**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

o0o

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

DỰ ĐOÁN KHÁCH HÀNG RỜI BỎ

(Customer Churn Prediction)

Nguyễn Thành Trung -20203915

**TP. HÀ NỘI, THÁNG 1 NĂM 2023**

**LỜI CẢM ƠN**

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Đại học Bách khoa Hà Nội đã đưa môn học Nhập môn Trí tuệ nhân tạo vào chương trình giảng dạy. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ đã nhiệt tình giảng dạy và luôn góp ý, truyền đạt những kiến thức quý báu trong cả những tiết học ở trường và ngoài giờ, giúp đỡ em trong suốt thời gian học tập vừa qua để nhóm em có thể hoàn thành bài tập lớn môn học này một cách tốt nhất có thể. Bài tập lớn môn Nhập môn Trí tuệ nhân tạo với đề tài “Dự đoán khách hàng rời bỏ” là kết quả của quá trình tìm tòi, trau dồi của nhóm em. Trong quá trình hoàn thành báo cáo, nhóm em chắc chắn không tránh khỏi những thiếu sót và nhóm rất mong nhận được những lời nhận xét từ thầy để bài báo cáo được hoàn thiện hơn. Cuối cùng, nhóm em xin kính chúc thầy có thật nhiều sức khỏe, hạnh phúc và luôn luôn thành công trong sự nghiệp giảng dạy.

**MỤC LỤC**

1. [GIỚI THIỆU 4](#_bookmark0)
   1. [Tổng quan 4](#_bookmark1)
   2. [Dữ liệu 4](#_bookmark2)
2. [PHƯƠNG PHÁP 6](#_bookmark3)
   1. [Hồi quy Logistic (Logistic Regression) 6](#_bookmark4)
   2. [Cây quyết định (Decision Tree) 7](#_bookmark5)
   3. [Rừng ngẫu nhiên (Random Forest) 9](#_bookmark6)
   4. [Gaussian Naïve Bayes 9](#_bookmark7)
   5. [K láng giềng gần nhất (K Nearst Neighbour) 11](#_bookmark8)
   6. [Stacking Model 11](#_bookmark9)
3. [PHƯƠNG PHÁP ĐÁNH GIÁ 13](#_bookmark10)
   1. [Kiểm định chéo bằng StratifiedKFold 13](#_bookmark11)
   2. [Các độ đo chính xác (Accuracy, Precision, Recall, F1-score) 13](#_bookmark12)
4. [THỰC HIỆN 15](#_bookmark13)
   1. [Chia tập dữ liệu 15](#_bookmark14)
   2. [Khai phá dữ liệu 16](#_bookmark15)
   3. [Tiền xử lý dữ liệu 22](#_bookmark16)
   4. [Tạo đoạn mã cơ sở (cùng với pipeline dữ liệu) 23](#_bookmark17)
   5. [Huấn luyện 23](#_bookmark18)
   6. [Đánh giá 25](#_bookmark19)
5. [TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN 27](#_bookmark20)

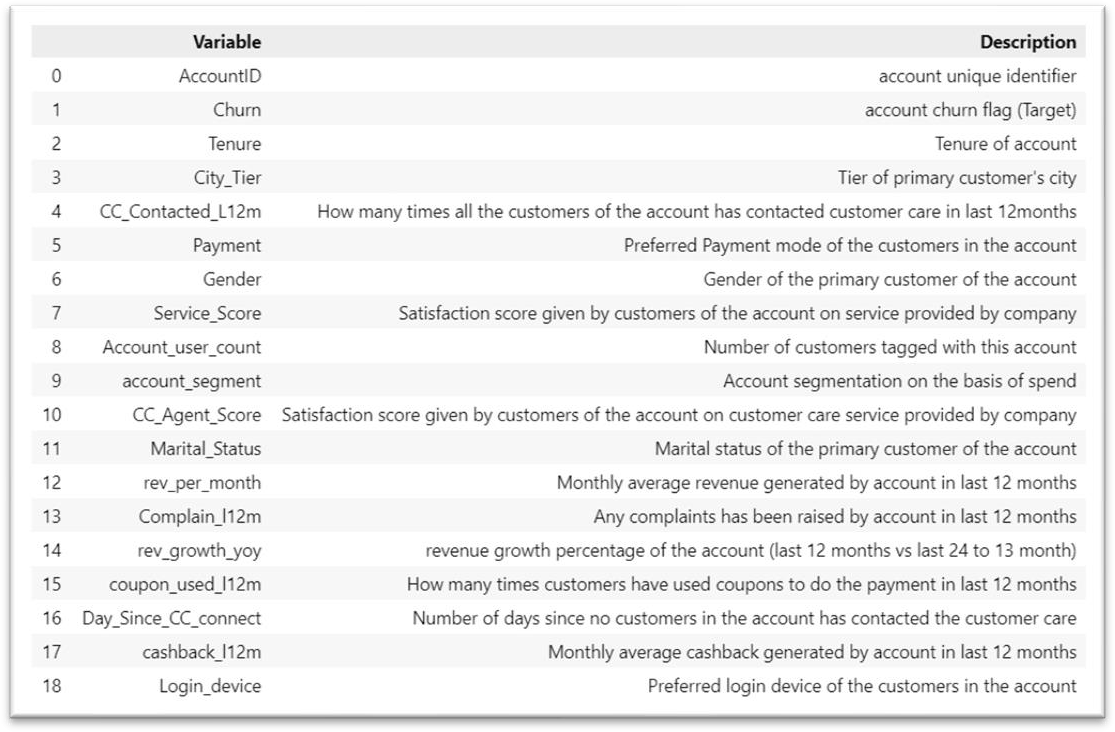
# GIỚI THIỆU

## Tổng quan

Một nhà cung cấp dịch vụ DTH (Direct to Home) trên thị trường đang phải đối mặt với những thách thức để giữ chân khách hàng hiện tại do có sự cạnh tranh gay gắt trong tình hình hiện tại. Trong công ty này, một tài khoản có thể có nhiều khách hàng được gắn thẻ, vì thế khi mất một tài khoản, công ty có thể mất nhiều hơn một khách hàng. Đồng thời, chi phí để quáng cáo sản phẩm tới những khách hàng mới thường đắt đỏ hơn với việc giữ chân khách hàng cũ. Do đó, tài khoản rời đi là một vấn đề lớn cần giải quyết. Dự đoán khách hàng rời bỏ có nghĩa là phát hiện khách hàng nào có khả năng rời khỏi dịch vụ hoặc hủy đăng ký dịch vụ. Khi bạn có thể xác định những khách hàng có nguy cơ hủy bỏ, bạn nên biết chính xác hành động tiếp thị nào cần thực hiện cho từng khách hàng riêng lẻ để tối đa hóa cơ hội khách hàng sẽ ở lại.

## Dữ liệu

Tập dữ liệu được sử dụng trong đồ án môn học này được công khai trên Kaggle ([Capstone Project - Churn Prediction | Kaggle](https://www.kaggle.com/code/vsridevi/capstone-project-churn-prediction/data?scriptVersionId=110749928)) bao gồm 11260 dòng và 19 cột. Mỗi dòng trong tập dữ liệu chứa thông tin của 1 tài khoản đăng ký dịch vụ của DTH. Mỗi cột trong tập dữ liệu đại diện cho 1 thông tin liên quan đến tài khoản như các thông tin về nhân khẩu học, về sử dụng dịch vụ của tài khoản, … Dưới đây là mô tả của các cột trong tập dữ liệu:



* Thông tin khách hàng chủ yếu sử dụng tài khoản được thể hiện qua các cột:
  + City\_Tier: Cấp thành phố của khách hàng chính sử dụng tài khoản
  + Gender: Giới tính của khác hàng chính sử dụng tài khoản
  + Marital\_Status: Tình trạng hôn nhân của khách hàng chính sử dụng tài khoản
* Thông tin tài khoản được thể hiện qua các cột:
  + AccountID: Số định danh của tài khoản
  + account\_segment: Phân khúc của tài khoản dựa trên lượng chi tiêu
* Thông tin về hành vi sử dụng dịch vụ của tài khoản được thể hiện qua các cột:
  + Tenure: Nhiệm kỳ của tài khoản
  + CC\_Contacted\_L12m: Số lần các khách hàng sử dụng tài khoản liên hệ với dịch vụ chăm sóc khách hàng trong vòng 12 tháng qua
  + Payment: Phương thức thanh toán chủ yếu của khách hàng sử dụng tài khoản
  + Day\_Since\_CC\_connect: Số ngày không có khách hàng nào trong tài khoản liên hệ với bộ phận chăm sóc khách hàng
  + coupon\_used\_l12m: Số lần khách hàng sử dụng phiếu giảm giá để thanh toán trong 12 tháng qua
  + Account\_user\_count: Số lượng khách hàng cùng được gắn với tài khoản này
  + Login\_device: Thiết bị đăng nhập chủ yếu của khách hàng trong tài khoản
* Thông tin về đánh giá dịch vụ của tài khoản được thể hiện qua các cột:
  + Service\_Score: Điểm đánh giá đối với dịch vụ của công ty cung cấp
  + CC\_Agent\_Score: Điểm đánh giá đối với dịch vụ chăm sóc của công ty cung cấp
  + Complain\_l12m: Phàn nàn đến từ tài khoản trong 12 tháng qua
* Thông tin liên quan đến doanh thu của tài khoản được thể hiện qua các cột:
  + rev\_per\_month: Doanh thu trung bình hàng tháng được tạo bởi tài khoản trong 12 tháng qua
  + rev\_growth\_yoy: Phần trăm tăng trưởng doanh thu của tài khoản (12 tháng qua so với 24 đến 13 tháng qua)
  + cashback\_l12m: Tiền hoàn lại trung bình hàng tháng được tạo bởi tài khoản trong 12 tháng qua
* Cột mục tiêu, cũng là cột cần dự đoán:
  + Churn: Trạng thái rời bỏ của tài khoản

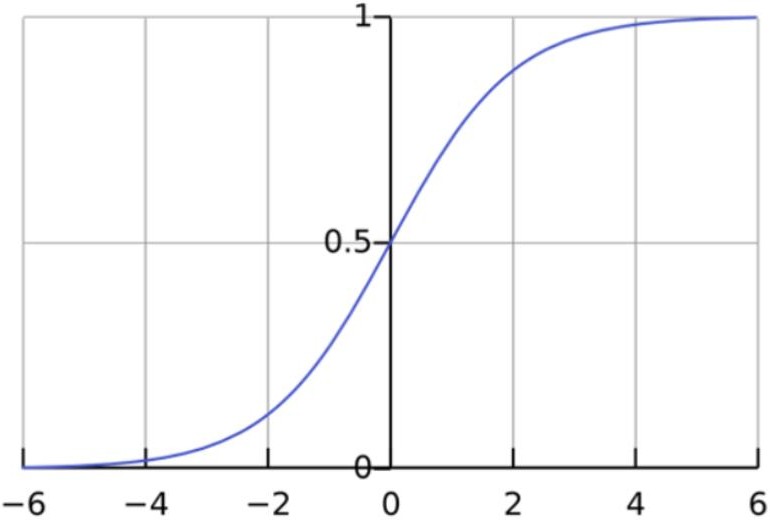
# PHƯƠNG PHÁP

## Hồi quy Logistic (Logistic Regression)

Mô hình hồi qui Logistic là sự tiếp nối ý tưởng của hồi qui tuyến tính vào các bài toán phân loại. Từ đầu ra của hàm tuyến tính chúng ta đưa vào hàm Sigmoid để tìm ra phân phối xác suất của dữ liệu. Lưu ý rằng hàm Sigmoid chỉ được sử dụng trong bài toán phân loại nhị phân. Đối với bài toán phân loại nhiều hơn hai nhãn, hàm Softmax (là một dạng hàm tổng quát của Sigmoid) sẽ được sử dụng. Đặc điểm của hàm Sigmoid là liên tục và khả vi trên R, miền giá trị nằm trong khoảng từ 0 đến 1, đây cũng là cách mà mô hình hồi quy logistic dự đoán đầu ra. Phương trình của hàm Sigmoid và hình ảnh của nó được biểu diễn như dưới đây:

1

𝜎(𝑥) = 1 + 𝑒−𝑥



Hàm hồi quy từ mô hình hồi quy Logistic có dạng như sau:

𝑃(𝑦 = 1|𝑿; 𝒘) = 𝜎(𝒘𝑇𝒙) = 1

1 + 𝑒−𝑾𝑇𝒙

, trong đó 𝒙 = [1, 𝑥1, 𝑥2, 𝑥3, … , 𝑥𝑛 ]𝑇 là vector đầu vào, 𝒘 = [𝑤0, 𝑤1, 𝑤2, … , 𝑤𝑛 ]𝑇 là vector trọng số và 𝑤0 là độ lệch và 𝑃(𝑦 = 1|𝑿; 𝒘) là xác suất để đầu là rơi vào giá trị 1. Tùy thuộc vào một ngưỡng nào đó được định trước, ta có thể quyết định nhã của đầu ra dự đoán là 0 hoặc 1, ví dụ với ngưỡng là 0.5 như sau thì đầu ra có thể là:

0, 𝑃(𝑦 = 1|𝑿; 𝒘) ≤ 0.5

{

1, 𝑃

(𝑦 = 1|𝑿; 𝒘)

> 0.5

Trong thư viện sklearn, hồi quy Logistic được cài đặt bằng Stochastic Gradient Descent (SGD), trong đó hàm mất mát được sử dụng là Cross Entropy, với giá trị mất mát tại điểm dữ liệu thứ i có dạng như sau:

𝐿(𝑦𝑖, 𝑦̂𝑖) = −𝑦𝑖 log 𝑦̂𝑖 − (1 − 𝑦𝑖) log(1 − 𝑦̂𝑖)

và giá trị mất mát trên toàn bộ tập dữ liệu là:

𝑛

𝑳(𝒚, 𝒚̂) = ∑ −𝑦𝑖 log 𝑦̂𝑖 − (1 − 𝑦𝑖) log(1 − 𝑦̂𝑖)

𝑖=1

Tùy thuộc vào cài đặt, ta cũng có thể thêm trọng số vào hàm mất mát, ví dụ như sau:

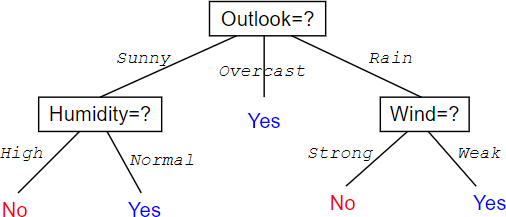
𝑛

𝑳(𝒚, 𝒚̂) = ∑ −𝑎𝑦𝑖 log 𝑦̂𝑖 − 𝑏(1 − 𝑦𝑖) log(1 − 𝑦̂𝑖)

𝑖=1

## Cây quyết định (Decision Tree)

Cây quyết định là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Mô hình cây quyết định mỗi nút biểu diễn một đặc trưng (feature), mỗi nhánh (branch) biểu diễn một quy luật( rule) và mỗi lá biểu biễn một kết quả. Một cây quyết định có thể được biểu diễn bằng một tập các luật IF-THEN. Khi có dữ liệu mới cần phần loại, bằng cách duyệt cây từ nút gốc đến một nút lá theo các điều kiện trên nhánh đã được học, cây sẽ cho ra nhãn lớp gắn với nút lá đó và gán nhãn đó cho dữ liệu mới. Dưới đây là hình ảnh ví dụ về 1 cây quyết định:



Cây quyết định có thể được xây dựng bằng thuật toán CART (Classification and Regression Tree). Cây phân loại và hồi qui (CART) là một kĩ thuật học máy có giám sát phổ

biến được áp dụng để dự đoán biến mục tiêu định tính (categorical target variable), tạo cây phân loại hoặc biến mục tiêu liên tục (continuous target variable).

Ý tưởng của thuật toán:

* Thuật toán sẽ xây dựng cây bắt đầu từ nút gốc
* Tại mỗi nút, chọn một thuộc tính với giá trị phân loại của nó (đánh giá bằng Gini) và chia các ví dụ học thành các nửa (tùy thuộc vào lựa chọn, nếu chia thành 2 nửa và giá trị phân loại là 10 thì có thể chia 1 nửa ≤ 10 và nửa còn lại >10), cụ thể như sau:
  + Với mỗi thuộc tính, tìm giá trị phân loại trong thuộc tính đó. Đối với thuộc tính số

(cũng là cách cài đặt trong thư viện sklearn), ta lấy các giá trị kiểm tra, là các giá trị trung bình giữa 2 giá trị liên tiếp trong thuộc tính đó. Nếu giá trị kiểm tra nào cho Gini Impurity thấp hơn sẽ được sử dụng để làm giá trị phân loại, sau đây là cách tính Gini Impurity. Trước hết, với mỗi giá trị kiểm tra, ta chia được dữ liệu thành các nửa, mỗi nửa ta sẽ tính giá trị Gini trên nửa đó như sau:

, trong đó 𝑝 = 𝑛𝑖, 𝑛

𝑛

𝐺𝑖𝑛𝑖 = 1 − ∑ 𝑝𝑖2

𝑖=1

là số lượng các nhãn i của nửa này và n là tổng số lượng các ví

𝑖 𝑛 𝑖

dụ trong nửa này. Khi đó Gini Impurity của giá trị kiểm tra này sẽ là:

𝑘

1

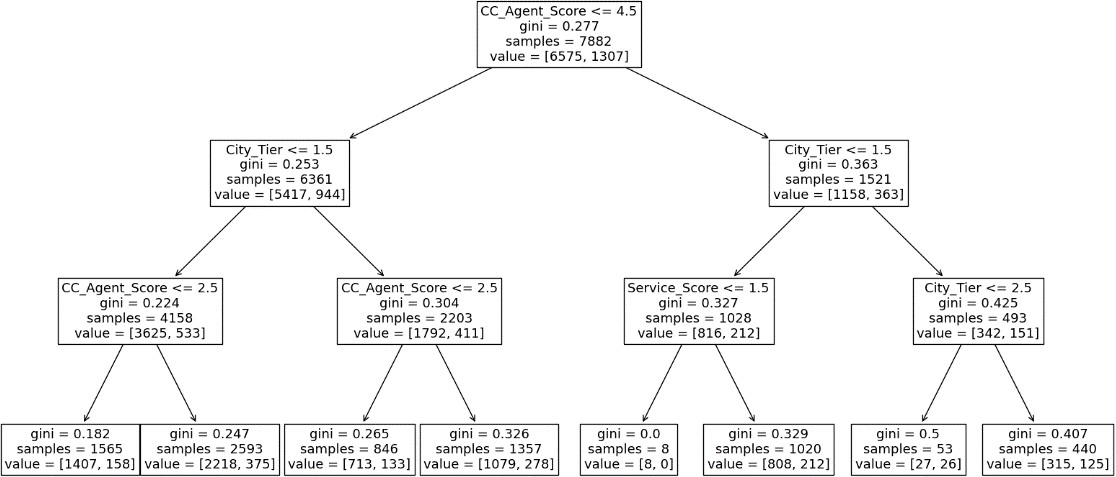
𝐺𝑖𝑛𝑖 𝐼𝑚𝑝𝑢𝑟𝑖𝑡𝑦 = 𝑁 ∑ 𝑁𝑘𝐺𝑖𝑛𝑖(𝑁𝑘)

𝑖=1

, trong đó N là tổng số lượng các ví dụ cho đến hiện tại, 𝑁𝑘 là số lượng ví dụ trong nửa thứ k, 𝐺𝑖𝑛𝑖(𝑁𝑘) là giá trị Gini trong nửa thứ k.

* + Chọn giá trị phân loại có Gini Impurity thấp nhất, sau đó tiến hành tạo các nút con theo giá trị phân loại này. Lặp lại quá trình tìm giá trị phân loại đối với mỗi nút con này cho đến khi vẽ xong toàn bộ cây (hoặc gặp điều kiện dừng).
  + Chú ý: đối với CART, mỗi thuộc tính có thể lặp lại nhiều lần (nhưng giá trị phân loại mỗi lần là khác nhau). Trong cài đặt của sklearn, do phải tránh trường hợp có quá nhiều thuộc tính, CART sẽ lựa chọn thuộc tính trước sau đó mới tìm giá trị phân loại của thuộc tính này (có thể lựa chọn ngẫu nhiên).

Dưới đây ví dụ về 1 cây quyết định CART:



.

## Rừng ngẫu nhiên (Random Forest)

Rừng ngẫu nhiên là một tập hợp các cây quyết định. Thuật toán rừng ngẫu nhiên học 1 tập các cây quyết định rồi sử dụng để dự đoán cho ví dụ và lấy kết quả là kết quả dự đoán đa số. Rừng ngẫu nhiên cũng đã được cài đặt trong thư viện sklearn, với các bước như sau:

* + - Với mỗi cây trong số lượng cây chọn trước:
      * Với tập huấn luyện có n mẫu (dòng hoặc ví dụ), lấy ngẫu nhiên n mẫu với kỹ thuật Bootstrapping, tức là sau khi lấy 1 mẫu thì vẫn giữ mẫu đấy ở trong tập huấn luyện, sau đó tiếp tục lấy như thế đến khi đủ n mẫu (hay lấy mẫu có trùng lặp).
      * Với mỗi nút, sử dụng ngẫu nhiên k thuộc tính (k nhỏ hơn số thuộc tính ban đầu, k được chọn trước và có thể điều chỉnh, mặc định của sklearn là k bằng căn bậc hai của số thuộc tính ban đầu) và huấn luyện tập n mẫu và k thuộc tính vừa chọn này.
    - Tổng hợp lại kết quả (lấy trung bình với rừng hồi quy và đa số với rừng phân loại) và đưa ra dự đoán.

Thuật toán rừng ngẫu nhiên thường có kết quả tốt hơn thuật toán cây quyết định do kết quả dự đoán cuối cùng được tham khảo từ nhiều mô hình cây.

## Gaussian Naïve Bayes

Naïve Bayes là một nhóm các thuật toán phân loại học máy có giám sát dựa trên định lý Bayes. Nó là một kỹ thuật phân loại đơn giản, nhưng có chức năng cao. Chúng được sử

dụng khi số chiều của đầu vào cao. Các vấn đề phân loại phức tạp cũng có thể được thực hiện bằng cách sử dụng trình phân loại Naive Bayes.

Xét bài toán phân loại với 𝐶 lớp 1,2, … , 𝐶. Giả sử có một điểm dữ liệu 𝑥 ∈ 𝑅𝑑. Ta cần tính xác suất để đầu ra là lớp 𝐶 biết rằng đầu vào là vector 𝑥, tức tính 𝑃(𝑐|𝑥). Biểu thức này, nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi lớp. Từ đó có thể giúp xác định lớp của điểm dữ liệu đó bằng cách chọn ra lớp có xác suất cao nhất:

𝐶= argmax 𝑝(𝑐|𝑥) = argmax 𝑝(𝑥|𝑐).𝑝(𝑐)

= argmax 𝑝(𝑥|𝑐). 𝑝(𝑐)

𝑐∈{1,...,𝐶}

𝑐 𝑝(𝑥) 𝑐

(vì mẫu số 𝑝(𝑥) không phụ thuộc vào 𝑐)

Để giúp cho việc tính toán được đơn giản, người ta thường giả sử một cách đơn giản nhất rằng các thành phần của biến ngẫu nhiên 𝑥 là [độc lập với nhau](https://machinelearningcoban.com/2017/07/09/prob/#-independence), nếu biết 𝑐 .Tức là:

𝑑

𝑝(𝑥|𝑐) = 𝑝( 𝑥1, 𝑥2, … , 𝑥𝑑 |𝑐) = ∏ 𝑝(𝑥𝑖|𝑐)

𝑖=1

Suy ra:

𝑑

𝐶 = 𝑎𝑟𝑔𝑚𝑎𝑥 𝑝(𝑐). ∏ 𝑝(𝑥𝑖|𝑐)

𝑐∈{1,...,𝐶}

𝑖=1

Việc tính toán 𝑝(𝑥𝑖|𝑐) phụ thuộc vào loại dữ liệu. Ở đây nhóm đã sử dụng [Gaussian Naïve](https://iq.opengenus.org/gaussian-naive-bayes/) [Bayes](https://iq.opengenus.org/gaussian-naive-bayes/) là một biến thể của [Naïve Bayes](https://iq.opengenus.org/text-classification-naive-bayes/) tuân theo phân phối chuẩn Gaussian. Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục.

Với mỗi chiều dữ liệu 𝑖 và một lớp 𝑐, 𝑥𝑖 tuân theo một phân phối chuẩn có kỳ vọng µ𝑐𝑖 và phương sai 𝜎2 .

𝑐𝑖

𝑝(𝑥 |𝑐) = 𝑝(𝑥 |µ , 𝜎2 ) = 1 𝑒𝑥𝑝(−

𝑖 𝑖 𝑐𝑖 𝑐𝑖

√2𝜋𝜎2

𝑐𝑖

(𝑥𝑖 − µ𝑐𝑖)2

2 )

2𝜎

𝑐𝑖

Trong đó bộ tham số 𝜃 = {µ𝑐𝑖, 𝜎2 } được xác định bằng Maximum Likelihood:

𝑐𝑖

𝑁

(µ𝑐𝑖, 𝜎2 ) = 𝑎𝑟𝑔𝑚𝑎𝑥 ∏ 𝑝(𝑥𝑖(𝑛)|µ𝑐𝑖, 𝜎2 )

𝑐𝑖

𝑐𝑖

µ𝑐𝑖,𝜎2

𝑛=1

𝑐𝑖

*Chú ý: Đây là cách tính của thư viện sklearn. Chúng ta cũng có thể đánh giá các tham số bằng MAP nếu biết trước priors của* µci, σ2 *.*

ci

## K láng giềng gần nhất (K Nearst Neighbour)

Thuật toán K láng giềng gần nhất (KNN) là một kĩ thuật học có giám sát (supervised learning) dùng để phân loại quan sát mới bằng cách tìm điểm tương đồng giữa quan sát mới này với dữ liệu sẵn có. KNN không thực hiện tính toán gì trong quá trình huấn luyện mà chỉ đơn giản là lưu lại các dữ liệu huấn luyện. Khi có dữ liệu cần dự đoán, KNN thực hiện tính khoảng cách từ điểm dữ liệu mới đến tất cả các điểm có trong dữ liệu sẵn có, lấy ra k điểm mà có khoảng cách ngắn nhất, sau đấy gán nhãn điểm dữ liệu mới theo đa số các nhãn xuất hiện.

Phép tính khoảng cách giữa 2 điểm có thể là Euclidian, Manhattan, Minkowski, ….

Dưới đây là ví dụ các công thức khoảng cách:

 Euclidean: 𝐷(𝑥, 𝑦) = √∑𝑛 (𝑥𝑖 − 𝑦𝑖)2

𝑖=1

 Manhattan: 𝐷(𝑥, 𝑦) = ∑𝑛 |𝑥𝑖 − 𝑦𝑖|

𝑖=1

 Minkowski: 𝐷(𝑥, 𝑦) = (∑𝑛

1

(𝑥𝑖 − 𝑦𝑖)𝑝)𝑝

𝑖=1

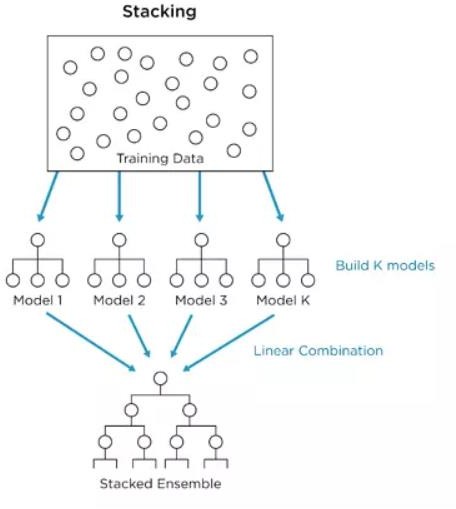
Dưới đây là thuật toán KNN:

1. Ta có D là tập các điểm dữ liệu đã được gắn nhãn và A là dữ liệu chưa được phân loại.
2. Đo khoảng cách từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D.
3. Chọn K (K là tham số được định nghĩa trước) điểm trong D mà có khoảng cách nhỏ nhất.
4. Đếm số lượng các nhãn trong các điểm vừa chọn, chọn ra nhãn có số lượng nhiều nhất. Nhãn của điểm dữ liệu mới chính là nhãn này.

## Stacking Model

Stacking hay tên đầy đủ là Stacked Generalization là một thuật toán học máy thuộc Ensemble Learning, kết hợp các dự đoán từ nhiều mô hình học máy trên cùng một tập dữ liệu.

Dưới đây là hình ảnh mô tả cho một mô hình stacking.



Trong sklearn, mô hình Stacking được cài đặt thông qua lớp StackingClassifier hoặc StackingRegressor. Lớp này có 2 phần cơ bản là base models và final model:

* Base models: Gồm tập các mô hình cơ sở học trực tiếp từ bộ dữ liệu và đưa ra dự đoán.
* Final model (hoặc Meta-model): Gồm 1 mô hình với đầu vào là các dự đoán ở base models, nhãn là nhãn gốc của tập dữ liệu. Tùy vào bài toán, mô hình này có thể là hồi quy Logistic hoặc hồi quy tuyến tính (hoặc bất kỳ mô hình nào khác tùy vào sử dụng).

*Chú ý: Trong sklearn, mô hình cuối (final model hoặc meta-model) sẽ được huấn luyện và đánh giá theo kiểm định chéo, trong khi đó mô hình cơ sở sẽ học toàn bộ dữ liệu.*

Như vậy Meta-model được huấn luyện dựa trên đầu ra dự đoán của các base-modes, các outputs này kết hợp với nhãn của bài toán tạo thành cặp dữ liệu đầu vào - đầu ra trong quá trình huấn luyện Meta-model. Có thể thấy Meta-model không học trực tiếp từ tập dữ liệu huấn luyện, tuy nhiên, việc dùng dữ liệu ban đầu thêm vào outputs của base-models vẫn hoàn toàn hợp lý, sẽ cấp thêm cho meta-model nhiều thông tin hơn về dữ liệu. Các base-models có những cách học khác nhau trên bộ dữ liệu, cho nên outputs hay errors của các base-models là không tương quan (uncorrelated) hay có độ tương quan thấp (low correlation). Đầu ra của base-models có thể là giá trị thực (cho bài toán Hồi quy) hoặc là các xác suất của nhãn trong bài toán phân loại.

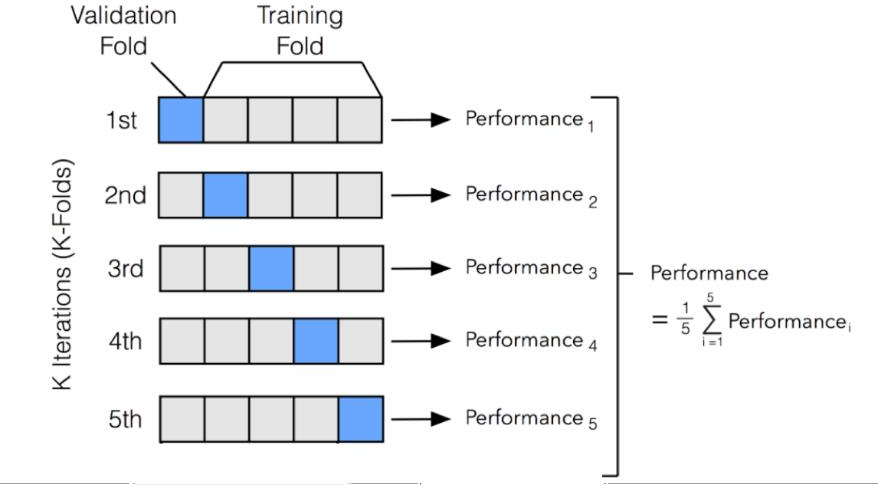
Base-models thường phức tạp và đa dạng, mỗi mô hình có cách học và giải quyết vấn đề khác nhau với cùng một bài toán như: Decision Tree, SVM, Neural Network,... và kể cả là các thuật toán ensemble khác như GBM, Random Forest,... Trái ngược với đó, Meta-model thường đơn giản hơn, dự đoán kết quả từ các kết quả dự đoán của base-models , và thường là:

* + Linear Regression cho bài toán hồi quy
  + Logistic Regression cho bài toán phân loại

# PHƯƠNG PHÁP ĐÁNH GIÁ

## Kiểm định chéo bằng StratifiedKFold

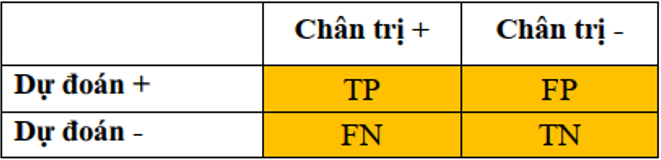
Kiểm định chéo bằng K-fold là một kỹ thuật dùng để đánh giá hiệu quả của mô hình. Với K-fold, tập dữ liệu huấn luyện sẽ được chia ra thành k phần phần tương đương nhau gọi là các folds. Sau đó, mô hình sẽ huấn luyện bằng cách sử dụng k – 1 folds và lấy fold còn lại để đánh giá. Điều này được lặp lại k lần, mỗi lần sử dụng một fold khác nhau làm tập thử nghiệm. Hiệu suất của mô hình sau đó được tính trung bình qua k lần lặp để cung cấp phép đo tổng thể. Dưới đây là hình minh họa của K-fold:



Đối với StratifiedKFold, mọi thứ vẫn giống với KFold trừ việc là khi chia thành các folds, mỗi folds đều sẽ giữ tỉ lệ nhãn (Yes/No, 1/0, …) tương đương với tập huấn luyện gốc.

## Các độ đo chính xác (Accuracy, Precision, Recall, F1-score)

Dưới đây hình ảnh confusion matrix đối với phân lớp nhị phân:



, trong đó:

* + - TP (True Positive): dương tính đúng (số nhãn + được dự đoán là +)
    - TN (True Negative): âm tính đúng (số nhãn - được dự đoán là -)
    - FP (False Positive): dương tính giả (số nhãn + được dự đoán là -)
    - FN (False Negative): âm tính giả (số nhãn – được dự đoán là +)

Các thang đo đánh giá tính hiệu quả của bài toán phân lớp được tính toán dựa trên confusion matrix được tính toán như sau:

* Độ chính xác (accuracy): xác định mức độ chính xác của kết quả phân lớp cho cả phân lớp âm và dương tính. Độ chính xác được tính theo công thức:

𝐴𝑐𝑐𝑢𝑟𝑎𝑛𝑐𝑦 =

𝑇𝑃 + 𝑇𝑁

𝑇𝑃 + 𝑇𝑁 + 𝐹𝑃 + 𝐹𝑁

* Độ nhạy (Recall): xác định mức độ chính xác của kết quả phân lớp cho cả phân lớp âm và dương tính. Độ chính xác được tính theo công thức:

𝑅𝑒𝑐𝑎𝑙𝑙 =

𝑇𝑃

𝑇𝑃 + 𝐹𝑁

* Độ chính xác phép đo (precision): xác định tỷ số dương tính được xác định đúng trên tổng số dương tính. Độ chính xác phép đo tính theo công thức

𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 =

𝑇𝑃

𝑇𝑃 + 𝐹𝑃

* Điểm F1 (F1 score): là trung bình điều hòa của giá trị độ nhạy và độ chính xác phép đo. Điểm F1 được tính theo công thức:

𝐹1 =

2𝑇𝑃

2𝑇𝑃 + 𝐹𝑃 + 𝐹𝑁

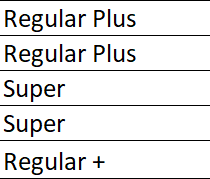
Đối với bài toán có sự mất cân bằng nhãn (nhãn 0 nhiều hơn nhãn 1 nhiều lần), độ đo Accuracy sẽ không mang nhiều ý nghĩa. Ví dụ, khi dự đoán cho bài toán khách hàng rời bỏ, có 99 khách hàng ở lại và 1 khách hàng rời bỏ, như vậy nếu mô hình dự đoán tất cả khách hàng đều ở lại thì sẽ đạt được Accuracy là 99%, một con số cực kỳ cao, tuy nhiên mục đích của bài toán đặt ra là tìm ra những người rời bỏ nhưng mô hình là chỉ dự đoán không rời bỏ, như vậy là không hề có ý nghĩa gì với mục đích đặt ra từ ban đầu. Chính vì vậy, nhóm lựa chọn tập trung cải thiện độ đo Recall và F1-score để lựa chọn mô hình.

# THỰC HIỆN

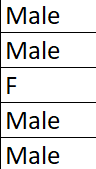
## Chia tập dữ liệu

Với tập dữ liệu ban đầu gồm 11260 điểm dữ liệu, nhóm chia tập dữ liệu ra làm 2 phần ghi lại trong 2 file là train.csv và test.csv theo tỷ lệ 8:2 và chia vẫn giữ tỷ lệ nhãn trong cả tập train và test.

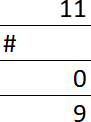
Mục đích của việc chia ngay từ ban đầu là để tránh data leakage, hay thông tin từ tập test bị rò rỉ trong quá trình huấn luyện, đồng thời cũng là để mô phỏng 1 phần dữ liệu tương lai, vì thế trong quá trình phân tích và huấn luyện, nhóm sẽ hoàn toàn không sử dụng gì đến tập kiểm tra. Tuy nhiên, do tập dữ liệu có sự không đồng nhất giữa các điểm dữ liệu ở các cột và chứa các ký tự vô nghĩa, vì thế nhóm sẽ sửa trước điều này trước khi chia (mặc dù có chút vi phạm so với ý nghĩa việc chia ra ban đầu). Dưới đây là hình ảnh trong dữ liệu không đồng nhất này:



Hình ảnh từ cột account\_segment



Hình ảnh tử cột Gender



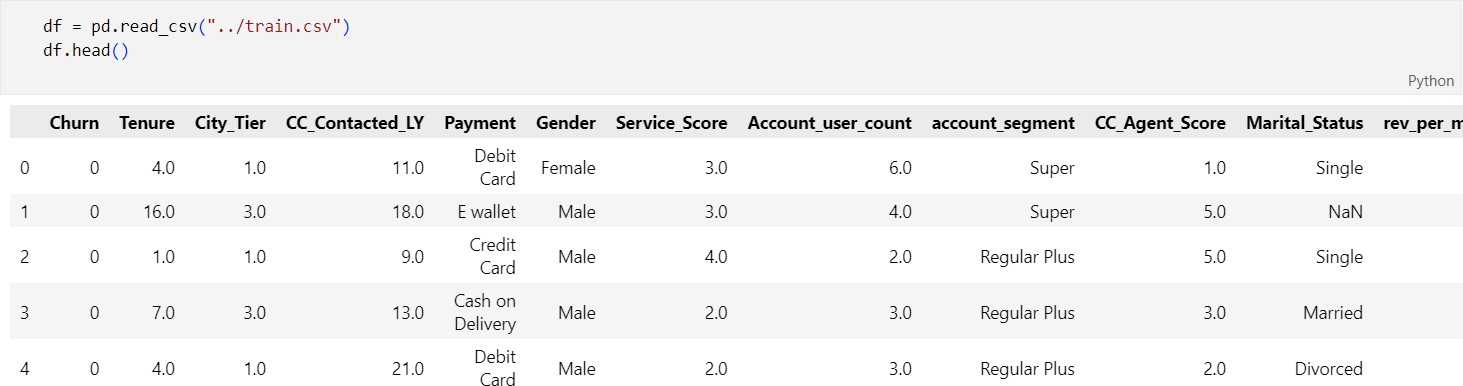
Ký hiệu lạ trong cột Tenure

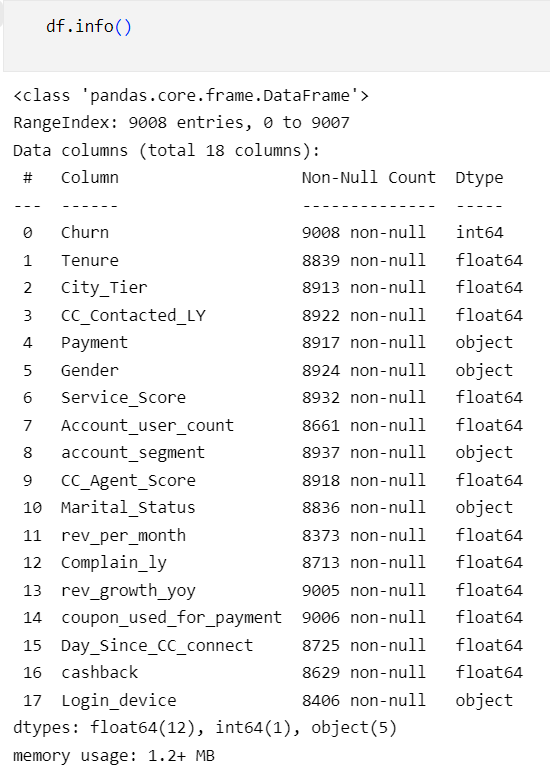
Cụ thể, các bước này có trong file **split\_data.py** đính kèm

## Khai phá dữ liệu

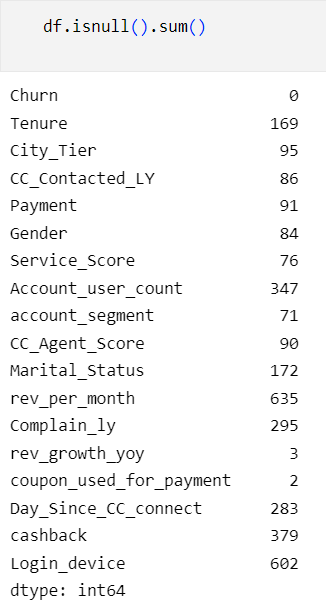
Bắt đầu từ đây, nhóm chỉ sử dụng tập dữ liệu trong train.csv

* + - Hình ảnh tổng quan về dữ liệu:



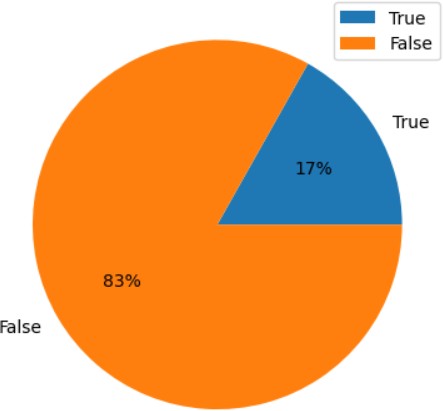


* Như vậy, tập dữ liệu huấn luyện có 9008 dòng, chứa cả các cột định tính và định lượng và tồn tại giá trị rỗng trong tập.
  + - Dưới đây là số lượng giá trị rỗng (null) có trong tập dữ liệu:



Đối với các giá trị rỗng này, nhóm sẽ điền vào là giá trị xuất hiện nhiều nhất trong cột (most\_frequent), các giá trị điền ở tập huấn luyện cũng sẽ được giữ lại để điền vào tập test.

* + - Hình ảnh phân bố nhãn của tập huấn luyện:

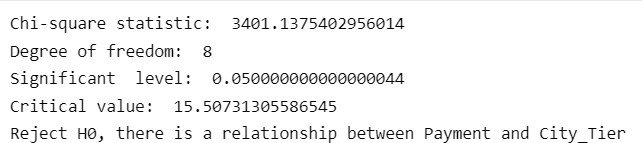


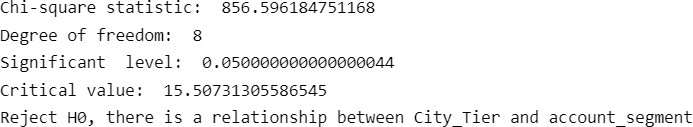
Sự phân bố cho thấy tập dữ liệu có sự mất cân bằng nhãn khi mà đa số (83%) dòng đều có nhãn là 0 (False hay chưa rời bỏ). Đây cũng là đặc trưng của những bài toán dạng này. Vì vậy, nhóm chủ yếu sử dụng các mô hình dạng cây (tree-based models) cho tập dữ liệu này. (Bên cạch đó, nhóm đã thử sử dụng undersampling và oversampling cho bài toán nhưng kết quả không có tiến triển gì)

* + - Với kiểu dữ liệu được hiển thị như trên thì có thể chia tập dữ liệu thành định tính đối với kiểu là object và còn lại là định lượng, tuy nhiên sau khi tìm hiểu và thử nghiệm thì nhóm quyết định để cột City\_Tier vào cột định tính do việc phân loại cấp thành phố trên các cấp thành phố cho biết khách hàng đang nằm trong vùng nào của quốc gia, tức là thành phố cấp 1 không có nghĩa là bé hơn thành phố cấp 2 hay có sự so sánh giữa cấp của thành phố. Việc này đã làm cải thiện hiệu năng của các mô hình học máy. Như vậy cột định tính bao gồm các cột như hình dưới đây (và các cột còn lại sẽ là định lượng):



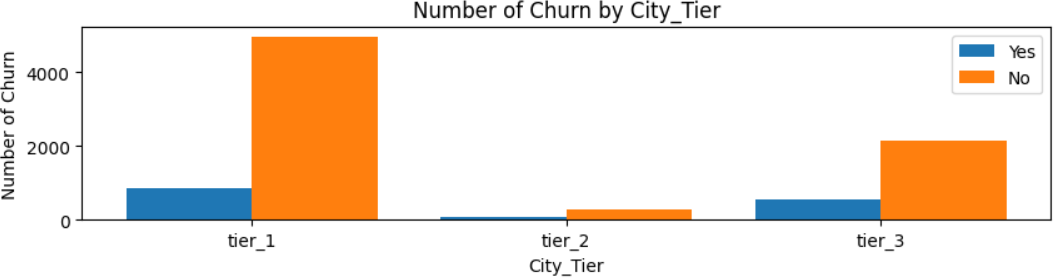
Tiếp theo, đối với các cột định tính, nhóm thực hiện kiểm định khi bình phương (Chi- squared test) để tìm ra sự tương qua giữa các cột, và các cột so với nhãn. Đối với sự tương quan giữa các cột và nhãn, các cột đều có mối liện hệ đối vỡi nhãn dựa trên giá trị kiểm định. Trong khi đó, một số cột có sự tương quan lẫn nhau với giá trị kiểm định rất cao, dưới đây là hình ảnh các cột đó:



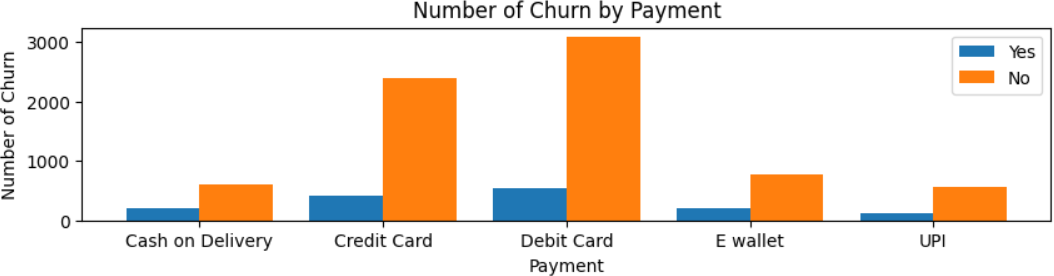


Giá trị kiểm định giữa các cột trên cao hơn rất nhiều so với các cặp cột khác. Đối với điều này, nhóm giả thiết rằng cột City\_Tier (cũng dựa trên việc phân các cấp thành phố) đang cho thấy cấp càng thấp (là gần 1) thì cho thấy mức sống cao hơn nên phương thức thanh toán cũng sử dụng chủ yếu là thẻ (thay vì tiền mặt) và account\_segment cũng chủ yếu là Super. Nhóm thử gộp các cột này lại thành một cột mới, ví dụ như City\_Tier là Tier\_1 và Payment là Debit Card thì sẽ có một giá trị mới là Tier\_1\_x\_Debit Card. Tuy nhiên, nhóm gặp vấn đề ở đoạn này. Mặc dù trên tập validation, các mô hình đều cho kết quả tang lên, tuy nhiên nhóm đã không áp được vào tập test do trong quá trình ghép 2 cột ở tập test đã sinh ra các điểm dữ liệu nằm ngoài từ điển của tập train nên one-hot đã báo lỗi, đồng thời do cũng chưa tìm được cách xử lý mới ngoài việc này nên nhóm đã để lại như cũ.

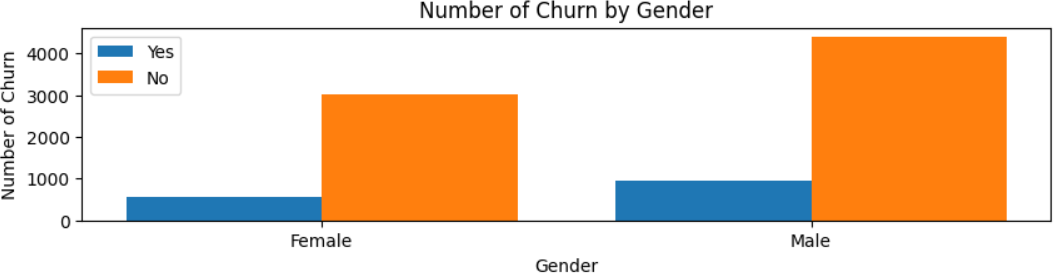
* + - Dưới đây là phân bố nhãn theo các cột định tính:



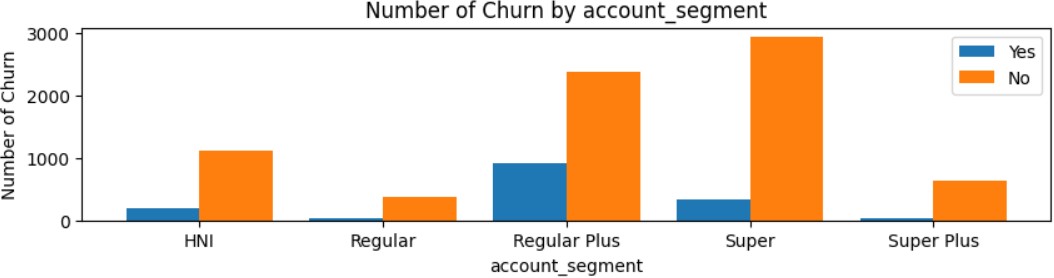
Có thể thấy phân bố của các giá trị cũng lệch khi chủ yếu tập trung ở thành phố cấp 1, cũng là thành phố có mức sống cao (có thể là vì vậy mà cũng có lượng tài khoản nhiều hơn).



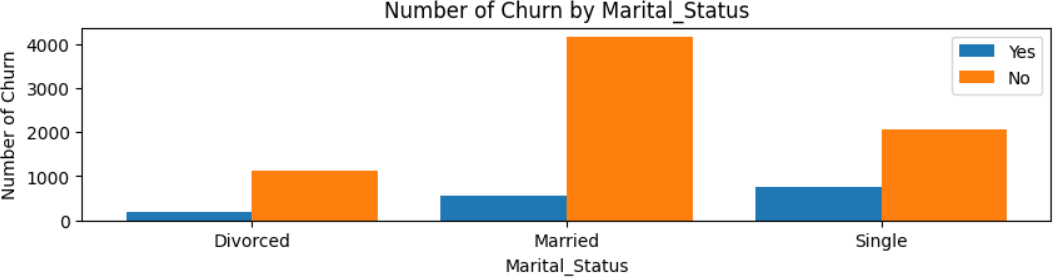
Phương thức thanh toán chủ yếu ở đây là thẻ ghi nợ và thẻ tín dụng (cao ở cả nhãn rời bỏ và không). Việc phân bố lệch vào 2 giá trị này so với cột City\_Tier cũng có thể là nguyên do khiến giá trị kiểm định cao như vậy.



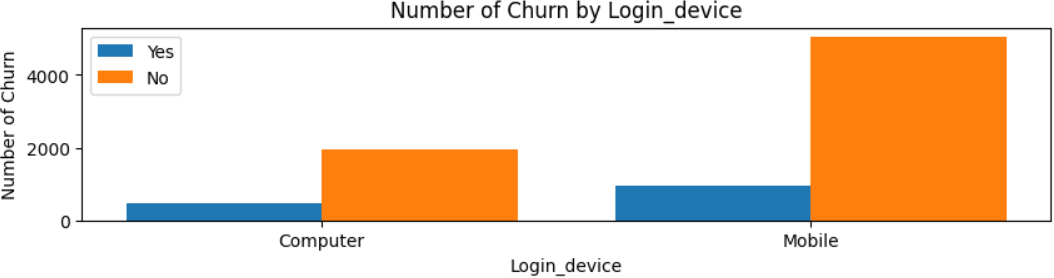
Tương tự như vậy, cột Genderr cũng chưa có khác biệt gì.



Đến đây, đã có sự khác biệt hơn khi các tài khoản thuộc phân khúc Regular Plus đang có tỷ lệ rời bỏ cao hơn hẳn các phân khúc khác.



Ở đây có vẻ như những người độc thân lại có xu hướng rời bỏ cao hơn



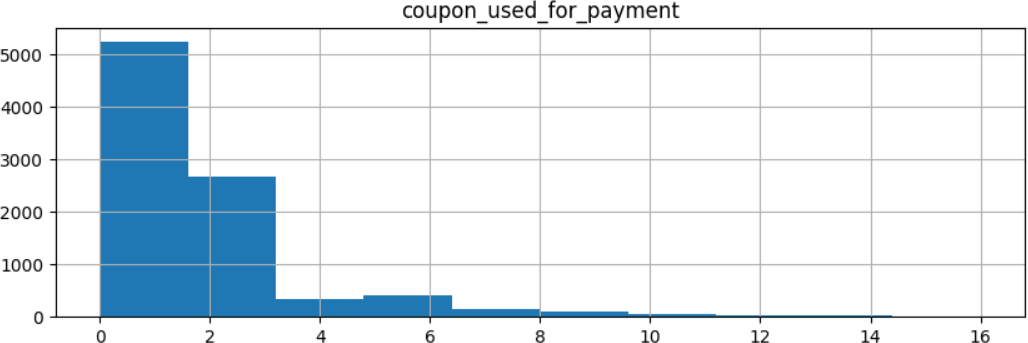
Cuối cùng là thiết bị đăng nhập, không có gì khác biệt (nhưng có vẻ người đăng nhập bằng máy tính đang rời bỏ cao, có thể vì tính bất tiện của thiết bị)

* + - Tới lượt các cột dạng định lượng, trước hết là ma trận tương quan để thể hiện mối quan hệ giữa các cột:



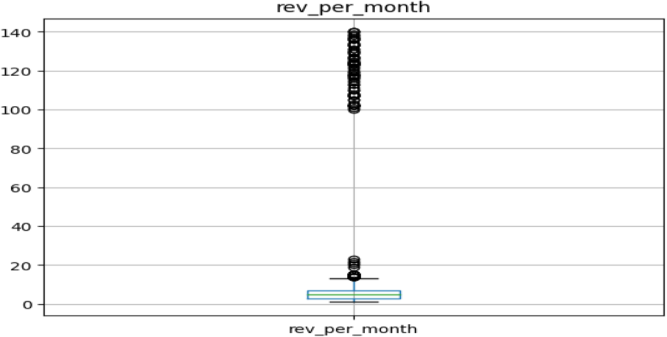
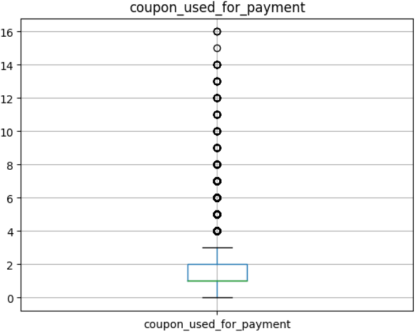
Nhìn chung, có hai cặp cột có hệ số tương quan cao nhất là cặp Service\_Score và Acount\_user\_count và cặp Day\_Since\_CC\_connect và coupon\_used\_for\_payment. Tuy nhiên do hệ số này không thực sự quá cao (chỉ với 0.33 và 0.36) nên nhóm không làm gì đối với 2 cặp cột này.

* + - Đối với phân bố từng cột một (hình ảnh tất cả các cột nằm trong file) , các cột chủ yếu có dạng phân bố log normal, phân bố chuẩn hoặc đều. Đối với điều này, nhóm đã giữ nguyên không có tác động gì đến các cột.



Ví dụ hình ảnh cột coupon\_used\_for\_payment

Cuối cùng là kiểm tra ngoại lai trong các cột định lượng, nhóm kiểm tra trực quan bằng Boxplot và kiểm định Z, cho thấy được sự tồn tại của ngoại lai trong các cột này. Các giá trị này nhóm sử dụng phươ ng pháp IQR để tìm kiếm, sau đó điền lại bằng cận trên (upper whisker) đối với các giá trị vượt ngưỡng này và cận dưới (lower whisker) đối với các giá trị thấp hơn ngưỡng này. Điều này cũng đã làm tăng hiệu quả mô hình. Dưới đây là hình ảnh boxplot ở một số cột và hình ảnh xác định ngoại lai qua kiểm định Z:





## Tiền xử lý dữ liệu

* + - Các bước ở đây đều đã một phần liệt kê ở trên, nhóm thực hiện bao gồm
      * Điền lại các giá trị không đồng nhất
      * Điền các giá trị rỗng
      * Xử lý Outlier

## Tạo đoạn mã cơ sở (cùng với pipeline dữ liệu)

Đầu tiên, nhóm sẽ tạo pipeline dữ liệu bằng cách sử dụng lớp Pipeline của sklearn (thư viện học máy bằng ngôn ngữ Python). Các phần trong pipeline này bao gồm imputer (điền vào các giá trị rỗng), onehot (biến đổi các giá trị trong cột định tính thành dạng số để có thể sử dụng mô hình), outlier\_handling (xử lý ngoại lai), scale (chuẩn hóa khoảng giá trị giữa các cột định lượng), cụ thể có trong file Datapipeline.py. Việc tạo pipeline này là để tránh việc khi tiến hành kiểm thử dẫn đến việc data leakage, đảm bảo không có nhầm lẫn cho việc này xảy ra.

Tiếp đó, để lựa chọn mô hình cũng như tham số, nhóm sử dụng kiểm định chéo bằng StratifiedKFold với 10 folds và có xào lại dữ liệu mỗi lần chia.

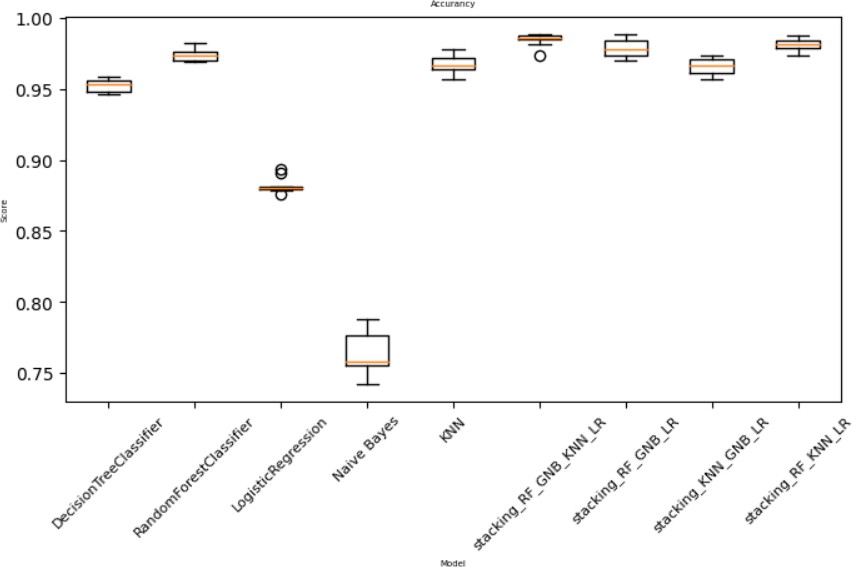
Việc tạo đoạn mã này cũng để nhóm có chung một cách viết mã nên sẽ đồng nhất hơn khi gộp lại.

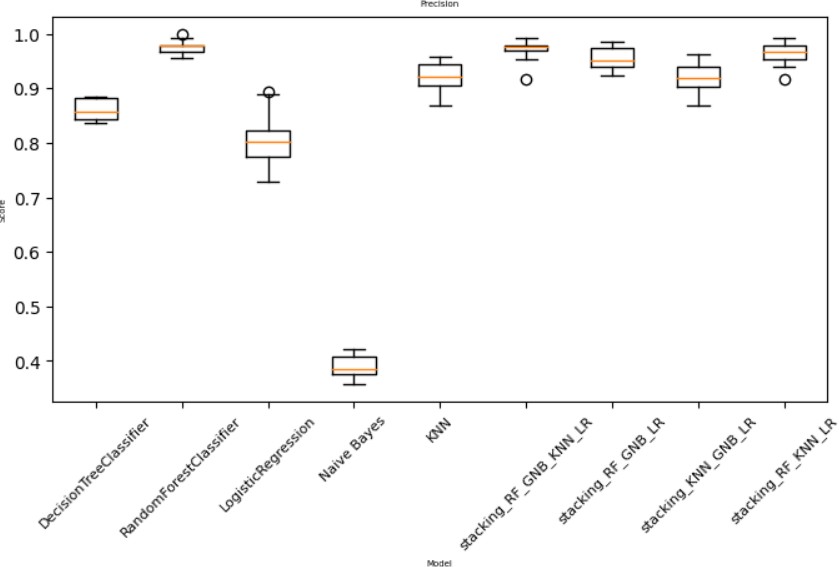
## Huấn luyện

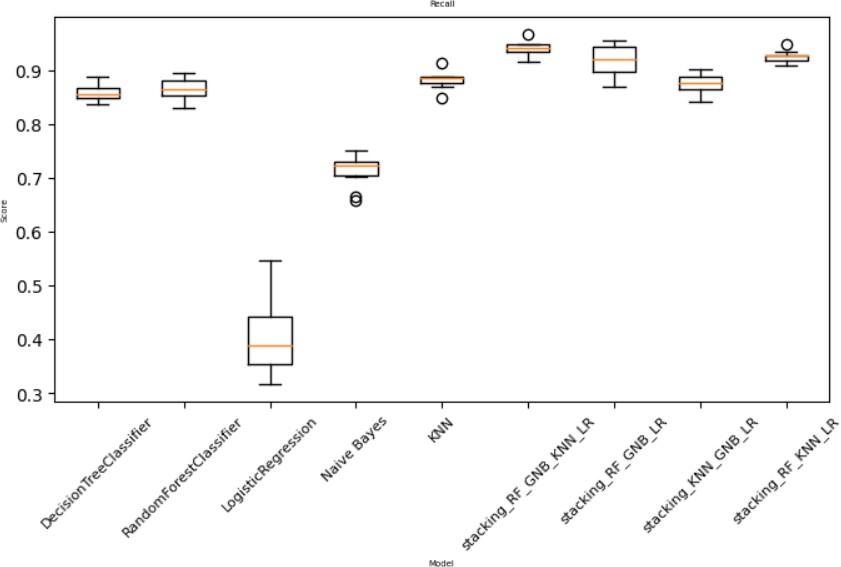
Các thành viên nhóm thử nghiệm nhiều lần (tự lựa chọn tham số) để tìm ra các mô hình phù hợp sau đó được tổng hợp lại, các mô hình được sử dụng như đã đề cập ở phần phương pháp, bao gồm: Hồi quy Logistic, Cây quyết định, Rừng ngẫu nhiên, Naive Bayes, K láng giềng gần nhất và Stacking Model. Trong đó, đối với Stacking Model, nhóm thực hiện ghép lần lượt các cặp mô hình cơ sở và mô hình phân loại như sau (RF – Rừng ngẫu nhiên, GNB – Gaussian Naïve Bayes, KNN – K láng giềng gần nhất, LR – Hồi quy Logistic):

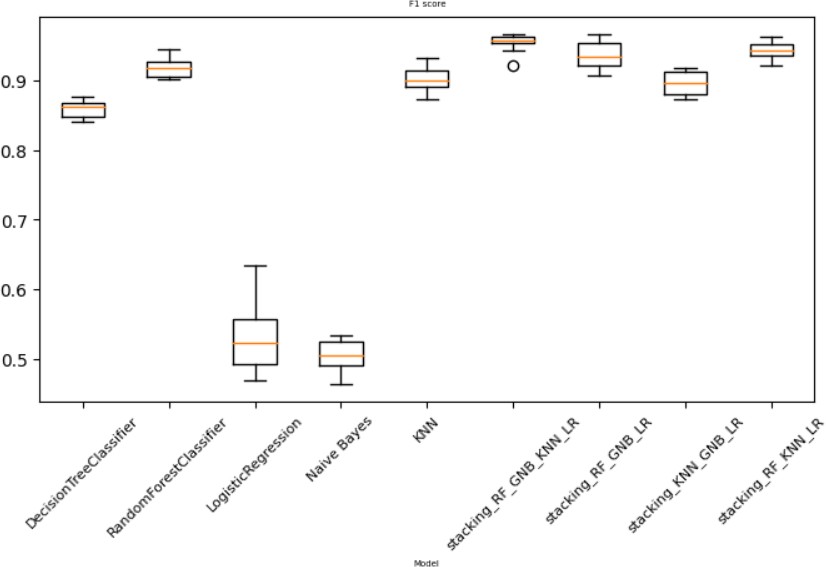
|  |  |
| --- | --- |
| **Mô hình cơ sở** | **Mô hình phân loại** |
| RF, GNB, KNN | LR |
| RF, GNB | LR |
| KNN, GNB | LR |
| RF, KNN | LR |

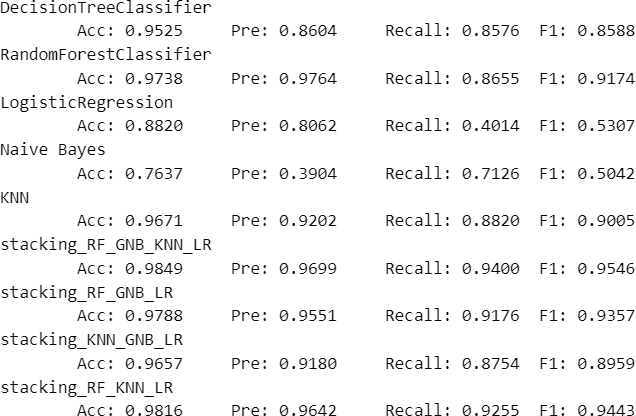
Dưới đây là kết quả và kết quả trung bình sau 10 folds trên tập validation của các mô hình này:







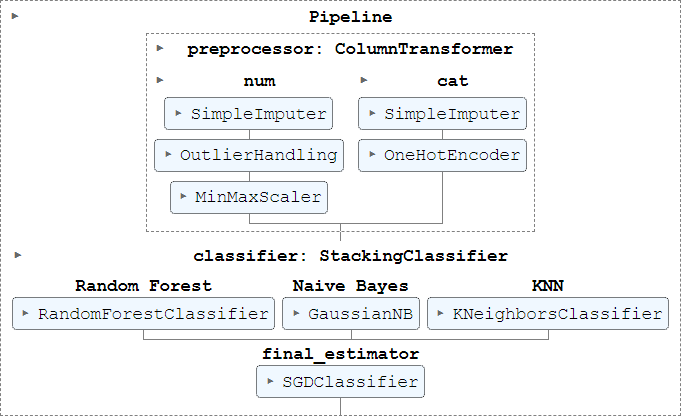


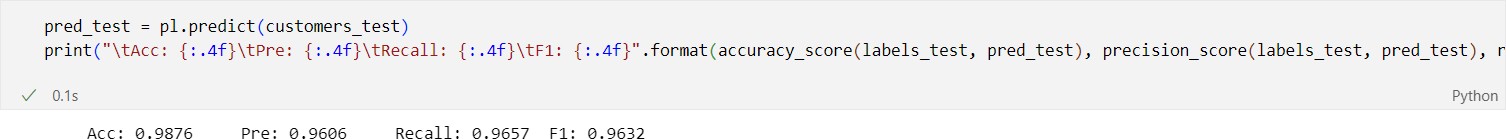


Mặc dù với tiêu chí lựa chọn Recall và F1 cao, nhưng có thể thấy, mô hình stacking với các mô hình cơ sở là RF, GNB, KNN và mô hình phân loại là LR cho kết quả vượt trội trên tất cả các độ đo với các mô hình đơn lẻ và mô hình stacking khác, vì vậy, nhóm quyết định sử dụng nó lên trên tập test.

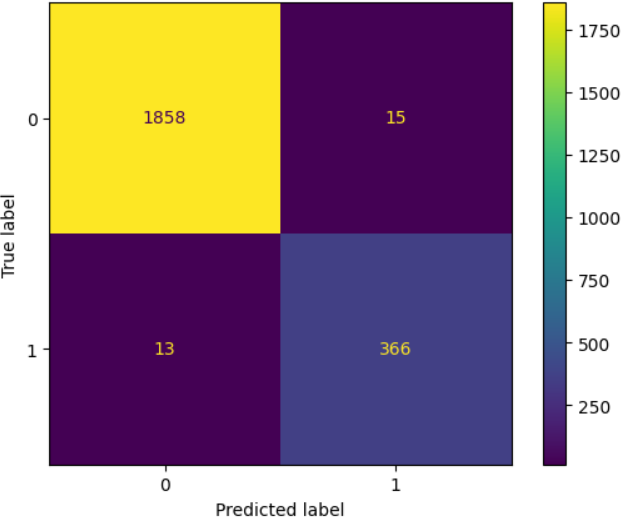
## Đánh giá

Áp dụng pipeline đã tạo ở phía trên, nhóm dễ dàng kiểm tra với tập test và cho được kết quả cao hơn mong đợi, cao hơn cả trên tập train (cũng không biết nên gọi đây là gì, nhưng có vẻ tập test khá tương tự với tập train). Dưới đây là hình ảnh pipeline dữ liệu sau khi thêm mô hình được lựa chọn bên trên và hình ảnh kết quả:





Và confusion matrix của kết quả (nhãn 1 là rời bỏ):



# TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN

* + - Có thể thấy, mô hình cuối cùng cho kết quả cao với cả tập validation và test. Nhìn chung, từ việc khai phá dữ liệu, nhóm khách hàng có khả năng rời bỏ cao tập trung nhiều ở phân khúc Regular Plus (hay có thể coi lượng người có lượng chi tiêu tương đối cao) và nhóm người độc thân.
    - Từ đồ án môn học này, nhóm đã tìm hiểu và làm quen với các bước cơ bản để xây dựng nên một mô hình học máy, các thuật toán trong học máy (nhiều mô hình nhóm không đề cập do không có cải thiện gì nhiều).
    - Về khó khăn của nhóm:
      * Các thành viên nhóm mới làm quen với một mô hình học máy, cũng như khi nhóm chọn lựa ngôn ngữ lập trình chính là ngôn ngữ lập trình nhóm mới tiếp cận - Python, vì thế cần thời gian để tìm hiểu từ đầu.
      * Thiếu kiến thức liên quan đến ngành (kinh tế, dịch vụ, tài chính …) khiến khó có thể đưa ra và phát hiện sâu được những insight lớn từ dữ liệu để cải thiện mô hình. Cũng vì vậy, nhóm tập trung chạy thử nghiệm nhiều lần, nhiều cách xử lý khiến tiêu tốn nhiều thời gian (mặc dù không phải cách nào cũng cải thiện được hiệu quả, thậm thí giảm đi),
      * Khó khăn trong việc tìm ra các bộ tham số phù hợp cho bài toán cũng rất khó tìm và mất nhiều thời gian.
      * Trong quá trình thực hiện, khi nhóm thực hiện kết hợp đặc trưng, nhóm gặp vấn đề khi tập test có giá trị không tồn tại trong từ điển của tập train, do đó dùng one- hot mã hóa đã dẫn đến lỗi, mặc dù kết quả cao hơn với tâp validation nhưng nhóm chưa tìm ra cách để áp dụng lên tập test hiệu quả nên đã bỏ qua đoạn này.
    - Về hướng phát triển:
      * Nhóm hướng tới việc tìm hiểu thêm kiến thức chuyên ngành vào để cố gắng tối ưu hóa trong việc chọn lựa và kết hợp đặc trưng.
      * Tìm hiểu và cài đặt các phương pháp mã hóa cho dạng dữ liệu định tính (hiện tại đang sử dụng one-hot và gặp vấn đề).
      * Tiếp tục tìm các mô hình học máy khác như mạng nơ ron, CatBoost, …
      * Tìm thêm cách xử lý dữ liệu mất cân bằng nhãn.