BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

BÀI TẬP LỚN

HỌC MÁY CƠ BẢN

DỰ ĐOÁN SỰ HÀI LÒNG CỦA KHÁCH HÀNG NGÂN HÀNG

NGÀNH: KHOA HỌC MÁY TÍNH

SINH VIÊN: **NGUYỄN VĂN ĐẠT**

MÃ SINH VIÊN: **12423061**

LỚP: **12423TN**

GIẢNG VIÊN: **PGS.TS.NGUYỄN VĂN HẬU**

HƯNG YÊN – 2025

NHẬN XÉT

Nhận xét của giảng viên:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GIẢNG VIÊN

NGUYỄN VĂN HẬU

LỜI CAM ĐOAN

Em xin cam đoan bài tập lớp môn Học máy cơ bản "Dự đoán sự hài lòng của khách hàng ngân hàng" là sản phẩm của bản thân. Những phần sử dụng tài liệu tham khảo trong bài tập lớn đã được nêu rõ trong phần tài liệu tham khảo. Các số liệu, kết quả trình bày trong bài tập lớn là hoàn toàn trung thực, nếu sai em xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của bộ môn và nhà trường đề ra.

LỜI CẢM ƠN

Để có thể hoàn thành bài tập lớn này, lời đầu tiên em xin phép gửi lời cảm ơn tới bộ môn Học máy cơ bản, Khoa Công nghệ thông tin – Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện đồ án môn học này. Đặc biệt em xin chân thành cảm ơn thầy Nguyễn Văn Hậu đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện bài tập lớn vừa qua.

Mặc dù em đã có cố gắng, nhưng với trình độ còn hạn chế, trong quá trình thực hiện đề tài không tránh khỏi những thiếu sót. Em hy vọng sẽ nhận được những ý kiến nhận xét, góp ý của thầy cô về những kết quả triển khai trong bài tập lớn.

Em xin trân trọng cảm ơn!

NỘI DUNG

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: TỔNG QUAN BÀI TOÁN

1.1 Giới thiệu bài toán

Trong bối cảnh cạnh tranh ngày càng gay gắt của ngành ngân hàng, việc nâng cao mức độ hài lòng của khách hàng đóng vai trò quan trọn trong việc duy trì lợi thế cạnh tranh và phát triển bền vững. Sự hài lòng của khách hàng không chỉ phản ánh chất lượng dịch vụ mà còn ảnh hưởng trực tiếp đến khả năng giữ chân khách hàng, uy tin thương hiệu và hiệu quả kinh doanh của ngân hàng.

Bài toán dự đoán sự hài lòng của khách hàng ngân hàng tập trung vào việc phân tích các dữ liệu liên quan đến đặc điểm khách hàng, hành vi giao dịch và trải nghiệm sử dụng dịch vụ nhằm xác định mức độ hài lòng của ho. Thông qua việc áp dụng các phương pháp phân tích dữ liệu và học máy, hệ thống có thể dự đoán sớm nhóm khách hàng có nguy cơ không hài lòng, từ đó hỗ trợ ngân hàng đưa ra các giải pháp cải thiện chất lượng dịch vụ và các nân hoá trải nghiệm người dùng.

Việc xây dựng mô hình dự đoán sự hài lòng của khách hàng ngân hàng không chỉ mang lại giá trị thực tiễn cho các tổ chức tài chính mà còn góp phần nâng cao hiệu quả quản lý, tối ưu hoá chiến lược chăm sóc khách hàng và thúc đẩy sự phát triển ổn định của hệ thống ngân hàng trong bối cảnh chyển đổi số hiện nay.

1.2 Tập dữ liệu

Dữ liệu được lấy từ Kaggle:
https://www.kaggle.com/datasets/adammaus/predicting-churn-for-bank-customers

```
df = pd.read_csv(path)
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	I
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	

Hình 1.1: Tổng quan về tập dữ liệu

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Hình 1.2: Thông tin của tập dữ liệu

- Tập dữ liệu có các đặc trưng sau:
    + RowNumber: Số thứ tự
    + CustomerId: Mã khách hàng
    + Surname: Họ khách hàng
    + CreditScore: Điểm tín dụng
    + Geography: Quốc gia
    + Gender: Giới tính
    + Age: Độ tuổi
    + Tenure: Số năm gắn bó với ngân hàng
    + Balance: Số dư tài khoản trung bình
    + NumOfProducts: Số lượng sản phẩm/dịch vụ sử dụng

+ HashCrCard: Có thẻ tín dụng
+ IsActiveMember: Là thành viên tích cực
+ EstimatedSalary: Mức lương ước tính
+ Exited: Rời bỏ ngân hàng

- Tập dữ liệu là một tệp ở định dạng csv bao gồm 10000 hàng × 14 cột
- Mô tả dữ liệu:

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	2500.75	5.000500e+03	7.500250e+03	10000.00
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	15628528.25	1.569074e+07	1.575323e+07	15815690.00
CreditScore	10000.0	6.505288e+02	96.653299	350.00	584.00	6.520000e+02	7.180000e+02	850.00
Age	10000.0	3.892180e+01	10.487806	18.00	32.00	3.700000e+01	4.400000e+01	92.00
Tenure	10000.0	5.012800e+00	2.892174	0.00	3.00	5.000000e+00	7.000000e+00	10.00
Balance	10000.0	7.648589e+04	62397.405202	0.00	0.00	9.719854e+04	1.276442e+05	250898.09
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	1.00	1.000000e+00	2.000000e+00	4.00
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	0.00	1.000000e+00	1.000000e+00	1.00
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	0.00	1.000000e+00	1.000000e+00	1.00
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	51002.11	1.001939e+05	1.493882e+05	199992.48
Exited	10000.0	2.037000e-01	0.402769	0.00	0.00	0.000000e+00	0.000000e+00	1.00

Hình 1.3: Mô tả tập dữ liệu

1.3 Tiền xử lý dữ liệu

- Loại bỏ những cột không cần thiết như RowNumber, CustomerId, Surname

```
df = df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)
df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Hình 1.4: Loại bỏ những cột không cần thiết

- Kiểm tra giá trị bị thiếu

```
df.isnull().sum()
```

```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Hình 1.5: Kiểm tra giá trị khuyết thiếu

- Sử dụng LabelEncoder cho dữ liệu danh mục

```
le = LabelEncoder()
df_cat = df[cat_vars].apply(le.fit_transform)
df_cat.head()
```

	HasCrCard	IsActiveMember	Geography	Gender
0	1	1	0	0
1	0	1	2	0
2	1	0	0	0
3	0	0	0	0
4	1	1	2	0

Hình 1.6: Mã hoá dữ liệu danh mục

- Sử dụng StandardScaler để chuẩn hoá dữ liệu dạng số

```
continuous_vars = ['CreditScore',  'Age', 'Tenure', 'Balance','NumOfProducts','EstimatedSalary']
scaler = StandardScaler()
df_continuous = pd.DataFrame(scaler.fit_transform(df[continuous_vars]), columns=continuous_vars)
df_continuous.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
0	-0.326221	0.293517	-1.041760	-1.225848	-0.911583	0.021886
1	-0.440036	0.198164	-1.387538	0.117350	-0.911583	0.216534
2	-1.536794	0.293517	1.032908	1.333053	2.527057	0.240687
3	0.501521	0.007457	-1.387538	-1.225848	0.807737	-0.108918
4	2.063884	0.388871	-1.041760	0.785728	-0.911583	-0.365276

Hình 1.7: Chuẩn hoá dữ liệu liên tục

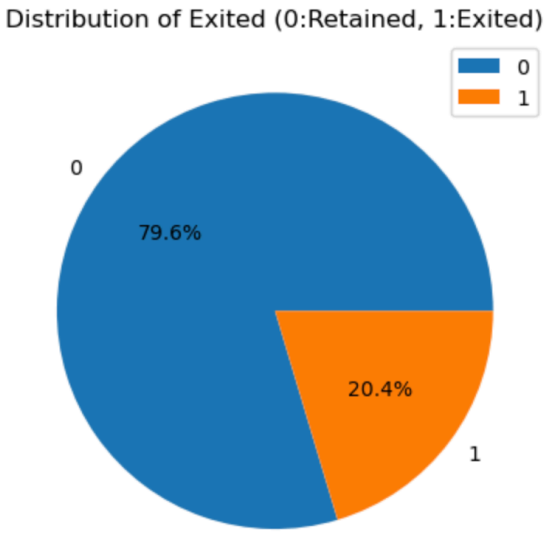
1.4 Phân tích và trực quan hoá dữ liệu

a) Số lượng khách hàng ở lại và rời đi

```
[9]: df['Exited'].value_counts()
```

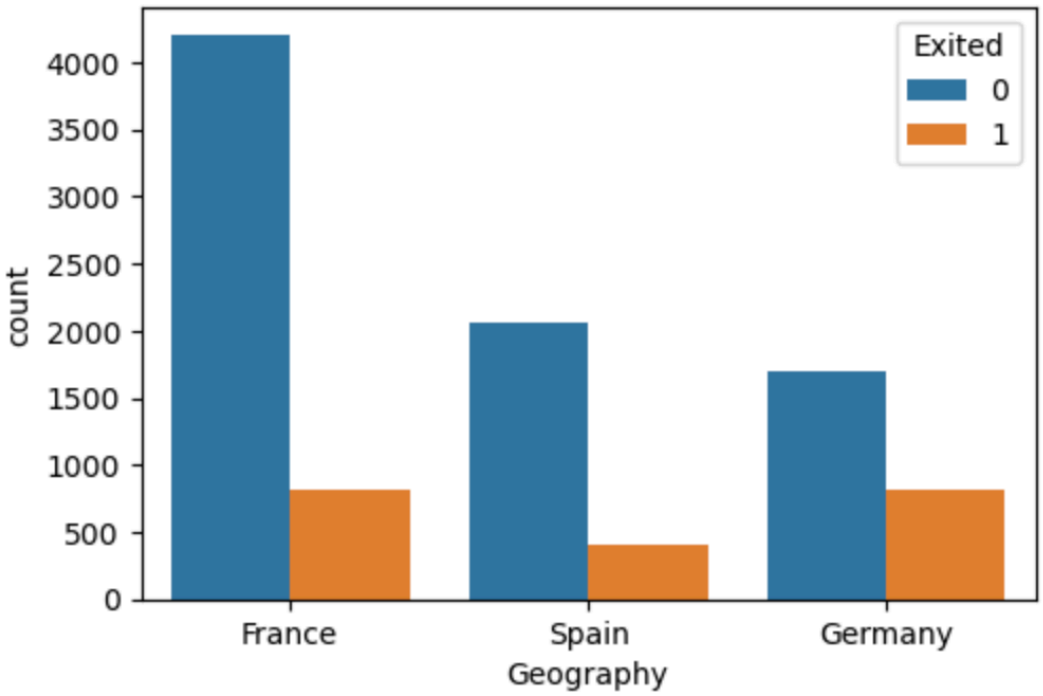
```
[9]: Exited
     0    7963
     1    2037
     Name: count, dtype: int64
```

```
11]: plt.pie(df['Exited'].value_counts(), labels=df['Exited'].value_counts().index, autopct='%1.1f%%')
     plt.title('Distribution of Exited (0:Retained, 1:Exited)')
     plt.legend()
     plt.show()
```



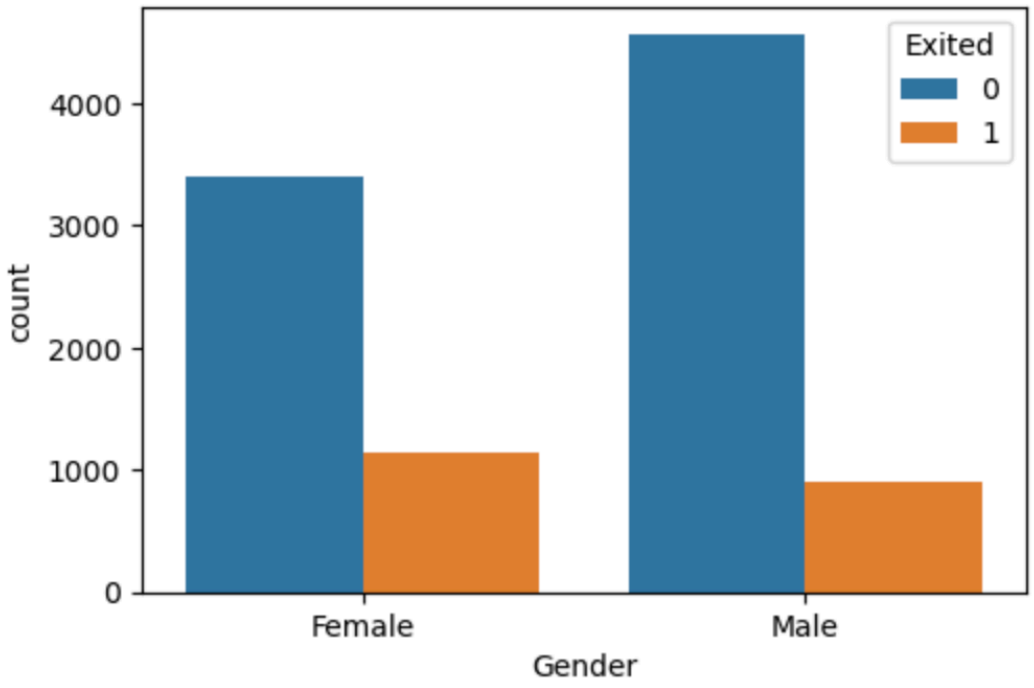
Hình 1.8: Trực quan hoá dữ liệu Exited

b) Trực quan hoá ảnh hưởng của Geography lên Exited



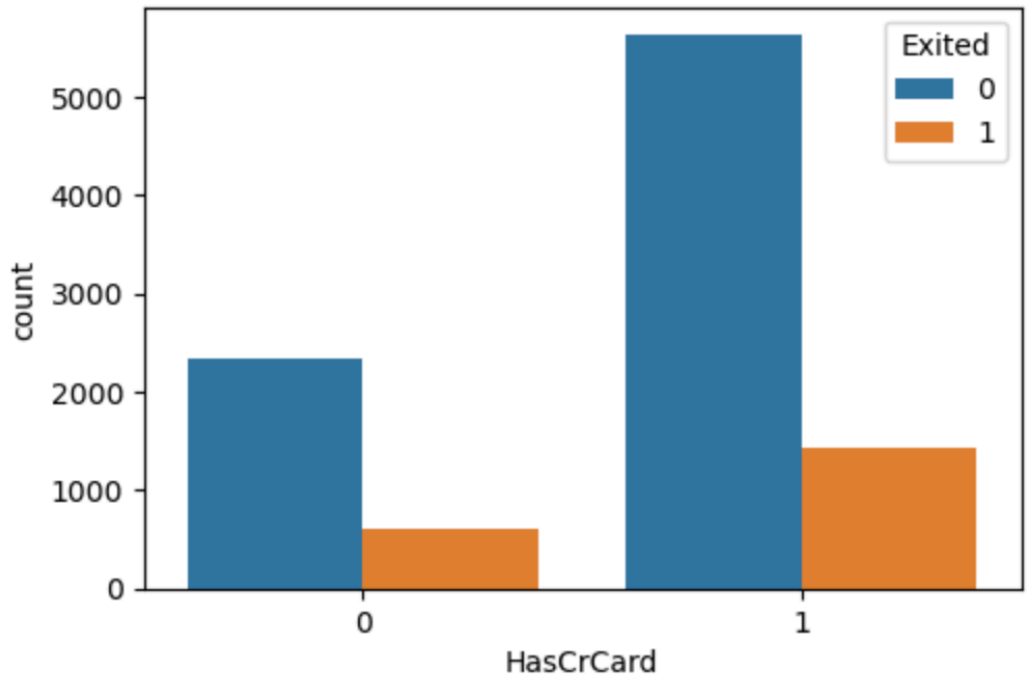
Hình 1.9: Trực quan hoá ảnh hưởng của Geography lên Exited

c) Trực quan hoá ảnh hưởng của Gender lên Exited



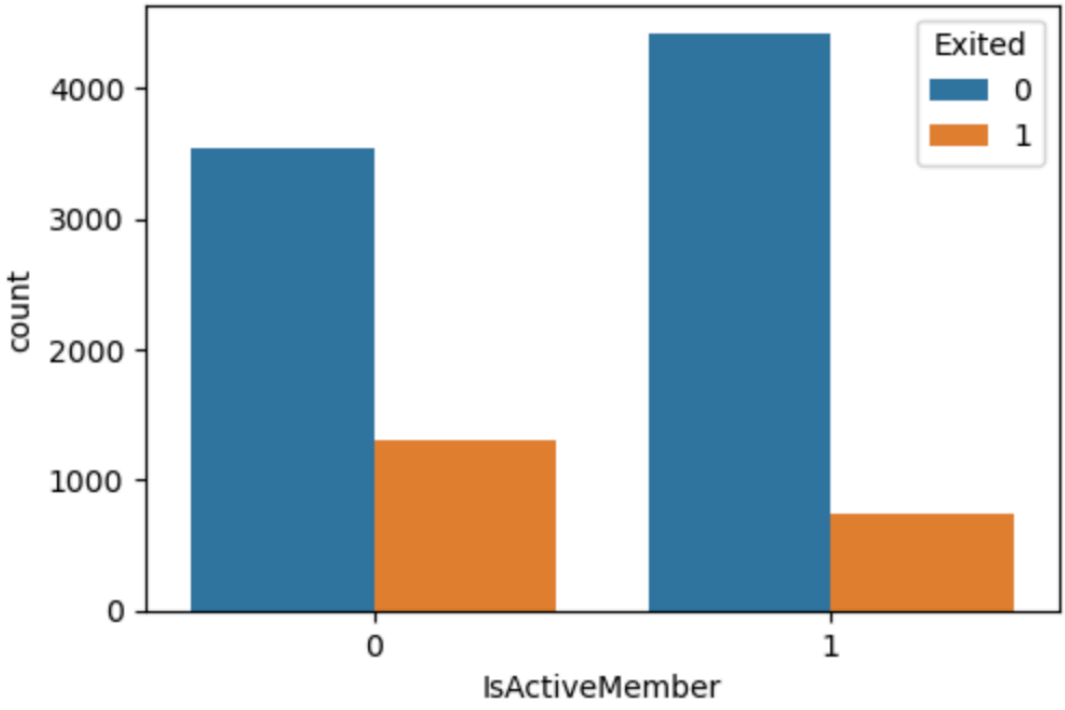
Hình 1.10: Trực quan hoá ảnh hưởng của Gender lên Exited

d) Trực quan hoá ảnh hưởng của HasCrCard lên Extied



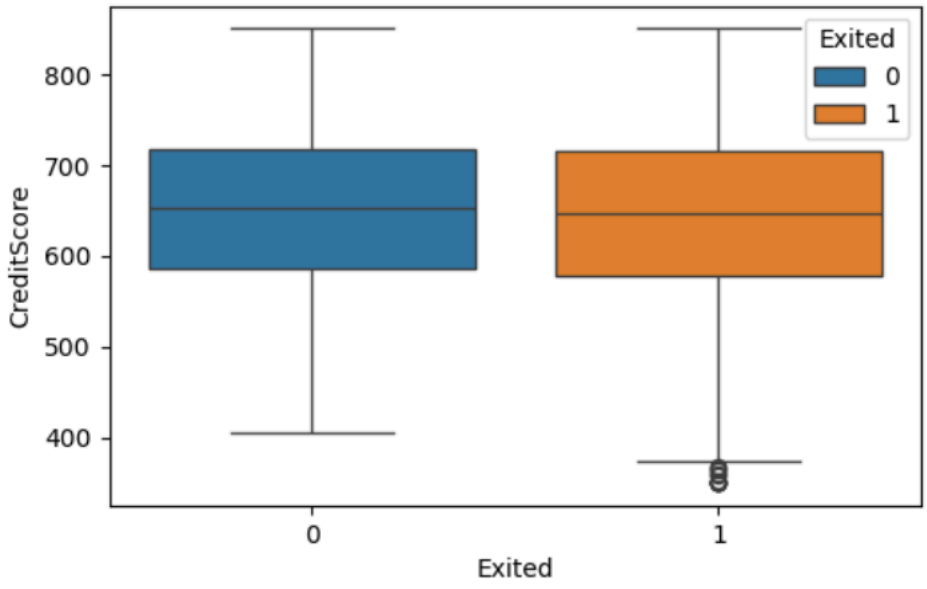
Hình 1.11: Trực quan hoá ảnh hưởng của HasCrCard lên Extied

e) Trực quan hoá ảnh hưởng của IsActiveMember lên Exited



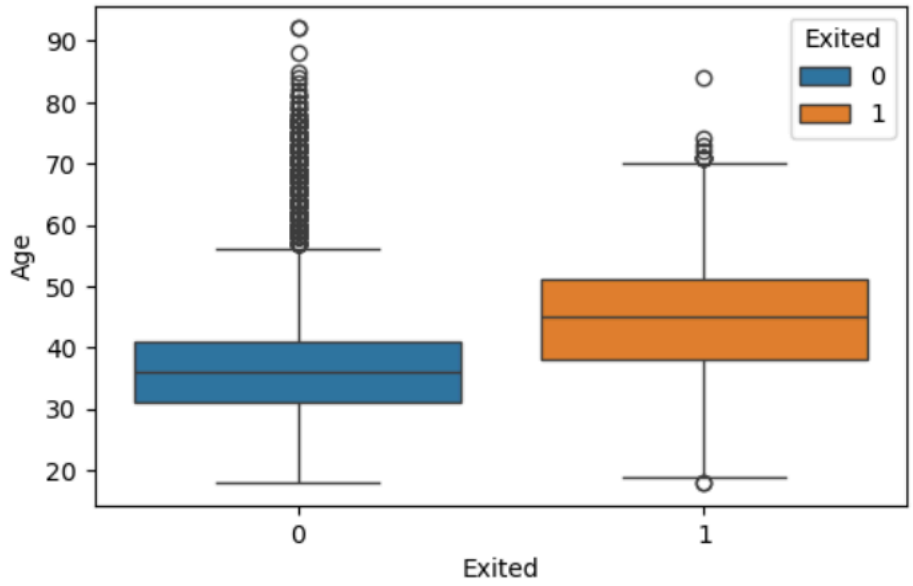
Hình 1.12: Trực quan hoá ảnh hưởng của IsActiveMember lên Exited

f) Trực quan hoá ảnh hưởng của CreditScore lên Extied



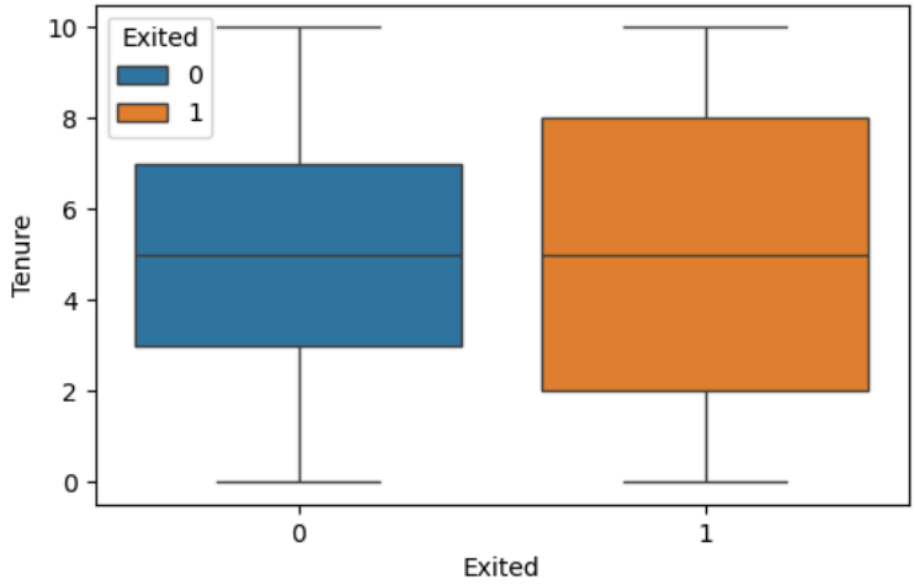
Hình 1.13: Trực quan hoá ảnh hưởng của CreditScore lên Extied

g) Trực quan hoá ảnh hưởng của Age lên Exited



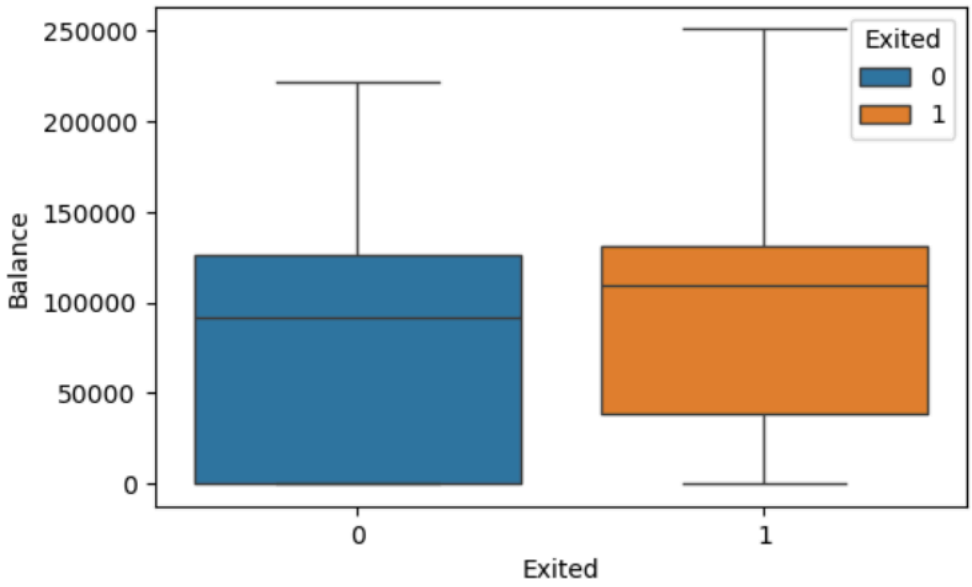
Hình 1.14: Trực quan hoá ảnh hưởng của Age lên Exited

h) Trực quan hoá ảnh hưởng của Tenure lên Exited



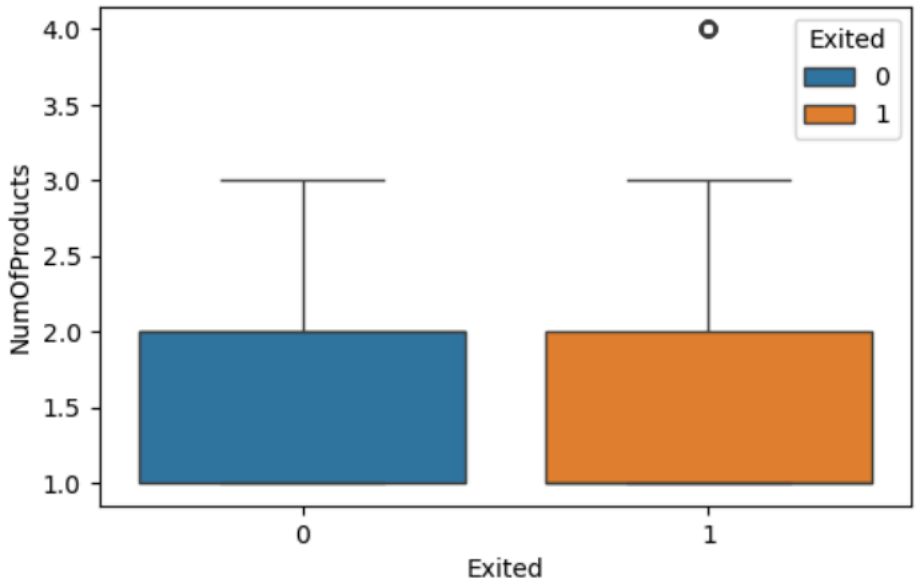
Hình 1.15: Trực quan hoá ảnh hưởng của Tenure lên Exited

i) Trực quan hoá ảnh hưởng của Balance lên Exited



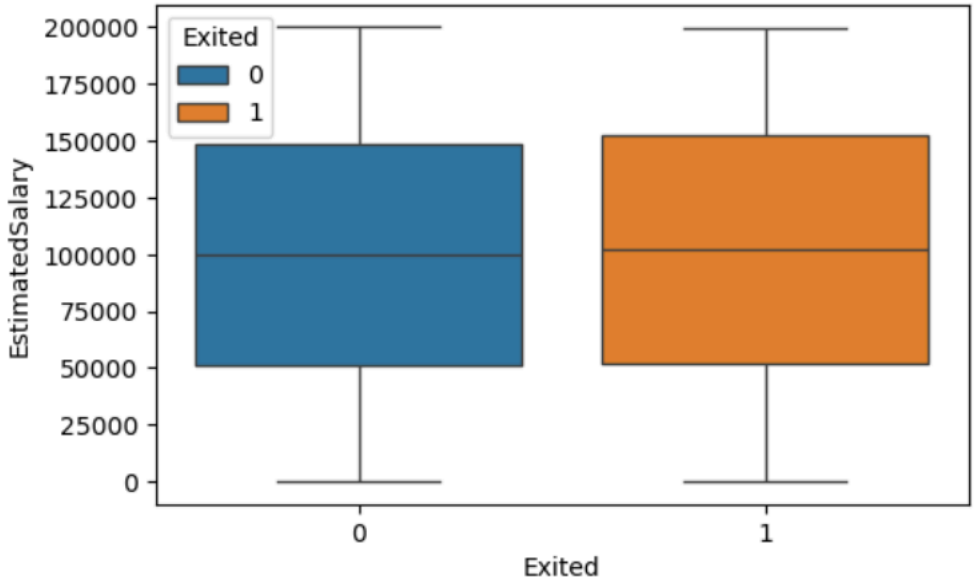
Hình 1.16: Trực quan hoá ảnh hưởng của Balance lên Exited

j) Trực quan hoá ảnh hưởng của NumOfProducts lên Exited



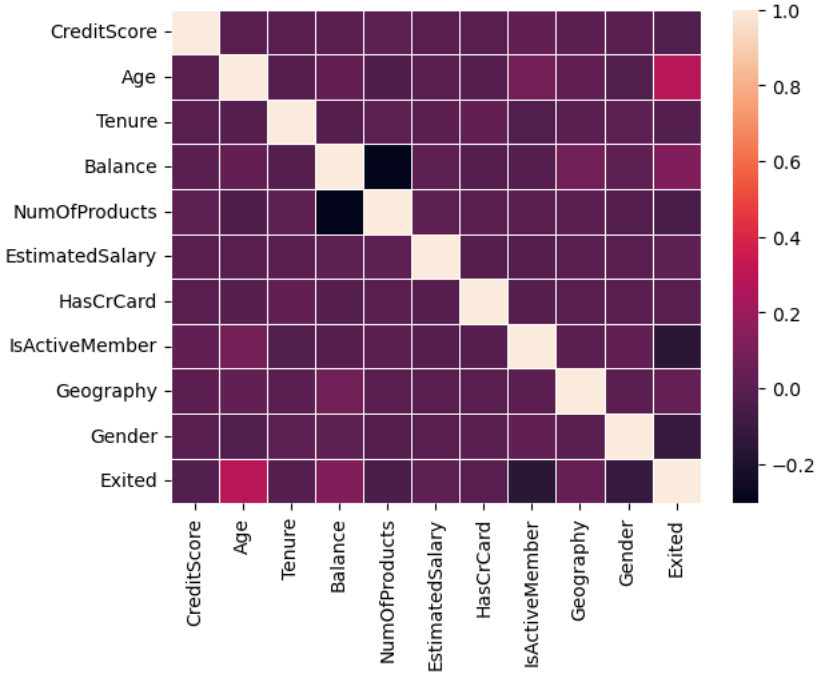
Hình 1.17: Trực quan hoá ảnh hưởng của NumOfProducts lên Exited

k) Trực quan hoá ảnh hưởng của EstimatedSalary lên Exited



Hình 1.18: Trực quan hoá ảnh hưởng của EstimatedSalary lên Exited

l) Ma trận tương quan giữa các đặc trưng



Hình 1.19: Ma trận tương quan giữa các đặc trưng

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Pandas

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và dễ hiểu. Tên thư viện được lấy từ "panel data" (dữ liệu bảng). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có thể không đồng nhất) và dữ liệu chuỗi thời gian¹. Thư viện này thường được sử dụng để thao tác, phân tích và làm sạch dữ liệu. Pandas cung cấp nhiều cấu trúc dữ liệu và các phép toán hỗ trợ thao tác dữ liệu số và dữ liệu thời gian². Nó là một công cụ toàn diện để xử lý dữ liệu trong Python, phù hợp với nhiều nhiệm vụ phân tích và quản lý dữ liệu. Thư viện Pandas trong Python là một thư viện mã nguồn mở cung cấp hỗ trợ mạnh mẽ cho việc thao tác dữ liệu. Nó cũng là một bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình Python. Thư viện này được sử dụng rộng rãi trong cả nghiên cứu và phát triển các ứng dụng khoa học dữ liệu. Thư viện này sử dụng một cấu trúc dữ liệu độc đáo được gọi là Dataframe. Pandas cung cấp rất nhiều hàm để thao tác và làm việc trên cấu trúc dữ liệu này. Chính sự linh hoạt và hiệu quả đã làm cho Pandas được sử dụng rộng rãi.

Tại sao nên chọn pandas?

- Pandas phù hợp với nhiều loại dữ liệu khác nhau
- Dữ liệu dạng bảng với các cột được nhập không đồng đều, chẳng hạn như trong bảng SQL hoặc bảng tính Excel
- Dữ liệu chuỗi thời gian có thứ tự và không có thứ tự (không nhất thiết phải có tần số cố định)
- Dữ liệu ma trận tùy ý (được nhập đồng đều hoặc không đồng đều) với nhãn hàng và cột
- Bất kỳ dạng tập dữ liệu quan sát/thống kê nào khác. Dữ liệu thực tế không cần phải được gắn nhãn trong cấu trúc dữ liệu pandas
- Pandas được xây dựng trên NumPy. Hai cấu trúc dữ liệu chính của pandas, Series (1 chiều) và DataFrame (2 chiều), xử lý các trường hợp sử dụng điển

hình nhất trong tài chính, thống kê, khoa học xã hội và nhiều lĩnh vực kỹ thuật.

Ưu điểm của pandas:

- Dễ dàng xử lý dữ liệu bị mất mát, được biểu diễn dưới dạng NaN, trong dữ liệu dấu phẩy động cũng như dữ liệu dấu phẩy cố định theo ý muốn của người dùng: bỏ qua hoặc chuyển thành 0
- Có thể thay đổi kích thước: các cột có thể được chèn và xóa khỏi DataFrame và các đối tượng có chiều cao hơn
- Căn chỉnh dữ liệu tự động và rõ ràng: các đối tượng có thể được căn chỉnh rõ ràng với một tập hợp nhãn hoặc người dùng có thể chỉ cần bỏ qua các nhãn và để Series, DataFrame, v.v. tự động căn chỉnh dữ liệu cho bạn trong các phép tính
- Định hình lại và xoay trục linh hoạt của tập dữ liệu
- Cắt lát dựa trên nhãn thông minh, lập chỉ mục nâng cao và chọn tập con của các tập dữ liệu lớn
- Hiệu suất tốt

Cài đặt thư viện Pandas:

- Sử dụng pip và nhập lệnh: pip install pandas
- Hoặc với Anaconda, sử dụng lệnh: conda install pandas
- Khai báo thư viện Pandas: import pandas as pd

2.1.1 Series

Series([data, index, dtype, name, copy, . . .])

Một Series là một mảng đa chiều tương tự như mảng Numpy hoặc một cột của bảng, nhưng nó bao gồm thêm một nhãn để đánh dấu bảng. Series có thể được khởi tạo thông qua NumPy, các kiểu Dict hoặc các kiểu dữ liệu vô hướng thông thường. Series có nhiều thuộc tính như index, array, value, dtype, v.v. Bạn có thể thực hiện chuyển đổi Series sang một dtype cụ thể, tạo bản sao của bảng, trả về giá

trị boolean của một phần tử, chuyển đổi Series từ DatetimeIndex sang PeriodIndex, v.v.

2.1.2 DataFrame

DataFrame([data, index, columns, dtype, copy])

DataFrame là một cấu trúc dữ liệu hai chiều có nhãn với các cột và hàng giống như bảng tính hoặc bảng. Giống như Series, DataFrame có thể chứa bất kỳ loại dữ liệu nào. Một điều quan trọng cần nhấn mạnh là tất cả các cột trong dataframe đều là các Series của Pandas. Vì vậy, DataFrame là sự kết hợp của nhiều Series hoạt động như các cột! DataFrame được sử dụng rộng rãi và là một trong những cấu trúc dữ liệu quan trọng nhất.

2.2 Matplotlib

Matplotlib là một thư viện đồ thị dành cho ngôn ngữ lập trình Python và mở rộng thư viện số học NumPy. Nó cung cấp API hướng đối tượng để nhúng đồ thị vào các ứng dụng sử dụng các bộ công cụ GUI đa năng như Tkinter, wxPython, Qt hoặc GTK. Matplotlib cho phép bạn tạo và hiển thị đồ thị, hình ảnh và các đồ họa khác, rất hữu ích cho việc trực quan hóa dữ liệu trong Python. Một phần quan trọng của Matplotlib là Pyplot, một mô-đun cung cấp các hàm đơn giản để thêm các thành phần đồ thị như đường thẳng, hình ảnh, văn bản, v.v. vào khung vẽ (trục). Matplotlib là một trong những thư viện Python phổ biến nhất được sử dụng để trực quan hóa dữ liệu. Nó là một thư viện đa nền tảng để tạo đồ thị 2D từ dữ liệu trong mảng. Matplotlib được viết bằng Python và sử dụng NumPy, phần mở rộng toán học của Python. Nó cung cấp API hướng đối tượng để nhúng đồ thị vào các ứng dụng và sử dụng các bộ công cụ GUI của Python như PyQt và WxPython hoặc Tkinter. Nó cũng có thể được sử dụng trong các trình thông dịch Python và IPython, Jupyter Notebooks và máy chủ web.

Matplotlib có một giao diện tên là Pylab, được thiết kế để giống với MATLAB - một ngôn ngữ lập trình độc quyền được phát triển bởi MathWorks. Matplotlib, cùng với NumPy, có thể được coi là phiên bản mã nguồn mở tương đương với MATLAB. Matplotlib ban đầu được viết bởi John D. Hunter vào năm 2003. Phiên bản ổn định hiện tại là 2.2.0, được phát hành vào tháng 1 năm 2018.

2.2.1 Matplotlib được dùng để làm gì?

Để thực hiện các suy luận thống kê cần thiết, việc trực quan hóa dữ liệu là rất quan trọng và Matplotlib là một giải pháp như vậy dành cho người dùng Python. Đây là một thư viện đồ thị rất mạnh mẽ, hữu ích cho những người làm việc với Python và NumPy. Mô-đun được sử dụng nhiều nhất của Matplotlib là Pyplot, cung cấp giao diện tương tự như MATLAB nhưng thay vào đó, nó sử dụng Python và là mã nguồn mở.

Để cài đặt Matplotlib, nếu bạn có Anaconda, chỉ cần gõ `conda install matplotlib` hoặc sử dụng công cụ pip `pip install matplotlib`.

2.2.2 Khái niệm chung

Một hình vẽ trong Matplotlib có thể được phân loại thành nhiều phần như sau:

- Figure: Nó giống như một cửa sổ chứa mọi thứ bạn sẽ vẽ trên đó.

- Axes: Các thành phần chính của một hình vẽ là các trục (các khung nhỏ hơn để vẽ). Một hình vẽ có thể chứa một hoặc nhiều trục. Nói cách khác, hình vẽ chỉ là vùng chứa, các trục là nơi các hình vẽ thực tế được vẽ.
- Axis: Chúng là các đối tượng giống như đường số và chịu trách nhiệm tạo ra các ranh giới của đồ thị.
- Artist: Mọi thứ bạn có thể thấy trên hình vẽ đều là đối tượng vẽ, chẳng hạn như đối tượng Văn bản, đối tượng Đường 2D, đối tượng Tập hợp. Hầu hết các Đối tượng vẽ đều được gắn với Trục.

2.2.3 Ưu điểm

Matplotlib là một thư viện tương tự như GNUplot. Ưu điểm chính so với GNUplot là Matplotlib là một module của Python. Do sự phổ biến ngày càng tăng của Python, Matplotlib cũng nhận được sự chú ý tương tự.

Một lý do khác khiến Matplotlib được ưa chuộng là nó được coi là một sự thay thế hoàn hảo cho MATLAB, nếu được sử dụng kết hợp với Numpy và Scipy. Trong khi MATLAB đắt tiền và là mã nguồn đóng, Matplotlib là phần mềm miễn phí và mã nguồn mở. Nó cũng là một ngôn ngữ hướng đối tượng. Hơn nữa, nó có thể được sử dụng với các bộ công cụ GUI đa năng như wxPython, Qt và GTK+.

Ngoài ra còn có một hàm "pylab", được thiết kế để giống với MATLAB. Điều này có thể giúp những người quen thuộc với MATLAB dễ dàng chuyển sang Matplotlib. Matplotlib có thể được sử dụng để tạo ra các hình ảnh chất lượng cao cho nhiều định dạng in ấn và môi trường tương tác trên nhiều nền tảng.

Một tính năng khác của Matplotlib là đường cong học tập của nó, có nghĩa là người dùng thường tiến bộ nhanh chóng sau khi bắt đầu. Trang web chính thức cho biết: "matplotlib cố gắng làm cho những việc khó khăn, phức tạp trở nên dễ dàng nhất có thể. Bạn có thể tạo hình ảnh, biểu đồ tần số, biểu đồ phổ, biểu đồ cột, biểu đồ lỗi, biểu đồ phân tán, v.v., chỉ với một vài dòng mã."

Matplotlib có một số giao diện để tương tác với thư viện matplotlib: API hướng đối tượng, Giao diện kịch bản (pyplot), Giao diện MATLAB (pylab). Cả pyplot và pylab đều là các giao diện nhẹ, nhưng Pyplot cung cấp giao diện thủ tục cho các thư viện vẽ đồ thị hướng đối tượng trong matplotlib. Các lệnh vẽ đồ thị của nó được thiết kế tương tự như của Matlab cả về tên gọi và ý nghĩa đối số. Thiết kế này đã làm cho việc sử dụng pyplot dễ dàng và dễ hiểu hơn, vì vậy trong các bài viết về Matplotlib, tôi sẽ sử dụng giao diện pyplot thay vì hai giao diện còn lại. Nếu chúng ta muốn can thiệp sâu hơn, với nhiều tùy chỉnh hơn, API hướng đối tượng sẽ là lựa chọn đúng đắn.

Để cài đặt Matplotlib nếu bạn có Anaconda, chỉ cần gõ `conda install matplotlib` hoặc sử dụng công cụ `pip install matplotlib`.
`pip install matplotlib`

2.3 Numpy

NumPy (Numeric Python) là một thư viện Python mã nguồn mở được sử dụng trong hầu hết các lĩnh vực khoa học và kỹ thuật. Nó là tiêu chuẩn thực tế để làm việc với dữ liệu số trong Python và là tiêu chuẩn cốt lõi của hệ sinh thái Python và PyData. NumPy hỗ trợ mạnh mẽ việc tính toán với ma trận, vectơ và các hàm đại số tuyến tính cơ bản. Điều này làm cho nó trở thành một công cụ quan trọng trong việc triển khai các thuật toán Học máy. Nó cho phép làm việc hiệu quả với ma trận

và mảng, đặc biệt là dữ liệu ma trận và mảng lớn, với tốc độ xử lý nhanh hơn nhiều lần so với chỉ sử dụng "Python cốt lõi".

2.3.1 Cài đặt

- pip install numpy

2.3.2 Các phép toán với Numpy

- Khai báo thư viện: import numpy as np
- Khởi tạo mảng:
  + arr = np.array([1,3,4,5,6]): Khởi tạo một mảng một chiều
  + arr1 = np.array([(4,5,6), (1,2,3)]): Khởi tạo một mảng hai chiều
  + arr2 = np.array(([(2,4,0,6), (4,7,5,6)],[(0,3,2,1), (9,4,5,6)],[(5,8,6,4), (1,4,6,8)])): Khởi tạo một mảng ba chiều
- Khởi tạo mảng với Hàm tích hợp:
  + np.zeros
  + np.ones((2,3,4), dtype = int): Mảng 3 chiều với phần tử là 1 và kích thước là 2x3x4.
  + np.arange(1,7,2): Mảng có các phần tử từ 1-6 và bước nhảy là 2.
  + np.full((2,3),5): Mảng 2 chiều với phần tử là 5 và kích thước là 2x3.
  + np.eye(4, dtype=int): Ma trận đơn vị có kích thước 4x4.
  + np.random.random((2,3)): Ma trận ngẫu nhiên có kích thước là 2x3.

2.3.3 Các thao tác mảng

- dtype: Kiểu dữ liệu của phần tử trong mảng
- shape: Kích thước của mảng
- size: Số lượng phần tử trong mảng
- ndim: Số chiều của mảng
- Truy cập các phần tử trong mảng: Các phần tử trong mảng được đánh số từ 0.arr[i]: Access to i^{th} of 1-dimensional array.
  + arr1[i,j]: Truy cập vào hàng thứ i và cột thứ j của mảng hai chiều..

+ arr2[n,i,j]: Truy cập vào n chiều, i hàng và j cột của mảng 3 chiều.
+ arr[a:b]: Truy cập các phần tử từ a đến b-1 trong mảng một chiều.
+ arr1[:,:i]: Truy cập các phần tử từ cột 0 đến i-1.

- Chức năng thống kê:
  + arr.max() or np.max(arr): Lấy giá trị lớn nhất trong mảng.
  + arr.min() or np.min(arr): Lấy giá trị nhỏ nhất trong mảng.
  + arr.sum() or np.sum(arr): Tóm tắt các phần tử trong mảng.
  + arr.mean() or np.mean(arr): Lấy giá trị trung bình của mảng
  + np.median(arr): Lấy giá trị trung bình của mảng.

2.4 Scikit-learn

Scikit-learn là một thư viện học máy mã nguồn mở hỗ trợ học có giám sát và không giám sát. Nó cũng cung cấp nhiều công cụ để huấn luyện mô hình, tiền xử lý dữ liệu, lựa chọn mô hình, đánh giá mô hình và nhiều tiện ích khác.

- Các công cụ đơn giản và hiệu quả để phân tích dữ liệu dự đoán
- Dễ tiếp cận với mọi người và có thể tái sử dụng trong nhiều ngữ cảnh
- Được xây dựng trên NumPy, SciPy và matplotlib
- Mã nguồn mở, có thể sử dụng thương mại - Giấy phép BSD

2.4.1 Cài đặt

- Pip install -U scikit-learn

2.4.2 Thao tác với scikit-learn

- Nhiệm vụ :
  + Classification:
    - Xác định đối tượng thuộc loại nào.
    - Ứng dụng: Phát hiện thư rác, nhận dạng hình ảnh.
    - Thuật toán: SVM, Random Forest, Logistic regression, ...
  + Regression:

- Dự đoán một thuộc tính có giá trị liên tục gắn liền với một đối tượng.
- Ứng dụng: Giá nhà, giá cổ phiếu.
- Thuật toán: Linear Regression, Lasso, Rigde, …

+ Clustering:
  - Tự động nhóm các đối tượng tương tự thành các tập hợp.
  - Ứng dụng: Phân khúc khách hàng, nhóm các kết quả thí nghiệm.
  - Thuật toán: k-Means, BSCAN, hierarchical clustering, ...

- Chức năng tích hợp hỗ trợ xử lý dữ liệu và mô hình:

+ Dimensionality reduction:
  - Giảm số lượng biến ngẫu nhiên cần xem xét.
  - Ứng dụng: Trực quan hóa, tăng hiệu quả.
  - Thuật toán: PCA, feature selection, …

+ Model selection:
  - So sánh, xác nhận và lựa chọn các tham số và mô hình.
  - Ứng dụng: Độ chính xác được cải thiện thông qua việc điều chỉnh tham số.
  - Thuật toán: Grid search, cross validation, metrics, ...

+ Preprocessing:
  - Trích xuất đặc trưng và chuẩn hóa.
  - Ứng dụng: Chuyển đổi dữ liệu đầu vào, chẳng hạn như văn bản, để sử dụng với các thuật toán máy học.
  - Thuật toán: Preprocessing, feature extraction, ...

CHƯƠNG 3: PHƯƠNG PHÁP

3.1. Lý thuyết

3.1.1 Mô hình Logistic Regression

- Hồi quy logistic là một mô hình thống kê được sử dụng cho các bài toán phân loại, dự đoán xác suất một trường hợp thuộc về một trong số các lớp. Đây là một thuật toán học máy có giám sát.Các loại LogisticRegression:
  + Nhị phân: Trong hồi quy Logistic nhị phân, chỉ có thể có hai loại biến phụ thuộc, chẳng hạn như 0 hoặc 1, Đạt hoặc Không đạt, v.v.
  + Đa biến: Trong hồi quy Logistic đa biến, có thể có 3 hoặc nhiều hơn các loại không được sắp xếp khác nhau của biến phụ thuộc, chẳng hạn như "mèo", "chó" hoặc "cừu".
- Khai báo mô hình:
  + Bước 1: Khai báo từ thư viện

    from sklearn.linear_model import LogisticRegression
  + Bước 2: Gán cho biến

    logistic_model = LogisticRegression()
- Tham số:
  + penalty{‘l1’, ‘l2’, ‘elasticnet’, None}, default=’l2’
    - None: không có hình phạt nào được thêm vào
    - ‘l2’: Thêm hình phạt L2 và đó là lựa chọn mặc định.
    - ‘l1’: Thêm hình phạt L1
    - ‘elasticnet’: Thêm cả L1 và L2
  + max_inter: Tổng số lần dữ liệu được khớp với mô hình

3.1.2 SVM model

- Máy hỗ trợ vectơ (SVM) là một thuật toán học máy mạnh mẽ được sử dụng rộng rãi cho cả phân loại tuyến tính và phi tuyến tính, cũng như các tác vụ hồi quy và phát hiện ngoại lệ. SVM có khả năng thích ứng cao, làm cho chúng phù hợp với nhiều ứng dụng khác nhau như phân loại văn bản, phân

loại hình ảnh, phát hiện thư rác, nhận dạng chữ viết tay, phân tích biểu hiện gen, phát hiện khuôn mặt và phát hiện bất thường.

- SVM đặc biệt hiệu quả vì chúng tập trung vào việc tìm siêu mặt phẳng phân tách tối đa giữa các lớp khác nhau trong đặc trưng mục tiêu, làm cho chúng mạnh mẽ cho cả phân loại nhị phân và đa lớp. Trong bản tóm tắt này, chúng ta sẽ khám phá thuật toán Máy hỗ trợ vectơ (SVM), các ứng dụng của nó và cách nó xử lý hiệu quả cả phân loại tuyến tính và phi tuyến tính, cũng như các tác vụ hồi quy và phát hiện ngoại lệ.
- Khai báo mô hình:
  + Bước 1: Khai báo từ thư viện

    from sklearn.svm import SVC
  + Bước 2: Gán cho biến

    svm_model = SVC()
- Tham số:
  + max_inter: Tổng số lần dữ liệu được khớp với mô hình

3.1.3 Phương pháp đánh giá

3.1.3.1 Accuracy

Hàm accuracy_score tính toán độ chính xác, hoặc là tỷ lệ (mặc định) hoặc là số lượng (normalize=False) các dự đoán chính xác.

Trong phân loại đa nhãn, hàm này trả về độ chính xác của tập con. Nếu toàn bộ tập hợp các nhãn được dự đoán cho một mẫu hoàn toàn khớp với tập hợp nhãn thực, thì độ chính xác của tập con là 1.0; ngược lại là 0.0.

Nếu y^i là giá trị được dự đoán của mẫu thứ i và yi là giá trị thực tương ứng, thì tỷ lệ các dự đoán chính xác trên nsamples được định nghĩa là:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

3.1.3.2 Precision

Precision là tỷ lệ TP / (TP + FP), trong đó TP là số lượng dương tính thật và FP là số lượng dương tính giả. Về mặt trực quan, độ chính xác thể hiện khả năng của bộ phân loại không gán nhãn là dương tính cho một mẫu thực tế là âm tính. Giá trị tốt nhất là 1 và giá trị tệ nhất là 0.

3.1.3.3 Recall

Độ nhạy (recall) là tỷ lệ TP / (TP + FN), trong đó TP là số lượng dương tính thật và FN là số lượng âm tính giả. Về mặt trực quan, độ nhạy thể hiện khả năng của bộ phân loại trong việc tìm ra tất cả các mẫu dương tính. Giá trị tốt nhất là 1 và giá trị tệ nhất là 0.

3.2. Code

- Chuẩn bị dữ liệu cho mô hình

```
df1 = pd.concat([df_continuous, df_cat, df['Exited']], axis=1)
df1.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	HasCrCard	IsActiveMember	Geography	Gender	Exited
0	-0.326221	0.293517	-1.041760	-1.225848	-0.911583	0.021886	1	1	0	0	1
1	-0.440036	0.198164	-1.387538	0.117350	-0.911583	0.216534	0	1	2	0	0
2	-1.536794	0.293517	1.032908	1.333053	2.527057	0.240687	1	0	0	0	1
3	0.501521	0.007457	-1.387538	-1.225848	0.807737	-0.108918	0	0	0	0	0
4	2.063884	0.388871	-1.041760	0.785728	-0.911583	-0.365276	1	1	2	0	0

Hình 3.1: Chuẩn bị dữ liệu training

- Chia dữ liệu thành 2 tập train và test theo tỷ lệ 80:20

```
X = df1.drop('Exited', axis=1)
y = df1['Exited']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
print(y_train.value_counts())
```

```
Exited
0    6356
1    1644
Name: count, dtype: int64
```

Hình 3.2: Chia thành 2 tập train và test

- Xử lý mất cân bằng dữ liệu:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()
X_res, y_res = smote.fit_resample(X_train, y_train)
print(y_res.value_counts())
```

```
Exited
0    6356
1    6356
Name: count, dtype: int64
```

Hình 3.3: Xử lý mất cân bằng dữ liệu trên tập train

- Huấn luyện mô hình
  + LogisticRegression:

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Hình 3.4: Huấn luyện mô hình Logistic Regression

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.815
Precision: 0.5966386554621849
Recall: 0.1806615776081425
F1 Score: 0.27734375
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.97      0.89      1607
           1       0.60      0.18      0.28       393

    accuracy                           0.81      2000
   macro avg       0.71      0.58      0.59      2000
weighted avg       0.78      0.81      0.77      2000
```

Hình 3.5: Kết quả trước khi xử lý mất cân bằng (LR)

```
lr = LogisticRegression()
lr.fit(X_res, y_res)
y_pred = lr.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.714
Precision: 0.3712230215827338
Recall: 0.6564885496183206
F1 Score: 0.4742647058823529
Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.73      0.80      1607
           1       0.37      0.66      0.47       393

    accuracy                           0.71      2000
   macro avg       0.63      0.69      0.64      2000
weighted avg       0.79      0.71      0.74      2000
```

Hình 3.6: Kết quả sau khi xử lý mất cân bằng (LR)

+ SVM(support vector machine)

```
svc = SVC(probability=True)
svc.fit(X_train, y_train)
```

Hình 3.7: Huấn luyện mô hình SVM

```
svc = SVC(probability=True)
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.858
Precision: 0.8044692737430168
Recall: 0.366412213740458
F1 Score: 0.5034965034965035
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.98      0.92      1607
           1       0.80      0.37      0.50       393

    accuracy                           0.86      2000
   macro avg       0.83      0.67      0.71      2000
weighted avg       0.85      0.86      0.84      2000
```

Hình 3.8: Kết quả trước khi xử lý mất cân bằng (SVM)

```
svc = SVC(probability=True)
svc.fit(X_res, y_res)
y_pred = svc.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.8005
Precision: 0.494949494949495
Recall: 0.7480916030534351
F1 Score: 0.5957446808510638
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.81      0.87      1607
           1       0.49      0.75      0.60       393

    accuracy                           0.80      2000
   macro avg       0.71      0.78      0.73      2000
weighted avg       0.84      0.80      0.81      2000
```

Hình 3.9: Kết quả sau khi xử lý mất cân bằng (SVM)

+ Random Forest:

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Hình 3.10: Huấn luyện mô hình RandomForest

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.8675
Precision: 0.7711864406779662
Recall: 0.4631043256997455
F1 Score: 0.5786963434022258
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      1607
           1       0.77      0.46      0.58       393

    accuracy                           0.87      2000
   macro avg       0.83      0.71      0.75      2000
weighted avg       0.86      0.87      0.85      2000
```

Hình 3.11: Kết quả trước khi xử lý mất cân bằng (RF)

```
rf = RandomForestClassifier()
rf.fit(X_res, y_res)
y_pred = rf.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.844
Precision: 0.6025316455696202
Recall: 0.6055979643765903
F1 Score: 0.6040609137055838
Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.90      0.90      1607
           1       0.60      0.61      0.60       393

    accuracy                           0.84      2000
   macro avg       0.75      0.75      0.75      2000
weighted avg       0.84      0.84      0.84      2000
```

Hình 3.12: Kết quả sau khi xử lý mất cân bằng (RF)

TÀI LIỆU THAM KHẢO

[1] Vũ Hưu Tiệp. *Machine Learning cơ bản.*

[2] CS229: Machine Learning - https://cs229.stanford.edu/

[3] AIO Exercise Book v2025 - https://tutorial.aivietnam.edu.vn/pdf/43/info

[4] Wes MCKinney. *Python for Data Analysis* - Python-for-Data-Analysis.pdf