ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Môn học: PYTHON CHO KHOA HỌC DỮ LIỆU

DỰ ĐOÁN TỈ LỆ KHÁCH HÀNG RỜI ĐI

Sinh viên thực hiện:

1. Lê Nho Hãn – 22110054
2. Tạ Quang Duy – 22110047
3. Nguyễn Lê Khánh Duy – 22110046

Giảng viên môn học:

HÀ VĂN THẢO

1 GIỚI THIỆU

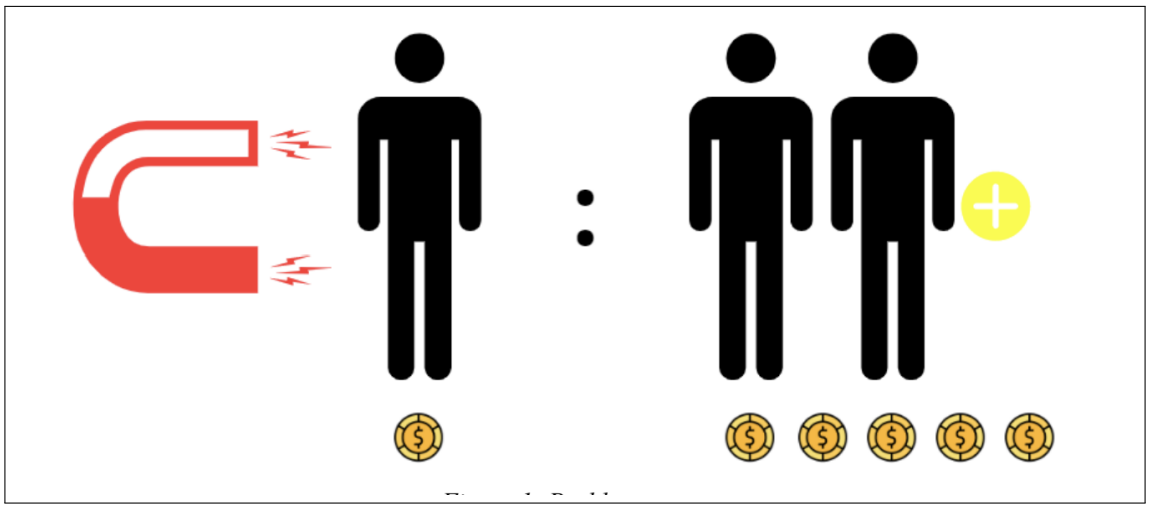
1.1 Giới thiệu đồ án

Khi bước vào kỷ nguyên dữ liệu, các doanh nghiệp thương mại phải đối mặt với sự cạnh tranh ngày càng tăng trên toàn cầu. Đại dịch năm 2020 càng làm gia tăng áp lực lên hệ thống quản lý mối quan hệ khách hàng của các doanh nghiệp. Để duy trì sự tăng trưởng doanh thu, các doanh nghiệp cần hiểu rõ hơn về nhu cầu và sở thích của khách hàng, đặc biệt là dự đoán tỷ lệ rời đi (churn) của khách hàng. Thông qua việc đặt mục tiêu và marketing, các doanh nghiệp có thể giảm thiểu thiệt hại doanh thu bằng cách giữ chân khách hàng có khả năng rời đi.

Customer Churn là mối quan tâm của nhiều doanh nghiệp thương mại điện tử. Bởi việc mất một khách hàng có thể dẫn đến những tác động tiêu cực như giảm thu nhập và lợi nhuận của doanh nghiệp, đồng thời có thể là dấu hiệu cảnh báo sự giảm sút về các sản phẩm, dịch vụ và chăm sóc khách hàng của doanh nghiệp đó.

"Don't Spend 5 Times More Attracting New Customers, Nurture The Existing Ones"

Dây là tựa đề một bài báo được đăng trên WORXPERTISE. Cho thấy việc giữ chân khách hàng thật sự cần thiết

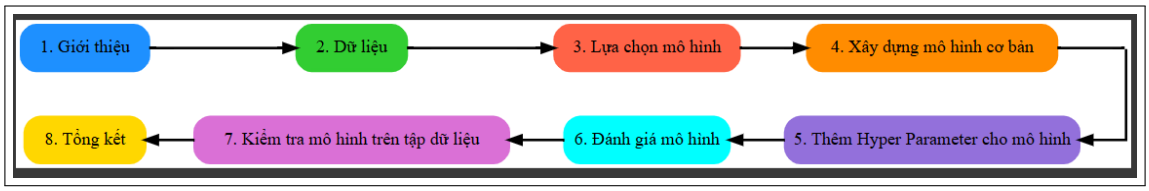


Tổng quan, mô hình giữ chân khách hàng có vẻ hiệu quả: doanh nghiệp tăng ngân sách marketing để giữ lại những khách hàng có khả năng rời đi, nhưng cuối cùng tạo ra đủ lợi nhuận để bù đắp chi phí bổ sung. Tuy nhiên, khi nhìn sâu vào vấn đề, mô hình giữ chân khách hàng lại tạo ra những vấn đề mới:

- Thứ nhất, không phải tất cả khách hàng có nguy cơ rời đi đều sẽ thực sự rời đi. Một số khách hàng có thể chỉ muốn thử xem liệu họ có thể nhận được các lợi ích thêm hay không, hoặc có thể ở lại dù các chỉ số của họ cho thấy họ có khả năng rời đi.
- Thứ hai, một số khách hàng sẽ rời đi bất kể họ nhận được khuyến mãi tài chính gì. Ví dụ, nếu một khách hàng quyết định ra nước ngoài định cư, họ có thể sẽ ngừng sử dụng dịch vụ bất chấp các ưu đãi giữ chân. Hoặc nếu các ưu đãi tài chính và các khuyến mãi không phải là sự quan tâm của 1 khách hàng, họ vẫn sẽ rời đi.

Vì vậy, các doanh nghiệp cần thiết kế và triển khai một mô hình dự đoán có thể ước tính được nguy cơ churn của khách hàng và can thiệp trước khi họ đưa ra quyết định cuối cùng. Mục tiêu chính của dự án này là áp dụng các kỹ thuật học máy tiên tiến để xây dựng mô hình dự đoán này.

1.2 Pipline



2 DỮ LIỆU

2.1 Giới thiệu

Giới thiệu về Dữ liệu

Dữ liệu The Telco Customer chứa thông tin về một công ty viễn thông hư cấu cung cấp dịch vụ điện thoại gia đình và Internet cho 7043 khách hàng tại California trên Kaggle. Dữ liệu này được sử dụng trong phân tích nhằm dự đoán hành vi rời bỏ (churn) của khách hàng, một vấn đề quan trọng trong việc quản lý và giữ chân khách hàng. Bộ dữ liệu bao gồm **7043 khách hàng** với **21 đặc điểm** khác nhau, được tổ chức thành các cột tương ứng với thông tin chi tiết về từng khách hàng.

Cấu trúc Dữ liệu
- **Churn**: Cột mục tiêu, chỉ ra liệu khách hàng có rời bỏ dịch vụ trong tháng qua hay không.
- **Dịch vụ Khách hàng đã đăng ký**: Các thông tin liên quan đến các dịch vụ mà khách hàng đang sử dụng, bao gồm:
- Dịch vụ điện thoại, nhiều đường dây điện thoại
- Dịch vụ internet, bảo mật trực tuyến, sao lưu trực tuyến
- Bảo vệ thiết bị, hỗ trợ kỹ thuật
- Dịch vụ phát trực tuyến TV và phim
- **Thông tin Tài khoản Khách hàng**:
- Thời gian sử dụng dịch vụ (thời gian khách hàng đã gắn bó)
- Loại hợp đồng, phương thức thanh toán
- Hóa đơn điện tử (có hoặc không), chi phí hàng tháng, tổng chi phí đã thanh toán
- **Thông tin Nhân khẩu học**:
- Giới tính, độ tuổi
- Trạng thái có đối tác hoặc người phụ thuộc

WA_Fn-UseC_-Telco-Customer-Churn.csv

2.2 Lấy dữ liệu

[ ]:
```
# Xử lý dữ liệu
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

```
# Trực quan hóa
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
# Lấy dữ liệu được lưu trên Google Drive
import gdown
gdown.download('https://drive.google.com/uc?
 ↪export=download&id=1G5ZBcJUxuEms_di057MhXPy4EUE6w3Bq',␣
 ↪'WA_Fn-UseC_-Telco-Customer-Churn.csv', quiet=False)
fil_path = '/sample_data'
```

```
Downloading...
From:
https://drive.google.com/uc?export=download&id=1G5ZBcJUxuEms_di057MhXPy4EUE6w3Bq
To: /content/WA_Fn-UseC_-Telco-Customer-Churn.csv
100%|| 978k/978k [00:00<00:00, 50.9MB/s]
```

[ ]:
```
# Đọc dữ liệu
data = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
data
```

[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
0	7590-VHVEG	Female	0	Yes	No	1	
1	5575-GNVDE	Male	0	No	No	34	
2	3668-QPYBK	Male	0	No	No	2	
3	7795-CFOCW	Male	0	No	No	45	
4	9237-HQITU	Female	0	No	No	2	
...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	
7039	2234-XADUH	Female	0	Yes	Yes	72	
7040	4801-JZAZL	Female	0	Yes	Yes	11	
7041	8361-LTMKD	Male	1	Yes	No	4	
7042	3186-AJIEK	Male	0	No	No	66	

	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	\
0	No	No phone service	DSL	No	...	
1	Yes	No	DSL	Yes	...	
2	Yes	No	DSL	Yes	...	
3	No	No phone service	DSL	Yes	...	
4	Yes	No	Fiber optic	No	...	
...	...	...	...	...	...	...
7038	Yes	Yes	DSL	Yes	...	
7039	Yes	Yes	Fiber optic	No	...	
7040	No	No phone service	DSL	Yes	...	
7041	Yes	Yes	Fiber optic	No	...	
7042	Yes	No	Fiber optic	Yes	...	

```
     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
...               ...         ...         ...             ...             ...
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95        1889.5
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70        151.65
...               ...                        ...            ...           ...
7038              Yes               Mailed check          84.80        1990.5
7039              Yes    Credit card (automatic)         103.20        7362.9
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40         306.6
7042              Yes  Bank transfer (automatic)         105.65        6844.5

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
...    ...
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]
```

[ ]:
```
# Xem tóm tắt thông tin về dữ liệu
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
```

```
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Có thể hiểu dữ liệu có các cột như sau:

+ CustomerID: Mã khách hàng.

+ gender: Giới tính của khách hàng.

+ SeniorCitizen: Có phải hoặc không phải người già, đặc biệt là người đã nghỉ hưu hoặc nhận lương hưu.

+ Partner: Có hay không có bạn đời.

+ Dependents: Có hay không có người phụ thuộc (là người mà khách hàng phải nuôi dưỡng).

+ tenure: Thời gian sử dụng dịch vụ.

+ PhoneService: Có hay không sử dụng dịch vụ điện thoại.

+ MultileLines: Có hay không sử dụng nhiều đường dây điện thoại.

+ InternetService: Loại dịch vụ Internet.

+ OnlineSecurity: Có hay không sử dụng dịch vụ bảo mật trực tuyến.

+ OnlineBackup: Có hay không sử dụng dịch vụ sao lưu trực tuyến. + DeviceProtection: Có hay không sử dụng dịch vụ bảo vệ thiết bị.

+ TechSupport: Có hay không sử dụng dịch vụ hỗ trợ kĩ thuật.

+ StreamingTV: Có hay không sử dụng dịch vụ truyền hình trực tuyến.

+ StreamingMovies: Có hay không sử dụng dịch vụ xem phim trực tuyến.

+ Contract: Loại hợp đồng.

+ PaperlessBilling: Có hay không sử dụng hóa đơn điện tử.

+ PaymentMethod: Phương thức thanh toán.

+ MonthlyCharges: Số tiền mà khách hàng phải trả mỗi tháng.

+ TotalCharges: Tổng số tiền mà khách hàng đã trả.

+ Churn: Khách hàng ngừng sử dụng dịch vụ hay vẫn còn sử dụng dịch vụ.

2.3 Xử lý và trực quan dữ liệu

2.3.1 Xử lý dữ liệu

[ ]:
```
# Đầu tiên kiểm tra các khoảng trắng tồn tại trong các cột chứa dữ liệu kiểu␣
↪'object' có thể ảnh hưởng đến tính toán.
empty_values = data.select_dtypes(include=['object']).apply(lambda x: x.str.
↪strip() == '').sum()
print(empty_values)
```

```
customerID           0
gender               0
Partner              0
Dependents           0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
TotalCharges        11
Churn                0
dtype: int64
```

[ ]:
```
# Cột TotalCharges chứa 11 khoảng trắng, các cột khác không có.
# Tìm hiểu các hàng này.
empty_rows = data[data['TotalCharges'].str.strip() == '']
print(empty_rows)
```

```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
488   4472-LVYGI  Female              0     Yes        Yes       0
753   3115-CZMZD    Male              0      No        Yes       0
```

```
936   5709-LVOEQ  Female              0     Yes        Yes       0
1082  4367-NUYAO    Male              0     Yes        Yes       0
1340  1371-DWPAZ  Female              0     Yes        Yes       0
3331  7644-OMVMY    Male              0     Yes        Yes       0
3826  3213-VVOLG    Male              0     Yes        Yes       0
4380  2520-SGTTA  Female              0     Yes        Yes       0
5218  2923-ARZLG    Male              0     Yes        Yes       0
6670  4075-WKNIU  Female              0     Yes        Yes       0
6754  2775-SEFEE    Male              0      No        Yes       0

     PhoneService     MultipleLines InternetService       OnlineSecurity  ...  \
488            No  No phone service             DSL                  Yes  ...
753           Yes                No              No  No internet service  ...
936           Yes                No             DSL                  Yes  ...
1082          Yes               Yes              No  No internet service  ...
1340           No  No phone service             DSL                  Yes  ...
3331          Yes                No              No  No internet service  ...
3826          Yes               Yes              No  No internet service  ...
4380          Yes                No              No  No internet service  ...
5218          Yes                No              No  No internet service  ...
6670          Yes               Yes             DSL                   No  ...
6754          Yes               Yes             DSL                  Yes  ...

         DeviceProtection          TechSupport          StreamingTV  \
488                   Yes                  Yes                  Yes
753   No internet service  No internet service  No internet service
936                   Yes                   No                  Yes
1082  No internet service  No internet service  No internet service
1340                  Yes                  Yes                  Yes
3331  No internet service  No internet service  No internet service
3826  No internet service  No internet service  No internet service
4380  No internet service  No internet service  No internet service
5218  No internet service  No internet service  No internet service
6670                  Yes                  Yes                  Yes
6754                   No                  Yes                   No

          StreamingMovies  Contract PaperlessBilling  \
488                    No  Two year              Yes
753   No internet service  Two year               No
936                   Yes  Two year               No
1082  No internet service  Two year               No
1340                   No  Two year               No
3331  No internet service  Two year               No
3826  No internet service  Two year               No
4380  No internet service  Two year               No
5218  No internet service  One year              Yes
6670                   No  Two year               No
6754                   No  Two year              Yes
```

```
                  PaymentMethod MonthlyCharges  TotalCharges Churn
488   Bank transfer (automatic)          52.55                No
753                Mailed check          20.25                No
936                Mailed check          80.85                No
1082               Mailed check          25.75                No
1340    Credit card (automatic)          56.05                No
3331               Mailed check          19.85                No
3826               Mailed check          25.35                No
4380               Mailed check          20.00                No
5218               Mailed check          19.70                No
6670               Mailed check          73.35                No
6754  Bank transfer (automatic)          61.90                No

[11 rows x 21 columns]
```

[ ]:
```
# Nhận thấy rằng TotalCharges ở những hàng này không có giá trị bởi vì số tháng gắn bó với dịch vụ (tenure) là 0, vì vậy sẽ được bằng giá trị 0.
# Chuyển dữ liệu ở cột này từ 'object' sang 'float64' sau đó thay thế NaN (Not a Number) bằng 0
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors = 'coerce')
data['TotalCharges'] = data['TotalCharges'].fillna(0)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
```

```
 19  TotalCharges      7043 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

[ ]:
```
print(data.isnull().sum())
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

2.3.2 Trực quan dữ liệu

[ ]:
```
# Thống kê cho dữ liệu kiểu số
data.describe()
```

[ ]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

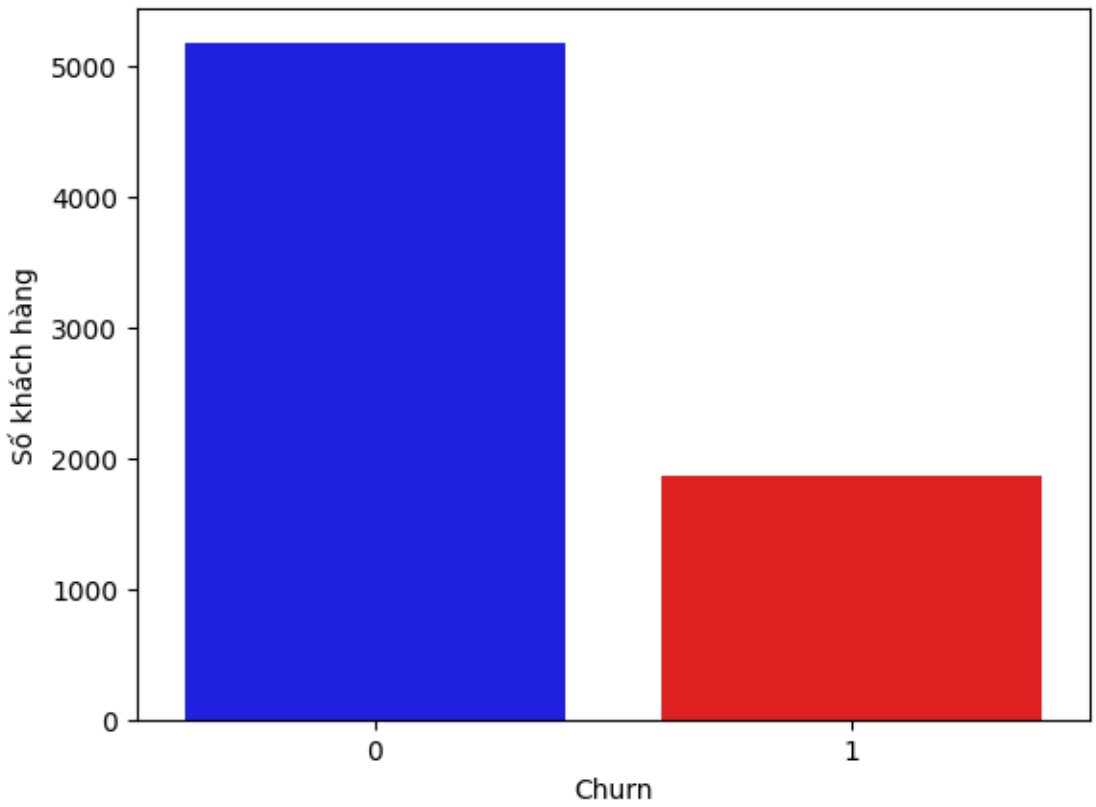
[ ]:
```
# Thống kê cho biến phân loại
data.describe(include = 'object')
```

[ ]:
```
        customerID gender Partner Dependents PhoneService MultipleLines  \
count         7043   7043    7043       7043         7043          7043
unique        7043      2       2          2            2             3
top     7590-VHVEG   Male      No         No          Yes            No
freq             1   3555    3641       4933         6361          3390

       InternetService OnlineSecurity OnlineBackup DeviceProtection  \
count             7043           7043         7043             7043
unique               3              3            3                3
top        Fiber optic             No           No               No
freq              3096           3498         3088             3095

       TechSupport StreamingTV StreamingMovies        Contract  \
count         7043        7043            7043            7043
unique           3           3               3               3
top             No          No              No  Month-to-month
freq          3473        2810            2785            3875

       PaperlessBilling     PaymentMethod Churn
count              7043              7043  7043
unique                2                 4     2
top                 Yes  Electronic check    No
freq               4171              2365  5174
```

[ ]:
```
# Đầu tiên chuyển Churn từ Yes/No thành 1/0 để dễ dàng vẽ các biểu đồ trực quan␣
↪xoay quanh Churn
data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})
```

[ ]:
```
# Xem sự chênh lệch giữa khách hàng rời bỏ dịch vụ so với khách hàng không rời␣
↪bỏ dịch vụ
sns.countplot(x = data['Churn'], palette = ['blue', 'red'])
plt.ylabel("Số khách hàng")
plt.show()
```

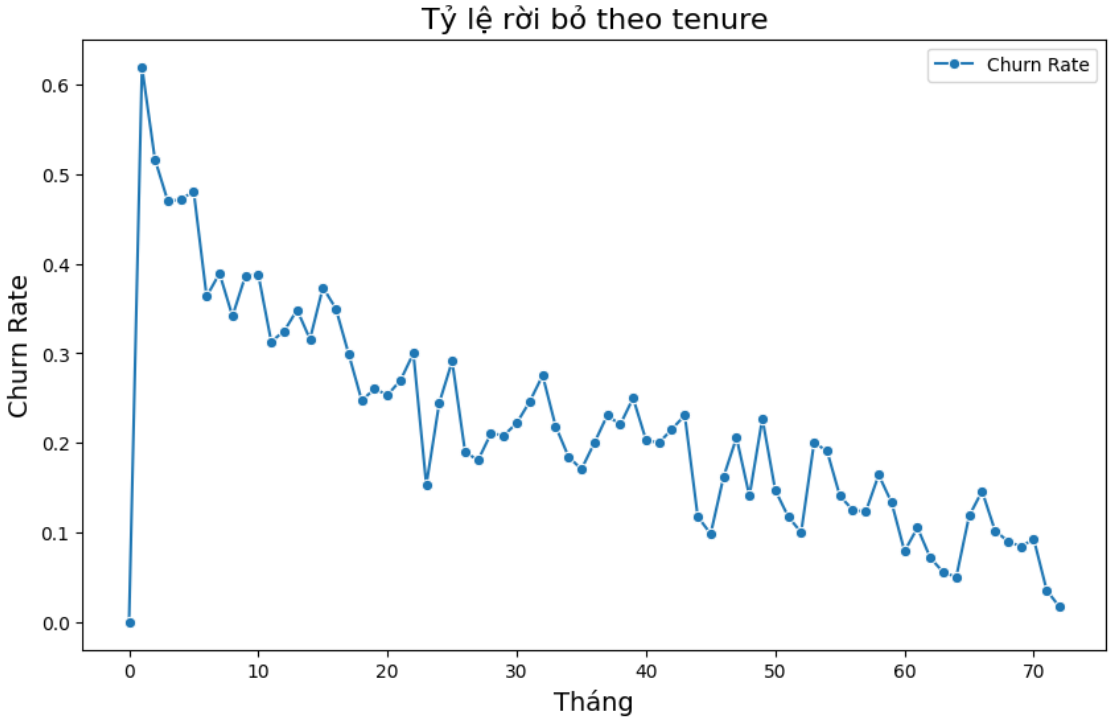
```
<ipython-input-12-2103414f44ab>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x = data['Churn'], palette = ['blue', 'red'])
```



[ ]:
```
# Tính tỷ lệ Churn theo tenure
churn_tenure = data.groupby('tenure')['Churn'].mean().reset_index()
print(churn_tenure)
# Biểu đồ churn rate theo tenure
plt.figure(figsize=(10, 6))
sns.lineplot(data=churn_tenure, x='tenure', y='Churn', marker='o', label='Churn␣
↪Rate')
plt.title('Tỷ lệ rời bỏ theo tenure', fontsize=16)
plt.xlabel('Tháng', fontsize=14)
plt.ylabel('Churn Rate', fontsize=14)
plt.show()
```

```
    tenure     Churn
0        0  0.000000
1        1  0.619902
2        2  0.516807
3        3  0.470000
4        4  0.471591
..     ...       ...
68      68  0.090000
69      69  0.084211
```

```
70       70  0.092437
71       71  0.035294
72       72  0.016575

[73 rows x 2 columns]
```



Trong 10 tháng đầu tiên, tỉ lệ rời bỏ khá cao và có sự biến động lớn. Điều này có thể cho thấy giai đoạn đầu là giai đoạn khách hàng đánh giá và quyết định có tiếp tục sử dụng dịch vụ hay không.

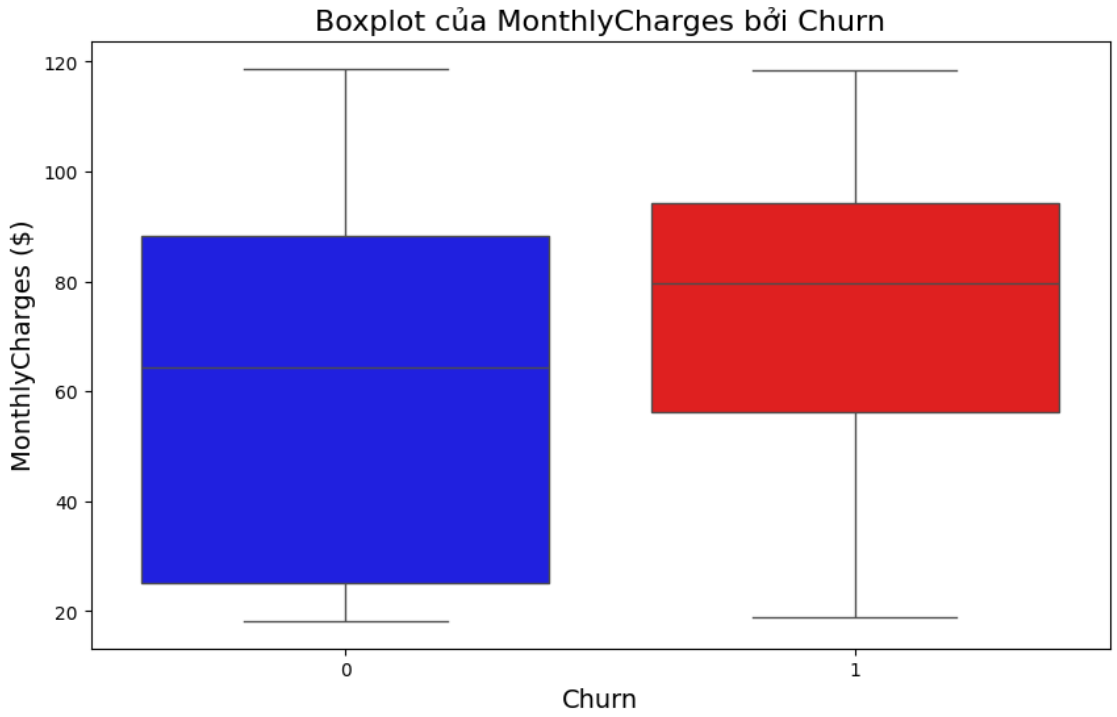
Khi thời gian gắn bó tăng, tỷ lệ rời bỏ giảm dần. Có thể là do khách hàng đã thích nghi hoặc hài lòng với dịch vụ.

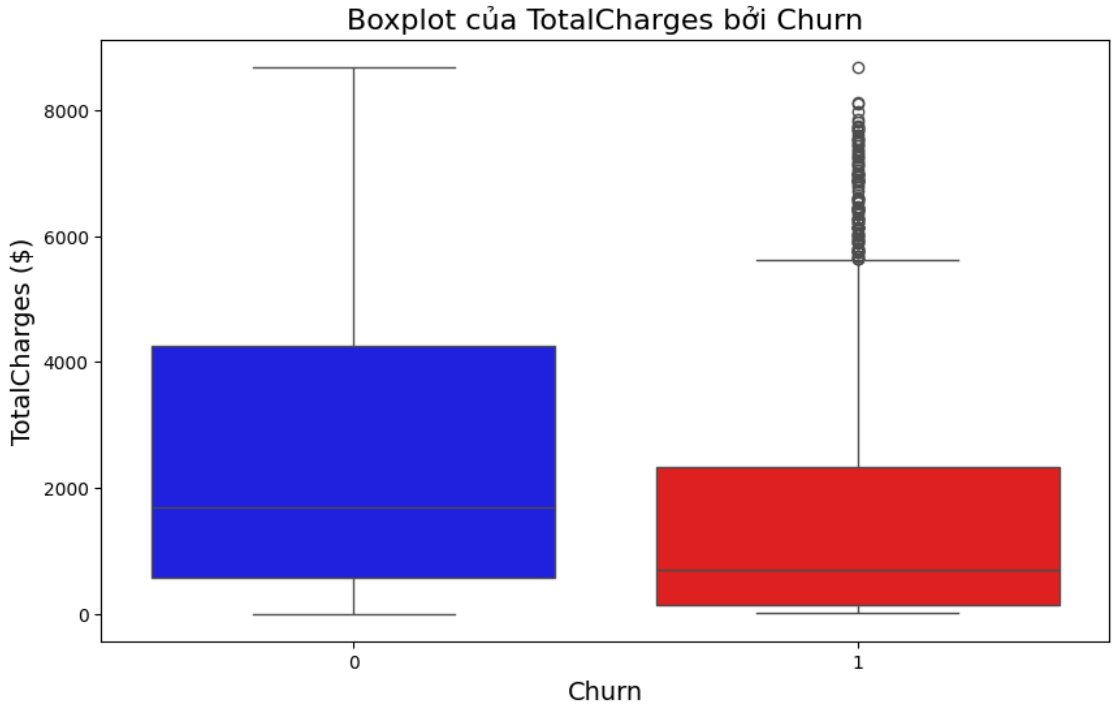
Có sự dao động nhẹ về tỷ lệ rời bỏ trong khoảng từ 10-30 tháng. Điều này có thể phản ánh các yếu tố như sự thay đổi của nhu cầu khách hàng hoặc tác động từ các yếu tố bên ngoài.

[ ]:
```python
# Vẽ biểu đồ Churn xoay quanh Charges
charge_columns = ['MonthlyCharges', 'TotalCharges']
for column in charge_columns:
    plt.figure(figsize = (10, 6))
    sns.boxplot(x ='Churn', y = column, data = data, palette = ['blue', 'red'])
    plt.title(f'Boxplot của {column} bởi Churn', fontsize = 16)
    plt.xlabel('Churn', fontsize = 14)
    plt.ylabel(column + ' ($)', fontsize = 14)
```

```
<ipython-input-14-0c4289c71267>:5: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x ='Churn', y = column, data = data, palette = ['blue', 'red'])
<ipython-input-14-0c4289c71267>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x ='Churn', y = column, data = data, palette = ['blue', 'red'])
```





Ở biểu đồ hộp của MonthlyCharges:

+ Đối với khách hàng vẫn còn sử dụng dịch vụ (Churn = 0), phần lớn khách hàng trong nhóm này có phí hàng tháng nằm trong khoảng từ 20 USD đến 80 USD.

+ Đối với khách hàng đã rời bỏ dịch vụ (Churn = 1), phần lớn khách hàng trong nhóm này có phí hàng tháng nằm trong khoảng từ 70 USD đến 100 USD, cho thấy chi phí cao có thể là yếu tố dẫn đến việc rời bỏ dịch vụ.

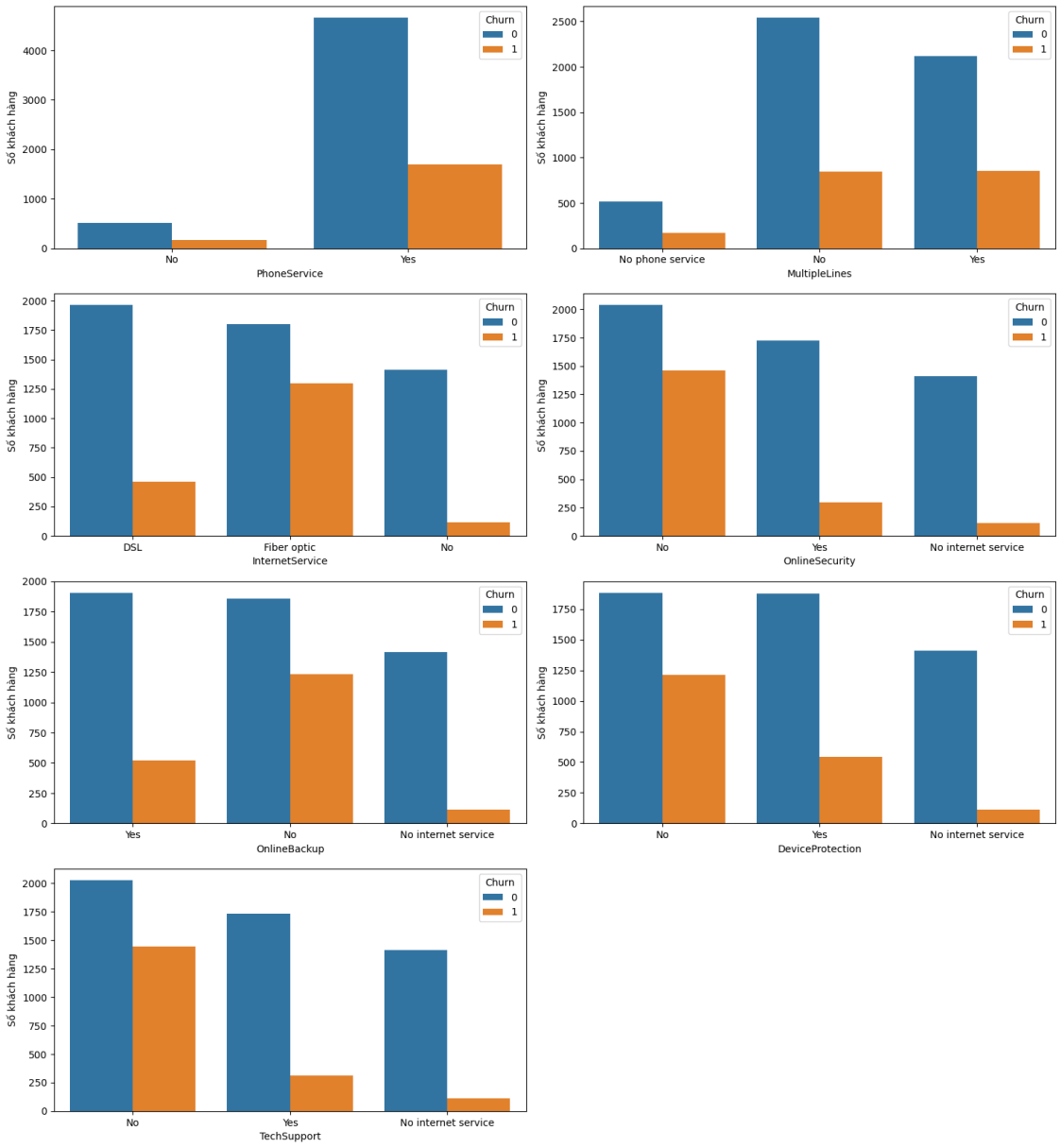
Ở biểu đồ hộp của TotalCharges:

+ Đối với khách hàng vẫn còn sử dụng dịch vụ (Churn = 0), phần lớn khách hàng trong nhóm này có tổng chi phí (TotalCharges) nằm trong khoảng từ 0 USD đến 4,000 USD.

+ Đối với khách hàng đã rời bỏ dịch vụ (Churn = 1), phần lớn tổng chi phí nằm trong khoảng từ 500 USD đến 2,500 USD, nhưng có một số giá trị ngoại lai cao vượt mức 4,000 USD, cho thấy một số khách hàng dù chi tiêu nhiều nhưng vẫn quyết định rời bỏ dịch vụ.

[ ]:
```
# Biểu đồ của Churn xoay quanh các biến phân loại
plt.figure(figsize=(15,40))
category_columns = ['PhoneService', 'MultipleLines', 'InternetService',␣
 ↪'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport']
for i,col in enumerate(category_columns):
    plt.subplot(10,2,i+1)
    sns.countplot(x = data[col], hue = data['Churn'], orient='h')
    plt.ylabel('Số khách hàng')
```

```
plt.tight_layout()
plt.show()
```



Khách hàng có xu hướng rời bỏ dịch vụ (churn) nhiều hơn khi họ không sử dụng các dịch vụ bổ sung như bảo mật trực tuyến (OnlineSecurity), sao lưu dữ liệu (OnlineBackup), hỗ trợ kỹ thuật (TechSupport), hoặc khi họ sử dụng dịch vụ Internet (InternetService).

Điều này cho thấy các dịch vụ bổ sung có thể ảnh hưởng đáng kể đến tỷ lệ giữ chân khách hàng.

2.4 Chuẩn bị dữ liệu để áp dụng các mô hình

2.4.1 Chuyển đổi dữ liệu

[ ]:
```
# Các cột chứa dữ liệu dạng phân loại và không phải ở dạng số, ta sẽ chuyển các␣
↪dữ liệu phân loại này về dạng số bằng Label Encoder
# Đầu tiên tạo danh sách các cột này:
classify_column =  ['gender', 'Partner', 'Dependents', 'PhoneService',␣
↪'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
                    'DeviceProtection', 'TechSupport', 'StreamingTV',␣
↪'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']
```

[ ]:
```
# Tiếp đến, chúng ta chuyển các giá trị phân loại trong các cột thành số
for column in classify_column:
    if column in data.columns:
        le = LabelEncoder()  # Tạo một LabelEncoder mới cho mỗi cột
        data[column] = le.fit_transform(data[column])

        # Hiển thị ánh xạ chuyển đổi
        print(f"Column: {column}")
        for index, class_value in enumerate(le.classes_):
            print(f"   Giá trị ban đầu: {class_value} -> Giá trị sau khi chuyển␣
↪đổi: {index}")
        print("\n")

data
```

```
Column: gender
   Giá trị ban đầu: Female -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Male -> Giá trị sau khi chuyển đổi: 1


Column: Partner
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 1


Column: Dependents
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 1


Column: PhoneService
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 1
```

```
Column: MultipleLines
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No phone service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2


Column: InternetService
   Giá trị ban đầu: DSL -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Fiber optic -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 2


Column: OnlineSecurity
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No internet service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2


Column: OnlineBackup
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No internet service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2


Column: DeviceProtection
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No internet service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2


Column: TechSupport
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No internet service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2


Column: StreamingTV
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No internet service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2


Column: StreamingMovies
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: No internet service -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 2
```

```
Column: Contract
   Giá trị ban đầu: Month-to-month -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: One year -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Two year -> Giá trị sau khi chuyển đổi: 2


Column: PaperlessBilling
   Giá trị ban đầu: No -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Yes -> Giá trị sau khi chuyển đổi: 1


Column: PaymentMethod
   Giá trị ban đầu: Bank transfer (automatic) -> Giá trị sau khi chuyển đổi: 0
   Giá trị ban đầu: Credit card (automatic) -> Giá trị sau khi chuyển đổi: 1
   Giá trị ban đầu: Electronic check -> Giá trị sau khi chuyển đổi: 2
   Giá trị ban đầu: Mailed check -> Giá trị sau khi chuyển đổi: 3
```

[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
0	7590-VHVEG	0	0	1	0	1	
1	5575-GNVDE	1	0	0	0	34	
2	3668-QPYBK	1	0	0	0	2	
3	7795-CFOCW	1	0	0	0	45	
4	9237-HQITU	0	0	0	0	2	
...	...	...	...	...	...	...	
7038	6840-RESVB	1	0	1	1	24	
7039	2234-XADUH	0	0	1	1	72	
7040	4801-JZAZL	0	0	1	1	11	
7041	8361-LTMKD	1	1	1	0	4	
7042	3186-AJIEK	1	0	0	0	66	

	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	\
0	0	1	0	0	...	
1	1	0	0	2	...	
2	1	0	0	2	...	
3	0	1	0	2	...	
4	1	0	1	0	...	
...	...	...	...	...	...	
7038	1	2	0	2	...	
7039	1	2	1	0	...	
7040	0	1	0	2	...	
7041	1	2	1	0	...	
7042	1	0	1	2	...	

```
      DeviceProtection  TechSupport  StreamingTV  StreamingMovies  Contract  \
0                    0            0            0                0         0
1                    2            0            0                0         1
2                    0            0            0                0         0
3                    2            2            0                0         1
4                    0            0            0                0         0
...                ...          ...          ...              ...       ...
7038                 2            2            2                2         1
7039                 2            0            2                2         1
7040                 0            0            0                0         0
7041                 0            0            0                0         0
7042                 2            2            2                2         2

      PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
0                    1              2           29.85         29.85      0
1                    0              3           56.95       1889.50      0
2                    1              3           53.85        108.15      1
3                    0              0           42.30       1840.75      0
4                    1              2           70.70        151.65      1
...                ...            ...             ...           ...    ...
7038                 1              3           84.80       1990.50      0
7039                 1              1          103.20       7362.90      0
7040                 1              2           29.60        346.45      0
7041                 1              3           74.40        306.60      1
7042                 1              0          105.65       6844.50      0

[7043 rows x 21 columns]
```

[ ]:
```
# Loại bỏ cột customerID vì không có ảnh hưởng đến các mô hình học máy
data = data.drop(columns = 'customerID')
```

2.4.2 Cân bằng dữ liệu

Phương pháp được áp dụng trong bài này là kỹ thuật Oversampling thông qua SMOTE (Synthetic Minority Over-sampling Technique).

SMOTE (Synthetic Minority Over-sampling Technique) là một kỹ thuật tăng cường mẫu cho các lớp thiểu số trong các bài toán phân loại không cân bằng.

SMOTE tạo ra các mẫu mới cho lớp thiểu số bằng cách nội suy giữa các điểm dữ liệu trong lớp thiểu số. Cụ thể, nó chọn một điểm dữ liệu ngẫu nhiên từ lớp thiểu số và tạo ra các điểm mới bằng cách kết hợp nó với các điểm láng giềng gần nhất của nó. Phương pháp này giúp cải thiện hiệu suất của các mô hình phân loại khi xử lý các tập dữ liệu không cân bằng, giúp mô hình nhận diện tốt hơn lớp thiểu số.

[ ]:
```
X = data.drop(columns = "Churn")
# X là tập các features chúng ta dùng dự đoán.
y = data["Churn"].values
```

```
# Y là target, kết quả chúng ta muốn dự đoán
```

[ ]:
```
from imblearn.over_sampling import SMOTE
from collections import Counter

smote = SMOTE(random_state=42) #Khởi tạo SMOTE
X_resampled, y_resampled = smote.fit_resample(X, y) # Cân bằng dữ liệu với SMOTE
# In kết quả sau khi cân bằng
print("Trước khi cân bằng:", Counter(y))
print("Sau khi cân bằng với SMOTE:", Counter(y_resampled))
```

```
Trước khi cân bằng: Counter({0: 5174, 1: 1869})
Sau khi cân bằng với SMOTE: Counter({0: 5174, 1: 5174})
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:474: FutureWarning:
`BaseEstimator._validate_data` is deprecated in 1.6 and will be removed in 1.7.
Use `sklearn.utils.validation.validate_data` instead. This function becomes
public and is part of the scikit-learn developer API.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/utils/_tags.py:354:
FutureWarning: The SMOTE or classes from which it inherits use `_get_tags` and
`_more_tags`. Please define the `__sklearn_tags__` method, or inherit from
`sklearn.base.BaseEstimator` and/or other appropriate mixins such as
`sklearn.base.TransformerMixin`, `sklearn.base.ClassifierMixin`,
`sklearn.base.RegressorMixin`, and `sklearn.base.OutlierMixin`. From scikit-
learn 1.7, not defining `__sklearn_tags__` will raise an error.
  warnings.warn(
```

2.4.3 Chia dữ liệu thành tập train và tập test

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,␣
↪test_size = 0.3, random_state = 4, stratify = y_resampled)
# Chia dữ liệu thành tập train và test với tỉ lệ 7:3
```

[ ]:
```
col=['TotalCharges','tenure','MonthlyCharges']
```

[ ]:
```
scaler = StandardScaler()

# Fit scaler trên tập huấn luyện
scaler.fit(X_train[col])

# Sử dụng scaler đã fit để chuẩn hóa
X_train[col] = scaler.transform(X_train[col])
X_test[col] = scaler.transform(X_test[col])
```

[ ]:
```
print('After train test split, the training set has {} rows and {} columns.'.
 ↪format(X_train.shape[0], X_train.shape[1]))
print('After train test split, the train has {} labels.'.format(y_train.
 ↪shape[0]))
print('After train test split, the test set has {} rows and {} columns.'.
 ↪format(X_test.shape[0], X_test.shape[1]))
print('After train test split, the test has {} labels.'.format(y_test.shape[0]))
```

```
After train test split, the training set has 7243 rows and 19 columns.
After train test split, the train has 7243 labels.
After train test split, the test set has 3105 rows and 19 columns.
After train test split, the test has 3105 labels.
```

[ ]:
```
X_train.head()
```

[ ]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	\
5799	0	0	0	0	-0.619236	0	
7456	0	0	1	0	0.092496	0	
10007	0	0	0	0	-0.954169	1	
1993	0	0	1	1	-1.121636	1	
8804	0	0	0	0	-1.121636	1	

	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	\
5799	1	0	0	0	
7456	0	0	0	1	
10007	0	1	0	1	
1993	0	0	0	0	
8804	0	1	0	0	

	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	\
5799	0	0	0	2	0	
7456	0	0	0	0	0	
10007	0	0	0	2	0	
1993	0	0	0	0	0	
8804	0	0	0	0	0	

	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
5799	1	2	-1.164530	-0.738193
7456	0	0	-0.617786	-0.235188
10007	1	2	0.566605	-0.704650
1993	1	2	-0.792991	-0.922276
8804	1	2	0.088420	-0.910715

3 LỰA CHỌN MÔ HÌNH

3.1 Mục tiêu

Tìm kiếm mô hình phù hợp nhất để phân loại chính xác giữa khách hàng rời bỏ và không rời bỏ và xác định những khách hàng có khả năng rời bỏ cao trong các mô hình học máy có giám sát.

Hiện nay, có rất nhiều thuật toán khai phá dữ liệu có thể áp dụng cho bài toán này. Tuy nhiên, mỗi thuật toán đều có những ưu và nhược điểm riêng, phù hợp với các tình huống và loại dữ liệu khác nhau. Sau khi xem xét các đặc điểm của bài toán và bộ dữ liệu, chúng tôi quyết định lựa chọn ba thuật toán chính:

- Logistic Regression
- Decision Tree.
- Random Forest.

Trong quá trình huấn luyện, chúng tôi cải thiện hiệu suất mô hình bằng cách liên tục kiểm tra kết quả dự đoán của mô hình với kết quả thực tế trong tập dữ liệu xác thực.

Sau khi hoàn thành huấn luyện, chúng tôi áp dụng mô hình vào tập kiểm tra và tạo ra các kết quả dự đoán thực tế.

3.1.1 Logistic Regression

- Đây là một thuật toán cổ điển và phổ biến trong các bài toán dự đoán phân loại.
- Logistic Regression hoạt động tốt với các tập dữ liệu nhỏ.
- Hồi quy logistic thường được sử dụng khi chúng ta muốn tìm mối quan hệ giữa các biến độc lập và một biến phụ thuộc dựa trên xác suất.

3.1.2 Decision Tree

Là thuật toán học có giám sát, dùng cho phân loại và hồi quy. Dữ liệu được chia nhỏ qua các nút dựa trên điều kiện từ các đặc trưng.

Ưu điểm:

- Dễ hiểu, dễ triển khai.
- Không cần chuẩn hóa dữ liệu.
- Xử lý tốt dữ liệu số và danh mục.

3.1.3 Random Forest

Random Forest là một thuật toán học máy ensemble (tập hợp) kết hợp nhiều Decision Trees để cải thiện độ chính xác và giảm overfitting. Nó sử dụng bagging (bootstrap aggregating) để tạo ra nhiều mô hình cây quyết định từ các mẫu dữ liệu ngẫu nhiên.

Ưu điểm:

- Giảm overfitting: Do kết hợp nhiều cây quyết định.
- Khả năng tổng quát tốt: Hiệu quả cao trên nhiều loại dữ liệu khác nhau.

- Dễ sử dụng: Không cần chuẩn hóa dữ liệu và dễ áp dụng với dữ liệu phức tạp.
- Đo lường tầm quan trọng của đặc trưng: Random Forest có thể đánh giá độ quan trọng của từng đặc trưng.

3.2 Các chỉ số đánh giá mô hình

Các định nghĩa:

Lớp Positive: churn.

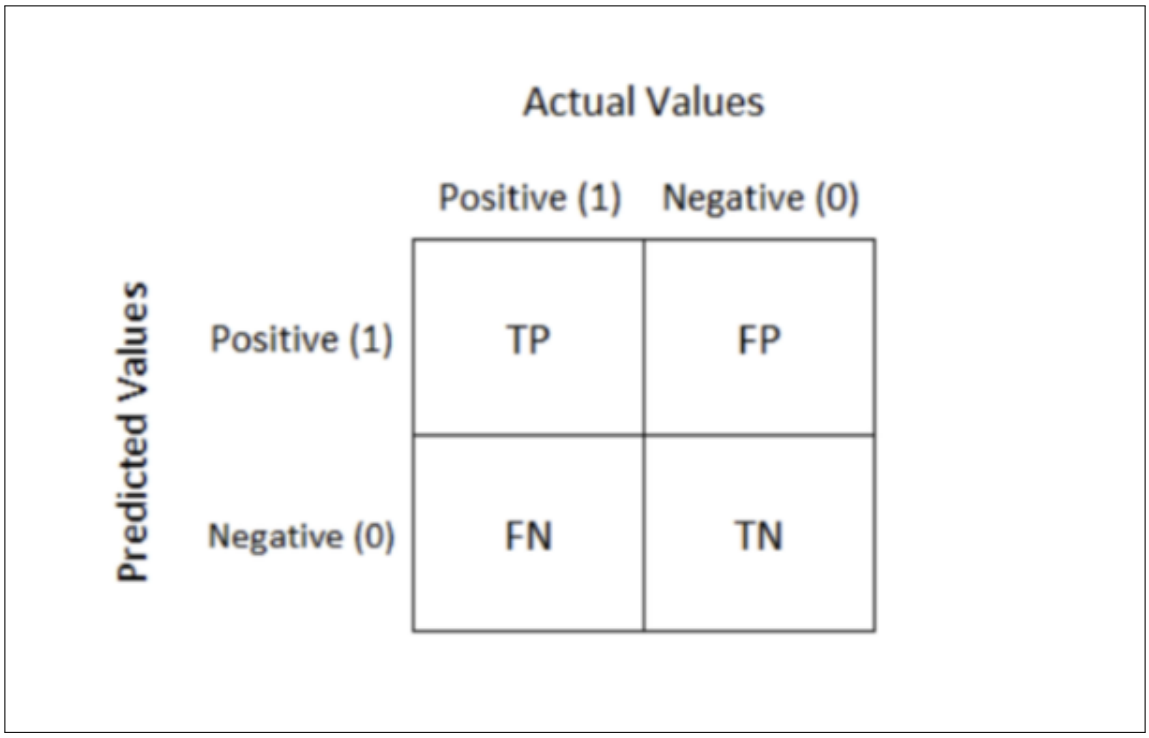
Lớp Negative : stayed.

True Positive (TP): Là kết quả khi mô hình dự đoán đúng lớp Positive.

Tương tự, True Negative (TN) là kết quả khi mô hình dự đoán đúng lớp Negative.

False Positive (FP): Là kết quả khi mô hình dự đoán sai lớp Positive.

False Negative (FN) là kết quả khi mô hình dự đoán sai lớp Negative.



1. Accuracy (Độ chính xác)

Định nghĩa: Độ chính xác là tỷ lệ dự đoán đúng so với tổng số dự đoán. Tính toán bằng công thức:

$$\text{Accuracy} = \frac{\text{Số dự đoán đúng}}{\text{Tổng số dự đoán}}$$

Ý nghĩa: Độ chính xác cho biết phần trăm dự đoán mà mô hình thực hiện đúng.

2. Precision (Độ chính xác của nhóm positive):

Định nghĩa: Precision đo lường mức độ chính xác của các dự đoán positive, tức là tỷ lệ dự đoán positive thực sự đúng. Tính toán bằng công thức:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Ý nghĩa: Precision cho biết mức độ đáng tin cậy của các dự đoán positive. Một Precision cao nghĩa là khi mô hình dự đoán một mẫu là positive, có xác suất rất cao rằng mẫu đó thực sự là positive.

3. Recall :

Định nghĩa: Recall đo lường khả năng của mô hình trong việc phát hiện tất cả các mẫu Positive, tức là tỷ lệ các trường hợp Positive thực sự được mô hình nhận diện đúng. Công thức tính Recall là:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Ý nghĩa: Recall quan trọng trong các bài toán mà việc bỏ lỡ các trường hợp Positive (False Negatives) là nghiêm trọng, ví dụ như trong y tế khi mô hình không nhận diện được bệnh nhân mắc bệnh.

4. F1 Score (Điểm F1):

Đây được gọi là một trung bình điều hòa (harmonic mean) của các tiêu chí Precision và Recall. Nó có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị Precision và Recall và đồng thời nó có giá trị lớn nếu cả 2 giá trị Precision và Recall đều lớn. Chính vì thế F1-Score thể hiện được một cách khách quan hơn performance của một mô hình học máy.

Công thức tính F1 Score là:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. ROC AUC

ROC-AUC là công cụ mạnh để đánh giá và so sánh mô hình phân loại. AUC càng cao, mô hình càng phân biệt tốt giữa các lớp.

4 XÂY DỰNG MÔ HÌNH CƠ BẢN

4.1 K-FOLD CROSS-VALIDATION

K-Fold Cross-Validation (CV) là một phương pháp được sử dụng trong học máy để đánh giá hiệu suất của mô hình và giúp xác định khả năng tổng quát (generalization) của mô hình. Phương pháp này giúp giảm thiểu vấn đề overfitting, nơi mô hình quá phù hợp với dữ liệu huấn luyện và không thể tổng quát tốt khi gặp dữ liệu mới.

Ưu điểm của K-Fold Cross-Validation:

Giảm thiểu overfitting: Do sử dụng tất cả các phần của dữ liệu để huấn luyện và kiểm tra, phương pháp này giúp giảm sự phụ thuộc vào một phần dữ liệu duy nhất.

Hiệu suất ổn định: Đánh giá mô hình trên nhiều tập dữ liệu khác nhau giúp có được một ước tính chính xác hơn về hiệu suất của mô hình.

Sử dụng toàn bộ dữ liệu: Mỗi điểm dữ liệu đều được sử dụng để huấn luyện và kiểm tra, giúp mô hình tận dụng tối đa thông tin có sẵn.

4.2 Build Model

Đầu tiên, chúng tôi sử dụng Cross Validation để kiểm tra hiệu suất của các thuật toán.

[ ]:
```
# import các thư viện cần thiết
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
```

[ ]:
```
# Thêm các model vào list models
models = []
models.append(('Logistic Regression', LogisticRegression()))
models.append(('Random Forest', RandomForestClassifier()))
models.append(('Decision Tree Classifier', DecisionTreeClassifier()))
```

[ ]:
```
acc_results =[]
auc_results =[]
pre_results =[]
f1_results =[]
names = []

result_col = ["Algorithm", "ROC AUC", "Accuracy", "Precision", "f1"]
model_results = pd.DataFrame(columns = result_col)

i=0
# K- fold cross validation

for name, model in models:
    names.append(name)
    kfold = model_selection.KFold(n_splits=10)

    cv_acc_results = model_selection.cross_val_score(model, X_train, y_train,
                     scoring="accuracy")
    cv_auc_results = model_selection.cross_val_score(model, X_train, y_train,
                     scoring="roc_auc")
    cv_pre_results = model_selection.cross_val_score(model, X_train, y_train,
                     scoring="precision")
    cv_f1_results = model_selection.cross_val_score(model, X_train, y_train,
```

```
                      scoring="f1")
    acc_results.append(cv_acc_results)
    auc_results.append(cv_auc_results)
    pre_results.append(cv_pre_results)
    f1_results.append(cv_f1_results)

    model_results.loc[i] = [name,
                            round(cv_acc_results.mean()*100,2),
                            round(cv_auc_results.mean()*100,2),
                            round(cv_pre_results.mean()*100,2),
                            round(cv_f1_results.mean()*100,2)]
    i+=1

model_results.sort_values(by = ['ROC AUC'], ascending=False)
```

[ ]:

	Algorithm	ROC AUC	Accuracy	Precision	f1
1	Random Forest	84.50	92.08	84.22	84.96
0	Logistic Regression	80.55	88.90	78.27	81.30
2	Decision Tree Classifier	77.98	78.23	77.42	78.36

Random Forest là thuật toán ưu việt nhất trong việc dự đoán, đạt điểm số cao nhất trong tất cả các chỉ số.

Logistic Regression là một lựa chọn tốt với kết quả ổn định, tuy nhiên kém hơn so với Random Forest về độ chính xác và khả năng phân biệt giữa các lớp.

Decision Tree có hiệu suất kém hơn so với hai thuật toán còn lại, đặc biệt là khi xét đến độ chính xác và khả năng phân biệt các lớp, mặc dù nó có thể vẫn hữu ích trong các tình huống yêu cầu mô hình đơn giản và dễ hiểu

Bây giờ, ta thử thêm các hyper parameter cho các mô hình và xem có sự cải thiện

5 THÊM CÁC HYPER PARAMETER TỐT CHO TỪNG MÔ HÌNH CỤ THỂ

5.1 GridsearchCV

Để tối ưu hiệu suất của các mô hình, chúng tôi đã sử dụng Gridsearch CV, một kỹ thuật khám phá một cách có hệ thống các tổ hợp siêu tham số khác nhau để tìm cấu hình tốt nhất cho mỗi thuật toán. Cách tiếp cận này cho phép chúng tôi tinh chỉnh các tham số của mô hình và tối đa hóa độ chính xác dự đoán của chúng.

GridSearchCV (Grid Search Cross-Validation) là một phương pháp tìm kiếm hệ thống các siêu tham số (hyperparameters) tối ưu cho mô hình học máy. Thay vì phải chọn siêu tham số một cách thủ công, GridSearchCV tự động thử nghiệm tất cả các kết hợp có thể của các siêu tham số được chỉ định và đánh giá hiệu suất của mô hình qua quá trình cross-validation.

Cơ bản, GridSearchCV thực hiện các bước sau:

Xác định lưới siêu tham số: Cung cấp một tập hợp các giá trị cho từng siêu tham số của mô hình,

chẳng hạn như C trong Logistic Regression hoặc n_estimators trong Random Forest.

Thử nghiệm tất cả các kết hợp: GridSearchCV thử tất cả các kết hợp có thể có của các siêu tham số trong lưới đã định.

Đánh giá với Cross-Validation: Đối với mỗi kết hợp siêu tham số, GridSearchCV thực hiện k-fold cross-validation để đánh giá hiệu suất mô hình, giúp giảm thiểu vấn đề overfitting.

Chọn mô hình tốt nhất: Sau khi thử nghiệm tất cả các kết hợp và đánh giá, GridSearchCV sẽ trả về mô hình với siêu tham số tối ưu, giúp cải thiện hiệu suất của mô hình.

[ ]:
```
# Các thư viện cần thiết
import tensorflow as tf

from sklearn import feature_selection
from sklearn import model_selection
from sklearn import metrics

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

from sklearn.metrics import recall_score, confusion_matrix, precision_score,␣
↪f1_score, accuracy_score, classification_report
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc, roc_auc_score, roc_curve
from sklearn.metrics import make_scorer, log_loss
from sklearn.metrics import average_precision_score
#!pip install scikit-plot
```

5.2 Logistic Regression

[ ]:
```
modelL=LogisticRegression()
```

[ ]:
```
modelL.fit(X_train,y_train)
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
pred_trainL = modelL.predict(X_train)
pred_testL=modelL.predict(X_test)
```

[ ]:
```
acc_trainL = accuracy_score(pred_trainL, y_train)
acc_testL = accuracy_score(pred_testL, y_test)
print(f'Training accuracy {acc_trainL: .3f}')
print(f'Testing accuracy {acc_testL: .3f}')
```

```
Training accuracy  0.806
Testing accuracy  0.805
```

5.2.1 Điều chỉnh các parameters

[ ]:
```
tuned_parameters = [{'C': [10.0, 100.0, 1000.0, 10000.0],
                     'max_iter': [500, 1000]}] # Các parameters cần điều chỉnh
clfL = GridSearchCV(LogisticRegression(), tuned_parameters,
                    verbose=1, n_jobs=-1)
clfL.fit(X_train,y_train)

print("\nBest parameters found:")
print(clfL.best_params_) # in ra hyperparameter tốt nhất

print("\nGrid scores:")
meansL = clfL.cv_results_['mean_test_score'] # Trung bình độ chính xác
stdsL = clfL.cv_results_['std_test_score'] # Độ lệch chuẩn độ chính xác
# Đối với mỗi tổ hợp hyperparameter, hiển thị trung bình +/- 2 độ lệch chuẩn.
for mean, std, params in zip(meansL, stdsL, clfL.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r" %(mean, std * 2, params))
```

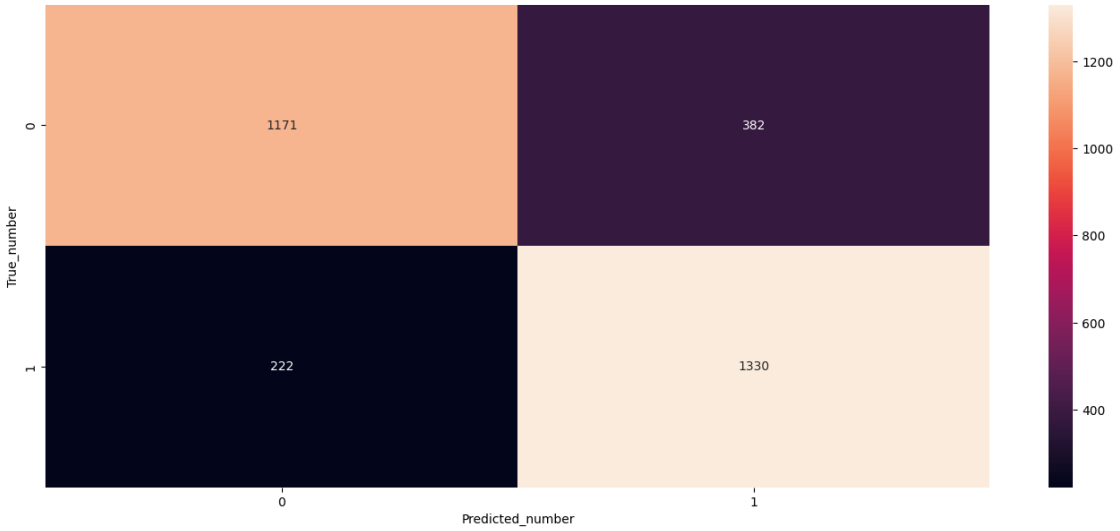
```
Fitting 5 folds for each of 8 candidates, totalling 40 fits

Best parameters found:
{'C': 10.0, 'max_iter': 500}

Grid scores:
0.806 (+/-0.004) for {'C': 10.0, 'max_iter': 500}
0.806 (+/-0.004) for {'C': 10.0, 'max_iter': 1000}
0.806 (+/-0.004) for {'C': 100.0, 'max_iter': 500}
0.806 (+/-0.004) for {'C': 100.0, 'max_iter': 1000}
0.806 (+/-0.004) for {'C': 1000.0, 'max_iter': 500}
0.806 (+/-0.004) for {'C': 1000.0, 'max_iter': 1000}
0.806 (+/-0.004) for {'C': 10000.0, 'max_iter': 500}
0.806 (+/-0.004) for {'C': 10000.0, 'max_iter': 1000}
```

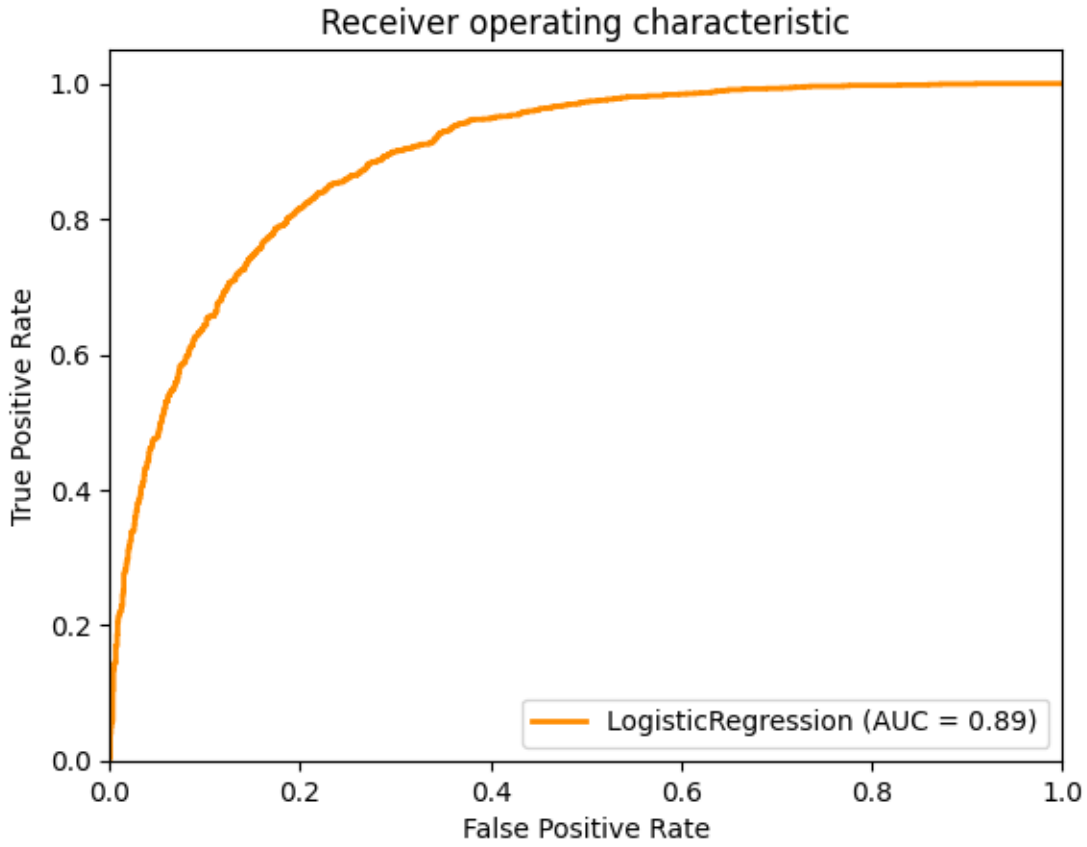
5.2.2 Đánh giá mô hình dựa trên các tham số tốt nhất tìm được.

[ ]:
```
# Xác định các giá trị hyper-parameter đã tìm ra
C = 10
max_iter = 500
# Train and test model
good_modelL = LogisticRegression(C = C, max_iter=max_iter) # create model
print(good_modelL) # in tham số của mô hình
good_modelL.fit(X_train,y_train) # train model
predL = good_modelL.predict(X_test) # Dự đoán kết quả cho tập test
pred_trainL = good_modelL.predict(X_train)
acc_trainL = accuracy_score(pred_trainL, y_train) # Độ chính xác trên tập train
print(f'Training accuracy = {acc_trainL: .3f}')
print("Results on test data")
accL = accuracy_score(y_test, predL) # Độ chính xác trên tập test
```

```
precL = precision_score(y_test, predL) # precision trên tập test
recaL = recall_score(y_test, predL) # recall trên tập test
print(f'Test accuracy = {accL: .4f}') # làm tròn 4 chữ số tập phân
print(f'Test precision = {precL: .4f}') # làm tròn 4 chữ số tập phân
print(f'Test recall = {recaL: .4f}') # làm tròn 4 chữ số tập phân
print("Classification report:")
print(classification_report(y_test, predL))

# Confusion Matrix
conf_mat = tf.math.confusion_matrix(labels=y_test,predictions=predL)
plt.figure(figsize = (17,7))
sns.heatmap(conf_mat, annot=True,fmt='d')
plt.xlabel('Predicted_number')
plt.ylabel('True_number')


fpr, tpr, thresholds = roc_curve(y_test, good_modelL.predict_proba(X_test)[:,␣
↪1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='LogisticRegression (AUC =␣
↪%0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
LogisticRegression(C=10, max_iter=500)
Training accuracy =  0.807
Results on test data
Test accuracy =  0.8055
Test precision =  0.7769
Test recall =  0.8570
Classification report:
              precision    recall  f1-score   support

           0       0.84      0.75      0.79      1553
           1       0.78      0.86      0.81      1552

    accuracy                           0.81      3105
   macro avg       0.81      0.81      0.80      3105
weighted avg       0.81      0.81      0.80      3105
```


1171
382
222
1330
0
1
True_number
0
1
Predicted_number
1200
1000
800
600
400


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
LogisticRegression (AUC = 0.89)

5.3 Decision Tree Classifier

[ ]:
```
modelD= DecisionTreeClassifier()
```

[ ]:
```
modelD.fit(X_train, y_train)
```

[ ]:
```
DecisionTreeClassifier()
```

[ ]:
```
pred_train_D = modelD.predict(X_train)
pred_test_D=modelD.predict(X_test)
```

[ ]:
```
acc_train_D = accuracy_score(pred_train_D, y_train)
acc_test_D = accuracy_score(pred_test_D, y_test)
print(f'Training accuracy {acc_train_D: .3f}')
print(f'Testing accuracy {acc_test_D: .3f}')
```

```
Training accuracy  0.998
Testing accuracy  0.783
```

Ta thấy mô hình đang bị overfitting

###Điều chỉnh các parameters

[ ]:
```
tuned_parameters = [{'criterion':['gini', 'entropy'],
                     'max_leaf_nodes': range(5,25)}] # các tham số cần điều␣
↪chỉnh

clf_D = GridSearchCV(DecisionTreeClassifier(), tuned_parameters,
                     verbose=1, n_jobs=-1)
clf_D.fit(X_train, y_train)

print("\nBest parameters found:")
print(clf_D.best_params_) # in ra tham số tốt nhất

print("\nGrid scores:")
means_D = clf_D.cv_results_['mean_test_score'] # Trung bình độ chính xác
stds_D = clf_D.cv_results_['std_test_score'] # Độ lệch chuẩn độ chính xác
# Đối với mỗi tổ hợp hyperparameter, hiển thị trung bình +/- 2 độ lệch chuẩn.
for mean, std, params in zip(means_D, stds_D, clf_D.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r" %(mean, std * 2, params))
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits

Best parameters found:
{'criterion': 'gini', 'max_leaf_nodes': 24}

Grid scores:
0.767 (+/-0.034) for {'criterion': 'gini', 'max_leaf_nodes': 5}
0.767 (+/-0.034) for {'criterion': 'gini', 'max_leaf_nodes': 6}
0.780 (+/-0.025) for {'criterion': 'gini', 'max_leaf_nodes': 7}
```

```
0.784 (+/-0.026) for {'criterion': 'gini', 'max_leaf_nodes': 8}
0.786 (+/-0.027) for {'criterion': 'gini', 'max_leaf_nodes': 9}
0.786 (+/-0.026) for {'criterion': 'gini', 'max_leaf_nodes': 10}
0.788 (+/-0.021) for {'criterion': 'gini', 'max_leaf_nodes': 11}
0.792 (+/-0.013) for {'criterion': 'gini', 'max_leaf_nodes': 12}
0.792 (+/-0.007) for {'criterion': 'gini', 'max_leaf_nodes': 13}
0.795 (+/-0.009) for {'criterion': 'gini', 'max_leaf_nodes': 14}
0.795 (+/-0.009) for {'criterion': 'gini', 'max_leaf_nodes': 15}
0.795 (+/-0.011) for {'criterion': 'gini', 'max_leaf_nodes': 16}
0.795 (+/-0.011) for {'criterion': 'gini', 'max_leaf_nodes': 17}
0.795 (+/-0.011) for {'criterion': 'gini', 'max_leaf_nodes': 18}
0.795 (+/-0.014) for {'criterion': 'gini', 'max_leaf_nodes': 19}
0.796 (+/-0.014) for {'criterion': 'gini', 'max_leaf_nodes': 20}
0.796 (+/-0.019) for {'criterion': 'gini', 'max_leaf_nodes': 21}
0.798 (+/-0.015) for {'criterion': 'gini', 'max_leaf_nodes': 22}
0.798 (+/-0.016) for {'criterion': 'gini', 'max_leaf_nodes': 23}
0.799 (+/-0.013) for {'criterion': 'gini', 'max_leaf_nodes': 24}
0.757 (+/-0.007) for {'criterion': 'entropy', 'max_leaf_nodes': 5}
0.762 (+/-0.017) for {'criterion': 'entropy', 'max_leaf_nodes': 6}
0.763 (+/-0.021) for {'criterion': 'entropy', 'max_leaf_nodes': 7}
0.766 (+/-0.021) for {'criterion': 'entropy', 'max_leaf_nodes': 8}
0.766 (+/-0.021) for {'criterion': 'entropy', 'max_leaf_nodes': 9}
0.774 (+/-0.022) for {'criterion': 'entropy', 'max_leaf_nodes': 10}
0.780 (+/-0.030) for {'criterion': 'entropy', 'max_leaf_nodes': 11}
0.784 (+/-0.028) for {'criterion': 'entropy', 'max_leaf_nodes': 12}
0.786 (+/-0.026) for {'criterion': 'entropy', 'max_leaf_nodes': 13}
0.788 (+/-0.027) for {'criterion': 'entropy', 'max_leaf_nodes': 14}
0.788 (+/-0.027) for {'criterion': 'entropy', 'max_leaf_nodes': 15}
0.790 (+/-0.021) for {'criterion': 'entropy', 'max_leaf_nodes': 16}
0.792 (+/-0.015) for {'criterion': 'entropy', 'max_leaf_nodes': 17}
0.793 (+/-0.013) for {'criterion': 'entropy', 'max_leaf_nodes': 18}
0.793 (+/-0.013) for {'criterion': 'entropy', 'max_leaf_nodes': 19}
0.793 (+/-0.013) for {'criterion': 'entropy', 'max_leaf_nodes': 20}
0.793 (+/-0.013) for {'criterion': 'entropy', 'max_leaf_nodes': 21}
0.793 (+/-0.015) for {'criterion': 'entropy', 'max_leaf_nodes': 22}
0.793 (+/-0.015) for {'criterion': 'entropy', 'max_leaf_nodes': 23}
0.792 (+/-0.015) for {'criterion': 'entropy', 'max_leaf_nodes': 24}
```

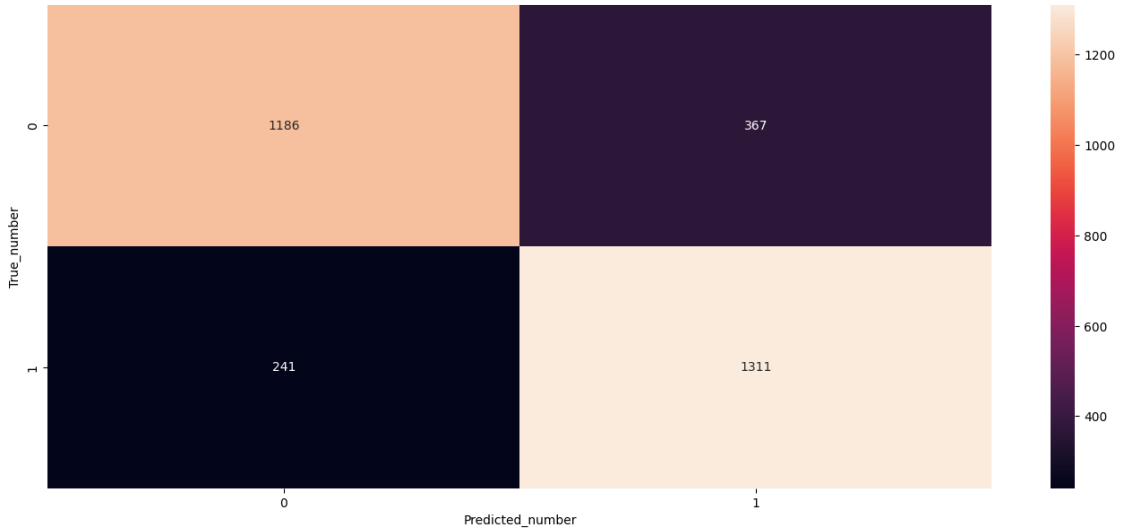
5.3.1 Đánh giá mô hình dựa trên các tham số tốt nhất tìm được.

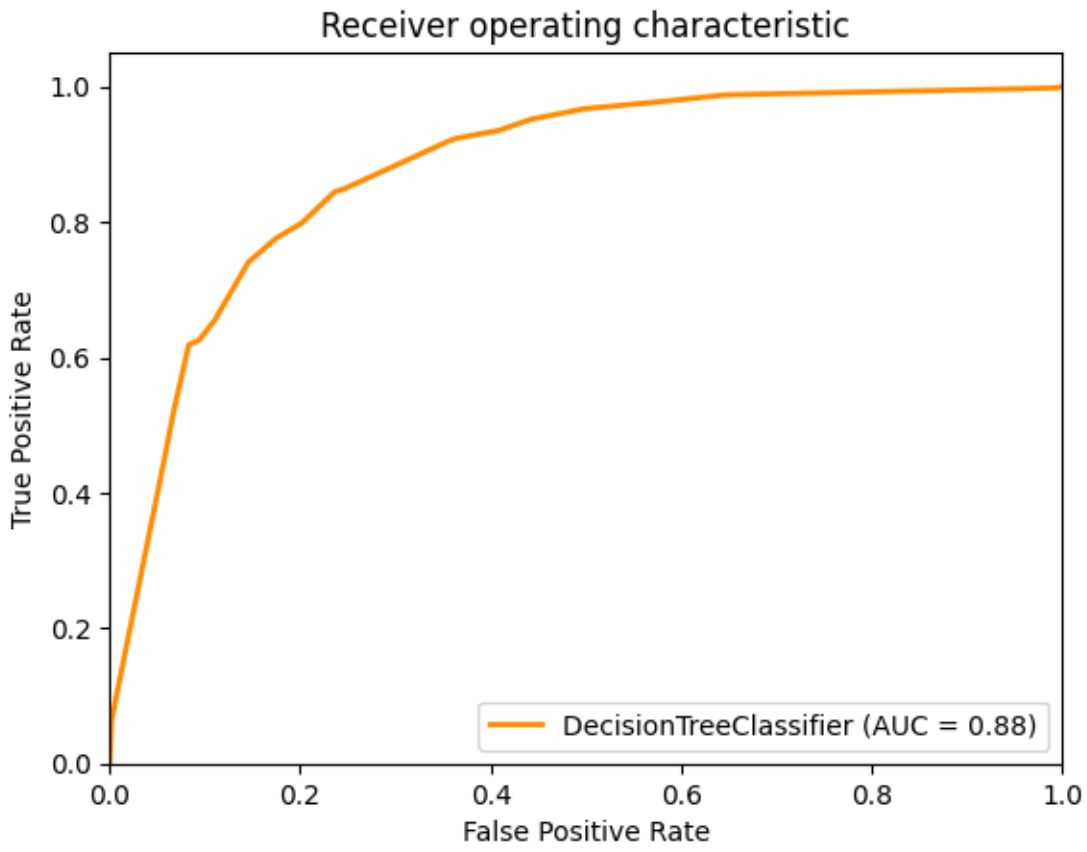
[ ]:
```
# Xác định các giá trị hyper-parameter đã tìm ra
criterion = 'gini'
max_leaf_nodes = 24
# Train and test model
good_model_D = DecisionTreeClassifier(criterion=criterion,
                                      max_leaf_nodes=max_leaf_nodes) # create␣
 ↪model
```

```
print(good_model_D) # in các tham số model
good_model_D.fit(X_train, y_train) # train model
pred_D = good_model_D.predict(X_test) # dự đoán kết quả trên tập test
pred_trainL = good_model_D.predict(X_train)
acc_trainL = accuracy_score(pred_trainL, y_train)
print(f'Training accuracy {acc_trainL: .3f}')
print("Results on test data")
acc_D = accuracy_score(y_test, pred_D) # accuracy trên tập test
prec_D = precision_score(y_test, pred_D) # precision trên tập test
reca_D = recall_score(y_test, pred_D) # recall trên tập test
print(f'Test accuracy = {acc_D: .4f}') # làm tròn 4 chữ số tập phân
print(f'Test precision = {prec_D: .4f}') # làm tròn 4 chữ số tập phân
print(f'Test recall = {reca_D: .4f}') # làm tròn 4 chữ số tập phân
print("Classification report:")
print(classification_report(y_test, pred_D))

# Confusion Matrix
conf_mat = tf.math.confusion_matrix(labels=y_test,predictions=pred_D)
plt.figure(figsize = (17,7))
sns.heatmap(conf_mat, annot=True,fmt='d')
plt.xlabel('Predicted_number')
plt.ylabel('True_number')

fpr, tpr, thresholds = roc_curve(y_test, good_model_D.predict_proba(X_test)[:,␣
 ↪1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='DecisionTreeClassifier (AUC␣
 ↪= %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
DecisionTreeClassifier(max_leaf_nodes=24)
Training accuracy  0.809
Results on test data
Test accuracy =  0.8042
Test precision =  0.7813
Test recall =  0.8447
Classification report:
              precision    recall  f1-score   support
```

0	0.83	0.76	0.80	1553
1	0.78	0.84	0.81	1552
accuracy			0.80	3105
macro avg	0.81	0.80	0.80	3105
weighted avg	0.81	0.80	0.80	3105





Sau khi điều chỉnh các tham số, ta đã cải thiện vấn đề overfitting của mô hình

##Random Forest

```
[ ]: modelR = RandomForestClassifier()
```

```
[ ]: modelR.fit(X_train, y_train)
```

```
[ ]: RandomForestClassifier()
```

```
[ ]: pred_train = modelR.predict(X_train)
     pred_test=modelR.predict(X_test)
```

```
[ ]: acc_train = accuracy_score(pred_train, y_train)
     acc_test = accuracy_score(pred_test, y_test)
     print(f'Training accuracy {acc_train: .3f}')
     print(f'Testing accuracy {acc_test: .3f}')
```

```
Training accuracy  0.998
Testing accuracy  0.849
```

Mô hình này cũng đang bị overfitting

###Điều chỉnh các Parameters

[ ]:
```
tuned_parameters = [{'n_estimators': [25, 50, 75, 100],
                     'max_features': [5, 10, 15],
                     'max_leaf_nodes': [8, 16, 24]}] # Các tham số cần điều␣
↪chỉnh

clfR = GridSearchCV(RandomForestClassifier(), tuned_parameters,
                    verbose=1, n_jobs=-1)
clfR.fit(X_train, y_train)

print("\nBest parameters found:")
print(clfR.best_params_) # in ra các thám số tốt nhất

print("\nGrid scores:")
means = clfR.cv_results_['mean_test_score'] # Trung bình độ chính xác
stds = clfR.cv_results_['std_test_score'] # Độ lệch chuẩn độ chính xác
# Đối với mỗi tổ hợp hyperparameter, hiển thị trung bình +/- 2 độ lệch chuẩn.
for mean, std, params in zip(means, stds, clfR.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r" %(mean, std * 2, params))
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits

Best parameters found:
{'max_features': 15, 'max_leaf_nodes': 24, 'n_estimators': 100}

Grid scores:
0.796 (+/-0.013) for {'max_features': 5, 'max_leaf_nodes': 8, 'n_estimators':
25}
0.797 (+/-0.015) for {'max_features': 5, 'max_leaf_nodes': 8, 'n_estimators':
50}
0.797 (+/-0.017) for {'max_features': 5, 'max_leaf_nodes': 8, 'n_estimators':
75}
0.796 (+/-0.014) for {'max_features': 5, 'max_leaf_nodes': 8, 'n_estimators':
100}
0.803 (+/-0.014) for {'max_features': 5, 'max_leaf_nodes': 16, 'n_estimators':
25}
0.803 (+/-0.013) for {'max_features': 5, 'max_leaf_nodes': 16, 'n_estimators':
50}
0.804 (+/-0.010) for {'max_features': 5, 'max_leaf_nodes': 16, 'n_estimators':
75}
0.804 (+/-0.013) for {'max_features': 5, 'max_leaf_nodes': 16, 'n_estimators':
100}
0.810 (+/-0.013) for {'max_features': 5, 'max_leaf_nodes': 24, 'n_estimators':
25}
0.812 (+/-0.012) for {'max_features': 5, 'max_leaf_nodes': 24, 'n_estimators':
50}
0.812 (+/-0.009) for {'max_features': 5, 'max_leaf_nodes': 24, 'n_estimators':
```

```
75}
0.813 (+/-0.006) for {'max_features': 5, 'max_leaf_nodes': 24, 'n_estimators':
100}
0.794 (+/-0.015) for {'max_features': 10, 'max_leaf_nodes': 8, 'n_estimators':
25}
0.798 (+/-0.011) for {'max_features': 10, 'max_leaf_nodes': 8, 'n_estimators':
50}
0.797 (+/-0.009) for {'max_features': 10, 'max_leaf_nodes': 8, 'n_estimators':
75}
0.796 (+/-0.009) for {'max_features': 10, 'max_leaf_nodes': 8, 'n_estimators':
100}
0.805 (+/-0.006) for {'max_features': 10, 'max_leaf_nodes': 16, 'n_estimators':
25}
0.805 (+/-0.006) for {'max_features': 10, 'max_leaf_nodes': 16, 'n_estimators':
50}
0.807 (+/-0.011) for {'max_features': 10, 'max_leaf_nodes': 16, 'n_estimators':
75}
0.804 (+/-0.006) for {'max_features': 10, 'max_leaf_nodes': 16, 'n_estimators':
100}
0.809 (+/-0.014) for {'max_features': 10, 'max_leaf_nodes': 24, 'n_estimators':
25}
0.811 (+/-0.014) for {'max_features': 10, 'max_leaf_nodes': 24, 'n_estimators':
50}
0.813 (+/-0.006) for {'max_features': 10, 'max_leaf_nodes': 24, 'n_estimators':
75}
0.810 (+/-0.008) for {'max_features': 10, 'max_leaf_nodes': 24, 'n_estimators':
100}
0.798 (+/-0.007) for {'max_features': 15, 'max_leaf_nodes': 8, 'n_estimators':
25}
0.798 (+/-0.012) for {'max_features': 15, 'max_leaf_nodes': 8, 'n_estimators':
50}
0.798 (+/-0.012) for {'max_features': 15, 'max_leaf_nodes': 8, 'n_estimators':
75}
0.798 (+/-0.010) for {'max_features': 15, 'max_leaf_nodes': 8, 'n_estimators':
100}
0.803 (+/-0.009) for {'max_features': 15, 'max_leaf_nodes': 16, 'n_estimators':
25}
0.803 (+/-0.007) for {'max_features': 15, 'max_leaf_nodes': 16, 'n_estimators':
50}
0.804 (+/-0.008) for {'max_features': 15, 'max_leaf_nodes': 16, 'n_estimators':
75}
0.804 (+/-0.007) for {'max_features': 15, 'max_leaf_nodes': 16, 'n_estimators':
100}
0.813 (+/-0.015) for {'max_features': 15, 'max_leaf_nodes': 24, 'n_estimators':
25}
0.812 (+/-0.015) for {'max_features': 15, 'max_leaf_nodes': 24, 'n_estimators':
50}
0.811 (+/-0.014) for {'max_features': 15, 'max_leaf_nodes': 24, 'n_estimators':
```

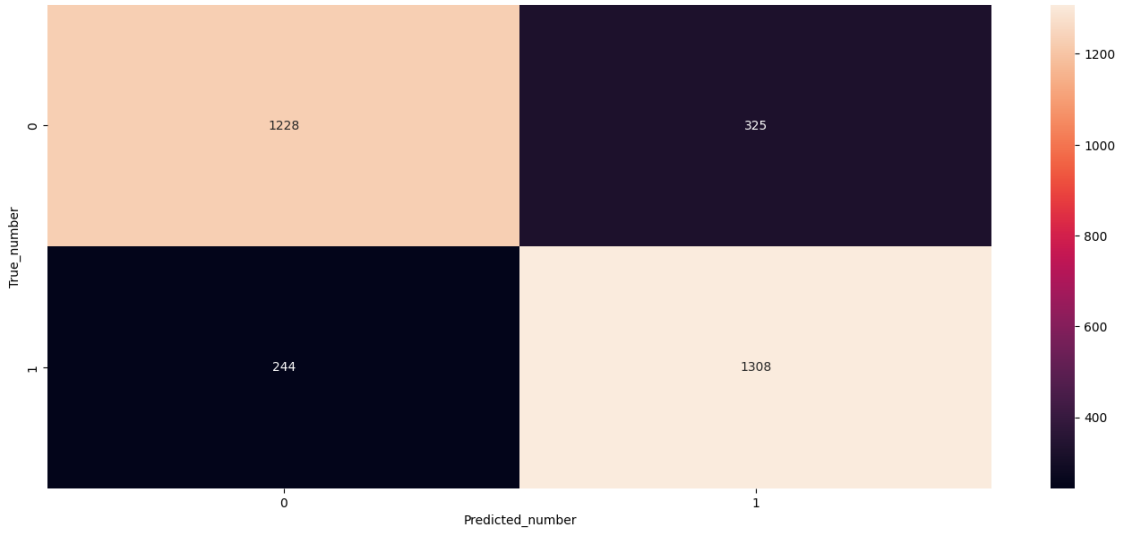
```
75}
0.814 (+/-0.013) for {'max_features': 15, 'max_leaf_nodes': 24, 'n_estimators':
100}
```

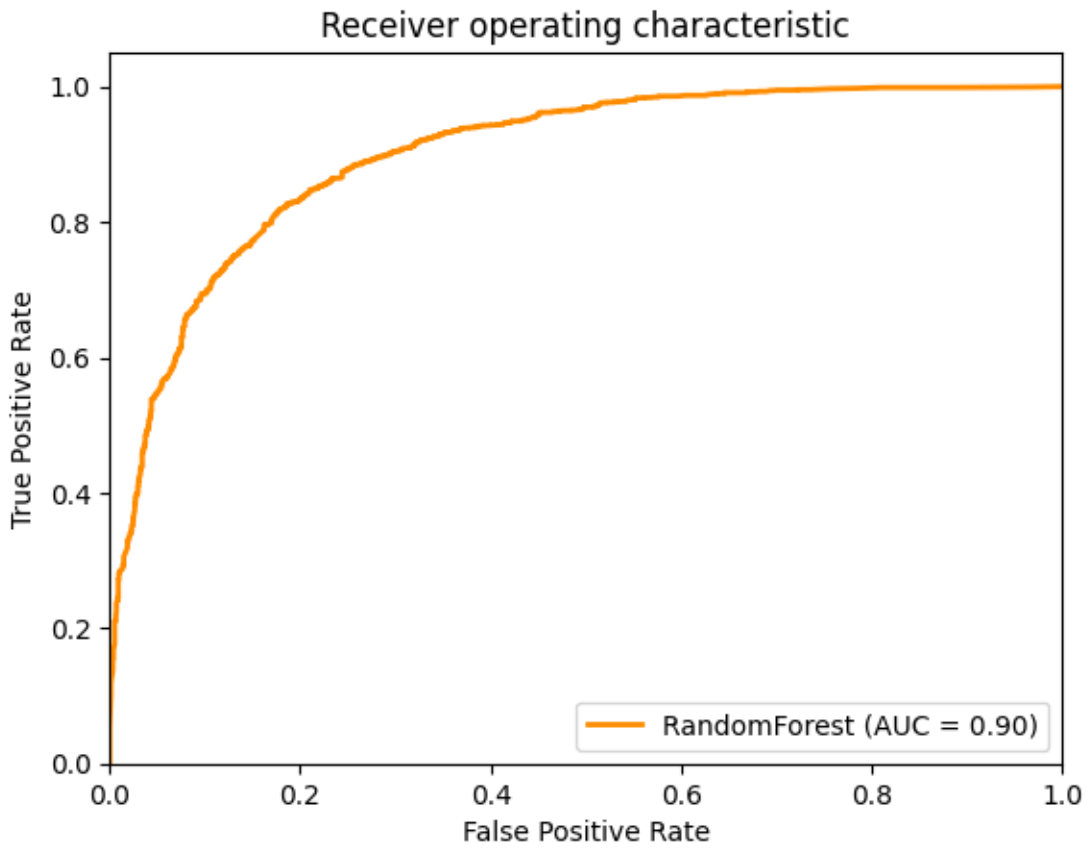
###Đánh giá RandomForest Classiffer với các tham số đã chọn.

[ ]:
```
# Xác định các giá trị hyper-parameter đã tìm ra
max_features = 15
max_leaf_nodes = 24
n_estimators = 75
# Train and test model
good_model_R = RandomForestClassifier(max_leaf_nodes = max_leaf_nodes,
                                      max_features = max_features,
                                      n_estimators=n_estimators, ) # create model
print(good_model_R) # in ra các tham số model
good_model_R.fit(X_train, y_train) # train model
pred = good_model_R.predict(X_test) # Dự đoán kết quả trên tập test
pred_trainL = good_model_R.predict(X_train)
acc_trainL = accuracy_score(pred_trainL, y_train)
print(f'Training accuracy {acc_trainL: .3f}')
print("Results on test data")
acc = accuracy_score(y_test, pred) # accuracy trên tập test
prec = precision_score(y_test, pred) # precision trên tập test
reca = recall_score(y_test, pred) # recall trên tập test
print(f'Test accuracy = {acc: .4f}') # làm tròn 4 chữ số tập phân
print(f'Test precision = {prec: .4f}') # làm tròn 4 chữ số tập phân
print(f'Test recall = {reca: .4f}') # làm tròn 4 chữ số tập phân
print("Classification report:")
print(classification_report(y_test, pred))

# Confusion Matrix
conf_mat = tf.math.confusion_matrix(labels=y_test,predictions=pred)
plt.figure(figsize = (17,7))
sns.heatmap(conf_mat, annot=True,fmt='d')
plt.xlabel('Predicted_number')
plt.ylabel('True_number')

fpr, tpr, thresholds = roc_curve(y_test, good_model_R.predict_proba(X_test)[:,␣
 ↪1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='RandomForest (AUC = %0.2f)'␣
 ↪% roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
RandomForestClassifier(max_features=15, max_leaf_nodes=24, n_estimators=75)
Training accuracy  0.823
Results on test data
Test accuracy =  0.8167
Test precision =  0.8010
Test recall =  0.8428
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.79      0.81      1553
           1       0.80      0.84      0.82      1552

    accuracy                           0.82      3105
   macro avg       0.82      0.82      0.82      3105
weighted avg       0.82      0.82      0.82      3105
```





Sau khi điều chỉnh các tham số, ta đã cải thiện vấn đề overfitting của mô hình

6 SỬ DỤNG K-FOLD CROSS-VALIDATION ĐỂ ĐÁNH GIÁ HIỆU NĂNG CỦA MÔ HÌNH

[ ]:
```
models_opt = []

models_opt.append(('Logistic Regression',␣
 ↪LogisticRegression(solver='liblinear',C = 10, max_iter=500, random_state =␣
 ↪0)))
models_opt.append(('Decision Tree Classifier', DecisionTreeClassifier(criterion␣
 ↪= 'gini', max_leaf_nodes=24, random_state = 0)))
models_opt.append(('Random Forest', RandomForestClassifier(max_leaf_nodes =␣
 ↪15,max_features = 24,
                                                    n_estimators =␣
 ↪75,criterion = 'entropy', random_state = 0)))
```

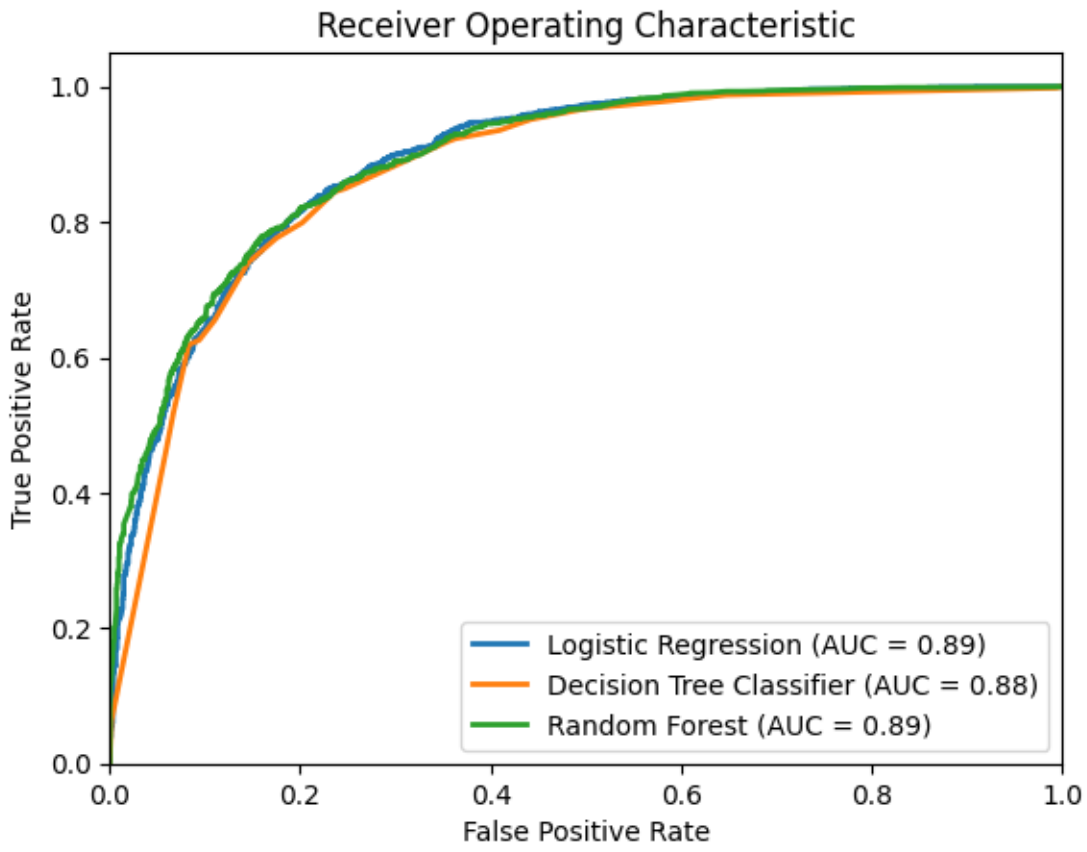
[ ]:
```
acc_results_opt =[]
auc_results_opt =[]
pre_results_opt =[]
f1_results_opt =[]
names_opt = []

result_col_opt = ["Algorithm", "ROC AUC", "Accuracy", 'Precision', 'F1 Scores']
model_results_opt = pd.DataFrame(columns = result_col_opt)

i=0
# K- fold cross validation

for name, model in models_opt:
    names_opt.append(name)
    kfold = model_selection.KFold(n_splits=10)

    cv_acc_results_opt = model_selection.cross_val_score(model, X_train,␣
 ↪y_train,
                    cv = kfold, scoring="accuracy")
    cv_auc_results_opt = model_selection.cross_val_score(model, X_train,␣
 ↪y_train,
                    cv = kfold, scoring="roc_auc")
    cv_pre_results_opt = model_selection.cross_val_score(model, X_train,␣
 ↪y_train,
                    cv = kfold, scoring="precision")
    cv_f1_results_opt = model_selection.cross_val_score(model, X_train, y_train,
                    cv = kfold, scoring="f1")
    acc_results_opt.append(cv_acc_results_opt)
    auc_results_opt.append(cv_auc_results_opt)
    pre_results_opt.append(cv_pre_results_opt)
    f1_results_opt.append(cv_f1_results_opt)
    model_results_opt.loc[i] = [name,
                            round(cv_auc_results_opt.mean()*100,2),
                            round(cv_acc_results_opt.mean()*100,2),
                            round(cv_pre_results_opt.mean()*100,2),
                            round(cv_f1_results_opt.mean()*100,2)]
    i+=1

model_results_opt.sort_values(by = ['ROC AUC'], ascending=False)
```

[ ]:

	Algorithm	ROC AUC	Accuracy	Precision	F1 Scores
2	Random Forest	89.16	80.22	79.57	80.41
0	Logistic Regression	88.96	80.55	78.20	81.31
1	Decision Tree Classifier	87.87	79.95	78.75	80.32

7 KIỂM TRA HIỆU NĂNG CỦA MÔ HÌNH TRÊN TẬP DỮ LIỆU KIỂM TRA

[ ]:
```
results_list = []
for name, model in models_opt:
    model.fit(X_train, y_train)  # Huấn luyện mô hình trên toàn bộ tập huấn␣
 ↪luyện
    y_pred = model.predict(X_test)  # Dự đoán nhãn cho tập test

    accuracy = metrics.accuracy_score(y_test, y_pred)
    precision = metrics.precision_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)
    recall = metrics.recall_score(y_test, y_pred)

    results_list.append({'Model': name, 'Accuracy': accuracy,
                         'Precision': precision, 'F1 Score': f1, 'Recall':␣
 ↪recall})
results = pd.DataFrame(results_list)

# Sắp xếp DataFrame theo giảm dần của ROC AUC
results = results.sort_values(by='Precision', ascending=False)

plt.figure()
for name, model in models_opt:
    fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)[:, 1])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, lw=2, label='%s (AUC = %0.2f)' % (name, roc_auc))

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()


print(results)
```



	Model	Accuracy	Precision	F1 Score	Recall
2	Random Forest	0.806119	0.792848	0.810334	0.828608
1	Decision Tree Classifier	0.804187	0.781287	0.811765	0.844716
0	Logistic Regression	0.805153	0.776739	0.814588	0.856314

8 TỔNG KẾT

Nhìn chung, không có sự khác biệt rõ ràng giữa các mô hình đã được sử dụng. Trong đó Random Forest cho kết quả dự đoán tốt nhất với tập dữ liệu. Hiệu suất mô hình được cải thiện thông qua việc sử dụng Gridsearch Cross Validation để điều chỉnh các tham số.

Mô hình học máy dự đoán tỷ lệ rời bỏ doanh nghiệp của khách hàng đã được nhóm xây dựng giúp doanh nghiệp xác định các yếu tố ảnh hưởng đến việc rời bỏ và dự đoán khả năng rời bỏ của từng khách hàng. Qua dự án này, gợi ý rằng các doanh nghiệp có thể tối ưu hóa chiến lược giữ chân khách hàng, cải thiện trải nghiệm dịch vụ và nâng cao hiệu quả kinh doanh.

Tài liệu tham khảo

- Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, P. (2000). SMOTE: Synthetic Minority Oversampling TEchnique. In International Conference of Knowledge Based Computer Systems, pp. 46–57. National Center for Software Technology, Mumbai, India, Allied Press.

- https://www.kaggle.com/datasets/blastchar/telco-customer-churn.
- https://www.kaggle.com/code/akinsanyajoel/customer-churn-prediction-for-a-telecom-company