TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN
HỆ THỐNG THƯƠNG MẠI THÔNG MINH

XÂY DỰNG MÔ HÌNH DỰ ĐOÁN TẬP KHÁCH HÀNG NGƯNG SỬ DỤNG DỊCH VỤ CỦA NGÂN HÀNG

Người hướng dẫn: **Th.S DƯƠNG HỮU PHÚC**

Người thực hiện: **LÊ NGUYỄN MINH TUẤN - 51800950**

TỪ HUY VẠN - 51800263

Lớp: **18050301**

Khóa: **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN
HỆ THỐNG THƯƠNG MẠI THÔNG MINH

XÂY DỰNG MÔ HÌNH DỰ ĐOÁN TẬP KHÁCH HÀNG NGƯNG SỬ DỤNG DỊCH VỤ CỦA NGÂN HÀNG

Người hướng dẫn: **Th.S DƯƠNG HỮU PHÚC**

Người thực hiện: **LÊ NGUYỄN MINH TUẤN - 51800950**

TỪ HUY VẠN - 51800263

Lớp: **18050301**

Khóa: **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

LỜI CẢM ƠN

Qua một học kỳ đầy khó khăn và vất vả, chúng em đã học tập được nhiều thứ từ môn 'Hệ thống thương mại thông minh'. Với sự giúp đỡ chỉ bảo và giảng dạy nhiệt tình của quý thầy cô, đặc biệt là quý cô Khoa Công nghệ thông tin Trường Đại học Tôn Đức Thắng.

Sau quãng thời gian tìm hiểu và thực hiện, chúng em đã hoàn thành đồ án cuối kỳ môn 'Hệ thống thương mại thông minh'. Em xin dành lời cảm ơn đặc biệt đến Thầy Dương Hữu Phúc đã giúp đỡ và chỉ bảo những thiếu sót trong quá trình thực hiện của chúng em.

Và đặc biệt, trong học kỳ này, Khoa đã tổ chức cho chúng em được tiếp cận với môn học mà theo hai em là rất hữu ích đối với sinh viên ngành Khoa Học Máy Tính cũng như tất cả các sinh viên thuộc các chuyên ngành Công Nghệ Thông Tin khác. Chúng em xin chân thành cảm ơn thầy Dương Hữu Phúc đã tận tâm hướng dẫn chúng em qua từng buổi học trên lớp thảo luận về lĩnh vực sáng tạo trong nghiên cứu khoa học.

Trong quá trình làm tiểu luận, khó tránh khỏi những sai sót, rất mong Thầy bỏ qua. Đồng thời em rất mong nhận được những ý kiến đóng góp của Thầy để chúng em có thể tiếp thu và cải thiện để nâng cao kiến thức của mình, đồng thời phục vụ tốt hơn cho quá trình nghiên cứu sau này.

Chúng em xin chân thành cảm ơn!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là thành quả của riêng chúng tôi và được sự hướng dẫn của ThS Dương Hữu Phúc. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Lê Nguyễn Minh Tuấn

Từ Huy Vạn

PHẦN NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

TP. Hồ Chí Minh, ngày tháng năm
(ký tên và ghi rõ họ tên)

Phần đánh giá của GV chấm bài

TP. Hồ Chí Minh, ngày tháng năm
(ký tên và ghi rõ họ tên)

LỜI MỞ ĐẦU

Với sự phát triển rất nhanh của xã hội hiện nay, định nghĩa về cách mạng khoa học kỹ thuật 4.0 không phải là điều quá xa lạ. Với lĩnh vực trí tuệ nhân tạo là một trong những trọng tâm chính, nó đang ảnh hưởng đến tất cả các lĩnh vực, nền kinh tế, các ngành kinh tế và ngành công nghiệp. Ngân hàng là một lĩnh vực lâu đời, bắt nguồn từ thế kỷ 13 và tại đây trong thế kỷ 21 trí tuệ nhân tạo đang bắt đầu đặt những ảnh hưởng đầu tiên của mình lên lĩnh vực lâu đời này.

Ngân hàng giờ đây là một trong những lĩnh vực cạnh tranh, sự cạnh tranh gay gắt này buộc các ngân hàng phải cải thiện chất lượng của mình nếu muốn giữ lấy khách hàng và trí tuệ nhân tạo với các mô hình học máy là một trong những công cụ hiệu quả nhất để làm được điều đó.

Trí tuệ nhân tạo hay còn được gọi là Artificial Intelligence được hiểu như một ngành của khoa học máy tính liên quan đến việc tự động hóa các hành vi thông minh. AI là trí tuệ do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người. Trí tuệ nhân tạo khác với với việc lập trình logic trong các ngôn ngữ lập trình là ở việc ứng dụng các hệ thống học máy để mô phỏng trí tuệ con người trong các xử lý mà con người làm tốt hơn máy tính. Cụ thể, trí tuệ nhân tạo giúp máy tính có được những trí tuệ của con người như biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu ngôn ngữ, tiếng nói, biết học và tự thích nghi, v.v...

Với sức mạnh tính toán vượt xa con người cùng với trí thông minh, trí tuệ nhân tạo là một công cụ tuyệt vời để thống kê và thực hiện các phép toán phức tạp từ đó giúp con người có nhiều nguồn thông tin hơn để vạch ra những chiến lược cụ thể trong khối ngành kinh tế. Và lĩnh vực ngân hàng không phải ngoại lệ.

Trong phạm vi đồ án, chúng em sẽ trực quan hóa các yếu tố ảnh hưởng đến quyết định của khách hàng khi ra quyết định liệu có ngưng sử dụng dịch vụ của ngân hàng hay không và từ các yếu tố đó xây dựng các mô hình Machine Learning để dự đoán xem với các yếu tố từ thông tin của khách hàng, liệu khách hàng có ngưng

sử dụng dịch vụ hay vẫn tiếp tục gắn bó với ngân hàng.

Việc làm đồ án là giúp cho chúng em có cơ hội áp dụng, và được tổng hợp các kiến thức của chúng em đã được học trên lớp. Nhưng với vốn kinh nghiệm ít ỏi và kiến thức chưa sâu, sai sót là không thể tránh khỏi. Rất mong có thể nhận được những đánh giá và chỉ bảo nhiệt tình của quý Thầy, Cô.

Chúng em xin cảm ơn thầy Dương Hữu Phúc đã nhiệt tình hướng dẫn và hoàn thiện đồ án đúng hạn !

Mục lục

Danh sách hình vẽ

BẢNG PHÂN CÔNG CÔNG VIỆC

MSSV	HỌ TÊN SINH VIÊN	CÔNG VIỆC
51800950	Lê Nguyễn Minh Tuấn	Report & Coding
51800263	Từ Huy Vạn	Report & Tableau

Bảng 1: Bảng phân công công việc

Chương 1

TỔNG QUAN ĐỀ TÀI

1.1 GIỚI THIỆU ĐỀ TÀI

Với việc chi phí để có được những khách hàng mới ngày càng gia tăng nên vì vậy việc duy trì các khách hàng cũ trở nên vô cùng quan trọng. Vì vậy các ngân hàng cần phải dự đoán trước các khách hàng nào đang có nguy cơ rời bỏ ngân hàng để kịp thời có chiến lược giữ chân khách hàng.

Bộ dữ liệu được nghiên cứu bao gồm 10000 khách hàng đang sử dụng và ngưng sử dụng dịch vụ ngân hàng từ độ tuổi 26 đến 73 bao gồm nam và nữ.

1.2 PHÁT BIỂU BÀI TOÁN

Bài toán đặt ra là làm cách nào có thể dự đoán được khách hàng nào có khả năng ngừng sử dụng dịch vụ ngân hàng trong tương lai hay không. Các yếu tố nào làm ảnh hưởng tới quyết định của khách hàng, điều này thôi thúc các doanh nghiệp tìm ra các phương pháp để dự đoán hành vi của khách hàng dựa trên dữ liệu có sẵn và các phương pháp Machine Learning là một công cụ hiệu quả để giải quyết các vấn đề trên.

1. Các mô hình Machine Learning sẽ tính xác suất khách hàng có khả năng bỏ đi dựa theo dữ liệu truyền vào của khách hàng đó.

2. Phân tích được thuộc tính nào của khách hàng giữ vai trò trọng yếu trong quyết định cuối cùng của khách hàng

1.3 MỤC TIÊU ĐỀ TÀI

Vậy Machine Learning là gì ?

Machine learning hay học máy là một kỹ thuật cho phép máy tính có khả năng cải thiện chính bản thân chúng dựa trên dữ liệu mẫu (training data) hoặc dựa vào kinh nghiệm (những gì đã được học). Bằng cách áp dụng Machine learning có thể dự đoán được khách nào đang có nguy cơ bỏ đi.

Dựa vào bộ dữ liệu kèm theo, trực quan hóa dữ liệu bằng Tableau từ đó phân tích được cách yếu tố ảnh hưởng tới quyết định của khách hàng kết hợp với việc huấn luyện các mô hình Machine Learning để cuối cùng có những công cụ hiệu quả dự đoán được hành vi của khách hàng

1.4 PHẠM VI ĐỀ TÀI

Các thuật toán Machine Learning được áp dụng trong đề tài:

1. Logistics Regression
2. K-Nearest Neighbors
3. Adaptive Boosting
4. Random Forest
5. Gradient Boosting

Các thuật toán này được áp dụng để phân tích trong bộ dữ liệu kèm theo, liên quan đến lĩnh vực ngân hàng và mô hình để dự đoán là phân loại nhị phân. Nghĩa là khách hàng vẫn sẽ sử dụng dịch vụ ngân hàng hoặc ngưng sử dụng

1.5 Ý NGHĨA KHOA HỌC & THỰC TIỄN

1.5.1 Ý nghĩa khoa học

Đề tài này sẽ giúp làm sáng tỏ đặc điểm và thực trạng của việc khách hàng từ bỏ dịch vụ ngân hàng. Bằng cách nắm bắt và làm rõ các khía

cạnh chính trong mối quan hệ giữa khách hàng và ngân hàng ta có thể xây dựng một kho dữ liệu cục bộ về khách hàng. Điều này sẽ làm nền tảng không chỉ cho việc dự đoán rời đi của khách hàng mà còn giúp trong những vấn đề khác như là tính toán giá trị lâu dài của khách hàng, gợi ý các dịch vụ phù hợp cho khách hàng.

1.5.2 Ý nghĩa thực tiễn

Nếu có thể dự đoán được khách hàng nào chuẩn bị ngừng sử dụng ngân hàng thì ngân hàng có thể thay đổi các chiến lược cũng như các ưu đãi để giữ chân khách hàng. Như vậy ngân hàng có thể phát triển mối quan hệ với khách hàng mới và giữ chân những khách hàng cũ nhằm tiết kiệm chi phí hơn. Đồng thời việc phân tích dữ liệu trên sẽ hỗ trợ cho ngân hàng cải thiện về mức độ phục vụ khách hàng.

1.6 CẤU TRÚC BÁO CÁO

Báo cáo được chia thành các chương như sau :

Chương 1 : Tổng quan đề tài

Chương 2 : Tổng quan về giải thuật

Chương 3 : Dữ liệu thực nghiệm

Chương 4 : Thực nghiệm

Chương 5 : Kết luận

Chương 6 : Tài liệu tham khảo

Trong quá trình làm bài, chúng em đã cố gắng trình bày một cách đầy đủ, dễ hiểu nhất, đồng thời chúng em đã nỗ lực trong quá trình tìm hiểu tài liệu và viết bài tiểu luận này, nhưng không thể tránh khỏi những sai lầm, thiếu sót được. Chúng em, mong sẽ nhận được ý kiến đóng góp từ phía thầy cô.

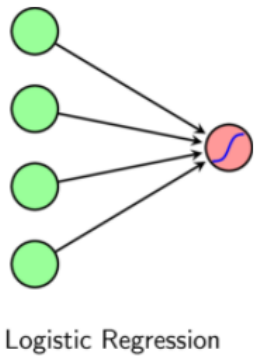
Chương 2

TỔNG QUAN VỀ GIẢI THUẬT

2.1 LOGISTICS REGRESSION

2.1.1 Tổng quan về Logisctics Regression

Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...). Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tình. Thuật toán trên dùng hàm sigmoid logistic để đưa ra đánh giá theo xác suất. Ví dụ: Khối u này 80% là lành tính, giao dịch này 90% là gian lận, ...



Hình 2.1: Biểu diễn Logistic Regression theo Neural Network

Ưu điểm

- Đơn giản, hiệu quả và được sử dụng rộng rãi
- Không yêu cầu mở rộng các tính năng
- Cung cấp điểm xác suất để quan sát
- Không đòi hỏi sức mạnh tính toán cao
- Tính toán nhanh trong việc phân loại các biến không xác định

Nhược điểm

- Nếu số lượng data ít hơn số lượng thuộc tính thì sẽ xảy ra overfitting
- Không thể giải quyết vấn đề phi tuyến tính
- Không hoạt động tốt với các biến độc lập không tương quan với biến mục tiêu
- Khó tính toán các mối quan hệ phức tạp, các thuật toán như Neural Networks cho kết quả tốt hơn Logistic Regression
- Yêu cầu các điểm dữ liệu được tạo ra một cách độc lập với nhau

2.1.2 Model Implement

```
# import Logistics Regression
from sklearn.linear_model import LogisticRegression

# model implement
lr = LogisticRegression(random_state=42, solver='liblinear', multi_class='ovr')
lr.fit(X_train,y_train)

# predict from model
y_pred = lr.predict(X_test)
```

2.2 K-NEAREST NEIGBORS

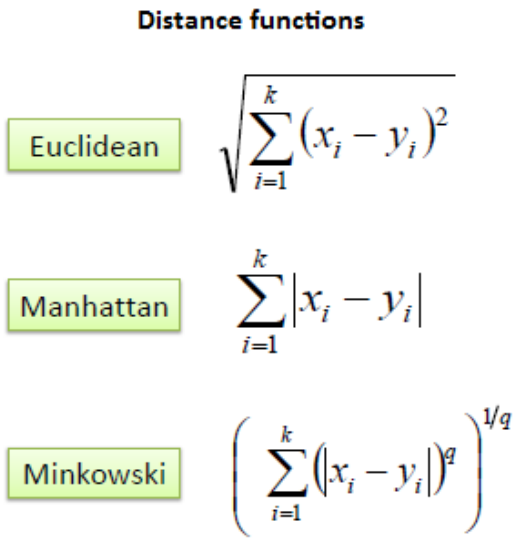
2.2.1 Tổng quan về K-Nearest Neigbors

KNN (K-Nearest Neighbors) là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ

tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.

K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán là Classification và Regression.

Thuật toán KNN cho rằng những dữ liệu tương tự nhau sẽ tồn tại gần nhau trong một không gian, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất. Việc tìm khoảng cách giữa 2 điểm cũng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp. Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính:



Hình 2.2: Các cách cơ bản để tính khoảng cách giữa 2 điểm dữ liệu

Ưu điểm

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

Nhược điểm

- KNN rất nhạy cảm với nhiễu khi K nhỏ.

- Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xá.
- Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.

2.2.2 Model Implement

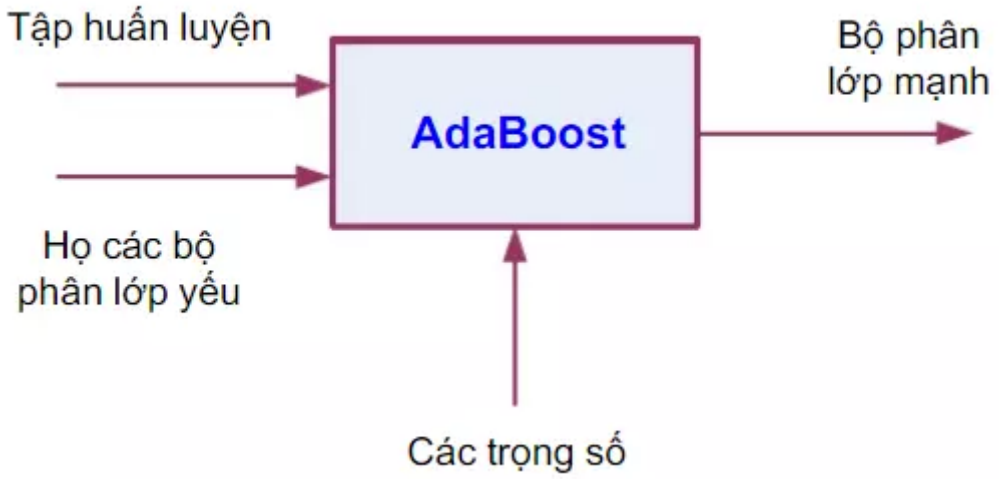
```
# import KNeigborsClassifier
from sklearn.neighbors import KNeighborsClassifier

# model implement
lr = LogisticRegression(random_state=42, solver='liblinear', multi_class='ovr')
lr.fit(X_train,y_train)

# predict from model
y_pred = lr.predict(X_test)
```

2.3 ADAPTIVE BOOSTING

2.3.1 Tổng quan về Adaptive Boosting

Adaptive Boost hay còn gọi là AdaBoost là một mô hình nằm trong nhánh Ensemble Learning của Machine Learning và là mô hình đầu tiên của nhánh Boosting trong Essemble Learning. AdaBoost có tên đầy đủ là Adaptive Boosting, do Yoav Freund và Robert Schapire công bố. Với ý tưởng là sử dụng các cây quyết định để đánh trọng số cho các điểm dữ liệu, từ đố tối thiểu hóa trọng số các điểm bị phân loại sai (trọng số lớn), để tăng hiệu suất của mô hình. AdaBoost là một thuật toán học mạnh, giúp đẩy nhanh việc tạo ra một bộ phân loại mạnh bằng cách chọn các đặc trưng tốt trong một họ các bộ phân loại yếu (thường là Decistion Tree) và kết hợp chúng lại tuyến tính bằng cách sử dụng các trọng số, Điều này thật sự cải thiện dần độ chính xác nhờ áp dụng hiệu quả một chuỗi các bộ phân loại yếu.

AdaBoost tiến hành train các mô hình mới dựa trên việc đánh lại trọng số cho các điểm dữ liệu hiện tải, nhằm giúp các mô hình mới có thể tập trung hơn vào các mẫu dữ liệu đang bị học sai, từ đó làm giảm giá trị loss của mô hình.



Hình 2.3: Các bước hoạt động của Adaptive Boosting

Ưu điểm

- AdaBoost khó dẫn đến tình trạng overfit do các thông số đầu vào không được tối ưu hóa chung
- Độ chính xác của các bộ phân loại yếu có thể được cải thiện bằng cách sử dụng AdaBoost
- Ngày này, ngoài là một mô hình rất tốt để phân loại nhị phân, AdaBoost còn được sử dụng để phân loại text và hình ảnh

Nhược điểm

- AdaBoost cần một bộ data với chất lượng cao. Một bộ data với nhiều điểm nhiễu cần được xử lý trước khi sử dụng mô hình AdaBoost.
- Chỉ có thể dùng để giải quyết các vấn đề về phân loại

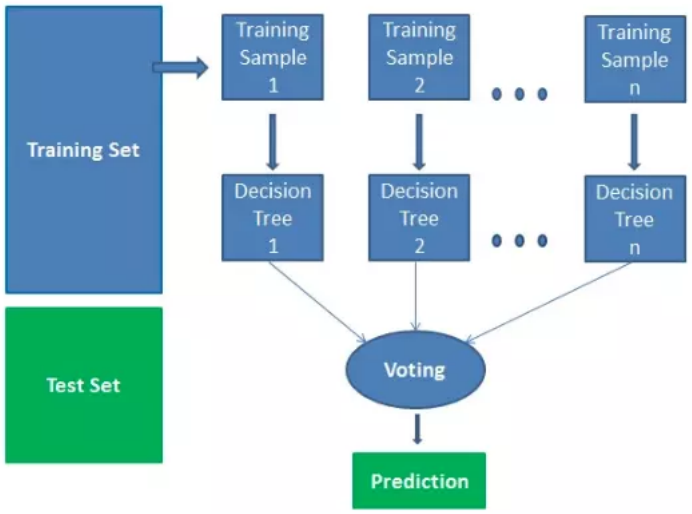
2.3.2 Model Implement

```
# import Adaptive Boosting
from sklearn.ensemble import AdaBoostClassifier

# model implement
ada = AdaBoostClassifier(n_estimators=100, random_state=0)
ada.fit(X_train,y_train)

# predict from model
y_pred = ada.predict(X_test)
```

2.4 RANDOM FOREST

2.4.1 Tổng quan về Random Forest

Random Forest là một mô hình thuộc Ensemble Learning. Random Forest được xây dựng từ nhiều cây quyết định (Decision Tree), tuy nhiên mỗi cây quyết định sẽ khác nhau (random). Sau đó kết quả được tổng hợp từ các mẫu ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu.



Hình 2.4: Các bước hoạt động của Random Forest

Thuật toán hoạt động theo bốn bước:

- Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
- Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi cây quyết định
- Bỏ phiếu cho mỗi kết quả dự đoán
- Chọn kết quả có tỉ lệ bỏ phiếu cao nhất

Ưu điểm

- Random Forest sử dụng nhiều decision tree và kết hợp các output của các decision tree. Điều này làm giảm overfit cho model và đồng thời giảm phương sai, từ đó tăng độ chính xác.

- Có thể được dùng cho cả các vấn đề về hồi quy và phân loại
- Có thể hoạt động tốt cả với các giá trị liên tục và phân loại
- Dễ dàng xử lý các điểm dữ liệu bị mất.
- Không yêu cầu scaling input

Nhược điểm

- Độ phức tạp cao do phải khởi tạo rất nhiều Decision Tree. Điều này dẫn tới việc hao phí tài nguyên
- Đi kèm với việc hao phí tài nguyên là cần nhiều thời gian để huấn luyện mô hình

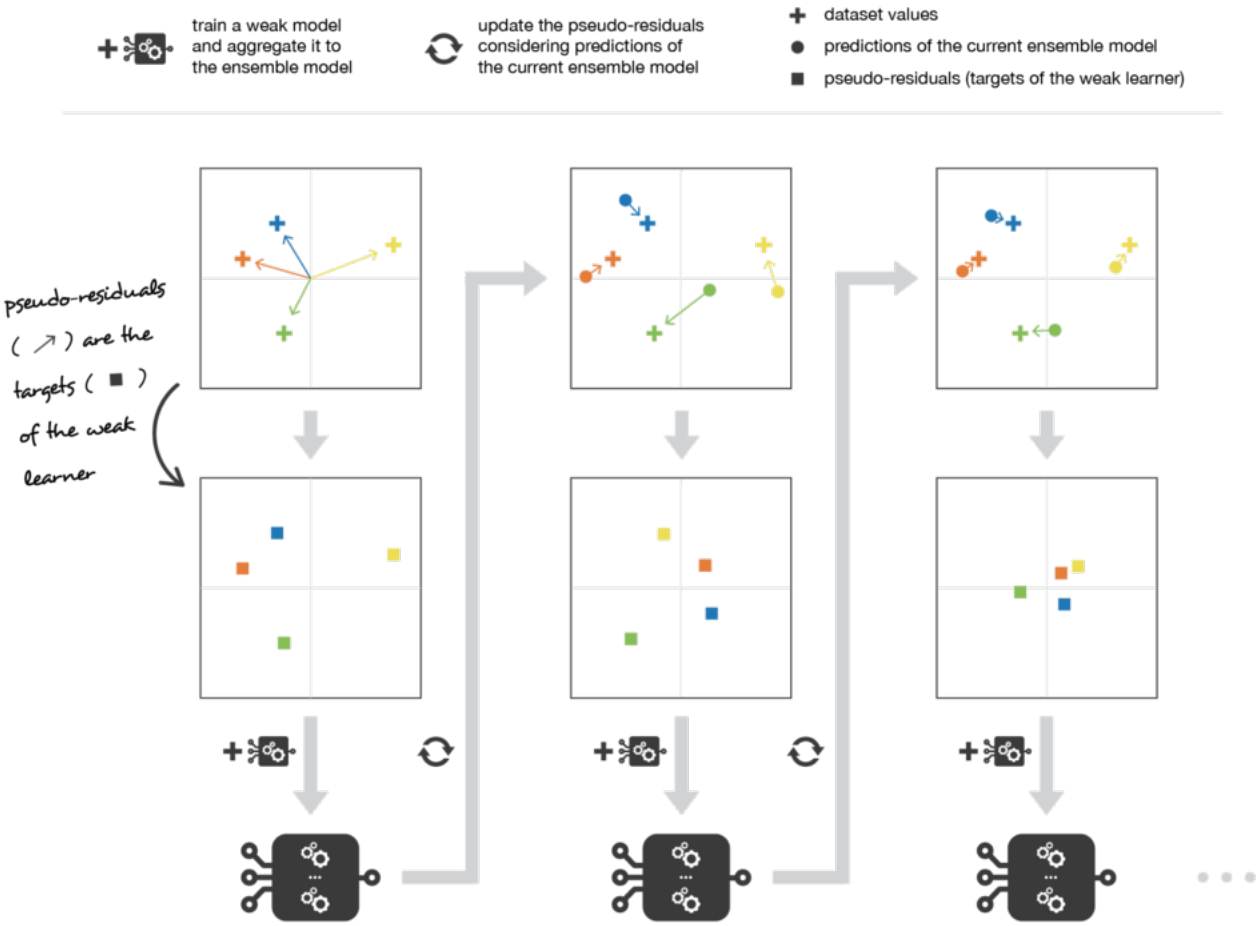
2.4.2 Model Implement

```
# import Random Forest
from sklearn.ensemble import RandomForestClassifier

# model implement
rfc = RandomForestClassifier(random_state=42)
rfc.fit(X_train,y_train)

# predict from model
y_pred = rfc.predict(X_test)
```

2.5 GRADIENT BOOSTING

2.5.1 Tổng quan về Gradient Boosting

Giống với AdaBoost, cùng thuộc nhánh boosting của Essemble Learning, nhưng Gradient Boosting được xem là một trong những thuật toán mạnh mẽ nhất được sử dụng trong các mô hình hồi quy và phân loại. Được tạo nên từ Gradient Descent và Boosting. 'Gradient' nghĩa là bạn có thể sử đạo hàm bậc hai hoặc đạo hàm nhiều bậc cho cùng một function. Gradient Boosting có ba phần chính, lần lượt là: additive model, loss function và một bộ phân loại yếu.



Hình 2.5: Cách hoạt động của Gradient Boosting

Ưu điểm

- Vì Gradient Boosting được xem như là một trong những mô hình mạnh của nhánh boosting nên độ chính xác rất cao.
- Tính linh hoạt cao, có thể tối ưu hóa trên nhiều loss function khác nhau và có nhiều các tham số để tùy chỉnh model.
- Không yêu cầu tiền xử lý dữ liệu.
- Dễ dàng xử lý các điểm dữ liệu bị mất.

Nhược điểm

- Gradient Boosting sẽ có thể gây ra overfit.
- Gradient Boosting cần nhiều tài nguyên để tính toán, có thể gây hao phí thời gian và bộ nhớ.

2.5.2 Model Implement

```
# import Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier

# model implement
gbc = GradientBoostingClassifier(n_estimators=100, learning_rate=0.5, max_depth=5,
    random_state=42)
gbc.fit(X_train,y_train)

# predict from model
y_pred = gbc.predict(X_test)
```

Chương 3

DỮ LIỆU THỰC NGHIỆM

3.1 PHÂN TÍCH BỘ DỮ LIỆU

3.1.1 Nguồn gốc data

Dữ liệu được lấy từ trang web

https://www.kaggle.com/sakshigoyal7/credit-card-customers

Bộ dữ liệu đã được thêm vào 2 thuộc tính mới là Home-ownership và Employees xác định khách hàng có việc làm hay không và hình thức sở hữu nhà của khách hàng đó

3.1.2 Chi tiết các thuộc tính

Bộ dữ liệu bao gồm 10.000 khách hàng gồm 22 thuộc tính:

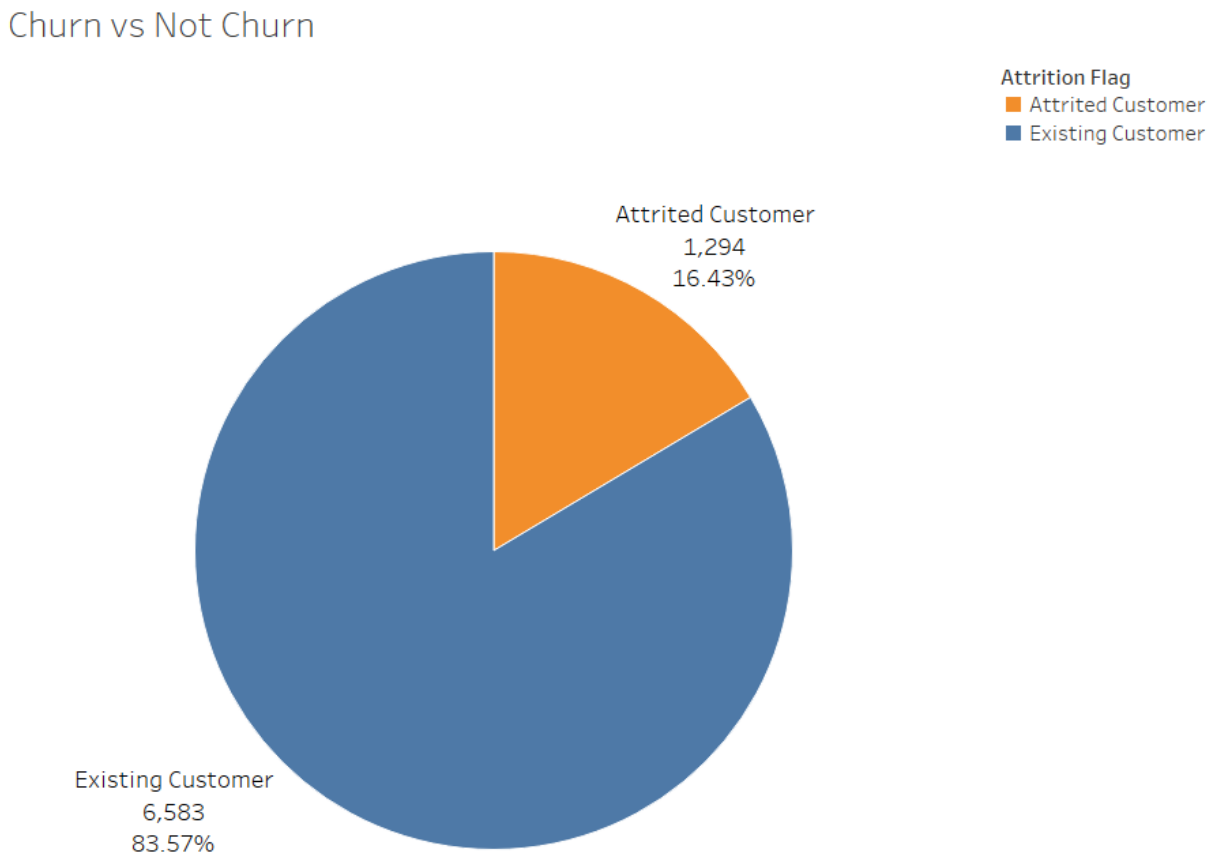
- CLIENTNUM: số id khách hàng
- Attrition_Flag: phân loại khách hàng còn sử dụng hay đã ngưng
- Customer_Age: Số tuổi khách hàng tính theo năm
- Gender: Giới tính khách hàng (Nam – Nữ)
- Dependent_count: Số người phụ thuộc
- Education_Level: Trình độ học vấn
- Marital_Status: Tình trạng hôn nhân
- Income_Category: Thu nhập hàng năm
- Card_Category: Loại thẻ ngân hàng

- Months_on_book: Thời gian dùng dịch vụ ngân hàng tính theo tháng
- Total_Relationship_Count: Các dịch vụ khác của ngân hàng mà khách hàng sử dụng
- Months_Inactive_12_mon: Số tháng không sử dụng dịch vụ trong vòng 12 tháng
- Contacts_Count_12_mon : Số lần liên hệ với ngân hàng trong vòng 12 tháng
- Credit_Limit: Hạn mức tín dụng
- Total_Revolving_Bal: Số tiền mượn không trả ở mỗi cuối kì thanh toán
- Avg_Open_To_Buy: Trung bình mức tối đa mà khách hàng có thể trả sử dụng thẻ ngân hàng (mức tối đa bằng hạn mức tín dụng trừ cho số dư)
- Total_Amt_Chng_Q4_Q1: thay đổi trong số tiền trong mỗi giao dịch Q4 -> Q1
- Total_Trans_Amt: Tổng số tiền giao dịch trong 12 tháng
- Total_Trans_Ct: Tổng số lần giao dịch trong 12 tháng
- Total_Ct_Chng_Q4_Q1: Thay đổi trong tổng số lần giao dịch Q4 -> Q1
- Avg_Utilization_Ratio: Tỉ lệ dùng dịch vụ
- Home_ownership: Quyền sở hữu nhà của khách hàng
- Employees: Khách hàng có việc làm hay không

3.1.3 Trình bày bộ dữ liệu bằng Tableau

Churn vs Not Churn

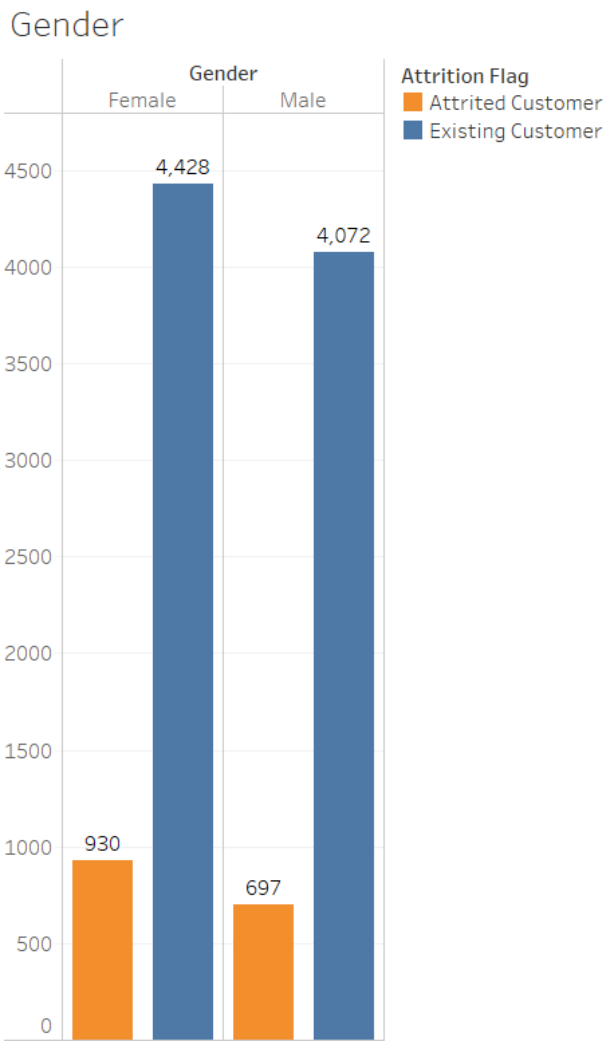
Phân bố khách hàng theo hai nhãn đang sử dụng và ngưng sử dụng dịch vụ ngân hàng



Hình 3.1: Churn vs Not Churn

Gender

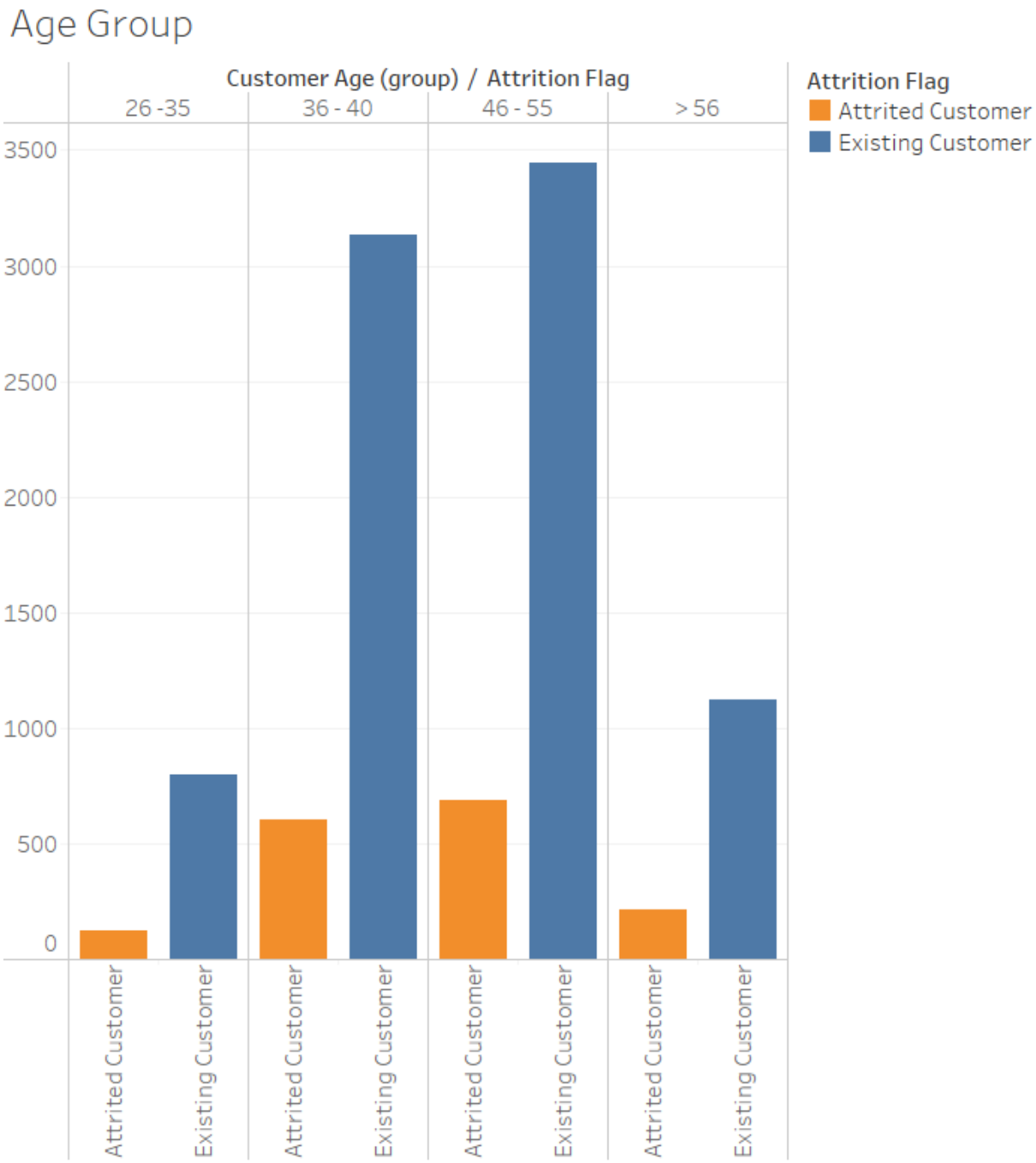
Phân bổ khách hàng theo giới tính có thể thấy rằng số lượng khách hàng nữ bỏ đi nhiều hơn nam



Hình 3.2: Gender

Age group

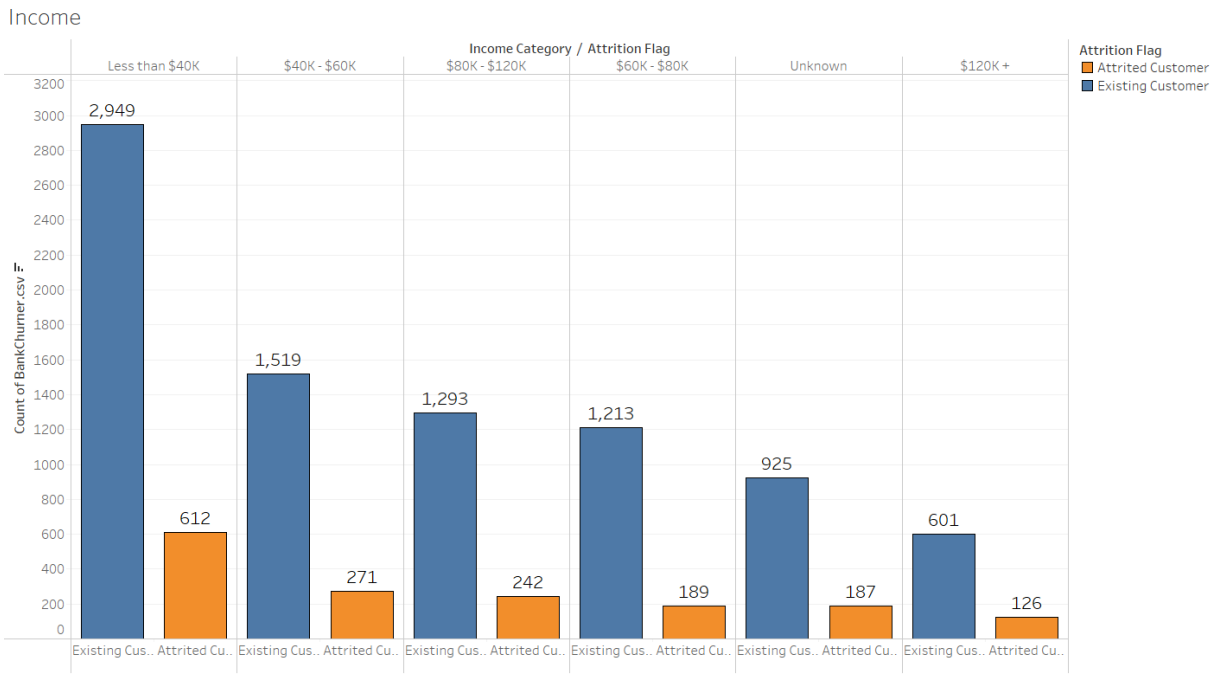
Phân bổ khách hàng theo độ tuổi thì có thể thấy trong nhóm tuổi 36-40, 46-55 ngưng sử dụng ngân hàng nhiều nhất.



Hình 3.3: Age group

Income

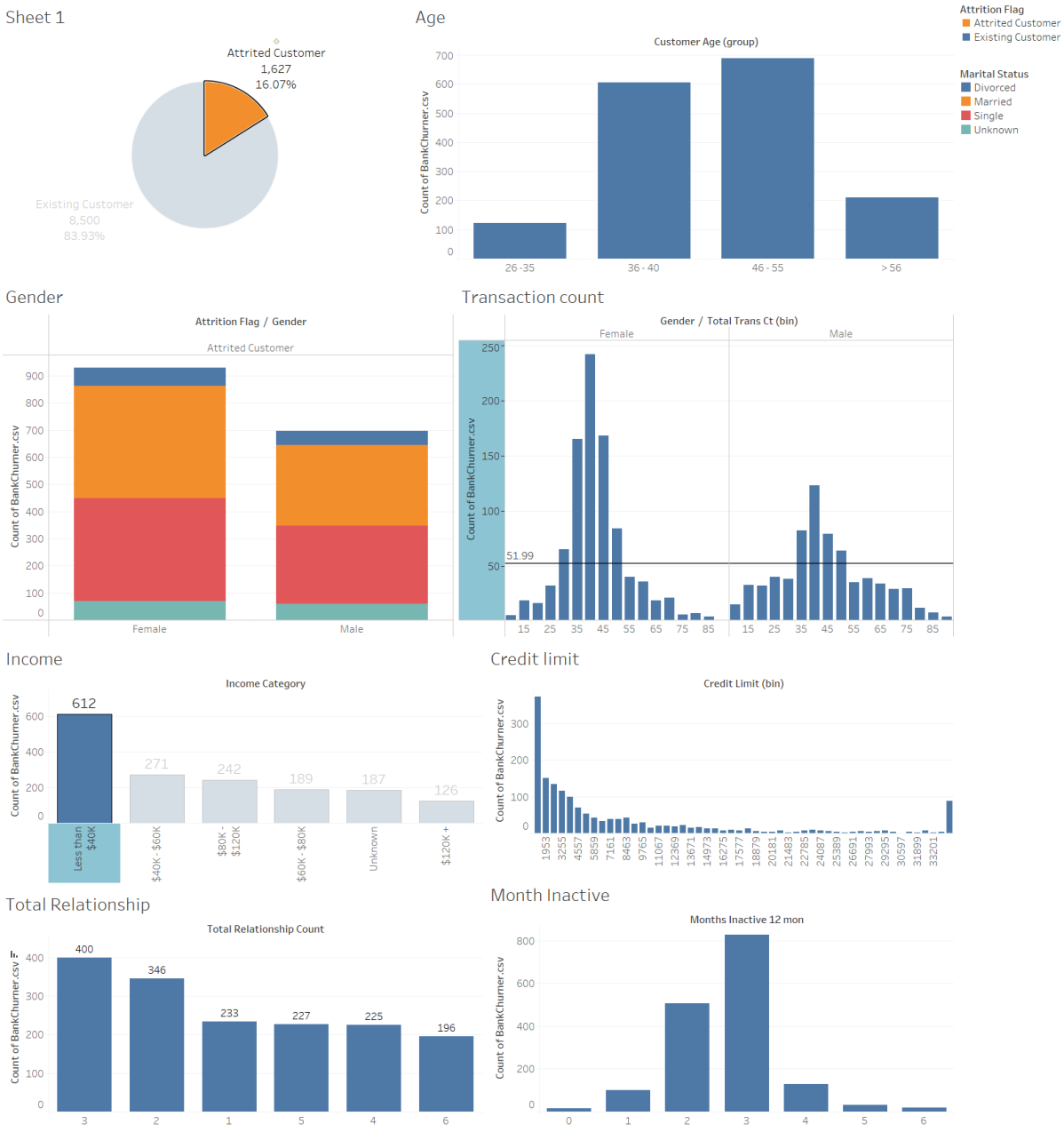
Phân bổ khách hàng theo thu nhập thì có thể thấy những khách hàng có thu nhập dưới 40K$ mỗi năm ngừng sử dụng ngân hàng là nhiều nhất.



Hình 3.4: Income

Churn Data

Khi xem xét dữ liệu của các khách hàng bỏ đi thì các khách hàng nữ trong độ tuổi 46-55 và có thu nhập dưới 40K$ hàng năm chiếu ưu thế và đồng thời tổng số giao dịch giao động từ 35 tới 45 lần trong 1 năm.



Hình 3.5: Churn Data

Overview of Data

Tổng quan về toàn bộ dữ liệu.

Hình 3.6: Data Overview

3.2 TIỀN XỬ LÝ DỮ LIỆU

Ở phần tiền xử lý dữ liệu chỉ đơn giản là dùng label encoder để chuẩn hóa dữ liệu

```
# import library
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

# Data preprocess
le.fit(df['Gender'])
df['Gender'] = le.transform(df['Gender'])
le.fit(df['Education_Level'])
df['Education_Level'] = le.transform(df['Education_Level'])
le.fit(df['Income_Category'])
df['Income_Category'] = le.transform(df['Income_Category'])
le.fit(df['Marital_Status'])
df['Marital_Status'] = le.transform(df['Marital_Status'])
le.fit(df['Card_Category'])
df['Card_Category'] = le.transform(df['Card_Category'])
le.fit(df['home_ownership'])
df['home_ownership'] = le.transform(df['home_ownership'])
```

```
le.fit(df['employees'])
df['employees'] = le.transform(df['employees']
le.fit(df['Attrition_Flag'])
df['Attrition_Flag'] = le.transform(df['Attrition_Flag'])
```

CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book
768805383	Existing Customer	45	M	3	High School	Married	$60K - $80K	Blue	39
818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44
713982108	Existing Customer	51	M	3	Graduate	Married	$80K - $120K	Blue	36
769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	34
709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - $80K	Blue	21

Hình 3.7: 10 cột đầu tiên trước khi tiền xử lý

CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book
768805383	1	45	1	3	3	1	2	0	39
818770008	1	49	0	5	2	2	4	0	44
713982108	1	51	1	3	2	1	3	0	36
769911858	1	40	0	4	3	3	4	0	34
709106358	1	40	1	3	5	1	2	0	21

Hình 3.8: 10 cột đầu tiên sau khi tiền xử lý

```
# Choose feature to train
X_feature = {'Customer_Age', 'Gender', 'Dependent_count', 'Education_Level', '
    Marital_Status', 'Income_Category', 'Card_Category', 'Credit_Limit', '
    Months_on_book', 'Total_Relationship_Count', 'Months_Inactive_12_mon', '
    Contacts_Count_12_mon', 'Total_Revolving_Bal', 'Avg_Open_To_Buy', '
    Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt', 'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1'
    , 'Avg_Utilization_Ratio', 'home_ownership', 'employees'}
X_feature = list(X_feature)
Y_feature = 'Attrition_Flag'

# X and y as inputs
X = df[X_feature]
y = df[Y_feature]
```

3.3 CÁC PHƯƠNG PHÁP ĐÁNH GIÁ GIẢI THUẬT

3.3.1 Giới thiệu về các phương pháp đánh giá

Khi xây dựng các mô hình Machine Learning, chúng ta cần xác định xem mô hình đó có tốt hay không. Nhưng làm cách nào để đánh giá hiệu quả một mô hình và so sánh hiệu năng giữa các mô hình với nhau một cách khách quan nhất?

Có rất nhiều mô hình đánh giá classification.Tùy vào những bài toán khác nhau mà chúng ta sử dụng các phương pháp các nhau. Các phương pháp thường được sử dụng là: accuracy score, confusion matrix, ROC curve, Area Under the Curve, Precision and Recall, F1 Score, etc. Tuy nhiên, trong phạm vi đồ án môn học này, bọn em sẽ sử dụng các phương pháp đánh giá lần lượt là accurancy score, confusion matrix, F1 score, Precision and Recall.

Các phương pháp đánh giá này sẽ được sử dụng trên tập dữ liệu chia theo KFold với n_splits=5. Và bộ dữ liệu gồm 2 nhãn để đánh giá. Giả sử ta gọi y_pred là tập kết quả đầu ra được mô hình dự đoán và y_test là tập kết quả thực.

3.3.2 Accuracy Score

Accuracy score là cách đơn giản và thường được sử dụng nhất. Cách đánh giá này đơn giản là tính tỉ lệ giữa số điểm được dự đoán đúng với tổng số điểm trong tập dữ liệu kiểm thử.

```
import numpy as np

def acc(y_true, y_pred):
    correct = np.sum(y_true == y_pred)
    return float(correct)/y_true.shape[0]

y_true = np.array([0, 0, 0, 0, 1, 1, 1, 1, 0, 0])
y_pred = np.array([0, 1, 0, 1, 1, 1, 0, 1, 1, 0])
print('accuracy = ', acc(y_true, y_pred))
```

```
 accuracy =  0.6
```

3.3.3 Confusion Matrix

Accuracy chỉ cho chúng ta biết về phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại nhiều nhất, và lớp nào thường bị phân loại nhầm vào lớp khác. Ta sử dụng confusion matrix để đánh giác các giá trị trên.

Confusion matrix là một ma trận vuông với kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j số lượng điểm **lẽ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j**. Như vậy với phân loại nhị phân, ta sẽ có số điểm tại hàng thứ 0 cột thứ 0 là số điểm dự đoán đúng của nhãn 0, hàng thứ 0 cột thứ 1 là số điểm lẽ ra thuộc vào nhãn 0 nhưng lại được dự đoán là thuộc vào nhãn 1, tương tự với các hàng và cột còn lại.

```
from sklearn.metrics import confusion_matrix
cnf_matrix = confusion_matrix(y_true, y_pred)
print('Confusion matrix:')
print(cnf_matrix)
normalized_confusion_matrix = cnf_matrix/cnf_matrix.sum(axis = 1, keepdims = True)
print('\nConfusion matrix (with normalizatrion:)')
print(normalized_confusion_matrix)
```

```
Confusion matrix:
[[3 3]
 [1 3]]

Confusion matrix (with normalizatrion:)
[[0.5  0.5 ]
 [0.25 0.75]]
```

3.3.4 Precision và Recall

True/False Positive/Negative

Ta định nghĩa True Positive(TP), False Positive(FP), True Negative(TN), False Negative(FN). Nếu xét theo confusion matrix sẽ được biểu diễn như sau:

```
                  |      Predicted      |      Predicted      |
                  |     as Positive     |     as Negative     |
------------------|---------------------|---------------------|
 Actual: Positive | True Positive (TP)  | False Negative (FN) |
------------------|---------------------|---------------------|
 Actual: Negative | False Positive (FP) | True Negative (TN)  |
------------------|---------------------|---------------------|
```

Precision

Precision là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive. Ta có thể khái quát bằng công thức:

$$Precision = \frac{TP}{TP + FP}$$

Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao. Khi Precision = 1, mọi điểm tìm được đều thực sự là positive, tức là không có điểm negative nào lẫn vào kết quả. Tuy nhiên, Precision = 1 không đảm bảo mô hình là tốt, vì câu hỏi đặt ra là liệu mô hình đã tìm được tất cả các điểm positive hay chưa. Nếu một mô hình chỉ tìm được đúng mọt điểm positive mà nó chắc chắn nhất thì ta không thể gọi là một mô hình tốt.

Recall

Recall là tỉ lệ số điểm true positive trong số nhẽng điểm thực sự là positive. Khái quát bằng công thức như sau:

$$Recall = \frac{TP}{TP + FN}$$

Recall cao đồng nghĩa với việc True Positive cao, tức là tỉ lệ bỏ sót các điểm thực sự positive là thấp. Khi Recall = 1, mọi điểm positive đều được tìm thấy. Tuy nhiên, đại lượng này lại không đo liệu có bao nhiêu điểm negative lẫn trong đó. Nếu mô hình phân loại mọi điểm là

positive thì chắc chắn Recall = 1, tuy nhiên dễ nhận ra đây là một mô hình cực tồi.

3.3.5 F1-Score

F1-Score hay còn gọi là F_1 score, là harmonic mean của precision và recall. Nghĩa là, F1-Score sẽ lấy cả FP và FN để tính toán. Mặc dù F1-Score khó hiểu hơn nhiều so với Accuracy nhưng F1 hữu dụng hơn nhiều so với accuracy, đặc biệt khi mà dữ liệu đầu vào bị lệch nhiều về một nhãn. Accuracy sẽ rất hiệu quả nếu FP và FN tương đương với nhau. Nếu FP và FN lệnh nhau quá nhiều, F1 sẽ là sự lựa chọn nên được ưu tiên.

Công thức tính F1-Score:

$$F1 - Score = \frac{2(Precision * Recall)}{Precision + Recall}$$

3.4 TỔNG QUAN VỀ SMOTE

3.4.1 SMOTE là gì ?

SMOTE là viết tắt của Synthetic Minority Oversampling Technique. Là một kỹ thuật thống kê làm tăng số lượng dữ liệu của bộ dữ liệu một cách cân bằng. Ý tưởng của mô hình là tạo ra nhiều điểm dữ liệu từ các thuộc tính phân lớp có số lượng ít mà đầu vào cung cấp. SMOTE không thay đổi số lượng của các thuộc tính phân lớp chiếm đa số.

Các điểm dữ liệu mới không phải chỉ là bản sao chép của các điểm dữ liệu có sẵn, thay vào đó sử dụng giải thuật để tạo ra các điểm mới bằng cách kết hợp các feature của điểm gốc và feature của các điểm lân cận.

SMOTE coi toàn bộ bộ dữ liệu đầu vào làm input nhưng chỉ tăng số lượng của dữ liệu có thuộc tính phân lớp ít.

3.4.2 Tại sao lại cần sử dụng SMOTE ?

Trong thực tế, có nhiều bộ dữ liệu bị mất cân bằng ở các thuộc tính phân lớp. Điều này gây khó khăn cho các mô hình machine learning, đòi hỏi sự cân bằng ở các thuộc tính phân lớp để các mô hình không có sự thiên vị trong các bài toán phân loại. SMOTE là một cách để làm

tăng số lượng dữ liệu của các thuộc tính phân lớp có số lượng ít hoặc hiếm.

SMOTE trả về một bộ data mới bao gồm bộ data cũ kết hợp với lượng dữ liệu được thêm bởi SMOTE

Chương 4

THỰC NGHIỆM

4.1 IMPORT CÁC THƯ VIỆN VÀ ĐỌC DỮ LIỆU

```
# Import library
import pandas as pd
from sklearn.model_selection import train_test_split,cross_val_score, KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier,
    GradientBoostingClassifier
import seaborn as sn
import matplotlib.pyplot as plt
import random
from sklearn.linear_model import LogisticRegression
!pip install imbalanced-learn
from imblearn.over_sampling import SMOTE

# Read Input from CSV
df = pd.read_csv('BankChurner.csv')
# Drop 2 unnecessary columns
df = df.drop(['
    Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon
    _Dependent_count_Education_Level_Months_Inactive_12_mon_1','
    Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon
    _Dependent_count_Education_Level_Months_Inactive_12_mon_2'], axis = 1)
```

Tạo dữ liệu giả để đánh giá mô hình

```
dummy_inputs = {}
number_of_feature = 5
for feature in X:
  dummy_inputs[feature] = []
for index in range(number_of_feature):
```

```
  for feature in X:
    dummy_inputs[feature].append(random.choice(X[feature]))
dummy_inputs = pd.DataFrame(dummy_inputs, index=[i for i in range(number_of_feature)
    ])
dummy_inputs
```

Education_Level	Gender	Months_on_book	Credit_Limit	Total_Revolving_Bal	Card_Category	Months_Inactive_12_mon	employees	Total_Trans_Ct	Customer_Age
5	0	39	7225.0	2517	0	1	0	94	42
2	0	36	19136.0	989	0	4	1	33	51
0	1	34	29856.0	1250	0	2	1	61	52
5	0	50	2393.0	0	0	1	1	78	56
5	1	42	34516.0	0	0	3	0	55	45

Hình 4.1: 10 cột đầu tiên của dữ liệu giả dùng để đánh giá

4.2 ĐÁNH GIÁ TRÊN CÁC MÔ HÌNH

Các mô hình được đánh giá bằng KFold với splits = 5

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
```

4.2.1 Logistic Regression

Không sử dụng SMOTE

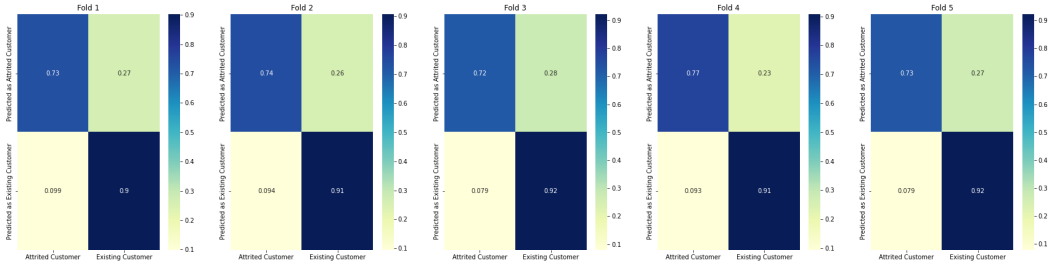
```
lr = LogisticRegression(random_state=42, solver='liblinear', multi_class='ovr')
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]
    lr.fit(X_train,y_train)
    y_pred = lr.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_lr = lr.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
    inverse_transform(dummy_outputs_lr)))
    print(classification_report(y_pred, y_test))
```

```
    print("*"*60)
    count += 1
```

Kết quả của Logistics Regression không sử dụng SMOTE

```
# Result at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.45      0.73      0.56       200
           1       0.97      0.90      0.93      1826

    accuracy                           0.88      2026
   macro avg       0.71      0.82      0.75      2026
weighted avg       0.92      0.88      0.90      2026
# Result at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.49      0.74      0.59       221
           1       0.97      0.91      0.94      1805

    accuracy                           0.89      2026
   macro avg       0.73      0.82      0.76      2026
weighted avg       0.91      0.89      0.90      2026
# Result at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.54      0.72      0.62       233
           1       0.96      0.92      0.94      1792

    accuracy                           0.90      2025
   macro avg       0.75      0.82      0.78      2025
weighted avg       0.91      0.90      0.90      2025
# Result at Fold 4
Predict dummy inputs at Fold 4: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.49      0.77      0.60       213
           1       0.97      0.91      0.94      1812

    accuracy                           0.89      2025
   macro avg       0.73      0.84      0.77      2025
weighted avg       0.92      0.89      0.90      2025
# Result at Fold 5
```

```
Predict dummy inputs at Fold 5: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
                precision    recall  f1-score   support

           0       0.57      0.73      0.64       251
           1       0.96      0.92      0.94      1774

    accuracy                           0.90      2025
   macro avg       0.76      0.83      0.79      2025
weighted avg       0.91      0.90      0.90      2025
```



Hình 4.2: Đánh giá Logistics Regression qua confusion matrix không sử dụng SMOTE

Có sử dụng SMOTE

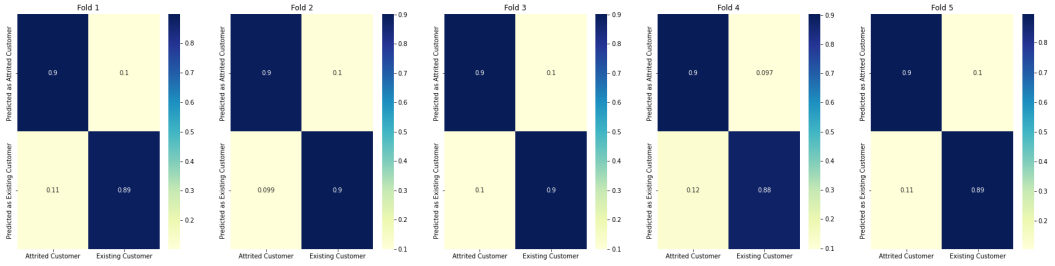
```
oversample = SMOTE()
X, y = oversample.fit_resample(X,y)

lr_smote = LogisticRegression(random_state=42, solver='liblinear', multi_class='ovr')
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    lr_smote.fit(X_train,y_train)
    y_pred = lr_smote.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_lr_smote = lr_smote.predict(dummy_inputs)
```

```
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
    inverse_transform(dummy_outputs_lr_smote)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của Logistics Regression có sử dụng SMOTE

```
# Result at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.45      0.73      0.56       200
           1       0.97      0.90      0.93      1826

    accuracy                           0.88      2026
   macro avg       0.71      0.82      0.75      2026
weighted avg       0.92      0.88      0.90      2026
# Result at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.49      0.74      0.59       222
           1       0.97      0.91      0.94      1804

    accuracy                           0.89      2026
   macro avg       0.73      0.82      0.76      2026
weighted avg       0.91      0.89      0.90      2026
# Result at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.54      0.72      0.62       234
           1       0.96      0.92      0.94      1791

    accuracy                           0.90      2025
   macro avg       0.75      0.82      0.78      2025
weighted avg       0.91      0.90      0.90      2025
# Result at Fold 4
Predict dummy inputs at Fold 4: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.49      0.79      0.60       205
           1       0.97      0.91      0.94      1820

    accuracy                           0.89      2025
```

```
   macro avg       0.73      0.85      0.77      2025
weighted avg       0.93      0.89      0.90      2025
# Result at Fold 5
Predict dummy inputs at Fold 5: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.56      0.74      0.64       247
           1       0.96      0.92      0.94      1778

    accuracy                           0.90      2025
   macro avg       0.76      0.83      0.79      2025
weighted avg       0.91      0.90      0.90      2025
```



Hình 4.3: Đánh giá Logistics Regression qua confusion matrix có sử dụng SMOTE

4.2.2 K-Nearest Neighbors

Không sử dụng SMOTE

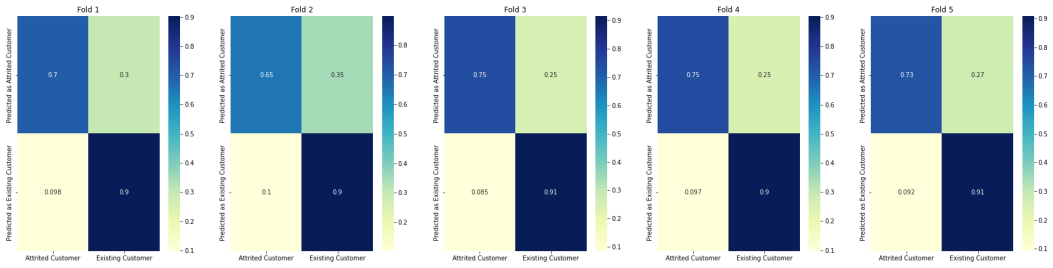
```
knn = KNeighborsClassifier(n_neighbors=20)
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    knn.fit(X_train,y_train)
    y_pred = knn.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
```

```
    dummy_outputs_knn = knn.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
    inverse_transform(dummy_outputs_knn)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của KNN không sử dụng SMOTE

```
# Predict at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.46      0.70      0.55       214
           1       0.96      0.90      0.93      1812

    accuracy                           0.88      2026
   macro avg       0.71      0.80      0.74      2026
weighted avg       0.91      0.88      0.89      2026
# Predict at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.45      0.65      0.53       230
           1       0.95      0.90      0.92      1796

    accuracy                           0.87      2026
   macro avg       0.70      0.77      0.73      2026
weighted avg       0.90      0.87      0.88      2026
# Predict at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.50      0.75      0.60       207
           1       0.97      0.91      0.94      1818

    accuracy                           0.90      2025
   macro avg       0.73      0.83      0.77      2025
weighted avg       0.92      0.90      0.91      2025
# Predict at Fold 4
Predict dummy inputs at Fold 4: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.47      0.75      0.58       210
           1       0.97      0.90      0.93      1815

```

```
    accuracy                           0.89      2025
   macro avg       0.72      0.83      0.76      2025
weighted avg       0.92      0.89      0.90      2025
# Predict at Fold 5
Predict dummy inputs at Fold 5: ['Existing Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.48      0.73      0.58       215
           1       0.97      0.91      0.94      1810

    accuracy                           0.89      2025
   macro avg       0.72      0.82      0.76      2025
weighted avg       0.91      0.89      0.90      2025
```



Hình 4.4: Đánh giá KNN qua confusion matrix không sử dụng SMOTE

Có sử dụng SMOTE

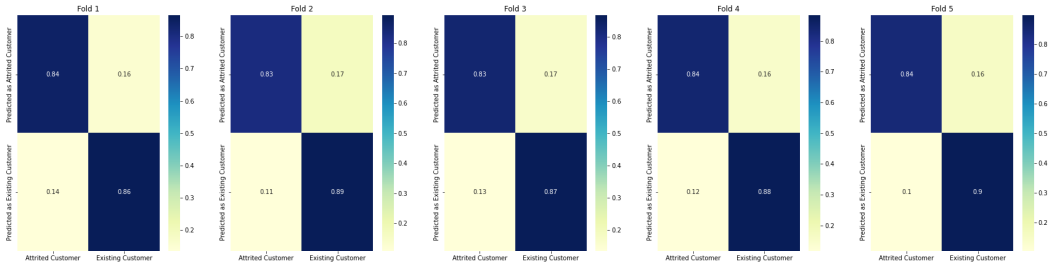
```
oversample = SMOTE()
X, y = oversample.fit_resample(X,y)

knn_smote = KNeighborsClassifier(n_neighbors=20)
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    knn_smote.fit(X_train,y_train)
    y_pred = knn_smote.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

```

```
    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_knn_smote = knn_smote.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
    inverse_transform(dummy_outputs_knn_smote)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của KNN có sử dụng SMOTE

```
# Predict at Fold 1
Predict dummy inputs at Fold 1: ['Attrited Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.87      0.84      0.86      1783
           1       0.83      0.86      0.85      1617

    accuracy                           0.85      3400
   macro avg       0.85      0.85      0.85      3400
weighted avg       0.85      0.85      0.85      3400
# Predict at Fold 2
Predict dummy inputs at Fold 2: ['Attrited Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.89      0.83      0.86      1756
           1       0.83      0.89      0.86      1644

    accuracy                           0.86      3400
   macro avg       0.86      0.86      0.86      3400
weighted avg       0.86      0.86      0.86      3400
# Predict at Fold 3
Predict dummy inputs at Fold 3: ['Attrited Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.87      0.83      0.85      1775
           1       0.82      0.87      0.84      1625

    accuracy                           0.85      3400
   macro avg       0.85      0.85      0.85      3400
weighted avg       0.85      0.85      0.85      3400
# Predict at Fold 4
Predict dummy inputs at Fold 4: ['Attrited Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.89      0.84      0.87      1830
```

```
           1       0.83      0.88      0.86      1570

    accuracy                           0.86      3400
   macro avg       0.86      0.86      0.86      3400
weighted avg       0.86      0.86      0.86      3400
# Predict at Fold 5
Predict dummy inputs at Fold 5: ['Attrited Customer' 'Existing Customer' 'Existing
    Customer' 'Attrited Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.91      0.84      0.87      1847
           1       0.83      0.90      0.86      1553

    accuracy                           0.87      3400
   macro avg       0.87      0.87      0.87      3400
weighted avg       0.87      0.87      0.87      3400
```



Hình 4.5: Đánh giá KNN qua confusion matrix có sử dụng SMOTE

4.2.3 Adaptive Boosting

Không sử dụng SMOTE

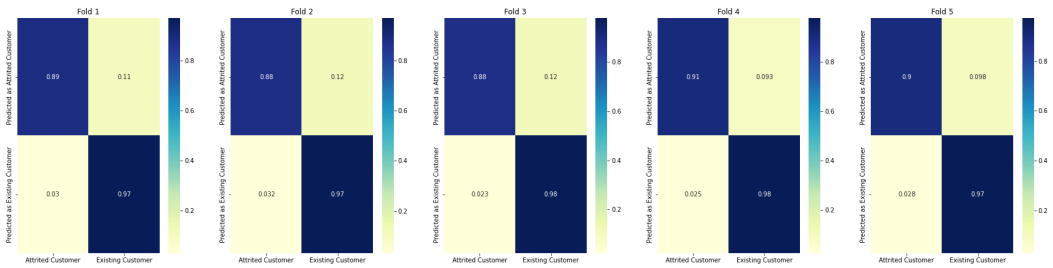
```
ada = AdaBoostClassifier(n_estimators=100, random_state=0)
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    ada.fit(X_train,y_train)
    y_pred = ada.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])
```

```
15
16      ax[count].set_title("Fold "+ str(count + 1))
17      sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
18      dummy_outputs_ada = ada.predict(dummy_inputs)
19      print("Predict dummy inputs at Fold "+str(count +1)+ ":" +str(le.
        inverse_transform(dummy_outputs_ada)))
20      print(classification_report(y_pred, y_test))
21      print("*"*60)
22      count += 1
```

Kết quả của Adaptive Boosting không sử dụng SMOTE

```
1  # Result at Fold 1
2  Predict dummy inputs at Fold 1: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Existing Customer']
3                  precision    recall  f1-score   support
4
5               0       0.84      0.89      0.86       310
6               1       0.98      0.97      0.97      1716
7
8       accuracy                           0.96      2026
9      macro avg       0.91      0.93      0.92      2026
10  weighted avg       0.96      0.96      0.96      2026
11 # Result at Fold 2
12 Predict dummy inputs at Fold 2: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Existing Customer']
13                 precision    recall  f1-score   support
14
15              0       0.84      0.88      0.86       318
16              1       0.98      0.97      0.97      1708
17
18      accuracy                           0.95      2026
19     macro avg       0.91      0.92      0.91      2026
20  weighted avg       0.95      0.95      0.95      2026
21 # Result at Fold 3
22 Predict dummy inputs at Fold 3: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Attrited Customer']
23                 precision    recall  f1-score   support
24
25              0       0.87      0.88      0.88       306
26              1       0.98      0.98      0.98      1719
27
28      accuracy                           0.96      2025
29     macro avg       0.92      0.93      0.93      2025
30  weighted avg       0.96      0.96      0.96      2025
31 # Result at Fold 4
32 Predict dummy inputs at Fold 4: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Attrited Customer']
33                 precision    recall  f1-score   support
34
```

```
           0       0.87      0.91      0.89       321
           1       0.98      0.98      0.98      1704

    accuracy                           0.96      2025
   macro avg       0.93      0.94      0.93      2025
weighted avg       0.97      0.96      0.96      2025
# Result at Fold 5
Predict dummy inputs at Fold 5: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.85      0.90      0.88       305
           1       0.98      0.97      0.98      1720

    accuracy                           0.96      2025
   macro avg       0.92      0.94      0.93      2025
weighted avg       0.96      0.96      0.96      2025
```



Hình 4.6: Đánh giá Adaptive Boosting qua confusion matrix không sử dụng SMOTE

Có sử dụng SMOTE

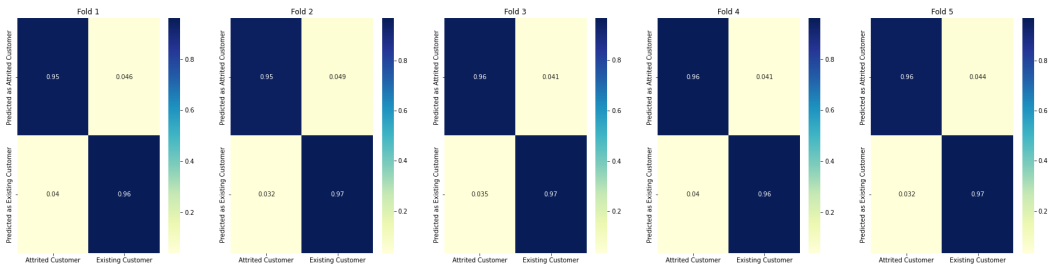
```
oversample = SMOTE()
X, y = oversample.fit_resample(X,y)

ada_smote = AdaBoostClassifier(n_estimators=100, random_state=0)
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    ada_smote.fit(X_train,y_train)
    y_pred = ada_smote.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
```

```
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_ada_smote = ada_smote.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+ ": " + str(le.inverse_transform(dummy_outputs_ada_smote)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của Adaptive Boosting có sử dụng SMOTE

```
# Result at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Attrited Customer' 'Existing Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.96      0.95      0.96      1738
           1       0.95      0.96      0.96      1662

    accuracy                           0.96      3400
   macro avg       0.96      0.96      0.96      3400
weighted avg       0.96      0.96      0.96      3400
# Result at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Attrited Customer' 'Existing Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      1662
           1       0.95      0.97      0.96      1738

    accuracy                           0.96      3400
   macro avg       0.96      0.96      0.96      3400
weighted avg       0.96      0.96      0.96      3400
# Result at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Attrited Customer' 'Attrited Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      1700
           1       0.96      0.97      0.96      1700

    accuracy                           0.96      3400
   macro avg       0.96      0.96      0.96      3400
weighted avg       0.96      0.96      0.96      3400
# Result at Fold 4
Predict dummy inputs at Fold 4: ['Existing Customer' 'Attrited Customer' 'Attrited
```

```
    Customer' 'Existing Customer' 'Attrited Customer']
33              precision    recall  f1-score   support
34
35           0       0.96      0.96      0.96      1736
36           1       0.96      0.96      0.96      1664
37
38    accuracy                           0.96      3400
39   macro avg       0.96      0.96      0.96      3400
40 weighted avg       0.96      0.96      0.96      3400
41 # Result at Fold 5
42 Predict dummy inputs at Fold 5: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
43              precision    recall  f1-score   support
44
45           0       0.97      0.96      0.96      1744
46           1       0.95      0.97      0.96      1656
47
48    accuracy                           0.96      3400
49   macro avg       0.96      0.96      0.96      3400
50 weighted avg       0.96      0.96      0.96      3400
```



Hình 4.7: Đánh giá Adaptive Boosting qua confusion matrix có sử dụng SMOTE

4.2.4 Random Forest

Không sử dụng SMOTE

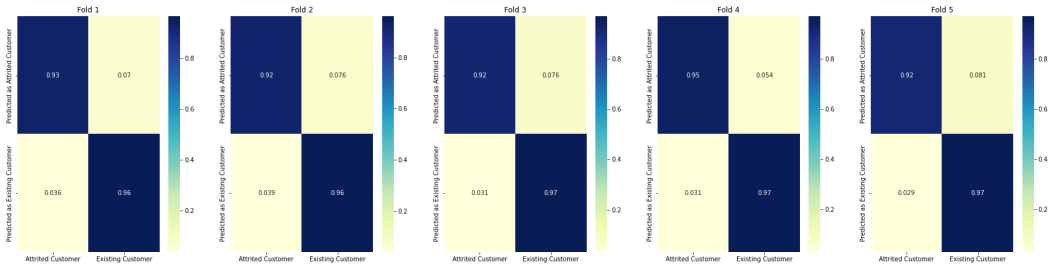
```
1  rfc = RandomForestClassifier(random_state=42)
2  ax = []
3  fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
4  count = 0
5  for train_index , test_index in kf.split(X):
6      X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
7      y_train , y_test = y[train_index] , y[test_index]
8
9      rfc.fit(X_train,y_train)
10     y_pred = rfc.predict(X_test)
11     cfmatrix = confusion_matrix(y_pred, y_test)
```

```
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_rfc = rfc.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
    inverse_transform(dummy_outputs_rfc)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của Random Forest không sử dụng SMOTE

```
# Result at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.81      0.93      0.86       284
           1       0.99      0.96      0.98      1742

    accuracy                           0.96      2026
   macro avg       0.90      0.95      0.92      2026
weighted avg       0.96      0.96      0.96      2026
# Result at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.80      0.92      0.86       288
           1       0.99      0.96      0.97      1738

    accuracy                           0.96      2026
   macro avg       0.89      0.94      0.91      2026
weighted avg       0.96      0.96      0.96      2026
# Result at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.82      0.92      0.87       276
           1       0.99      0.97      0.98      1749

    accuracy                           0.96      2025
   macro avg       0.91      0.95      0.92      2025
weighted avg       0.97      0.96      0.96      2025
# Result at Fold 4
```

```
Predict dummy inputs at Fold 4: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.84      0.95      0.89       295
           1       0.99      0.97      0.98      1730

    accuracy                           0.97      2025
   macro avg       0.91      0.96      0.93      2025
weighted avg       0.97      0.97      0.97      2025
# Result at Fold 5
Predict dummy inputs at Fold 5: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Existing Customer']
              precision    recall  f1-score   support

           0       0.84      0.92      0.88       296
           1       0.99      0.97      0.98      1729

    accuracy                           0.96      2025
   macro avg       0.91      0.94      0.93      2025
weighted avg       0.96      0.96      0.96      2025
```



Hình 4.8: Đánh giá Random Forest qua confusion matrix không sử dụng SMOTE

Có sử dụng SMOTE

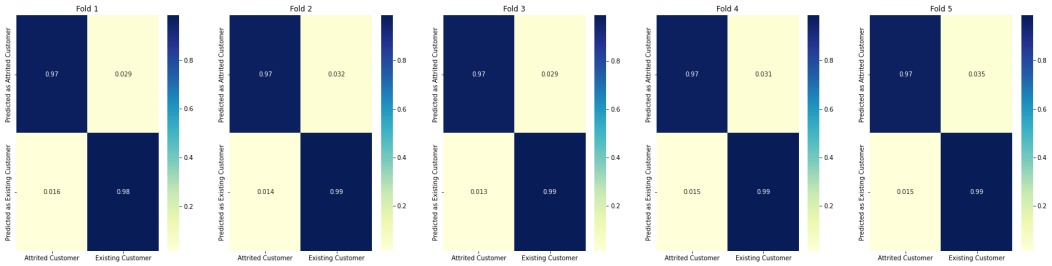
```
oversample = SMOTE()
X, y = oversample.fit_resample(X,y)

rfc_smote = RandomForestClassifier(random_state=42)
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    rfc_smote.fit(X_train,y_train)
```

```
    y_pred = rfc_smote.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_rfc_smote = rfc_smote.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.inverse_transform(dummy_outputs_rfc_smote)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của Random Forest có sử dụng SMOTE

```
# Result at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Attrited Customer' 'Existing Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1747
           1       0.97      0.98      0.98      1653

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
# Result at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Attrited Customer' 'Existing Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1665
           1       0.97      0.99      0.98      1735

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
# Result at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Attrited Customer' 'Existing Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1717
           1       0.97      0.99      0.98      1683

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
```

```
weighted avg       0.98      0.98      0.98      3400
# Result at Fold 4
Predict dummy inputs at Fold 4: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1760
           1       0.97      0.99      0.98      1640

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
# Result at Fold 5
Predict dummy inputs at Fold 5: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1757
           1       0.96      0.99      0.97      1643

    accuracy                           0.97      3400
   macro avg       0.97      0.98      0.97      3400
weighted avg       0.98      0.97      0.98      3400
```



Hình 4.9: Đánh giá Random Forest qua confusion matrix có sử dụng SMOTE

4.2.5 Gradient Boosting

Không sử dụng SMOTE

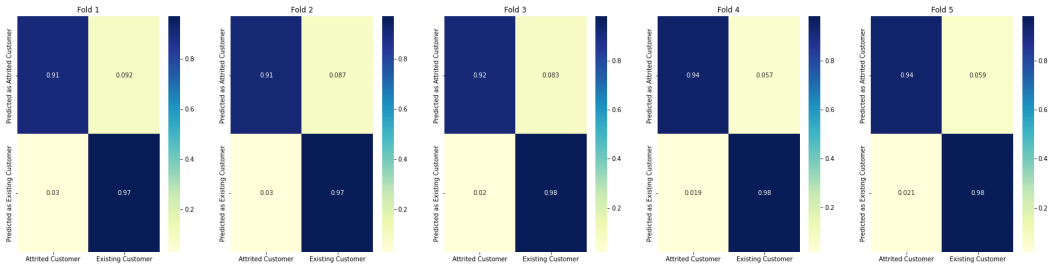
```
gbc = GradientBoostingClassifier(n_estimators=100, learning_rate=0.5, max_depth=5,
    random_state=42)
ax = []
fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
count = 0
for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]
```

```
8
9       gbc.fit(X_train,y_train)
10      y_pred = gbc.predict(X_test)
11      cfmatrix = confusion_matrix(y_pred, y_test)
12      normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
13      df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
        Customer", "Predicted as Existing Customer"],
14                           columns = ["Attrited Customer", "Existing Customer"])
15
16      ax[count].set_title("Fold "+ str(count + 1))
17      sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
18
19      dummy_outputs_gbc = gbc.predict(dummy_inputs)
20      print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
        inverse_transform(dummy_outputs_gbc)))
21      print(classification_report(y_pred, y_test))
22      print("*"*60)
23      count += 1
```

Kết quả của Gradient Boosting không sử dụng SMOTE

```
1  # Result at Fold 1
2  Predict dummy inputs at Fold 1: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Attrited Customer']
3                  precision    recall  f1-score   support
4
5               0       0.84      0.91      0.87       304
6               1       0.98      0.97      0.98      1722
7
8       accuracy                           0.96      2026
9      macro avg       0.91      0.94      0.93      2026
10 weighted avg       0.96      0.96      0.96      2026
11 # Result at Fold 2
12 Predict dummy inputs at Fold 2: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Existing Customer']
13                 precision    recall  f1-score   support
14
15              0       0.85      0.91      0.88       310
16              1       0.98      0.97      0.98      1716
17
18      accuracy                           0.96      2026
19     macro avg       0.92      0.94      0.93      2026
20 weighted avg       0.96      0.96      0.96      2026
21 # Result at Fold 3
22 Predict dummy inputs at Fold 3: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Existing Customer']
23                 precision    recall  f1-score   support
24
25              0       0.89      0.92      0.90       300
26              1       0.99      0.98      0.98      1725
```

```
27
28      accuracy                           0.97      2025
29     macro avg       0.94      0.95      0.94      2025
30  weighted avg       0.97      0.97      0.97      2025
31  # Result at Fold 4
32  Predict dummy inputs at Fold 4: ['Existing Customer' 'Attrited Customer' 'Existing
        Customer' 'Existing Customer' 'Existing Customer']
33                  precision    recall  f1-score   support
34
35               0       0.90      0.94      0.92       318
36               1       0.99      0.98      0.98      1707
37
38      accuracy                           0.97      2025
39     macro avg       0.95      0.96      0.95      2025
40  weighted avg       0.98      0.97      0.98      2025
41  # Result at Fold 5
42  Predict dummy inputs at Fold 5: ['Existing Customer' 'Attrited Customer' 'Existing
        Customer' 'Existing Customer' 'Existing Customer']
43                  precision    recall  f1-score   support
44
45               0       0.89      0.94      0.91       305
46               1       0.99      0.98      0.98      1720
47
48      accuracy                           0.97      2025
49     macro avg       0.94      0.96      0.95      2025
50  weighted avg       0.97      0.97      0.97      2025
```



Hình 4.10: Đánh giá Gradient Boosting qua confusion matrix không sử dụng SMOTE

Có sử dụng SMOTE

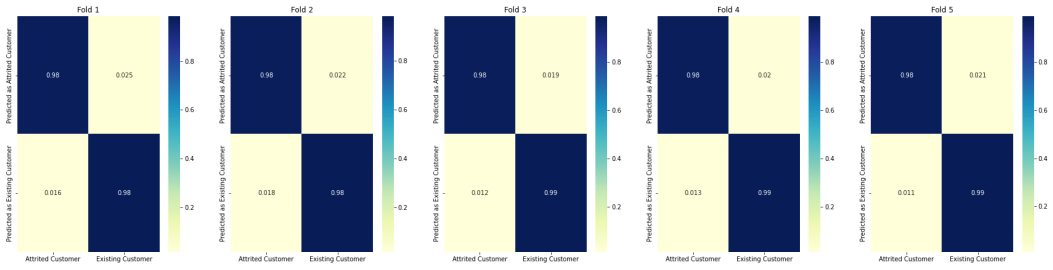
```
1  gbc_smote = GradientBoostingClassifier(n_estimators=100, learning_rate=0.5, max_depth
       =5, random_state=42)
2  ax = []
3  fig, (ax) = plt.subplots(1, 5, figsize=(30,7))
4  count = 0
5  for train_index , test_index in kf.split(X):
6      X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
```

```
    y_train , y_test = y[train_index] , y[test_index]

    gbc_smote.fit(X_train,y_train)
    y_pred = gbc_smote.predict(X_test)
    cfmatrix = confusion_matrix(y_pred, y_test)
    normalize_confusion_matrix = cfmatrix / cfmatrix.sum(axis = 1, keepdims = True)
    df_cm = pd.DataFrame(normalize_confusion_matrix, index = ["Predicted as Attrited
    Customer", "Predicted as Existing Customer"],
                         columns = ["Attrited Customer", "Existing Customer"])

    ax[count].set_title("Fold "+ str(count + 1))
    sn.heatmap(df_cm, cmap="YlGnBu", annot=True, ax=ax[count])
    dummy_outputs_gbc_smote = gbc_smote.predict(dummy_inputs)
    print("Predict dummy inputs at Fold "+str(count + 1)+": "+str(le.
    inverse_transform(dummy_outputs_gbc_smote)))
    print(classification_report(y_pred, y_test))
    print("*"*60)
    count += 1
```

Kết quả của Gradient Boosting có sử dụng SMOTE

```
# Result at Fold 1
Predict dummy inputs at Fold 1: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1740
           1       0.97      0.98      0.98      1660

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
# Result at Fold 2
Predict dummy inputs at Fold 2: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1641
           1       0.98      0.98      0.98      1759

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
# Result at Fold 3
Predict dummy inputs at Fold 3: ['Existing Customer' 'Attrited Customer' 'Existing
    Customer' 'Existing Customer' 'Attrited Customer']
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1703
           1       0.98      0.99      0.98      1697
```

```
27
28     accuracy                           0.98      3400
29    macro avg       0.98      0.98      0.98      3400
30 weighted avg       0.98      0.98      0.98      3400
31 # Result at Fold 4
32 Predict dummy inputs at Fold 4: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Existing Customer']
33                precision    recall  f1-score   support
34
35             0       0.99      0.98      0.98      1744
36             1       0.98      0.99      0.98      1656
37
38     accuracy                           0.98      3400
39    macro avg       0.98      0.98      0.98      3400
40 weighted avg       0.98      0.98      0.98      3400
41 # Result at Fold 5
42 Predict dummy inputs at Fold 5: ['Existing Customer' 'Attrited Customer' 'Existing
       Customer' 'Existing Customer' 'Existing Customer']
43                precision    recall  f1-score   support
44
45             0       0.99      0.98      0.98      1739
46             1       0.98      0.99      0.98      1661
47
48     accuracy                           0.98      3400
49    macro avg       0.98      0.98      0.98      3400
50 weighted avg       0.98      0.98      0.98      3400
```



Hình 4.11: Đánh giá Gradient Boosting qua confusion matrix có sử dụng SMOTE

Chương 5

KẾT LUẬN

Trong quá trình thực hiện đề tài "Dự đoán tập khách hàng ngưng sử dụng dịch vụ ngân hàng" nhóm chúng em đã thu được các kết quả sau:

• Thêm hiểu biết về ngân hàng và các vấn đề mà các ngân hàng đăng gặp phải

• Hiểu được việc rời đi của khách hàng có ảnh hưởng như thế nào đến ngân hàng

• Biết được các nhóm khách hàng nào có nguy cơ rồi đi nhiều nhất.

• Hiểu rõ hơn về tính ứng dụng của học máy để dự đoán và các thuật toán học máy được áp dụng trong đề tài

Tuy nhiên vẫn còn một số khuyết điểm sau:

• Vẫn còn lúng túng trong việc trực quan hóa dữ liệu trong tableau

• Chưa hiểu sâu về kiến trúc của các thuật toắn

Trong tương lai nhóm chúng em sẽ tìm cách tối ưu các mô hình đang sử dụng, cố gắng thực nghiệm dữ liệu trên các mô hình mới và phát triển thành ứng dụng triển khai được trong thực tế để có thể dự đoán được đúng thời gian khách hàng rời đi để có các chiến lược giữ lại khách hàng.

Tài liệu tham khảo

[1] Bui Tien, T. (2021, May 28). *Gradient Boosting - Tất tần tật về thuật toán mạnh mẽ nhất trong Machine Learning*. VIBLO ASIA. Retrieved December 9, 2021, from https://viblo.asia/p/gradient-boosting-tat-tan-tat-ve-thuat-toan-manh-me-nhat-trong-machine-learning-YWOZrN7vZQ0

[2] Cao Minh, H. (2021, July 2). *AdaBoost - Bước đi đầu của Boosting.* VIBLO ASIA. Retrieved December 9, 2021, from https://viblo.asia/p/adaboost-buoc-di-dau-cua-boosting-gAm5yrGwKdb

[3] CHOUDHURY, A. M. B. I. K. A. (2021, January 18). *AdaBoost Vs Gradient Boosting: A Comparison Of Leading Boosting Algorithms.* Analytics In Diamag. Retrieved December 9, 2021, from https://analyticsindiamag.com/adaboost-vs-gradient-boosting-a-comparison-of-leading-boosting-algorithms/

[4] Chowdhury, M. (2021, June 26). *Churn Analytics: Data Analysis to Machine learning.* Medium. Retrieved December 6, 2021, from https://medium.com/@mchowdhuryca/churn-analytics-from-data-analysis-to-machine-learning-95854d102ed6

[5] DHANYA, S. H. R. E. E. (2021, June 1). *AdaBoost : A Brief Introduction to Ensemble learning.* Analytics Vidhya. Retrieved December 10, 2021, from https://www.analyticsvidhya.com/blog/2021/06/adaboost-a-brief-introduction-to-ensemble-learning/

[6] Konstantin, T. (2021, May 1). *Bank Churn Data Exploration And Churn Prediction.* Kaggle. Retrieved November 5, 2021, from https://www.kaggle.com/thomaskonstantin/bank-churn-data-exploration-and-churn-prediction

[7] Kumar, N. (2019, February 23). *Advantages and Disadvantages of Random Forest Algorithm in Machine Learning.* The Professionals Point. Retrieved December 11, 2021, from

https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html

[8] Microsoft. (2021, August 23). *SMOTE.* Retrieved December 7, 2021, from https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/smote

[9] Pham Minh, H. (2021, January 17). *Ensemble learning và các biến thể (P1).* VIBLO ASIA. Retrieved December 9, 2021, from https://viblo.asia/p/ensemble-learning-va-cac-bien-the-p1-WAyK80AkKxX

[10] Pham Minh, P. (2021, May 20). *Giải thuật Adaboost ứng dụng trong nhận dạng biển số xe(Data Mining).* VIBLO ASIA. Retrieved December 9, 2021, from https://viblo.asia/p/giai-thuat-adaboost-ung-dung-trong-nhan-dang-bien-so-xedata-mining-07LKXxbPKV4

[11] R. (2021, November 15). *Know your customers to know your churners.* Kaggle. Retrieved November 6, 2021, from https://www.kaggle.com/renmellofrey/know-your-customers-to-know-your-churners

[12] *Random forest Algorithm in Machine learning: An Overview.* (2020, February 19). Great Learning. Retrieved December 10, 2021, from https://www.mygreatlearning.com/blog/random-forest-algorithm/

[13] Tuong, H. (2020, August 15). *Thuật Toán K-Nearest Neighbors (KNN) Siêu Cơ Bản.* CODELEARN. Retrieved December 7, 2021, from https://codelearn.io/sharing/thuat-toan-k-nearest-neighbors-knn

[14] Vu Huu, T. (2017a, January . *K-nearest neighbors.* Machine Learning Cơ Bản. Retrieved December 7, 2021, from https://machinelearningcoban.com/2017/01/08/knn/

[15] Vu Huu, T. (2017b, January 27). *Logistic Regression.* Machine Learning Cơ Bản. Retrieved December 8, 2021, from https://machinelearningcoban.com/2017/01/27/logisticregression/

[16] Vu Huu, T. (2018, January 3). *Các phương pháp đánh giá một hệ thống phân lớp.* Machine Learning Cơ Bản. Retrieved December 11, 2021, from https://machinelearningcoban.com/2017/08/31/evaluation/