật toán Cây quyết định là một trong những thuật toán học máy phổ biến nhất. Nó sử dụng một cấu trúc giống như cây và các kết hợp có thể có của chúng để giải quyết một vấn đề cụ thể. Cây quyết định là một mô hình học có giám sát, được áp dụng vào cả 2 bài toán phân lớp và hồi quy. Việc xây dựng một decision tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của Decision tree là nó có thể làm việc với các đặc trưng dạng categorical, thường là rời rạc và không có thứ tự. Decision tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng categorical và liên tục (numeric). Một điểm đáng lưu ý nữa là decision tree ít yêu cầu việc chuẩn hoá dữ liệu.

Một cây quyết định là một cấu trúc bao gồm nút gốc (root), các branch (nhánh) và các leaf node (nút lá). Mỗi nút bên trong biểu thị một phép thử trên một thuộc tính, mỗi nhánh biểu thị kết quả của một phép thử và mỗi nút lá chứa một nhãn lớp. Nút trên cùng trong cây là nút gốc.



Hình 2: Ví dụ về một cây quyết định

Vậy một cách tổng quát, một cây quyết định là một sơ đồ có dạng hình cây, ở đó mỗi internal node (non-leaf node – không phải nút lá) biểu thị một câu hỏi trên mỗi thuộc tính, mỗi branch (nhánh) đại diện cho một kết quả của câu hỏi, và mỗi leaf node (nút lá) giữ một nhãn lớp (class label). Các non-leaf node thường có hai hoặc nhiều node con (child node). Các child node này có thể là một leaf node hoặc một non-leaf node khác. Nút trên cùng của cây được gọi là root node (nút gốc).

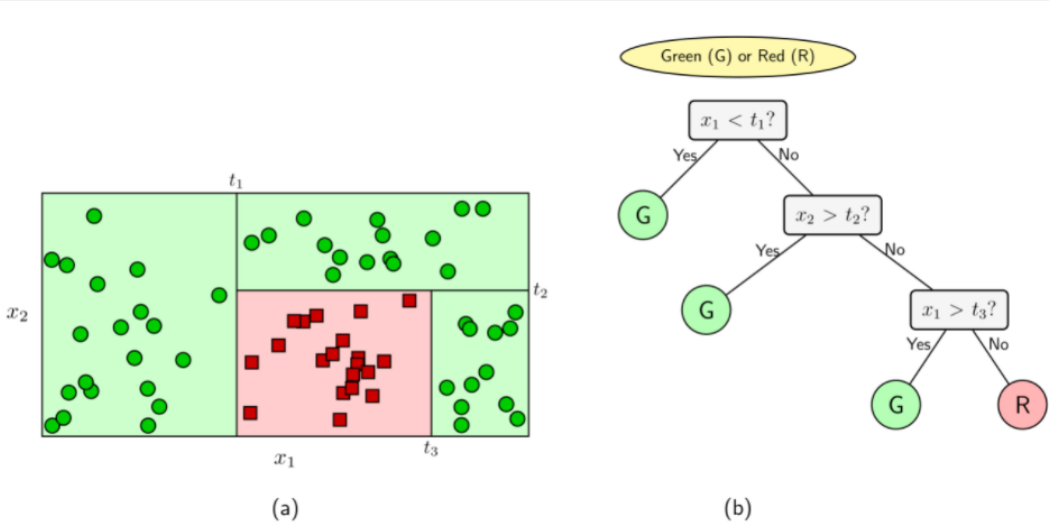
Một số thuật toán cây quyết định chỉ tạo ra cây nhị phân (binary tree) (Đó là mỗi internal node rẽ nhánh đến đúng 2 node khác hay tất cả các non-leaf node chỉ có hai child node). Trong khi một số khác tạo ra nonbinary tree.

3.1.2 Cây quyết định sử dụng như nào để phân lớp?

Cho một quan sát X mà nhãn lớp liên quan là chưa biết, các giá trị thuộc tính của quan sát đó được kiểm tra dựa trên cây quyết định. Một đường đi (path) từ gốc (root) đến một nút lá (leaf), chứa dự đoán lớp cho quan sát đó. Cây quyết định có thể dễ dàng chuyển đổi thành quy tắc phân lớp.

3.1.3 Tại sao phân loại cây quyết định rất phổ biến

Việc xây dựng các cây quyết định phân lớp không yêu cầu bất kỳ tri thức miền hoặc cài đặt tham số nào, và do đó thích hợp cho khám phá tri thức khám phá. Cây quyết định có thể xử lý các dữ liệu nhiều chiều. Sự thể hiện tri thức thu nhận của chúng ở dạng cây là trực quan và thường dễ hiểu với con người. Các bước học tập và phân loại của cây quyết định rất đơn giản và nhanh chóng. Nói chung, phân loại cây quyết định có độ chính xác tốt. Tuy nhiên, việc sử dụng thành công có thể phụ thuộc vào dữ liệu trong tay. Các thuật toán cây quyết định đã được sử dụng để phân loại trong nhiều lĩnh vực ứng dụng như y học, phân tích tài chính, thiên văn học và sinh học phân tử. Cây quyết định là cơ sở của một số hệ thống quy tắc thương mại.



Hình 3: Ví dụ về bài toán phân lớp sử dụng decision tree

3.2 Cây quyết định ID3

Từ cuối thập niên 1970 đến đầu thập niên 1980, J. Ross Quinlan, một nhà nghiên cứu machine learning, đã phát triển thuật toán cây quyết định với tên gọi ID3 (Iterative Dichotomiser). Quinlan sau đó đã trình bày C4.5 (một sự kế thừa của ID3), cái mà đã trở thành một chuẩn mực mà các thuật toán học có giám sát mới hơn thường được so sánh. Năm 1984, một nhóm các nhà thống kê (L. Breiman, J. Friedman, R. Olshen và C. Stone) công bố cuốn sách Classification and Regression Trees (CART) trong đó mô tả việc tạo ra các cây quyết định nhị phân (binary decision tree). ID3 và CART được phát minh độc lập với nhau tại cùng một thời điểm, nhưng theo một cách tiếp cận tương tự để học cây quyết định từ các bộ dữ liệu huấn luyện. Hai thuật toán nền tảng này đã tạo ra một loạt các công việc trên cây quyết định. Sau đây chúng ta sẽ tìm hiểu về thuật toán Iterative Dichotomiser 3 (ID3)

3.2.1 Ý tưởng

Trong ID3, chúng ta cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Với các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị

khác nhau, việc tìm được nghiệm tối ưu thường là không khả thi. Thay vào đó, một 47 phương pháp đơn giản thường được sử dụng là tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó. Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi child node. Việc chọn ra thuộc tính tốt nhất ở mỗi bước như thế này được gọi là cách chọn greedy (tham lam). Cách chọn này có thể không phải là tối ưu, nhưng trực giác cho chúng ta thấy rằng cách làm này sẽ gần với cách làm tối ưu. Ngoài ra, cách làm này khiến cho bài toán cần giải quyết trở nên đơn giản hơn.

Sau mỗi câu hỏi, dữ liệu được phân chia vào từng child node tương ứng với các câu trả lời cho câu hỏi đó. Câu hỏi ở đây chính là một thuộc tính, câu trả lời chính là giá trị của thuộc tính đó. Để đánh giá chất lượng của một cách phân chia, chúng ta cần đi tìm một phép đo.

Trước hết, thế nào là một phép phân chia tốt? Bằng trực giác, một phép phân chia là tốt nhất nếu dữ liệu trong mỗi child node hoàn toàn thuộc vào một class–khi đó child node này có thể được coi là một leaf node, tức ta không cần phân chia thêm nữa. Nếu dữ liệu trong các child node vẫn lẫn vào nhau theo tỉ lệ lớn, ta coi rằng phép phân chia đó chưa thực sự tốt. Từ nhận xét này, ta cần có một hàm số đo độ tinh khiết (purity), hoặc độ vẩn đục (impurity) của một phép phân chia. Hàm số này sẽ cho giá trị thấp nhất nếu dữ liệu trong mỗi child node nằm trong cùng một class (tinh khiết nhất), và cho giá trị cao nếu mỗi child node có chứa dữ liệu thuộc nhiều class khác nhau.

Một hàm số có các đặc điểm này và được dùng nhiều trong lý thuyết thông tin là hàm entropy.

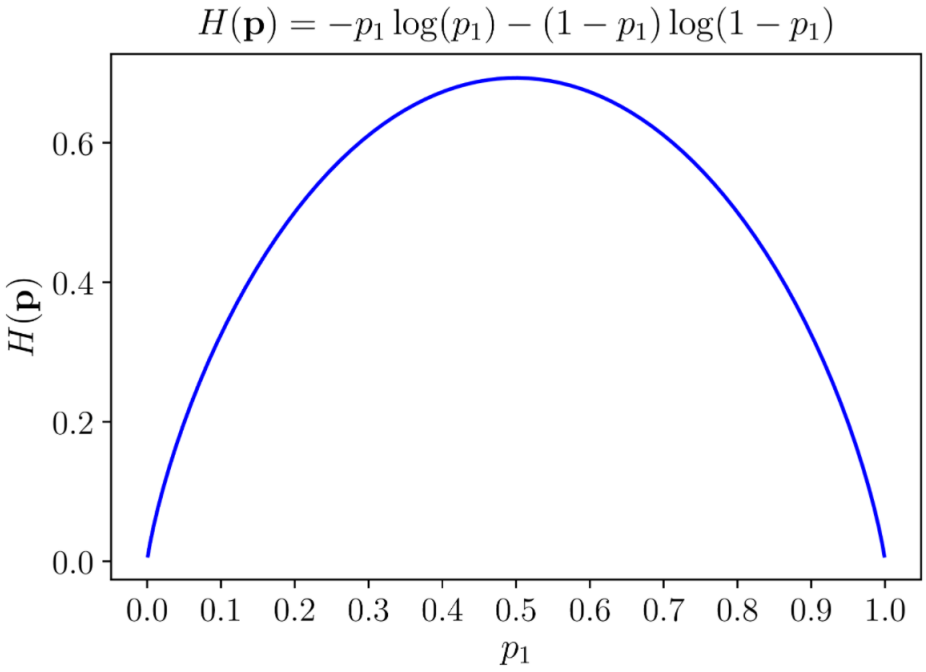
3.2.2 Hàm số Entropy

Cho một phân phối xác suất của biến rời rac $\boldsymbol{x}$ có thể nhận n giá trị khác nhau $x_l, x_2, \ldots, x_n$. Giả sử rằng xác suất để x nhận các giá trị này là $p_i = p\left(x = x_i\right)$ với $0 \leq p_i$

$\leq 1, \sum_{i=1}^{n} p_i = 1$. Ký hiệu phân phối này là $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. Entropy của phân phối này được định nghĩa là:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log(p_i)$$

trong đó log là logarit tự nhiên và quy ước $0 \log(0) = 0$.



Hình 4: Đồ thị hàm số entropy với n = 2

3.2.3 Thuật toán ID3

Xét một hài toán với C class khác nhau. Giả sử ta đang làm việc với một $non-$ leaf node với các điểm dữ liệu tạo thành một tập $\mathcal{S}$ với số phần tử là $|\mathcal{S}| = N$. Giả sử thêm rằng trong số N điểm dữ liệu này, $N_c, c = 1, 2, \ldots, C$ điểm thuộc vào class c được xấp xỉ bằng $\frac{N_c}{N}$ (maximum likelihood estimation). Như vậy, entropy tại node này được tính bởi:

$$H(\mathcal{S}) = -\sum_{c=1}^{C} \frac{N_c}{N} \log\left(\frac{N_c}{N}\right)$$

Tiếp theo, giả sử thuộc tính được chọn là x. Dựa trên x, các điểm dữ liệu trong $\mathcal{S}$ được phân ra thành K child node $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K$ với số điểm trong mỗi child node lần

lượt là $m_1, m_2, \ldots, m_K$. Ta định nghĩa:

$$H(x, \mathcal{S}) = \sum_{k=1}^{K} \frac{m_k}{N} H(\mathcal{S}_k)$$

là tổng có trọng số entroy của mỗi child node- -được tính tương tự như (2). Việc lấy trọng số này là quan trọng vì các node thường có số lượng điểm khác nhau. Tiếp theo, ta định nghĩa information gain dựa trên thuộc tính x

$$G(x, \mathcal{S}) = H(\mathcal{S}) - H(x, \mathcal{S})$$

Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên:

$$x^* = \arg\max_x G(x, \mathcal{S}) = \arg\min_x H(x, \mathcal{S})$$

tức thuộc tính khiến cho *information gain* đạt giá trị lớn nhất.

3.2.4 Điều kiện dừng

Trong các thuật toán decision tree nói chung và ID3 nói riêng, nếu ta tiếp tục phân chia các node chưa tinh khiết, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, tree có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng overfitting sẽ xảy ra.

Để tránh overfitting, một trong số các phương pháp sau có thể được sử dụng. Tại một node, nếu một trong số các điều kiện sau đây xảy ra, ta không tiếp tục phân chia node đó và coi nó là một leaf node:

- Nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
- Nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
- Nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế chiều sâu của tree này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.

- Nếu tổng số leaf node vượt quá một ngưỡng nào đó.
- Nếu việc phân chia node đó không làm giảm entropy quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

Ngoài các phương pháp trên, một phương pháp phổ biến khác được sử dụng để tránh overfitting là *pruning*, tạm dịch là *cắt tỉa*.

3.2.5 Pruning

Pruning là một kỹ thuật regularization để tránh overfitting cho decision tree nói chung. Trong pruning, một decision tree sẽ được xây dựng tới khi mọi điểm trong training set đều được phân lớp đúng. Sau đó, các leaf node có chung một non-leaf node sẽ được cắt tỉa và non-leaf node đó trở thành một leaf-node, với class tương ứng với class chiếm đa số trong số mọi điểm được phân vào node đó. Việc cắt tỉa cây quyết định này có thể được xác định dựa vào các cách sau.

1. Dựa vào một validation set. Trước tiên, training set được tách ra thành một training set nhỏ hơn và một validation set. Decision tree được xây dựng trên training set cho tới khi mọi điểm trong training set được phân lớp đúng. Sau đó, đi ngược từ các leaf node, cắt tỉa các sibling node của nó và giữ lại node bố mẹ nếu độ chính xác trên validation set được cải thiện. Khi nào độ chính xác trên validation set không được cải thiện nữa, quá trình pruning dừng lại. Phương pháp này còn được gọi là *reduced error pruning*.

2. Dựa vào toàn bộ data set. Trong phương pháp này, ta không tách tập training ban đầu ra mà sử dụng toàn bộ dữ liệu trong tập này cho việc xây dựng decision tree. Một ví dụ cho việc này là cộng thêm một đại lượng regularization vào hàm mất mát. Đại lượng regularization sẽ lớn nếu số leaf node là lớn. Cụ thể, giả sử decision tree cuối cùng có K leaf node, tập hợp các điểm huấn luyện rơi vào mỗi leaf node lần lượt là $\mathcal{S}_1, \ldots, \mathcal{S}_K$. Khi đó, regularized loss của ID 3 có thể

được tính tương tự như (3):

$$\mathcal{L} = \sum_{k=1}^{K} \frac{|\mathcal{S}_k|}{|\mathcal{S}|} H(\mathcal{S}_k) + \lambda K$$

với $|\mathcal{S}_k|$ ký hiệu số phần tử của tập hợp $\mathcal{S}_k$ và $H(\mathcal{S}_k)$ chính là entropy của leaf node tương ứng với $\mathcal{S}_k$, được tính tương tự như (2), và λ là một số thực dương không quá lớn. Giá trị của hàm số này nhỏ nếu cả data loss-số hạng thứ nhất-nhỏ (entropy tại mỗi node là thấp), và regularization-số hạng thứ hai-cũng nhỏ (số leaf node là ít). Vì hàm mất mát trong (5) là một hàm rời rạc, rất khó để trực tiếp tối ưu hàm này. Việc tối ưu có thể được thực hiện thông qua pruning như sau. Trước hết, xây dựng một decision tree mà mọi điểm trong tập huấn luyện đều được phân loại đúng (toàn bộ các entopy của các node bằng 0). Lúc này data loss bằng 0 nhưng regularization có thể lớn, khiến cho $\mathcal{L}$ lớn. Sau đó, ta có thể tỉa dần các leaf node sao cho $\mathcal{L}$ giảm. Việc cắt tỉa được lặp lại đến khi $\mathcal{L}$ không thể giảm được nữa.

3.2.6 Giải thuật ID3

Thuật toán ID3 xây dựng cây quyết định bằng cách sử dụng giải thuật tìm kiếm tham lam (greedy search) đối với không gia cây quyết định có thể tìm được. Việc xây dựng các nút (node) theo chiến lược Top-Down từ trên xuống, xuất phát từ nút gốc (root).

Các bước thực hiện:

1. Bắt đầu với tập S ban đầu là nút gốc.
2. Trên mỗi lần lặp của thuật toán, nó sẽ lặp qua thuộc tính rất không được sử dụng của tập S và tính toán Entropy (H) và Độ lợi thông tin (IG) của thuộc tính này.
3. Sau đó, nó sẽ chọn thuộc tính có mức tăng Entropy nhỏ nhất hoặc thông tin là lớn nhất.

4. Tập hợp S sau đó được chia theo thuộc tính đã chọn để tạo ra một tập con dữ liệu.

5. Thuật toán tiếp tục lặp lại trên mỗi tập hợp con, chỉ xem xét các thuộc tính chưa từng được chọn trước đó.

6. Quá trình phát triển cây sẽ tiếp tục cho tới khi:

   - Cây quyết định phân loại hoàn toàn (perfectly classifies) các dữ liệu đầu vào.

   - Tất cả các thuộc tính được sử dụng.

Mã giả:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C, **then**
(3) return N as a leaf node labeled with the class C;
(4) **if** attribute list is empty **then**
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, attribute list) to **find** the “best” splitting criterion;
(7) label node N with splitting criterion;
(8) **if** splitting attribute is discrete-valued **and** multiway splits allowed **then** // not restricted to binary trees
(9) attribute list ← attribute list splitting attribute; // remove splitting attribute
(10) **for each** outcome j of splitting criterion // partition the tuples and grow sub-trees for each partition
(11) let Dj be the set of data tuples in D satisfying outcome j; // a partition (12) **if** Dj is empty **then**
(13) attach a leaf labeled with the majority class in D to node N;
(14) **else** attach the node returned by **Generate_decision_tree**(Dj, attribute list) to node N; **endfor**
(15) return N;

3.3 Xây dựng mô hình giải quyết bài toán

Bước đầu tiên, ta cần tiến hành tiền xử lý dữ liệu, do dữ liệu đầu vào không đầy đủ, cần có bước tiền xử lý dữ liệu để thuận lợi cho quá trình xây dựng mô hình. Dưới đây là hàm raw_data_preprocessing hỗ trợ cho việc tiền xử lý dữ liệu.

```
def raw_data_preprocessing(df):
  df['Communication'] = df['Communication'].fillna('none')
  df['Outcome'] = df['Outcome'].fillna('NoPrev')
  df['Job'] = df['Job'].fillna('management')
  df['Education'] = df['Education'].fillna('secondary')
  df['CallStart'] = pd.to_datetime(df['CallStart'] )
  df['CallEnd'] = pd.to_datetime(df['CallEnd'] )
  df['CallTime'] = (df['CallEnd'] - df['CallStart']).dt.total_seconds()
  df.drop(["CallStart","CallEnd"],axis=1, inplace= True)
  feature = ['Job','Marital','Education','LastContactMonth',
  'Outcome','Communication']
  df_onehot = pd.get_dummies(df[feature])
  df.drop(['Job','Marital','Education','LastContactMonth','Outcome',
  'Communication'], axis=1, inplace= True)
  df= pd.concat([df_onehot, df], axis=1)
 return df
```

Tiếp đến là xây dựng mô hình train dữ liệu, đầu tiên ta phân bộ dữ liệu đã gán nhãn ra thành 2 tập train và tập test sau đó tiến hành tạo mô hình bằng thư viện và đánh giá mô hình. Tạo ra mô hình cây quyết định dựa trên dữ liệu thực tế, sau đó tiến hành đánh giá các mô hình đó. Trên python, nếu sử dụng scikit-learn thì công việc rất dễ dàng bằng việc sử dụng class sklearn.tree.DecisionTreeClassifier,

```
df= pd.read_csv('carInsurance_train.csv')
df = raw_data_preprocessing(df)
x= df.drop(['CarInsurance','Id'],axis=1)
y= df['CarInsurance']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,
   random_state=2)
dt = tree.DecisionTreeClassifier(criterion='entropy',
max_depth=8, min_samples_split=5, random_state=20)
dt.fit(x_train,y_train)
joblib.dump(dt, 'decision_tree_model.joblib')
```

Cuối cùng là xây dựng một hàm xử lý dữ liệu đầu vào và dự đoán.

```
def pre_carinsurance(df1):
  dt = joblib.load('decision_tree_model.joblib')
  df2 = df1
  df1 = raw_data_preprocessing(df1)
  df1 = df1.drop(['CarInsurance','Id'],axis=1)
  pred = dt.predict(df1)
  pred = pd.DataFrame(pred, columns=["CarInsurance"])
  df2= pd.read_csv('carInsurance_test.csv')
  df2 = df2.drop(["CarInsurance"], axis=1)
  results = pd.concat([df2, pred], axis=1)
  return results
```

Sau khi chạy tập dự đoán, kết quả ta thu được là:

Như đã biết mô hình không thể đạt được độ chính xác là 100%, nhìn vào hình 5 ta có thể thấy, độ chính xác của mô hình lên đến **81.33%**. Mô hình đã được kiểm tra và không xảy ra tình trạng overfiting.

Nhìn chung thì mô hình cho ra kết quả rất tốt và có thể sử dụng để xây dựng giải

```
              precision    recall  f1-score   support

           0       0.85      0.84      0.85       732
           1       0.75      0.77      0.76       468

    accuracy                           0.81      1200
   macro avg       0.80      0.81      0.80      1200
weighted avg       0.81      0.81      0.81      1200


Confusion Matrix:
 [[614 118]
 [106 362]]

Accuracy:  0.8133333333333334
```

Hình 5: Kết quả chạy dự đoán mô hình

pháp phần mềm.

4 Triển khai phần mềm

Hệ thống dự đoán khả năng thành công của chiến dịch tiếp thị được xây dựng với mục đích đưa ra dự đoán những khách hàng nào có khả năng mau hay không mua bảo hiểm, phân tích các yếu tố đặc trưng dẫn đến các khả năng đó, từ đó hỗ trợ nhà quản lý đưa ra các chính sách tiếp thị.

- Nhà quản lý được toàn quyền quản lý dữ liệu trên hệ thống

- Xem kết quả dự báo và các thông kê đi kèm. Các kết quả dự báo hiển thị đủ thông tin khách hàng.

4.1 Phân tích hệ thống

4.1.1 Phân tích chức năng hệ thống

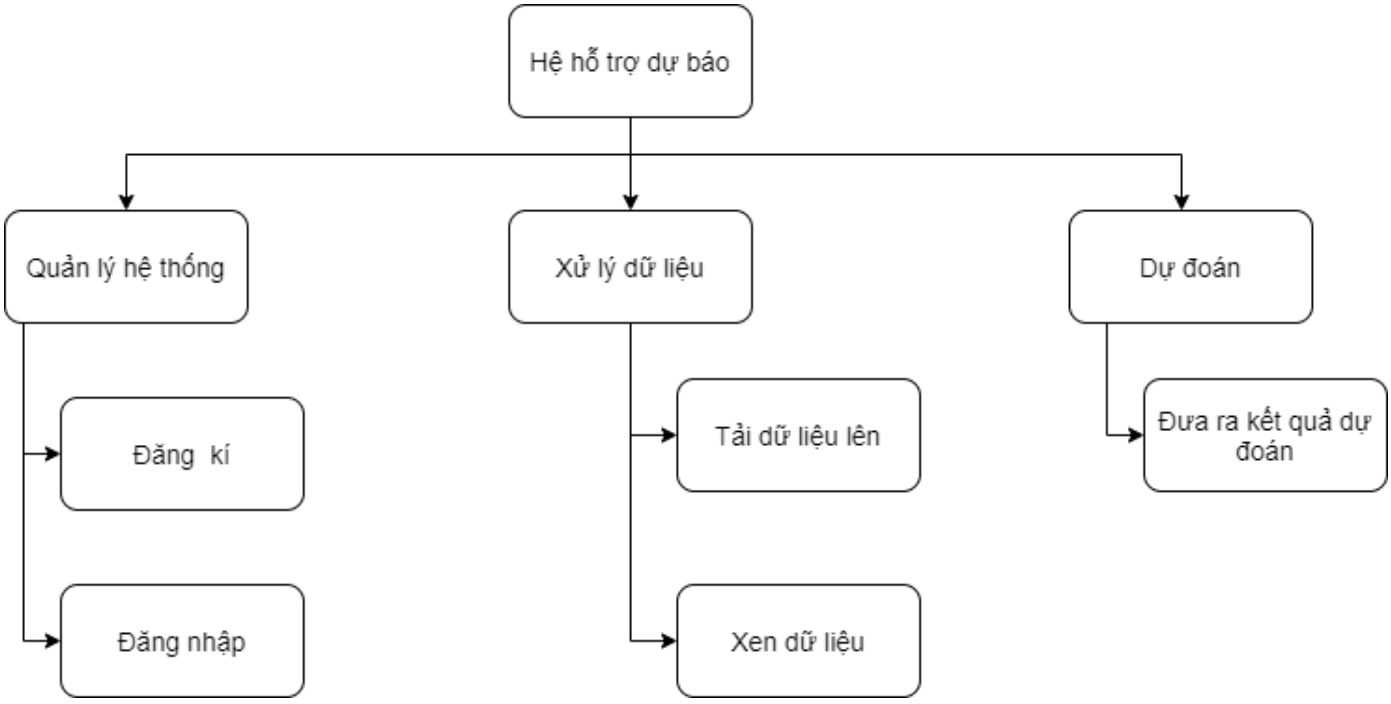
Hệ thống bao gồm một số chức năng chính:

- Đăng nhập, đăng kí.

- Tải dữ liệu lên, xem dữ liệu.

- Đưa ra kết quả dự đoán: có thể xem kết quả dự đoán trên hệ thống.

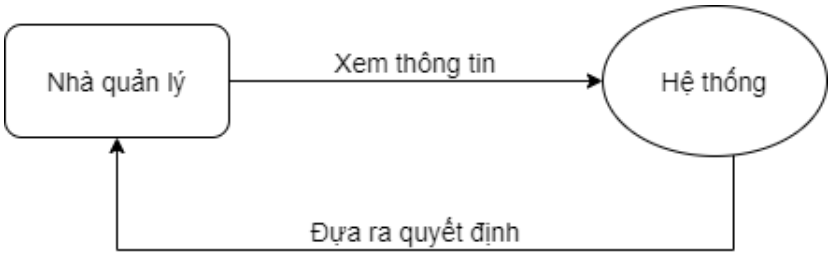
4.1.2 Biểu đồ phân cấp chức năng



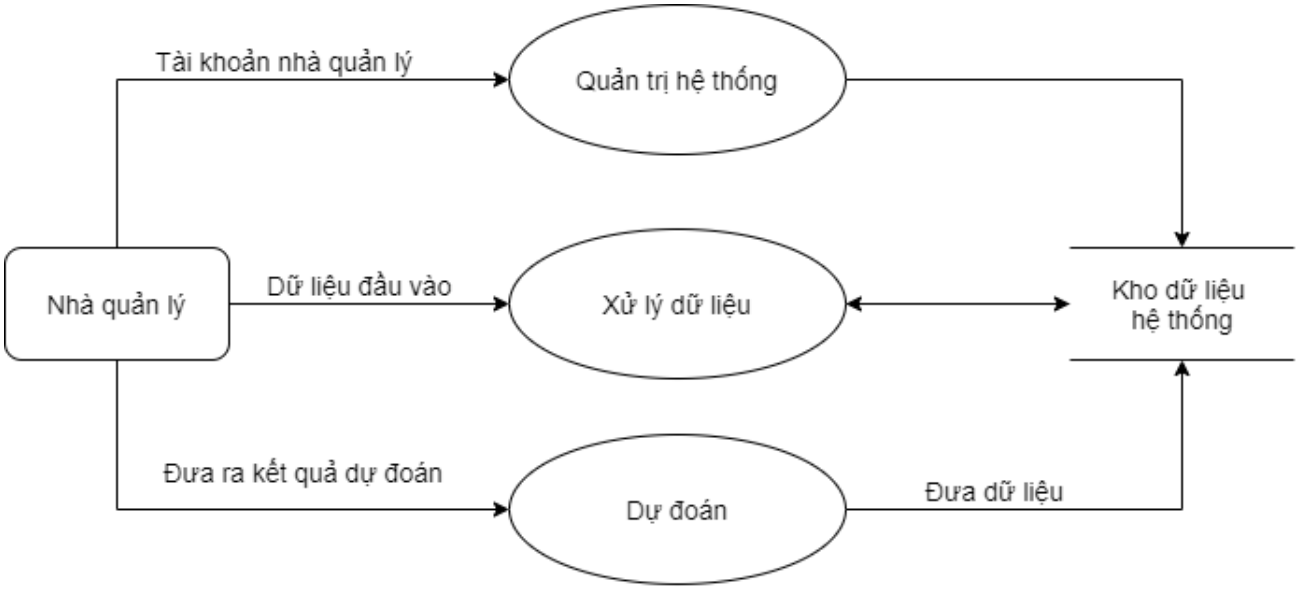
Hình 6: Biểu đồ phân cấp chức năng

4.1.3 Biểu đồ luồng dữ liệu

1. Biểu đồ luồng dữ liệu mức ngữ cảnh

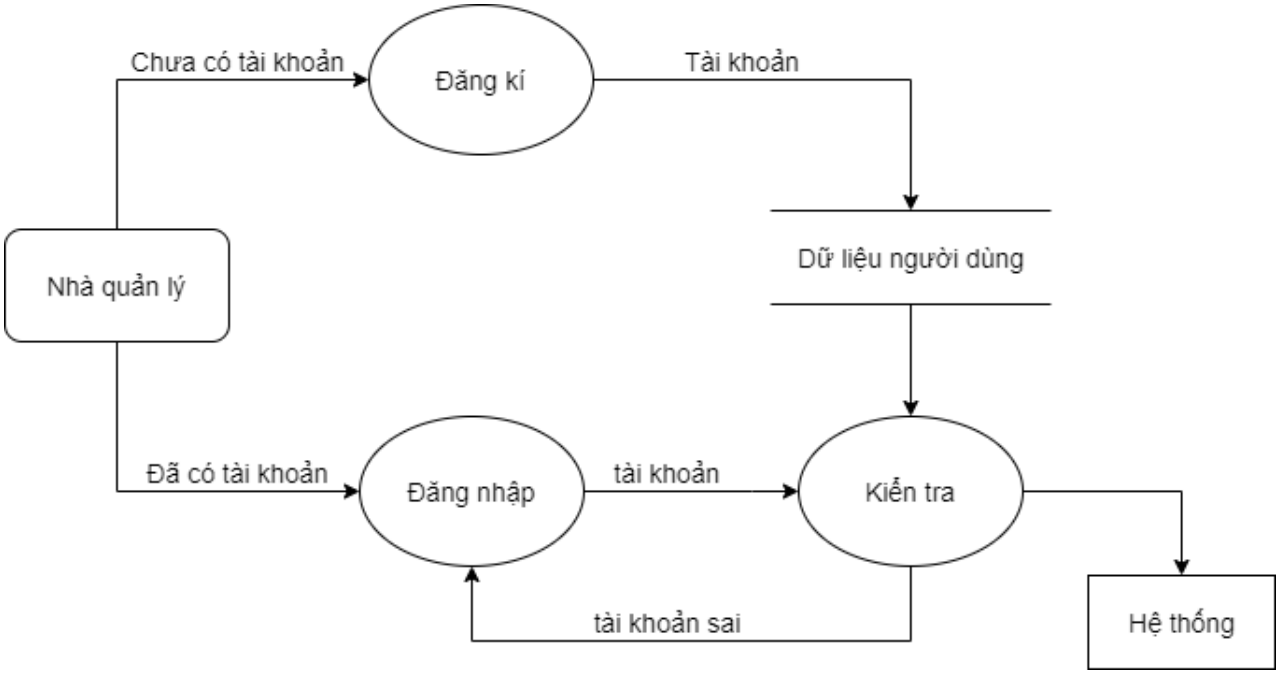


2. Biểu đồ luồng dữ liệu mức đỉnh

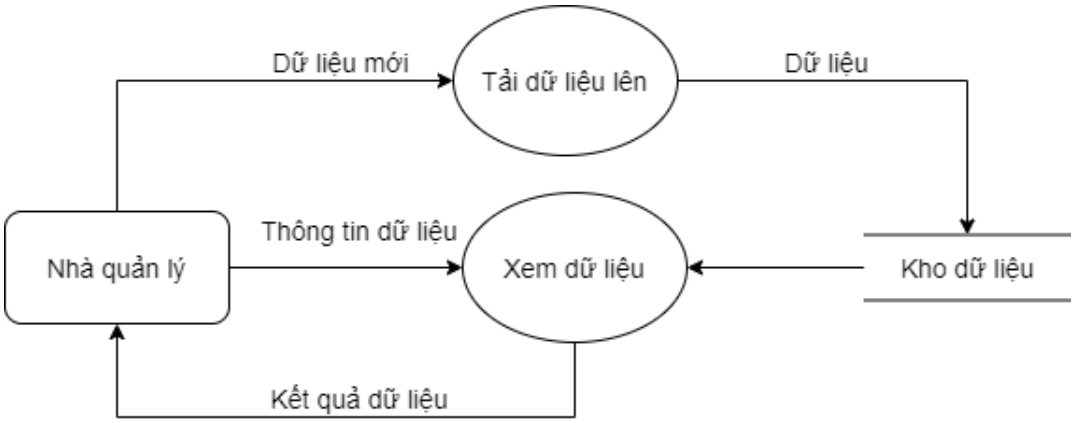


3. Biểu đồ luồng dữ liệu mức dưới đỉnh

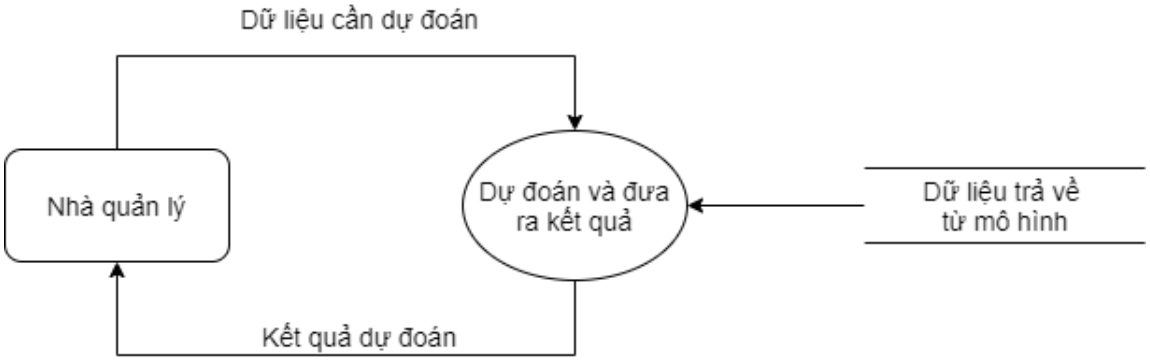
   (a) Chức năng quản trị hệ thống



(b) Chức năng xử lý dữ liệu

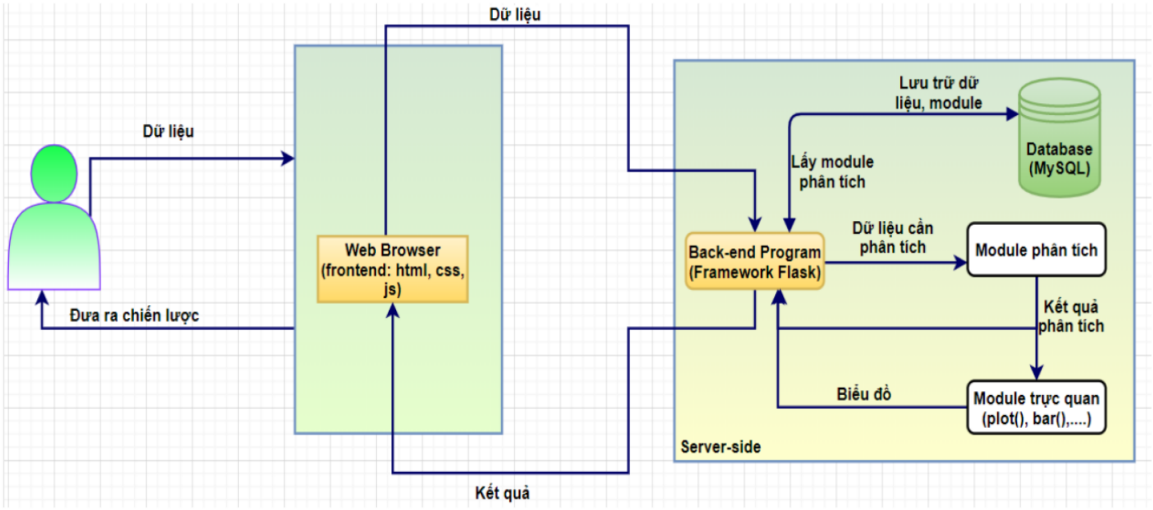


(c) Chức năng dự báo



4.2 Thiết kế hệ thống

4.2.1 Mô hình hệ thống



Hình 7: Mô hình hệ thống

4.2.2 Thiết kế về mặt giao diện

1. Font-end

(a) *HTML*

HTML là từ viết tắt của HyperText Markup Language (ngôn ngữ đánh dấu siêu văn bản) dùng mô tả cấu trúc của các trang Web và tạo ra các loại tài liệu có thể xem được trong trình duyệt.

HTML là ngôn ngữ cơ bản và phổ biến nhất và được dùng để tạo ra tất cả các website trên thế giới. Không một website nào có thể tồn tại mà không dùng HTML. Phiên bản hiện tại là HTML5 ưu việt hơn so với các phiên bản trước.

(b) *CSS*

CSS (viết tắt của Cascading Style Sheets) là một ngôn ngữ định dạng được sử dụng để mô tả trình bày các trang Web, bao gồm màu sắc, cách bố trí và phông

chữ. CSS cho phép chúng hiển thị nội dung tương thích trên các loại thiết bị có kích thước màn hình khác nhau, chẳng hạn như màn hình lớn, màn hình nhỏ như điện thoại hay máy tính bản. Nhờ vậy có thể tối ưu hóa văn bản của mình cho người sử dụng.

Chúng ta cần nhúng CSS vào tài liệu HTML. Nếu không, các định dạng CSS sẽ không ảnh hưởng đến các trình duyệt của bạn hiển thị tài liệu HTML.

(c) *Javascript*

Javascript là một ngôn ngữ nhanh và nhẹ chạy trong môi trường máy chủ lưu trữ (ví dụ: trình duyệt web), JavaScript có thể được kết nối với các đối tượng của môi trường để cung cấp kiểm soát chương trình đối với chúng. JavaScript cho phép bạn thực hiện những điều phức tạp trên các trang web như bản đồ tương tác. . . Javascript được hỗ trợ hầu như trên tất cả các trình duyệt như Firefox, Chrome, . . . thậm chí các trình duyệt trên thiết bị di động cũng có hỗ trợ.

(d) *Boostrap*

Bootstrap là một framework cho phép thiết kế website eponsive nhanh hơn và dễ dàng hơn Bootstrap là bao gồm các HTML templates, CSS templates và Javascript tạo ra những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác. Trong bootstrap có thêm các plugin Javascript trong nó. Giúp cho việc thiết kế reponsive của bạn dễ dàng hơn và nhanh chóng hơn.

2. Back-end

(a) *Python*

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ

có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

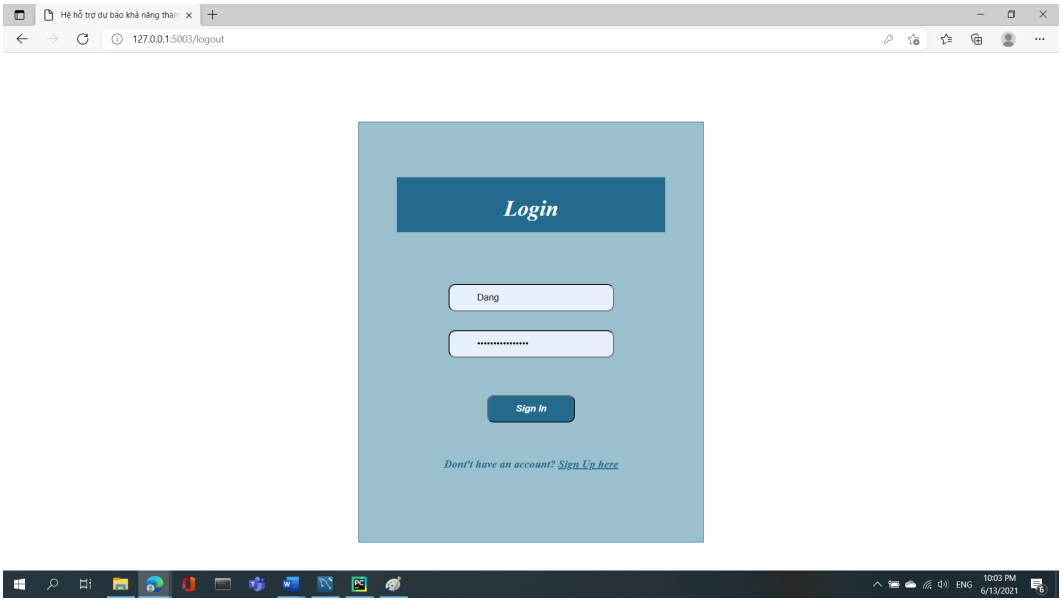
Python là một ngôn ngữ lập trình được dùng trong nhiều lĩnh vực như lập trình web, tạo nguyên mẫu phần mềm, dùng trong khoa học tính toán, như Machine learning với nhiều thư viện liên quan như NumPy, SciPy, ...

(b) *Flask*

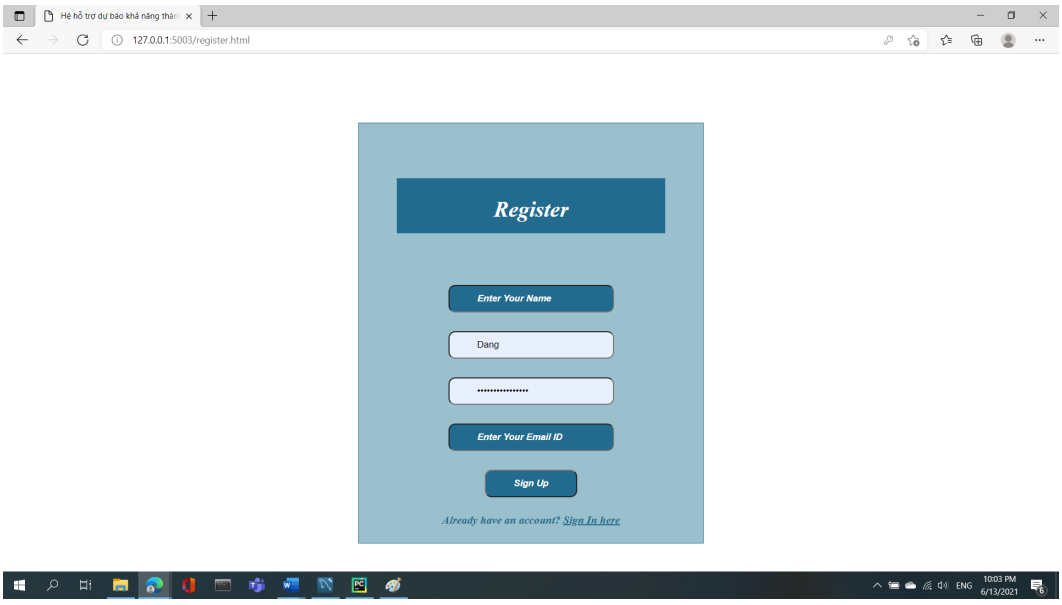
Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép bạn xây dựng các ứng dụng web từ đơn giản tới phức tạp. Nó có thể xây dựng các api nhỏ, ứng dụng web chẳng hạn như các trang web, blog, trang wiki hoặc một website dựa theo thời gian hay thậm chí là một trang web thương mại. Flask cung cấp cho bạn công cụ, các thư viện và các công nghệ hỗ trợ bạn làm những công việc trên. Flask là một môi trường độc lập, ít sử dụng các thư viện khác bên ngoài.

Flask đơn giản, dễ học, dễ sử dụng phù hợp với người mới bắt đầu. Vì vậy, em đã chọn Flask để làm back-end cho hệ thống.

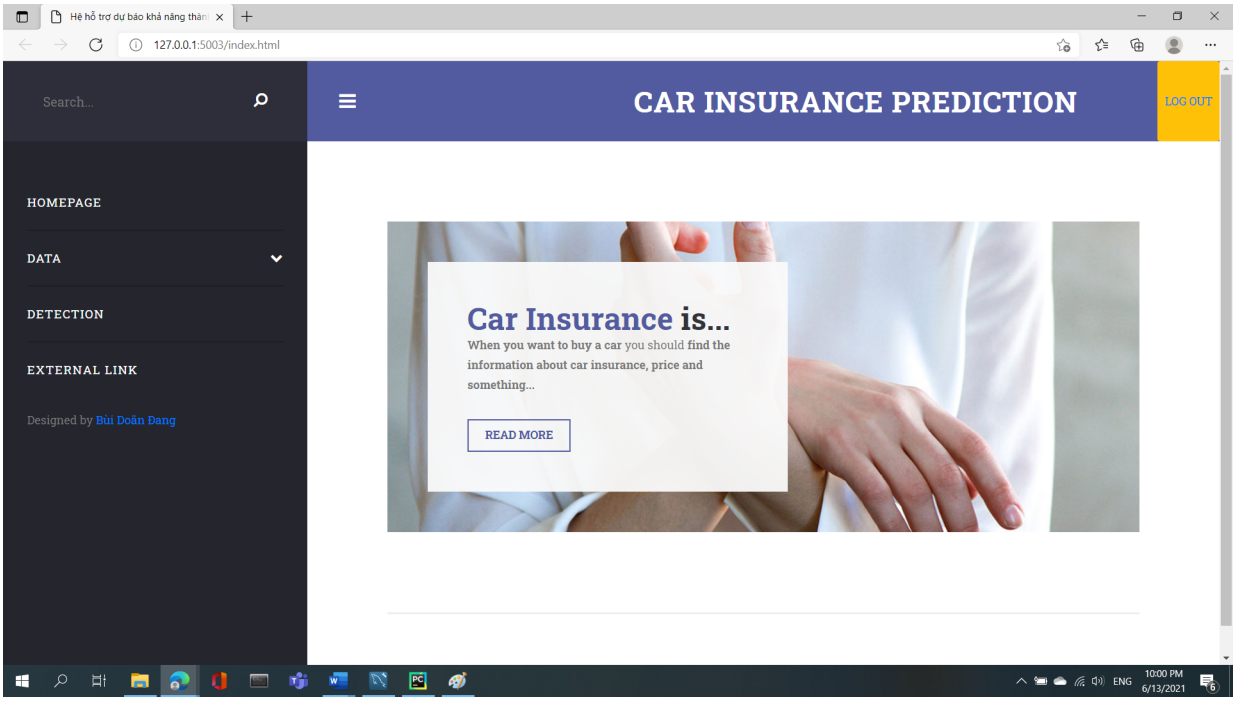
4.2.3 Giao diện website



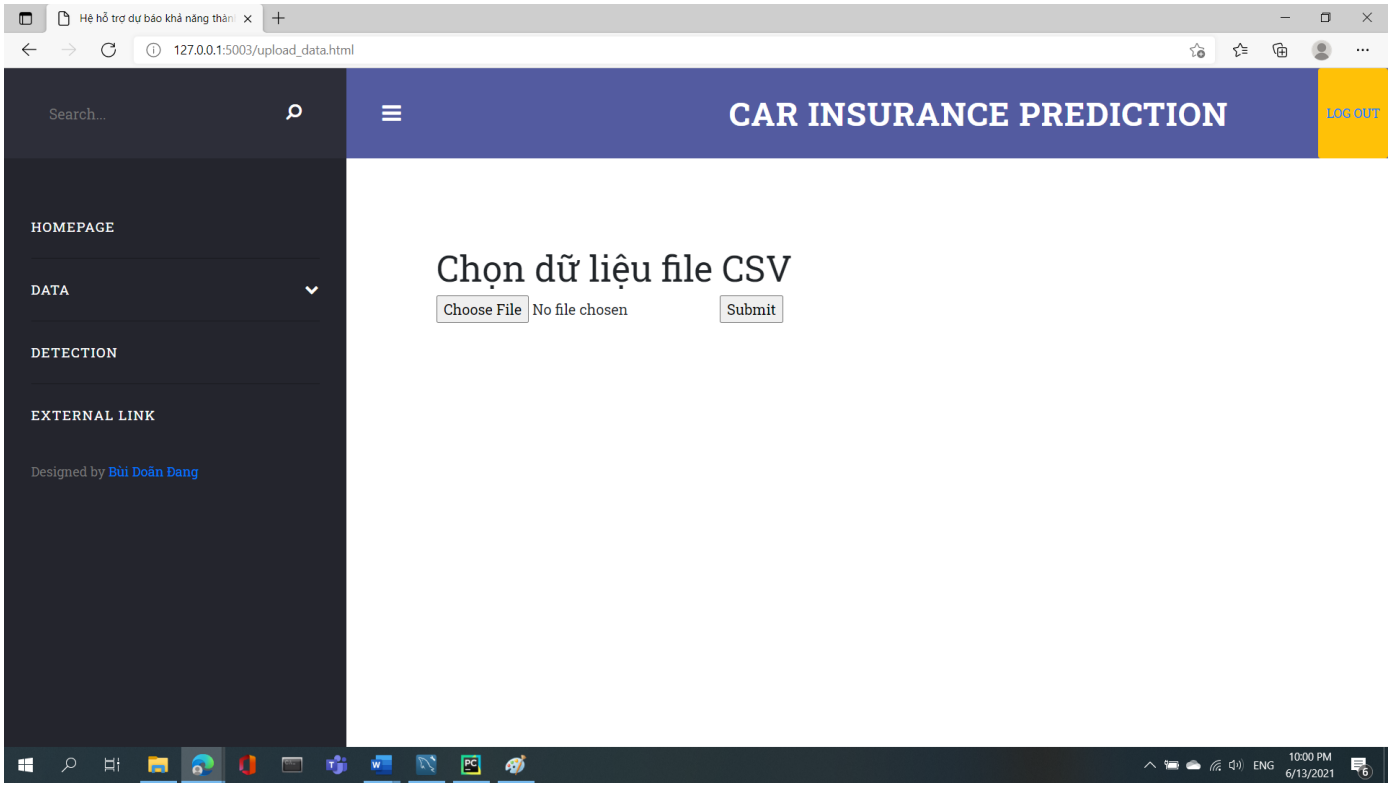
Hình 8: Giao diện đăng nhập



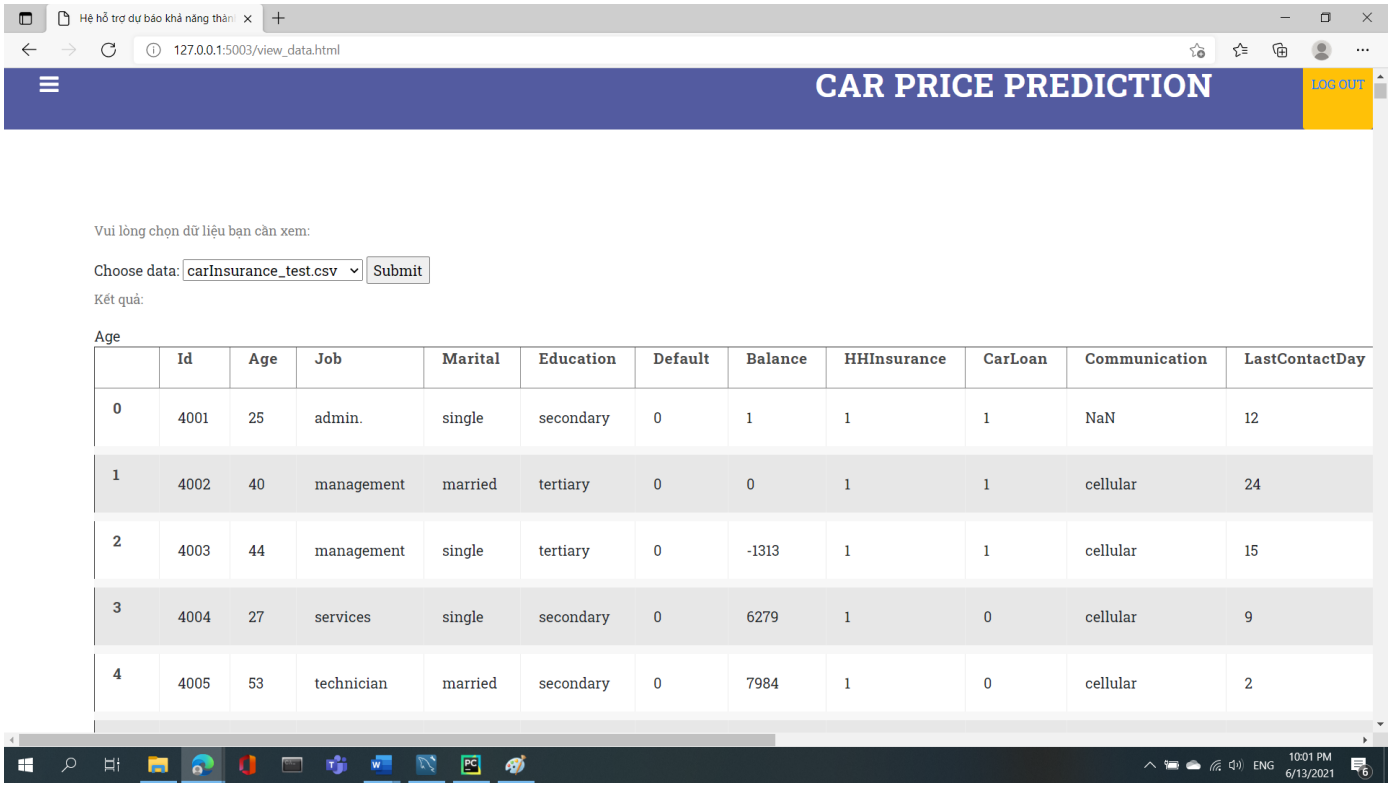
Hình 9: Giao diện đăng kí



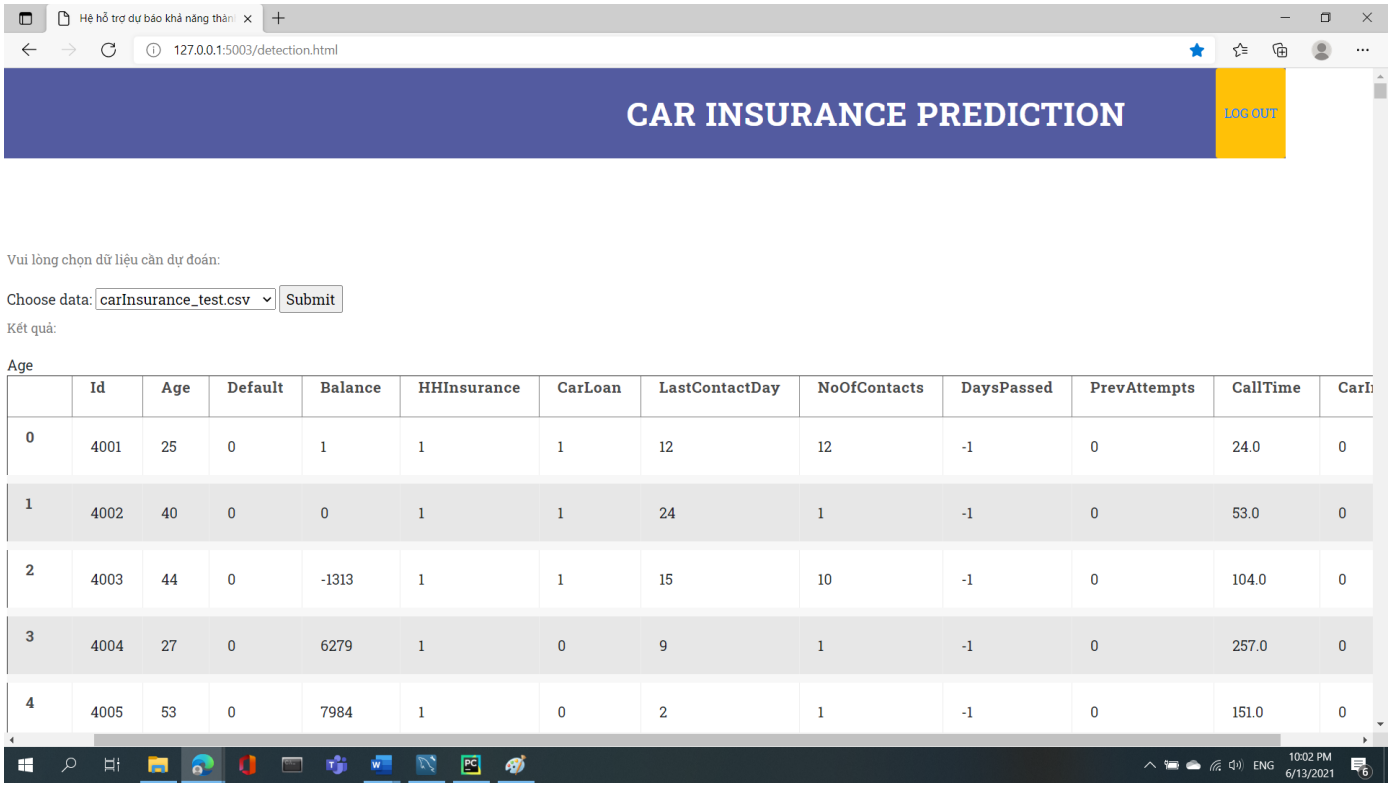
Hình 10: Giao diện trang chủ



Hình 11: Giao diện tải dữ liệu lên



Hình 12: Giao diện xem dữ liệu



Hình 13: Giao diện dự đoán

Kết luận

Trong báo cáo em đã giới thiệu bài toán, mô tả dữ liệu sử dụng, trình bày về cơ sở lý thuyết, xây dựng mô hình và quy trình phân tích thiết kế hệ thống dự đoán khả năng thành công của chiến dịch tiếp thị bảo hiểm ô tô. Tiến hành xây dựng và thiết kế hệ thống đưa vào vận hành và đã đáp ứng được các chức năng thiết kế của hệ thống. Trong quá trình viết báo cáo và xây dựng hệ thống khó tránh khỏi được những sai sót, em mong nhận được những góp ý chỉnh sửa từ Thầy để báo cáo của em được hoàn thiện hơn.

Hà Nội, tháng 6 năm 2021

Tác giả báo cáo

Bùi Doãn Đang

Tài liệu

[1] Tiep Vu Huu. 2018. *Machine Learning cơ bản*

[2] Jiawei, Micheline Kamber, Jian Pei Simon. *Data Mining: Concepts and Techniques Third Edition*. 225 Wyman Street, Waltham, MA 02451, USA, 225 Wyman Street, Waltham, MA 02451, USA, 2012

[3] https://www.kaggle.com/kondla/carinsurance

[4] https://en.wikipedia.org/wiki/Decision_tree_learning

[5] Internet