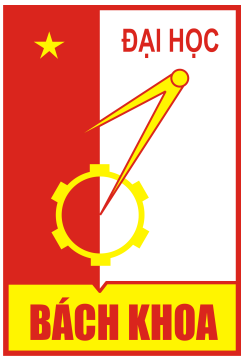
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO HỆ HỖ TRỢ QUYẾT ĐỊNH

ĐỀ TÀI: CUSTOMER CHURNING

Giảng viên hướng dẫn: TS. Lê Hải Hà
Sinh viên thực hiện: Phạm Thị Ngọc Anh
MSSV: 20206224
Mã lớp: 142314

HÀ NỘI – 2023

Lời cảm ơn

Trong suốt quá trình tìm hiểu và nghiên cứu đề tài này, em xin cảm ơn TS. Lê Hải Hà đã tận tình hướng dẫn để giúp em tiếp thu những kiến thức cần thiết để có thể hoàn thiện đề tài. Có thể nó sẽ còn nhiều thiếu sót, em mong thầy cô sẽ góp ý để báo cáo được hoàn thiện hơn. Em xin chân thành cảm ơn!

Hà Nội, ngày 30 tháng 7 năm 2023

Tác giả báo cáo

Phạm Thị Ngọc Anh

Mục lục

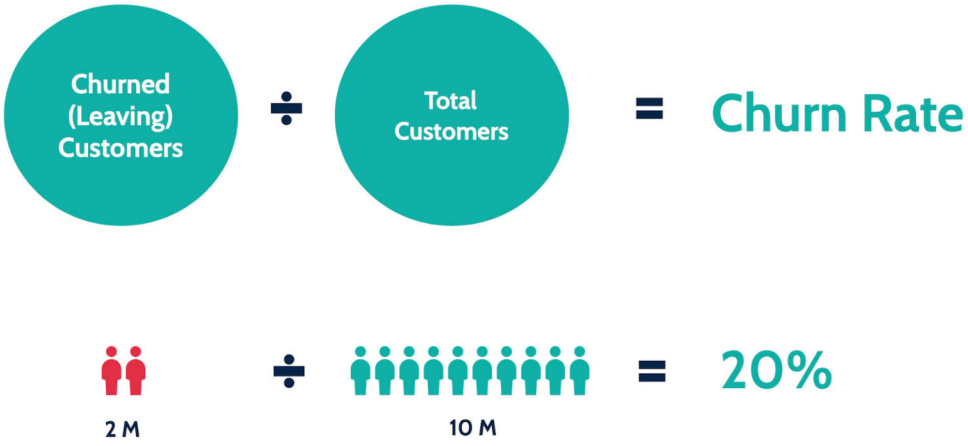
Chương 1

Tổng quan về bài toán

1.1 Mô tả bài toán

Customer Churn hiện đang là một thách thức quan trọng mà các doanh nghiệp phải đối mặt trong nhiều ngành công nghiệp. Trong thị trường cạnh tranh gay gắt ngày nay, việc giữ chân khách hàng hiện tại trở nên vô cùng quan trọng để duy trì sự thành công và phát triển bền vững.

Customer Churn được hiểu đơn giản là khi một khách hàng quyết định ngừng sử dụng một dịch vụ hay kết thúc hợp đồng (không muốn gia hạn) với một công ty nào đó. Customer Churn Rate (Tỷ lệ khách hàng rời bỏ) được tính theo công thức:



Hình 1.1 Công thức tính Tỷ lệ khách hàng rời bỏ

Hiện nay, có rất nhiều ngành nghề cần quan tâm đến chỉ số Customer Churn Rate. Trong đó có những ngành tiêu biểu như:

- Các công ty thu phí của người dùng theo tháng và doanh thu đến từ việc khách hàng tiếp tục duy trì sử dụng dịch vụ ngành viễn thông với các dịch vụ: mạng Internet, cước điện thoại hay truyền hình cáp...
- Các công ty về công nghệ cung cấp các dịch vụ như: hosting services, điện toán đám mây,...
- Game và các ứng dụng điện thoại với tỷ lệ rời bỏ được đánh giá dựa trên lượt tải xuống mới, có kích hoạt hay không kích hoạt.

Nguồn thu chính từ những công ty này đến từ phí sử dụng hàng tháng. Nếu lượng khách hàng hủy hợp đồng hay ngừng sử dụng dịch vụ hàng tháng cao hơn lượng khách hàng mới, đây chính là dấu hiệu của việc kinh doanh có vấn đề. Hiểu rõ nguyên nhân dẫn đến việc mất mát khách hàng và xây dựng các chiến lược hiệu quả để giảm thiểu tình trạng này đã trở thành yếu tố quan trọng đối với các doanh nghiệp với mong muốn tồn tại và phát triển trong thị trường kinh doanh biến đổi liên tục.

Việc khách hàng quyết định rời bỏ doanh nghiệp hoặc ngừng sử dụng sản phẩm/dịch vụ của họ phụ thuộc vào nhiều yếu tố khách nhau. Dưới đây là một số nguyên nhân phổ biến khiến khách hàng từ bỏ:

- Không hài lòng với chất lượng sản phẩm/dịch vụ: Khách hàng thường mong đợi nhận được sản phẩm hoặc dịch vụ chất lượng cao và nếu sản phẩm không đáp ứng được nhu cầu hoặc có vấn đề về chất lượng, họ có thể chọn chuyển sang đối thủ cạnh tranh.
- Giá cả không hợp lý: Nếu giá cả của sản phẩm/dịch vụ quá cao hoặc không tương xứng với giá trị mà khách hàng nhận được, họ có thể tìm kiếm giải pháp tốt hơn từ công ty khác.
- Cạnh tranh mạnh: Thị trường cạnh tranh có thể khiến khách hàng có nhiều lựa chọn hơn. Nếu đối thủ cung cấp các giải pháp tốt hơn hoặc giá cả cạnh tranh hơn, khách hàng có thể quyết định chuyển sang công ty khác.
- Thay đổi nhu cầu hoặc tình huống cá nhân: Có thể có sự thay đổi trong nhu cầu hoặc tình huống cá nhân của khách hàng khiến họ không còn cần sản

phẩm/dịch vụ của doanh nghiệp nữa.

- Vấn đề về thương hiệu: Nếu doanh nghiệp gặp vấn đề về uy tín hoặc danh tiếng trong mắt khách hàng, họ có thể không còn tin tưởng và chọn tìm kiếm nhà cung cấp khác.
- Không có tương tác và gắn kết: Nếu doanh nghiệp không duy trì tương tác và gắn kết với khách hàng của mình, họ có thể cảm thấy bị bỏ rơi và dễ dàng quyết định rời bỏ.

Để giảm thiểu Customer Churn, doanh nghiệp cần tập trung vào việc cải thiện chất lượng sản phẩm/dịch vụ, xây dựng mối quan hệ tốt hơn với khách hàng, và đảm bảo rằng họ đáp ứng đủ các nhu cầu và yêu cầu của khách hàng.

Dựa trên các công cụ và thuật toán để lập mô hình sau đó đưa ra dự đoán về sự rời bỏ của khách hàng ngân hàng, em hy vọng báo cáo này sẽ giúp cho doanh nghiệp chủ động giải quyết tình trạng mất mát khách hàng, dẫn đến nâng cao sự hài lòng của khách hàng, cải thiện tỷ lệ giữ chân khách hàng và cuối cùng là gia tăng lợi nhuận.

1.2 Thư viện và thuật toán

Với bài toán Customer Churn, em sẽ sử dụng các thư viện của Python để phân tích và xây dựng mô hình. Cụ thể:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

Hình 1.2 Import các thư viện sử dụng

- **Pandas** là một thư viện mã nguồn mở trong ngôn ngữ lập trình Python được sử dụng rộng rãi trong xử lý và phân tích dữ liệu. Thư viện này cung cấp các cấu trúc dữ liệu và công cụ cho việc làm việc với dữ liệu dạng bảng (như

DataFrame) và chuỗi dữ liệu (như Series). Pandas cho phép người lập trình thực hiện các thao tác mạnh mẽ và linh hoạt trên dữ liệu, giúp dễ dàng tạo, biến đổi, lọc và tổng hợp thông tin từ các bộ dữ liệu lớn.

- **Seaborn** là một thư viện mã nguồn mở trong ngôn ngữ lập trình Python, được sử dụng để trực quan hóa dữ liệu và tạo các biểu đồ đẹp mắt và thông qua nó có thể dễ dàng thực hiện các tùy chỉnh và tạo ra các biểu đồ chuyên nghiệp. Seaborn được xây dựng dựa trên thư viện Matplotlib, giúp tăng tính tương thích và cải thiện khả năng vẽ đồ thị trong Python.
- **Scikit-learn** là một thư viện cung cấp nhiều công cụ, thuật toán và hàm để thực hiện các tác vụ liên quan đến học máy như phân loại, hồi quy, gom cụm (clustering), giảm chiều dữ liệu (dimensionality reduction), và phân tích dữ liệu.

Chương 2

Xử lý dữ liệu

2.1 Tổng quan bộ dữ liệu

Báo cáo này sử dụng bộ dữ liệu được cung cấp miễn phí trên Kaggle với 10000 hồ sơ khách hàng, 14 trường dữ liệu:

```
[2]: df = pd.read_csv('Churn_Modelling.csv')
[3]: df.sample(10)
```

[3]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
2477	15617136	Mazzanti	451	Germany	Female	38	9	61482.47	1	1	1	167538.66	0
158	15623595	Clayton	586	Spain	Female	28	2	0.00	2	1	1	92067.35	0
1413	15588986	Grant	673	Germany	Female	29	4	99097.36	1	1	1	9796.69	0
6656	15703763	Sanderson	554	France	Male	44	7	85304.27	1	1	1	58076.52	0
3580	15670943	See	778	Germany	Male	31	9	182275.23	2	1	0	190631.23	0
542	15626578	Milne	622	France	Male	26	9	0.00	2	1	1	153237.59	0
9105	15683339	P'eng	656	Spain	Female	34	6	59877.33	1	1	0	14032.62	1
8688	15724764	Lawley	667	Germany	Female	42	10	64404.26	2	0	0	26022.37	0
6039	15700964	Pollard	624	Germany	Female	27	7	104848.68	1	1	1	167387.36	0
7188	15662152	Trevisan	552	France	Female	38	9	134105.01	1	0	0	57850.10	0

Hình 2.1 Bộ dữ liệu ban đầu

1. RowNumber: Số thứ tự
2. CustomerId: Mã số khách hàng
3. Surname: Tên khách hàng
4. CreditScore: Điểm tín dụng
5. Geography: Quốc gia nơi cư trú
6. Gender: Giới tính

7. Age: Tuổi
8. Tenure: Thời gian khách hàng sử dụng dịch vụ của ngân hàng
9. Balance: Số dư tài khoản
10. NumOfProducts: Số lượng dịch vụ của ngân hàng mà khách hàng đã sử dụng.
11. HasCrCard: Khách hàng đã có thẻ tín dụng (Credit card) hay chưa với 1 là đã có, 0 là chưa có.
12. IsActiveMember: Khách hàng có còn hoạt động hay không với 1 là còn, 0 là không.
13. EstimatedSalary: Mức lương ước tính
14. Exited: Khách hàng có rời bỏ hay không với 1 là có rời bỏ, 0 là không rời bỏ

Tính tỷ lệ Customer Churn:

```
df['Exited'].value_counts().to_frame()
```

Exited	count
0	7963
1	2037

```
format(df['Exited'].mean())
```

```
'0.2037'
```

Hình 2.2 Tỷ lệ khách hàng rời bỏ

Như vậy, trong tổng số 10000 khách hàng có 7963 khách hàng không rời bỏ và 2037 khách hàng rời bỏ. Tỷ lệ Customer Churn của ngân hàng này là 20,37%.

→ Ngân hàng hoạt động bình thường, không có dấu hiệu đáng báo động.

2.2 Tiền xử lý dữ liệu

Loại bỏ dữ liệu không cần thiết

Nhận thấy thuộc tính 'RowNumber', 'CustomerId', 'Surname' không gây ảnh hưởng đến quyết định rời bỏ dịch vụ của khách hàng nên ta sẽ tiến hành xoá 3 thuộc tính khỏi bộ dữ liệu.

```
[5]: df.drop(columns = ['RowNumber','CustomerId', 'Surname'], inplace = True)
```

```
[6]: df.sample(10)
```

[6]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
6750	618	France	Female	42	2	0.00	4	0	0	111097.39	1
252	681	France	Female	79	0	0.00	2	0	1	170968.99	0
9497	684	France	Female	25	1	0.00	2	0	1	144978.47	0
621	656	Spain	Female	40	10	167878.50	1	0	1	151887.16	0
4926	674	Germany	Female	36	6	100762.64	1	1	0	182156.86	0
8824	643	Spain	Female	35	6	0.00	2	1	1	41549.64	0
6815	606	Spain	Male	36	0	94153.56	1	0	1	120138.27	0
7562	685	France	Male	33	6	0.00	1	1	0	58458.26	0
4791	709	Spain	Male	35	2	0.00	2	1	0	104982.39	0
3564	847	France	Male	51	5	97565.74	1	0	0	144184.06	1

Hình 2.3 Bộ dữ liệu sau khi xoá

Tiếp theo ta sẽ quan sát đặc trưng thống kê của bộ dữ liệu, bao gồm: giá trị trung bình, min, max,...

```
[4]: df.describe()
```

[4]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exit
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.0000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.2037
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.4027
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.0000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.0000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.0000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.0000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.0000

Hình 2.4 Mô tả của bộ dữ liệu

Nhìn chung, bộ dữ liệu không có sự bất thường.

Kiểm tra trùng lặp

Sử dụng hàm duplicated() để kiểm tra số lượng bản ghi bị trùng lặp có trong bộ dữ liệu.

```
df.duplicated().sum()
```

```
0
```

Hình 2.5 Số bản ghi trùng lặp

Kiểm tra số giá trị khác nhau của thuộc tính

Ta tiếp tục quan sát tổng số giá trị unique của từng cột

```
for col in df.columns:
    print(col, ':', len(df[col].value_counts()))
```

```
CreditScore : 460
Geography : 3
Gender : 2
Age : 70
Tenure : 11
Balance : 6382
NumOfProducts : 4
HasCrCard : 2
IsActiveMember : 2
EstimatedSalary : 9999
Exited : 2
```

Hình 2.6 Số lượng giá trị unique

Chú ý tới một số thuộc tính như Geography chỉ có 3 giá trị dữ liệu khác nhau là France, Spain và Gemany. Gender chỉ có 2 giá trị là Male và Female. HasCrCard, IsActiveMember và Exited đều chỉ có 2 giá trị là 1 và 0.

Kiểm tra dữ liệu bị thiếu

Thực hiện tính phần trăm giá trị bị thiếu (null) theo từng cột. Quan sát kết quả ta thấy dữ liệu không bị thiếu nên không cần phải xử lý.

```
[14]: percent = df.isnull().sum() *100 / len(df)
      null_val = pd.DataFrame({'percent' : percent})
      null_val.sort_values('percent', ascending = False, inplace = True)
      print(mis_val)
```

	percent
CreditScore	0.0
Geography	0.0
Gender	0.0
Age	0.0
Tenure	0.0
Balance	0.0
NumOfProducts	0.0
HasCrCard	0.0
IsActiveMember	0.0
EstimatedSalary	0.0
Exited	0.0

Hình 2.7 Phần trăm giá trị null từng thuộc tính

2.3 Xử lý dữ liệu

Mã hoá One-hot

Mã hóa one-hot là một cách biến đổi nhanh chóng từ dữ liệu dạng hạng mục sang dạng số. Với cách mã hóa này, ta có thể xây dựng nhanh chóng các mô hình đơn giản như hồi quy tuyến tính hay SVM, các mô hình này bắt buộc giá trị đầu vào là ở dạng số.

Ở đây chúng ta cần phải chuyển đổi các thuộc tính "Geography" và "Gender" thành dạng nhị phân. Như đã tìm hiểu ở trên, thuộc tính "Geography" gồm 3 giá trị và thuộc tính "Gender" gồm 2 giá trị.

Sau khi sử dụng kỹ thuật OHE, tập dữ liệu sẽ thay đổi 2 cột "Geography" và "Gender". Trong đó:

- Cột "Geography", dữ liệu có thuộc tính “France” có giá trị bằng “0”, dữ liệu có thuộc tính ““Germany” có giá trị bằng “1” và dữ liệu có thuộc tính “Spain” có giá trị bằng “2”.
- Cột "Gender", dữ liệu có thuộc tính "Male" có giá trị bằng “1”, “Female” có giá trị bằng “0”.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Gender']= le.fit_transform(df['Gender'])
df['Geography']= le.fit_transform(df['Geography'])
df.head(10)
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0
5	645	2	1	44	8	113755.78	2	1	0	149756.71	1
6	822	0	1	50	7	0.00	2	1	1	10062.80	0
7	376	1	0	29	4	115046.74	4	1	0	119346.88	1
8	501	0	1	44	4	142051.07	2	0	1	74940.50	0
9	684	0	1	27	2	134603.88	1	1	1	71725.73	0

Hình 2.8 Bộ dữ liệu sau khi thực hiện OHE

Split dataset

Ta sẽ chia bộ dữ liệu thành hai phần: tập huấn luyện (training set) và tập kiểm tra (test set) với tỷ lệ tập kiểm tra 20% và tập huấn luyện 80%. Ta chia dữ liệu thành 2 phần: features và decided với features bao gồm các cột thuộc tính trừ cột "Exited".

```
from sklearn.model_selection import train_test_split
# Tách cột "Exited" ra khỏi bộ dữ liệu và lưu vào biến decided
decided = df['Exited']
# Loại bỏ cột "Exited" khỏi DataFrame để có tập dữ liệu huấn luyện
features_df = df.drop(columns=['Exited'])
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
train_df, test_df, train_decided, test_decided = train_test_split(features_df, decided, test_size=0.2, random_state=42)
# Tiếp tục chia dữ liệu tập huấn luyện thành tập huấn luyện và tập validation
train_df, valid_df, train_decided, valid_decided = train_test_split(train_df, train_decided, test_size=0.25, random_state=42)

print("Số dòng dữ liệu trong tập huấn luyện:", len(train_df))
print("Số dòng dữ liệu trong tập validation:", len(valid_df))
print("Số dòng dữ liệu trong tập kiểm tra:", len(test_df))
```

```
Số dòng dữ liệu trong tập huấn luyện: 6000
Số dòng dữ liệu trong tập validation: 2000
Số dòng dữ liệu trong tập kiểm tra: 2000
```

Hình 2.9 Kết quả khi thực hiện Split dataset

Tiếp theo, ta chia tập huấn luyện thành 2 phần. Như vậy ta có mẫu test chiếm 20%, mẫu validation chiếm là 20% và training chiếm 60%

Chuẩn hoá dữ liệu

Với bộ dữ liệu này ta cần thực hiện chuẩn hoá các thuộc tính "CreditScore", "Age", "Balance", "EstimatedSalary".

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
train_df[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']] = scaler.fit_transform(train_df[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']])
train_df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
8588	0.626553	2	0	-0.948125	7	0.026803	1	1	0	0.389943
3178	-1.143262	0	0	0.006684	4	0.538874	2	1	1	-1.026089
5200	-1.455583	1	1	0.293126	9	0.283178	2	1	0	-1.486725
8889	-0.747657	0	0	0.006684	9	0.833254	1	1	0	-0.246001
5789	0.387107	1	0	1.534377	1	0.000856	1	1	0	-1.006993

Hình 2.10 Kết quả chuẩn hoá trên tập Train

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
valid_df[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']] = scaler.fit_transform(valid_df[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']])
valid_df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
3188	1.540684	0	0	-0.638499	4	-0.008173	1	1	1	-1.094645
8293	1.180852	1	0	-1.014031	4	0.241532	1	0	1	-1.376405
1710	1.818270	2	1	-1.014031	8	0.950302	1	1	0	-0.378075
7510	1.509842	0	1	-0.262967	1	-1.217124	2	1	1	0.999297
1461	-0.412692	2	1	-0.450733	4	-1.217124	2	1	0	1.205094

Hình 2.11 Kết quả chuẩn hoá trên tập Valid

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
test_df[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']] = scaler.fit_transform(test_df[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']])
test_df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
6252	-0.512502	1	1	-0.677299	3	0.300846	2	0	0	-0.990969
4684	-0.236047	0	1	0.384298	1	-1.255997	2	1	1	0.829379
1731	-0.461306	2	0	0.480807	4	-1.255997	2	1	0	-0.699045
4742	-1.434019	1	1	1.928440	8	0.662139	2	1	1	1.252315
4521	-0.881109	2	0	-1.159844	7	0.756215	1	1	1	0.277492

Hình 2.12 Kết quả chuẩn hoá trên tập Test

Chương 3

Xây dựng và đánh giá mô hình dự báo

3.1 Logistic regression

Logistic regression là một kỹ thuật phân tích dữ liệu sử dụng toán học để tìm ra mối quan hệ giữa hai yếu tố dữ liệu. Sau đó, kỹ thuật này sử dụng mối quan hệ đã tìm được để dự đoán giá trị của những yếu tố đó dựa trên yếu tố còn lại. Dự đoán thường cho ra một số kết quả hữu hạn, như có hoặc không.

Hồi quy Logistic sử dụng hàm sigmoid (hàm logistic) để chuyển đổi kết quả từ đầu ra của mô hình về xác suất trong khoảng [0, 1]. Kết quả sau khi xử lý bởi hàm sigmoid thể hiện xác suất rơi vào một nhãn cụ thể. Nếu xác suất lớn hơn một ngưỡng (thường là 0.5), quan sát được dự đoán thuộc về lớp tích cực (positive class); nếu xác suất nhỏ hơn ngưỡng, quan sát được dự đoán thuộc về lớp tiêu cực (negative class)

Xây dựng mô hình Đầu tiên, ta xây dựng mô hình Logistic Regression trên tập huấn luyện, sau đó đánh giá mô hình trên tập kiểm tra

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Xây dựng mô hình Logistic Regression
model = LogisticRegression()
model.fit(train_df, train_decided)
```

Hình 3.1 Xây dựng mô hình Logistic Regression

Đánh giá mô hình Chúng ta sẽ sử dụng các độ đo thông thường như độ chính xác (accuracy), độ phân loại chính xác (precision), độ phủ (recall), và F1-score.

```python
# Dự đoán trên tập kiểm tra
predicted_decided = model.predict(test_df)

# Đánh giá mô hình bằng các độ đo thông thường
accuracy = accuracy_score(test_decided, predicted_decided)
precision = precision_score(test_decided, predicted_decided)
recall = recall_score(test_decided, predicted_decided)
f1 = f1_score(test_decided, predicted_decided)

# In kết quả đánh giá mô hình
print("Độ chính xác (Accuracy):", accuracy)
print("Độ phân loại chính xác (Precision):", precision)
print("Độ phủ (Recall):", recall)
print("F1-score:", f1)
```

```
Độ chính xác (Accuracy): 0.814
Độ phân loại chính xác (Precision): 0.5945945945945946
Độ phủ (Recall): 0.16793893129770993
F1-score: 0.2619047619047619
```

Hình 3.2 Đánh giá mô hình Logistic Regression

Mô hình hồi quy Logistic Regression đạt độ chính xác 81,4%, có nghĩa là mô hình đang dự đoán chính xác các điểm dữ liệu trong tập kiểm tra. Tuy nhiên F1-score thấp (26,19%), điều này thể hiện mô hình có xu hướng dự đoán lớp positive không tốt, và cần cải thiện để đạt được độ phân loại chính xác cao hơn và đảm bảo mô hình dự đoán tốt cả hai lớp (positive và negative)

Thường thì mô hình Logistic Regression có xu hướng tập trung vào lớp nhiều mẫu hơn và có thể bỏ sót lớp ít mẫu hơn. Bằng cách cân bằng trọng số, mô hình sẽ được huấn luyện sao cho có xu hướng tốt hơn trong việc phân loại các lớp thiểu số. e, sử dụng tham số "class_weight='balanced'" để cân bằng trọng số của các lớp

Thêm tham số "class_weight='balanced'" vào mô hình

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Xây dựng mô hình Logistic Regression
model = LogisticRegression(class_weight='balanced')
model.fit(train_df, train_decided)
```

```
LogisticRegression
LogisticRegression(class_weight='balanced')
```

Hình 3.3 Mô hình Logistic Regression thêm tham số

Các chỉ số đánh giá của mô hình sau khi thêm tham số

```
Độ chính xác (Accuracy): 0.7125
Độ phân loại chính xác (Precision): 0.3729050279329609
Độ phủ (Recall): 0.6793893129770993
F1-score: 0.48151487826871053
```

Hình 3.4 Đánh giá Mô hình Logistic Regression thêm tham số

Sau khi thêm tham số cân bằng thì chỉ số F1-score đã tăng lên thành 48.15%, như vậy mô hình đã đánh giá chính xác hơn ban đầu.

3.2 Random forest

Random Forest là thuật toán xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định.

Cách thức hoạt động của Random Forest:

1. Tạo dữ liệu mẫu ngẫu nhiên: Khi huấn luyện mô hình Random Forest, chúng ta tạo ngẫu nhiên một tập con của dữ liệu huấn luyện bằng cách lấy mẫu với thay thế. Quá trình này tạo ra nhiều bộ dữ liệu con có kích thước nhỏ hơn.

2. Xây dựng cây quyết định: Trên mỗi tập con dữ liệu, chúng ta xây dựng một cây quyết định độc lập. Cây quyết định là một mô hình học máy đơn giản sử dụng luật if-else để đưa ra các quyết định.

3. Kết hợp cây quyết định: Sau khi xây dựng nhiều cây quyết định, chúng ta kết hợp kết quả từ các cây này để ra quyết định cuối cùng. Trong bài toán phân

loại, quyết định cuối cùng thường dựa trên sự biểu quyết đa số (majority vote) của các cây. Trong bài toán hồi quy, quyết định cuối cùng thường là trung bình của các dự đoán từ các cây.

4. Tránh overfitting: Random Forest có khả năng giảm thiểu hiện tượng overfitting (quá khớp) thông qua việc sử dụng ngẫu nhiên các tập con dữ liệu và sử dụng trung bình dự đoán từ nhiều cây.

Random Forest cho thấy hiệu quả hơn so với thuật toán phân loại thường được sử dụng vì có khả năng tìm ra thuộc tính nào quan trọng hơn so với những thuộc tính khác.

Mô hình Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
model = RandomForestClassifier()
model.fit(train_df, train_decided)
# Dự đoán trên tập kiểm tra
predicted_decided = model.predict(test_df)
# Đánh giá mô hình bằng các độ đo thông thường
accuracy = accuracy_score(test_decided, predicted_decided)
precision = precision_score(test_decided, predicted_decided)
recall = recall_score(test_decided, predicted_decided)
f1 = f1_score(test_decided, predicted_decided)
# In kết quả đánh giá mô hình
print("Độ chính xác (Accuracy):", accuracy)
print("Độ phân loại chính xác (Precision):", precision)
print("Độ phủ (Recall):", recall)
print("F1-score:", f1)
```

Hình 3.5 Mô hình Random Forest

Các chỉ số đánh giá mô hình

```
Độ chính xác (Accuracy): 0.87
Độ phân loại chính xác (Precision): 0.7782426778242678
Độ phủ (Recall): 0.4732824427480916
F1-score: 0.5886075949367089
```

Hình 3.6 Đánh gia mô hình Random Forest

Độ chính xác của mô hình rất cao (87%) và F1-scored đạt 58,86%, có nghĩa mô hình hiệu quả hơn mô hình Logistic Regression

Sau khi thêm tham số cân bằng "class_weight='balanced'"

```
Độ chính xác (Accuracy): 0.8615
Độ phân loại chính xác (Precision): 0.7636363636363637
Độ phủ (Recall): 0.42748091603053434
F1-score: 0.5481239804241436
```

Hình 3.7 Đánh giá mô hình Random Forest khi thêm tham số

Các chỉ số đánh giá giảm so với ban đầu.

3.3 Support Vector Machine

SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồi thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" (hyper-plane) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.

Mô hình SVM

```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
#Xây dựng mô hình
model = SVC()
model.fit(train_df, train_decided)
# Dự đoán trên tập kiểm tra
predicted_decided = model.predict(test_df)
# Đánh giá mô hình bằng các độ đo thông thường
accuracy = accuracy_score(test_decided, predicted_decided)
precision = precision_score(test_decided, predicted_decided)
recall = recall_score(test_decided, predicted_decided)
f1 = f1_score(test_decided, predicted_decided)
# In kết quả đánh giá mô hình
print("Độ chính xác (Accuracy):", accuracy)
print("Độ phân loại chính xác (Precision):", precision)
print("Độ phủ (Recall):", recall)
print("F1-score:", f1)
```

```
Độ chính xác (Accuracy): 0.8175
Độ phân loại chính xác (Precision): 0.9375
Độ phủ (Recall): 0.07633587786259542
F1-score: 0.14117647058823532
```

Hình 3.8 Kết quả đánh giá mô hình SVM

Độ chính xác cao (81,75%), mô hình dự đoán khá chính xác quyết định rời bỏ của khách hàng. Tuy nhiên mô hình chưa hiệu quả (F1-score 14,12%)

3.4 Gradient Boosting

Gradient Boosting là một phương pháp học máy trong lĩnh vực học tập có giám sát (supervised learning) được sử dụng chủ yếu cho các bài toán phân loại (classification) và hồi quy (regression). Đây là một kỹ thuật tinh chỉnh mô hình ensemble (tập hợp) các mô hình yếu (weak learner) thành một mô hình mạnh mẽ hơn.

Cách thức hoạt động của Gradient Boosting:

1. Bắt đầu bằng việc xây dựng một mô hình yếu ban đầu, thường là một cây quyết định đơn giản.
2. Tính toán sai số (residual) giữa các giá trị thực tế và dự đoán ban đầu của mô hình yếu này.
3. Xây dựng một mô hình yếu tiếp theo (có thể là cây quyết định khác) để dự đoán sai số từ bước trước.
4. Cộng dồn các dự đoán của các mô hình yếu trước đó để tạo ra mô hình mạnh hơn cho từng lần dự đoán.
5. Quá trình xây dựng các mô hình yếu và cộng dồn dự đoán lặp lại nhiều lần (theo số lượng lặp (iterations) đã cho hoặc dừng lại khi sai số không còn giảm đáng kể).
6. Cuối cùng, mô hình Gradient Boosting tổng hợp các dự đoán từ các mô hình yếu để tạo ra một dự đoán cuối cùng.

```python
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Xây dựng mô hình Gradient Boosting
model = GradientBoostingClassifier()
model.fit(train_df, train_decided)
# Dự đoán trên tập kiểm tra
predicted_decided = model.predict(test_df)
# Đánh giá mô hình bằng các độ đo thông thường
accuracy = accuracy_score(test_decided, predicted_decided)
precision = precision_score(test_decided, predicted_decided)
recall = recall_score(test_decided, predicted_decided)
f1 = f1_score(test_decided, predicted_decided)
# In kết quả đánh giá mô hình
print("Độ chính xác (Accuracy):", accuracy)
print("Độ phân loại chính xác (Precision):", precision)
print("Độ phủ (Recall):", recall)
print("F1-score:", f1)
```

```
Độ chính xác (Accuracy): 0.864
Độ phân loại chính xác (Precision): 0.7552742616033755
Độ phủ (Recall): 0.455470737913486
F1-score: 0.5682539682539682
```

Hình 3.9 Kết quả đánh giá mô hình SVM

Mô hình có độ chính xác rất cao (86,4%) và F1-scored đạt 56,82%, chứng tỏ mô hình hoạt động khá hiệu quả

Kết luận

Kết quả đạt được

1. Tìm hiểu bài toán Customer Churning, dự đoán và đưa ra mô hình dự đoán về sự rời đi của khách hàng trong ngân hàng.
2. Tìm hiểu và biết sử dụng các thư viện
3. Vận dụng các kiến thức về thuật toán trong Machine Learning, Python, Hồi quy tuyến tính,... để xây dựng mô hình và đánh giá mô hình cho bài toán Customer Churning.

Kỹ năng đạt được

1. Nâng cao khả năng nghiên cứu, đọc hiểu tài liệu chuyên ngành.
2. Viết và trình bày bài báo khoa học liên quan đến nội dung báo cáo.
3. Soạn thảo đồ án bằng công cụ $LATEX$.
4. Tìm hiểu và biết sử dụng các thư viện Pandas, Seaborn, Scikit Learn.

Tài liệu tham khảo

[1] Slide bài giảng Hệ hỗ trợ Quyết định - TS. Lê Hải Hà

[2] https://www.kaggle.com/datasets/barelydedicated/bank-customer-churn-modeling

[3] https://scikit-learn.org/stable/

[4] https://machinelearningmastery.com/

[5] Andreas C. Müller and Sarah Guido. Book: *Introduction to Machine Learning with Python*. September 2016

[6] Daniel S. Putler and Robert E. Krider. Book: *Customer and Business Analytics: Applied Data Mining for Business Decision Making Using R*. May 2012.