BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

XÂY DỰNG MÔ HÌNH DỰ ĐOÁN KHÁCH HÀNG MỞ TÀI KHOẢN TIẾT KIỆM VÀ TỐI ƯU HÓA HỆ THỐNG SỬ DỤNG HỌC MÁY VÀ HỌC SÂU

Học phần: Máy Học
Nhóm Sinh Viên:

1. Phan Trần Sơn Bảo
2. Huỳnh Thị Cẩm Nhung
3. Lê Thị Tuyết Nhung
4. Nguyễn Đình Đại Nhơn

Chuyên Ngành: KHOA HỌC DỮ LIỆU
Khóa: K46

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

TP. Hồ Chí Minh, Ngày 22 tháng 04 năm 2023

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN

1.1. Giới Thiệu Về Đề Tài

Ngành tài chính liên tục phát triển và tiến hóa, và để các ngân hàng duy trì sự cạnh tranh, họ cần tìm cách mới và sáng tạo để thu hút khách hàng mới và tăng lợi nhuận. Một cách hiệu quả để làm điều này là khuyến khích khách hàng mở tài khoản tiết kiệm mới với lãi suất hấp dẫn cùng nhiều những ưu đãi khác nhau. Tuy nhiên, không phải tất cả khách hàng đều quan tâm đến việc mở tài khoản mới, điều này làm cho việc xác định những khách hàng có khả năng làm điều này trở thành một thách thức đối với các ngân hàng.

Để giải quyết khó khăn cũng như thách thức này, các ngân hàng cần hiểu sở thích và hành vi tài chính của khách hàng. Một cách để làm điều này là phân tích dữ liệu mà họ có về khách hàng. Bằng cách sử dụng các thuật toán học máy và học sâu, các ngân hàng có thể xây dựng mô hình dự đoán để xác định các mẫu trong hành vi của khách hàng, giúp họ tùy chỉnh chiến lược tiếp thị cho khách hàng cụ thể.

1.2. Phát Biểu Bài Toán

Mục tiêu của dự án này là xây dựng một mô hình dự đoán xác suất khách hàng mở tài khoản tiết kiệm mới bằng các thuật toán học máy và học sâu. Việc xây dựng mô hình này sẽ dựa trên đặc điểm dân số, lịch sử ngân hàng và sự tương tác của các chiến dịch tiếp thị. Mô hình sẽ được huấn luyện trên tập dữ liệu về thông tin khách hàng, bao gồm tuổi, thu nhập và lịch sử giao dịch. Bằng cách phân tích dữ liệu này, mô hình sẽ xác định các mẫu trong hành vi của khách hàng giúp các ngân hàng tập trung chiến dịch tiếp thị của mình hiệu quả hơn.

Ngoài việc xây dựng mô hình dự đoán, dự án này cũng sẽ thảo luận về lợi ích của việc sử dụng các thuật toán về mô hình dự đoán trong ngành tài chính. Nó sẽ khám phá cách các ngân hàng sử dụng những mô hình này để cải thiện trải nghiệm khách hàng và tăng lợi nhuận. Hơn nữa, dự án sẽ xem xét các thách thức và tiềm năng cũng như nhược điểm của việc sử dụng Học máy trong ngành ngân hàng và nhấn mạnh một số thực tiễn tốt nhất cho việc triển khai các thuật toán Học máy trong ngành tài chính.

1.3. Một Số Hướng Tiếp Cận Giải Quyết Bài Toán

Có nhiều hướng tiếp cận để giải quyết bài toán dự đoán khách hàng mở tài khoản tiết kiệm mới bằng các thuật toán Học máy. Một trong số đó là sử dụng các thuật toán phân loại như SVM, Random Forest, Decision Tree, Logistic Regression, Naive Bayes và KNN. Tuy nhiên, để tăng độ chính xác và hiệu quả của mô hình, các kỹ thuật học sâu như ANN, CNN và RNN cũng có thể được áp dụng. Ngoài ra, một số phương pháp khác như ensemble learning, active learning và transfer learning cũng có thể được sử dụng để tăng cường độ chính xác và giảm thiểu sự chênh lệnh của mô hình. Tuy nhiên, việc sử dụng các phương pháp này có thể yêu cầu thời gian và chi phí đầu tư cao hơn.

Đối với dự án này cùng với bộ dữ liệu này, chúng tôi đề xuất sử dụng thuật toán Máy Vecto hỗ trợ (SVM) và K-Nearest Neighbors (KNN). SVM là một thuật toán học máy giám sát phân tích dữ liệu cho việc phân loại và phân tích hồi quy. SVM tạo ra ranh giới giữa các điểm dữ liệu thuộc các lớp khác nhau. KNN là một thuật toán học máy giám sát khác, dự đoán phân loại của một điểm dữ liệu mới dựa trên khoảng cách của nó đến các điểm dữ liệu khác.

Cùng với đó, chúng tôi sử dụng thêm một mô hình học sâu là mô hình Mạng nơ-ron nhân tạo (ANN) để giải quyết bài toán này đạt được kết quả dự báo tốt nhất. ANN là một loại mô hình được xây dựng bằng các lớp liên kết với nhau của các nơ-ron, mô phỏng cấu trúc của não người. ANN đặc biệt phù hợp cho các tác vụ phân loại và phân tích hồi quy, nó có thể học các mối quan hệ phức tạp giữa dữ liệu đầu vào và đầu ra. Và để tối ưu hóa mô hình, chúng tôi đề xuất sử dụng một phương pháp gọi là phương pháp tinh chỉnh siêu tham số (Hyperparameter tuning) bằng Grid Search Cross Validation (GridSearchCV) của thư viện Scikit-learn. Grid Search là một kỹ thuật để chọn các tham số tốt nhất cho mô hình, bằng cách tìm kiếm và so sánh các tham số khác nhau. GridSearchCV sẽ lặp lại việc chạy mô hình với tất cả các tổ hợp tham số khác nhau được cung cấp, sau đó đánh giá kết quả để chọn ra mô hình tốt nhất. Các siêu tham số được tối ưu bao gồm dropout_rate, weight_constraint, neurons, reg_strength, batch_size và epochs.

Và đương nhiên, để có thể xây dựng được các mô hình một cách trơn tru như vậy, chúng tôi đã thực hiện các nhiệm vụ tiền xử lý dữ liệu trước đó bằng cách xử lý dữ liệu bị thiếu, các giá trị không xác định, biến phân loại, các giá trị ngoại lai, việc mất cân bằng dữ

liệu ngoài ý muốn và tỷ lệ hóa dữ liệu. Việc tiền xử lý dữ liệu và lựa chọn mô hình trong việc phát triển một mô hình dự đoán hiệu quả là vô cùng quan trọng. Nếu không tiền xử lý dữ liệu đúng cách, các thuật toán học máy có thể không thể dự đoán hành vi của khách hàng một cách chính xác.

1.4. Bộ Dữ Liệu

Bộ dữ liệu được thu thập thông qua các chiến dịch tiếp thị trực tiếp của một ngân hàng Bồ Đào Nha từ tháng 5 năm 2008 đến tháng 11 năm 2010. Dữ liệu thô bao gồm 41188 mẫu dữ liệu với 21 thuộc tính bao gồm thông tin về khách hàng, thông tin về sản phẩm ngân hàng và các biến số đánh giá khác.

age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
37	admin.	married	university.degree	no	yes	no	cellular	jul	wed	353	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
45	technician	married	professional.course	no	yes	no	cellular	jul	wed	95	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
34	blue-collar	married	basic.9y	no	yes	no	cellular	jul	wed	477	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
38	blue-collar	married	unknown	no	yes	no	cellular	jul	wed	131	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
33	admin.	married	high.school	no	yes	no	cellular	jul	wed	134	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
34	blue-collar	married	basic.9y	no	yes	no	cellular	jul	wed	85	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
33	admin.	married	high.school	no	no	no	cellular	jul	wed	254	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
45	technician	married	professional.course	no	no	no	cellular	jul	wed	394	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
38	blue-collar	married	unknown	no	no	no	cellular	jul	wed	618	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
54	services	married	high.school	no	no	no	telephone	jul	wed	250	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
54	services	married	high.school	no	no	no	cellular	jul	wed	82	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
35	management	married	university.degree	no	no	no	cellular	jul	wed	182	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
25	admin.	married	high.school	no	no	no	cellular	jul	wed	95	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
31	services	married	basic.9y	no	yes	no	cellular	jul	wed	537	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	yes
57	blue-collar	married	basic.6y	no	yes	no	cellular	jul	wed	64	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
31	services	married	basic.9y	no	no	no	cellular	jul	wed	679	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	yes
29	blue-collar	married	high.school	no	yes	no	telephone	jul	wed	42	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
53	admin.	married	university.degree	no	yes	no	cellular	jul	wed	618	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	yes	no	cellular	jul	wed	183	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	yes	no	cellular	jul	wed	296	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	no	no	cellular	jul	wed	206	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	yes	no	cellular	jul	wed	317	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	yes	no	cellular	jul	wed	107	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	yes	no	cellular	jul	wed	97	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	no	no	cellular	jul	wed	296	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
48	blue-collar	married	basic.4y	no	yes	no	cellular	jul	wed	74	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	housemaid	married	basic.4y	no	yes	no	cellular	jul	wed	402	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
48	blue-collar	married	basic.4y	no	yes	no	cellular	jul	wed	126	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
51	management	married	university.degree	no	yes	no	cellular	jul	wed	65	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
51	management	married	university.degree	no	yes	no	cellular	jul	wed	194	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
51	management	married	university.degree	no	yes	no	cellular	jul	wed	420	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
51	management	married	university.degree	no	yes	no	cellular	jul	wed	96	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
51	management	married	university.degree	no	no	no	cellular	jul	wed	175	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
48	blue-collar	married	basic.4y	no	no	no	cellular	jul	wed	128	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
34	blue-collar	married	basic.9y	no	yes	no	cellular	jul	wed	173	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
43	blue-collar	married	basic.6y	no	no	no	cellular	jul	wed	272	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no
30	admin.	married	high.school	no	yes	no	cellular	jul	wed	166	1	999	0	nonexistent	1.4	93.918	-42.7	4.962	5228	no

Hình 1: Bộ dữ liệu thô

Các thuộc tính của bộ dữ liệu bao gồm:

- 1 - age (numeric) : tuổi
- 2 - job : (categorical) : loại công việc
- 3 - marital : (categorical) : tình trạng hôn nhân
- 4 - education (categorical) : giáo dục
- 5 - default: (categorical) : có nợ xấu hay không?
- 6 - housing:(categorical) : có khoản vay mua nhà hay không?
- 7 - loan: (categorical) : có khoản vay cá nhân hay không?

- 8 - contact: (categorical) : loại phương tiện liên lạc
- 9 - month: (categorical) : tháng liên lạc cuối cùng trong năm
- 10 - day_of_week (categorical) : ngày liên lạc cuối cùng trong tuần
- 11 - duration: (numeric). thời lượng liên lạc cuối cùng, tính bằng giây
- 12 - campaign: (numeric) : số lần liên lạc được thực hiện trong chiến dịch này cho khách hàng này.
- 13 - pdays: (numeric) : số ngày trôi qua kể từ lần liên hệ cuối cùng với khách hàng trong chiến dịch tiếp thị trước đó.
- 14 - previous: (numeric) : số lần liên hệ đã thực hiện trước đó cho khách hàng này trong chiến dịch hiện tại.
- 15 - poutcome: (categorical) : kết quả của chiến dịch tiếp thị trước đó.
- 16 - emp.var.rate (numeric) : tỷ lệ biến động việc làm - chỉ báo theo quý.
- 17 - cons.price.idx: (numeric) : chỉ số giá tiêu dùng - chỉ báo hàng tháng.
- 18 - cons.conf.idx: (numeric) : chỉ số niềm tin người tiêu dùng - chỉ báo hàng tháng.
- 19 - euribor3m: (numeric) : tỷ lệ euribor 3 tháng - chỉ báo hàng ngày.
- 20 - nr.employed: (numeric) : số lượng nhân viên - chỉ báo theo quý.
- 21 – y : (binary) : Có khách hàng đã đăng ký kỳ hạn gửi tiền hay không?.

CHƯƠNG 2. TIỀN XỬ LÝ DỮ LIỆU

Đầu tiên, đọc file dữ liệu của nhóm bằng phương thức **pd.read_csv**:

```
# Load data
data = pd.read_csv('bank-additional-full.csv', sep = ';')
display(data)
```

H1: load data

Nhóm sẽ có một bảng dữ liệu như sau:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	nonexistent	-1.1	94.767	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	-1.1	94.767	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	nonexistent	-1.1	94.767	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	-1.1	94.767	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	failure	-1.1	94.767	

41188 rows × 21 columns

Hình 2: Bảng dữ liệu

Kiểm tra thông tin của bảng dữ liệu bằng lệnh **data.info()**, và các dòng dữ liệu bị khuyết thiếu dùng lệnh **data.isna().sum()**, nhóm sẽ thu được kết quả như sau:

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

Hình 3: Kiểm tra các cột có chứa dữ liệu khuyết thiếu

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

H4: kiểm tra thông tin bộ dữ liệu

Từ kết quả trên, nhóm nhìn thấy được bảng dữ liệu có 41188 dòng và chứa 21 cột, và kết quả không có dòng nào bị thiếu giá trị, nhưng có một vài điểm bất thường ngay trong bảng dữ liệu mà nhóm phát hiện ra được.

Nhận xét bảng dữ liệu: Nhóm nhìn thấy trong bảng có các giá trị '**unknown**', là các giá trị không xác định được trong một cột của dữ liệu, có thể là do từ phía khách hàng chưa biết chọn gì hoặc là do đó là giá trị bị nhiễu. Nhóm sẽ tiến hành một số kiểm tra để xem có bao nhiêu cột chứa giá trị '**unknown**' và cách xử lý các cột đó:

education	default	housing	loan
basic.4y	no	no	no
high.school	unknown	no	no
high.school	no	yes	no
basic.6y	no	no	no
high.school	no	no	yes
...	...	...	...
professional.course	no	yes	no
professional.course	no	no	no
university.degree	no	yes	no
professional.course	no	no	no
professional.course	no	yes	no

Nhóm sẽ tiến hành liệt kê ra các cột chứa các giá trị '**unknown**':

```
# Các cột chứa giá trị 'unknown' khác 'education'
categoricals = []
for i in data.columns:
  if (data[data[i]=='unknown'].shape[0] > 0) & (i != 'education'):
    categoricals.append(i)
categoricals
```

```
['job', 'marital', 'default', 'housing', 'loan']
```

H6: In ra các cột có chứa giá trị 'unknown'

Nhận xét: Có rất nhiều cột chứa giá trị '**unknown**' mà nhóm không biết được đó là do từ phía khách hàng chưa xác định thông tin của mình được hay là có một ý nghĩa nào khác, nên nhóm coi đó là các giá trị khuyết thiếu mà trong quá trình thu thập được dữ liệu người ta đã không bỏ trống mà điền trực tiếp vào ô đó là '**unknown**'. Và ngoài những cột trên có cột '**education**' cũng chứa giá trị 'unknown' nhưng không được liệt kê ở trên vì đây là thang đo thứ bậc có giá trị mặc định, nên sẽ được xử lý riêng bên ngoài.

```
data['education'].unique()
```

```
array(['basic.4y', 'high.school', 'basic.6y', 'basic.9y',
       'professional.course', 'unknown', 'university.degree',
       'illiterate'], dtype=object)
```

H7: Kiểm tra các giá trị trong cột ’education’

Từ các cột chứa giá trị ‘**unknown**’ được liệt kê ở trên đều có kiểu dữ liệu object, mục đích của việc liệt kê các cột này là chuẩn hóa dữ liệu đồng thời đưa các giá trị ‘**unknown**’ về giá trị NaN để nhóm có thể dễ dàng xử lý hơn.

Nhóm sẽ xử lý cột '**education**' trước, vì đây là cột có chứa giá trị '**unknown**' nhưng cũng là cột có ý nghĩa cấp bậc, nên không thể sử dụng hàm **LabelEncoder** để mã hóa được vì hàm này sẽ mã hóa ngẫu nhiên không theo thứ tự, sẽ mất đi ý nghĩa của cột '**education**'.

```
data['education'] = data['education'].replace(['basic.4y', 'high.school', 'basic.6y', 'basic.9y', 'professional.course',
                                               'university.degree', 'illiterate', 'unknown'], [4, 12, 6, 9, 14, 17, 0, np.nan])
```

H8: Chuẩn hóa cột ‘education’

Nhóm sẽ kết quả của cột **education** như sau:

education
4.0
12.0
12.0
6.0
12.0
...
14.0
14.0
17.0
14.0
14.0

H9: Kết quả sau khi chuẩn hóa cột ‘education’

Sau đó nhóm sẽ tiến hành xử lý các cột còn lại bằng hàm **LabelEncoder** để cho ra kết quả chuẩn hóa ngẫu nhiên và cũng tiện lợi hơn.

```
# Encoder các cột categorical
d = data.copy()
le = LabelEncoder()
for col in categoricals:
    le.fit(d[col])
    d[col+'_encoded'] = le.transform(d[col])
    d.loc[d[col]=='unknown', col+'_encoded'] = np.nan

d = d.drop(columns = categoricals, axis=1)
d
```

H10: chuẩn hóa các cột đã được liệu kê còn lại

Và nhóm sẽ thu lại được kết quả một bảng dữ liệu như sau:

y	job_encoded	marital_encoded	default_encoded	housing_encoded	loan_encoded
no	3.0	1.0	0.0	0.0	0.0
no	7.0	1.0	NaN	0.0	0.0
no	7.0	1.0	0.0	2.0	0.0
no	0.0	1.0	0.0	0.0	0.0
no	7.0	1.0	0.0	0.0	2.0
...	...	...	...	...	...
yes	5.0	1.0	0.0	2.0	0.0
no	1.0	1.0	0.0	0.0	0.0
no	5.0	1.0	0.0	2.0	0.0
yes	9.0	1.0	0.0	0.0	0.0
no	5.0	1.0	0.0	2.0	0.0

H11: kết quả chuẩn hóa

Khi này, nhóm sẽ thử kiểm tra các giá trị khuyết thiếu của các cột trong bảng dữ liệu lại một lần nữa:

```
# Kiểm tra một lần nữa các cột chứa dữ liệu bị thiếu
d.isna().sum().sort_values(ascending = False)
```

```
default_encoded    8597
education          1731
loan_encoded        990
housing_encoded     990
job_encoded         330
marital_encoded      80
cons.price.idx        0
y                     0
nr.employed           0
euribor3m             0
cons.conf.idx         0
age                   0
poutcome              0
previous              0
pdays                 0
campaign              0
duration              0
day_of_week           0
month                 0
contact               0
emp.var.rate          0
dtype: int64
```

H12: kiểm tra lại các giá trị khuyết thiếu trong cột

Nhận Xét: Vì các dòng dữ liệu chứa giá trị bị thiếu chiếm nhiều nhất là 20% so với số dòng từ bộ dữ liệu gốc (8597/41188), nên việc xóa các dòng dữ liệu bị thiếu sẽ có ảnh hưởng rất lớn đến việc phân lớp sau này cho kết quả không chính xác.

Chúng ta cùng kiểm tra lại các kiểu dữ liệu hiện đang có trong bảng dữ liệu và nếu nó có kiểu **object** thì nhóm sẽ tiến hành chuẩn hóa các dữ liệu còn lại:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              41188 non-null  int64
 1   education        39457 non-null  float64
 2   contact          41188 non-null  object
 3   month            41188 non-null  object
 4   day_of_week      41188 non-null  object
 5   duration         41188 non-null  int64
 6   campaign         41188 non-null  int64
 7   pdays            41188 non-null  int64
 8   previous         41188 non-null  int64
 9   poutcome         41188 non-null  object
 10  emp.var.rate     41188 non-null  float64
 11  cons.price.idx   41188 non-null  float64
 12  cons.conf.idx    41188 non-null  float64
 13  euribor3m        41188 non-null  float64
 14  nr.employed      41188 non-null  float64
 15  y                41188 non-null  object
 16  job_encoded      40858 non-null  float64
 17  marital_encoded  41108 non-null  float64
 18  default_encoded  32591 non-null  float64
 19  housing_encoded  40198 non-null  float64
 20  loan_encoded     40198 non-null  float64
dtypes: float64(11), int64(5), object(5)
memory usage: 6.6+ MB
```

H12: kiểm tra thông tin của bộ dữ liệu

Nhóm thấy được còn 5 cột chứa dữ liệu object, nên nhóm sẽ tiến hành chuẩn hóa các cột đó bằng hàm **OrdinalEncoder**:

```
# Mã hóa dữ liệu của các biến phân loại
from sklearn.preprocessing import OrdinalEncoder

ord_enc = OrdinalEncoder()
for i in d.columns:
  if d[i].dtypes == 'object':
    d[i] = ord_enc.fit_transform(d[[i]]).astype('int')
```

H13: Chuẩn hóa tự động các cột còn lại có KDL object

Nhóm sẽ thấy được kết quả như sau:

```
d.info()
```

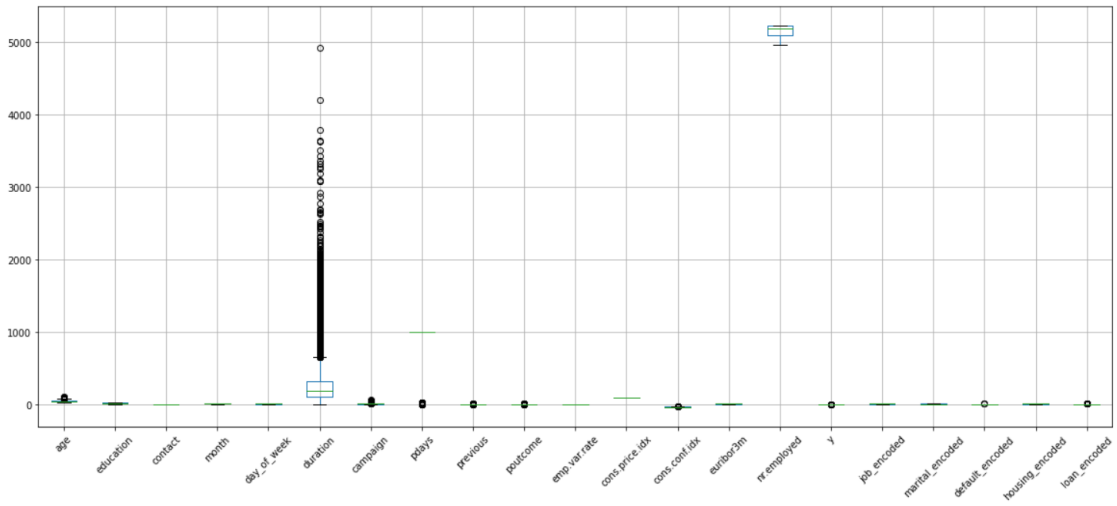
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   age              41188 non-null  int64  
 1   education        39457 non-null  float64
 2   contact          41188 non-null  int64  
 3   month            41188 non-null  int64  
 4   day_of_week      41188 non-null  int64  
 5   duration         41188 non-null  int64  
 6   campaign         41188 non-null  int64  
 7   pdays            41188 non-null  int64  
 8   previous         41188 non-null  int64  
 9   poutcome         41188 non-null  int64  
 10  emp.var.rate     41188 non-null  float64
 11  cons.price.idx   41188 non-null  float64
 12  cons.conf.idx    41188 non-null  float64
 13  euribor3m        41188 non-null  float64
 14  nr.employed      41188 non-null  float64
 15  y                41188 non-null  int64  
 16  job_encoded      40858 non-null  float64
 17  marital_encoded  41108 non-null  float64
 18  default_encoded  32591 non-null  float64
 19  housing_encoded  40198 non-null  float64
 20  loan_encoded     40198 non-null  float64
dtypes: float64(11), int64(10)
memory usage: 6.6 MB
```

H14: Kết quả chuẩn hóa

Tiếp theo nhóm sẽ tiến hành xử lý các outliers, vì không thể xóa các dữ liệu khuyết thiếu để tránh sai sót sau này nên nhóm sẽ xử lý outliers trước.

Nhóm sẽ kiểm tra các outliers bằng cách vẽ biểu đồ boxplot:

```
# Vẽ biểu đồ boxplot thể hiện các cột chứa các outliers
plt.figure(figsize=(20,8))
d.boxplot()
plt.xticks(rotation=45)
print()
```



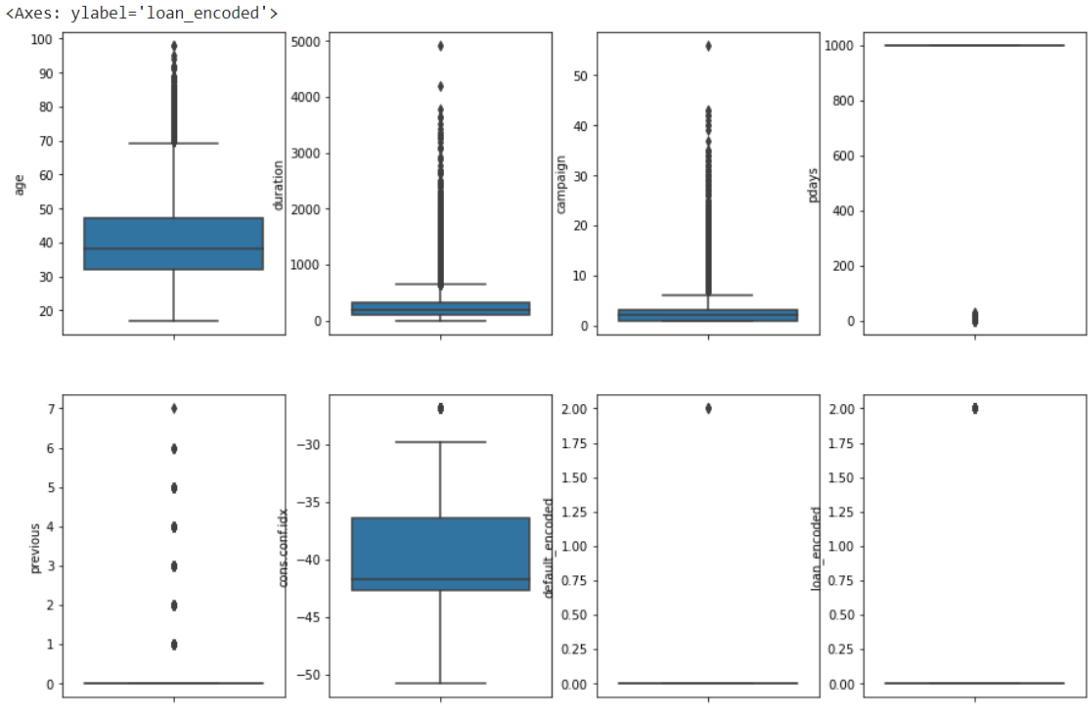
H15: biểu đồ boxplot để kiểm tra các cột dữ liệu có outliers

Nhận xét: Từ biểu đồ trên, nhóm nhìn thấy cột duration là chứa nhiều giá trị outliers nhất và các cột age, campaign, pdays, previous, cons.conf.idx, default_encoded, loan_encoded có chứa dữ liệu khuyết thiếu nhưng không đáng kể. Nên nhóm sẽ tiến hành kiểm tra từng biến để xem sẽ xử lý cột nào.

```
# Vẽ biểu đồ boxplot của từng biến
fig, axs = plt.subplots(nrows=2, ncols=4,figsize=(15,10), sharex=True)
sns.boxplot(y=d["age"],ax=axs[0][0])
sns.boxplot(y=d["duration"],ax=axs[0][1])
sns.boxplot(y=d["campaign"],ax=axs[0][2])
sns.boxplot(y=d["pdays"],ax=axs[0][3])
sns.boxplot(y=d["previous"],ax=axs[1][0])
sns.boxplot(y=d["cons.conf.idx"],ax=axs[1][1])
sns.boxplot(y=d["default_encoded"],ax=axs[1][2])
sns.boxplot(y=d["loan_encoded"],ax=axs[1][3])
```

H16: Kiểm tra từng cột có chứa outliers

Chúng ta sẽ nhìn thấy rõ các giá trị outliers trong các cột đã nêu trên:



H17: kết quả các cột chứa outliers

Nhận xét: Từ các biểu đồ trên các cột '**age**', '**duration**', '**previous**', '**campaign**' là chứa các outliers được nhìn thấy rõ nhất, nên nhóm sẽ tiến hành xử lý các outlier của các cột này.

```
# Xử Lý Outliers
df_clean = d.copy()
lower = 0.25
higher = 0.75
outlier_columns = ['age', 'duration', 'previous', 'campaign']
for column in outlier_columns:
    Q1 = d[column].quantile(.25)
    Q3 = d[column].quantile(.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    d = d[d[column] >= lower]
    d = d[d[column] <=upper]
```

H18: xử lý outliers

Tiếp theo nhóm sẽ tiến hành kiểm tra các outliers đã bị loại bỏ chưa bằng cách xem lại các thông tin có trong bảng dữ liệu:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30573 entries, 0 to 41186
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   age              30573 non-null  int64  
 1   education        29330 non-null  float64
 2   contact          30573 non-null  int64  
 3   month            30573 non-null  int64  
 4   day_of_week      30573 non-null  int64  
 5   duration         30573 non-null  int64  
 6   campaign         30573 non-null  int64  
 7   pdays            30573 non-null  int64  
 8   previous         30573 non-null  int64  
 9   poutcome         30573 non-null  int64  
 10  emp.var.rate     30573 non-null  float64
 11  cons.price.idx   30573 non-null  float64
 12  cons.conf.idx    30573 non-null  float64
 13  euribor3m        30573 non-null  float64
 14  nr.employed      30573 non-null  float64
 15  y                30573 non-null  int64  
 16  job_encoded      30322 non-null  float64
 17  marital_encoded  30526 non-null  float64
 18  default_encoded  23705 non-null  float64
 19  housing_encoded  29835 non-null  float64
 20  loan_encoded     29835 non-null  float64
dtypes: float64(11), int64(10)
memory usage: 5.1 MB
```

H19: kiểm tra lại thông tin bộ dữ liệu

Nhận xét: Nhìn vào kết quả trên, nhóm thấy các dòng giá trị đang từ 41186 còn 30573, các giá trị khuyết thiếu cũng đã giảm đi đáng kể.

Vì vẫn còn các giá trị khuyết thiếu trong dữ liệu, nhưng cũng chiếm không quá nhiều so với tổng các dòng dữ liệu, nên nhóm sẽ tiến hành điều giá trị trung bình của 9 dòng gần nhất vào dòng bị bỏ trống bằng hàm **KNNImputer**:

```python
from sklearn.impute import KNNImputer

# Khởi tạo đối tượng KNNImputer với n_neighbors = 9
imputer = KNNImputer(n_neighbors=9)

# Điền giá trị bị khuyết và trả về một mảng Numpy mới
new_array  = imputer.fit_transform(d)

# Tạo một DataFrame mới từ mảng Numpy
df = pd.DataFrame(data=new_array, columns=d.columns)
```

H20: Xử lý các giá trị khuyết thiếu

Sau đó, Nhóm sẽ kiểm tra các giá trị bị thiếu đã được xử lý chưa:

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

```
age               0
education         0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
job_encoded       0
marital_encoded   0
default_encoded   0
housing_encoded   0
loan_encoded      0
dtype: int64
```

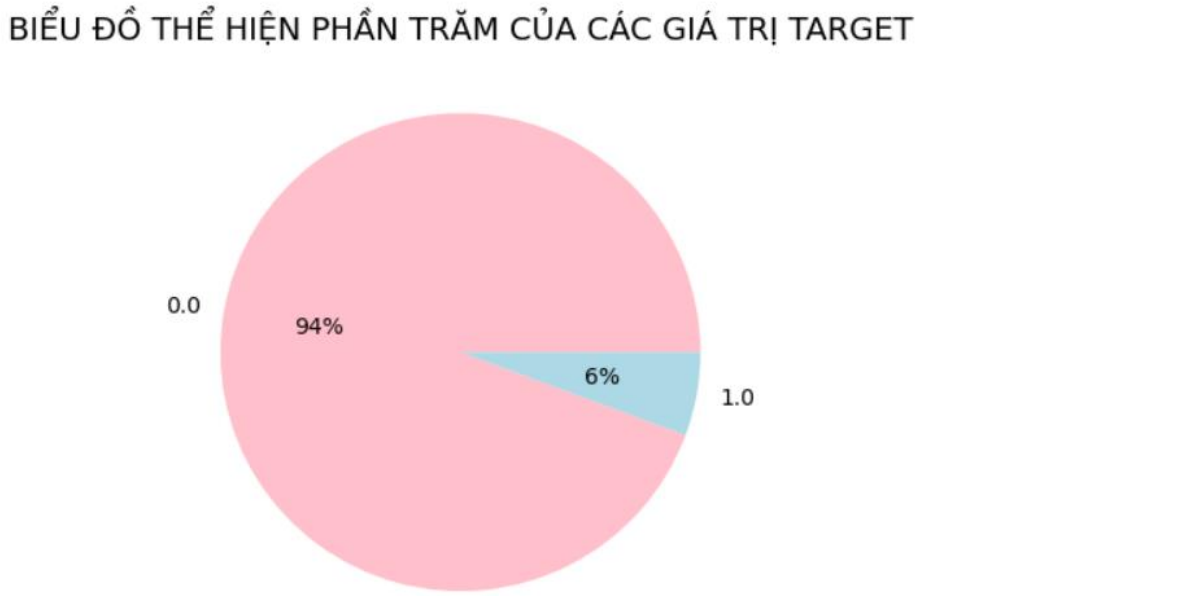
H21: kết quả sau xử lý

Ngoài xử lý các outliers và các giá trị khuyết thiếu trong bảng dữ liệu thô, nhóm còn phát hiện ra dữ liệu bị mất cân bằng, tức là biến target bị nghiêng về một giá trị quá nhiều, như vậy sẽ làm cho quá trình phân lớp sau này cho ra kết quả không chính xác.

Đầu tiên, nhóm sẽ kiểm tra mức độ data bị Imbalance trong bảng dữ liệu như thế nào bằng cách vẽ biểu đồ tròn để đưa ra kết quả:

```
[32] y = df["y"]
     X = df.drop(['y'], axis=1)
```

```
[36] #plt.figure(figsize = (14,8))
     plt.title('Biểu đồ thể hiện phần trăm của các giá trị target'.upper(), fontsize = 14)
     plt.pie(y.value_counts(),labels=y.unique(), autopct = '%.f%%', colors = {'lightblue', 'pink'})
     plt.show()
     #data đang bị imbalance nên cần balance lại data
```



H22: kiểm tra giá trị cột target bị imbalance

Nhận xét: Nhóm nhìn thấy phần trăm chênh lệch giữa các giá trị trong biến target là rất cao (chênh lệch khoảng 88%), trong đó giá trị 'yes' là chiếm tỷ lệ phần trăm là nhiều nhất. Khi đó, nhóm tiến hành xử lý data bị Imbalance, vì nó sẽ làm ảnh hưởng rất lớn đến kết quả dự báo, phân lớn sai sót sau này

Nhóm sẽ sử dụng phương thức **SMOTE** để xử lý vấn đề này:

```
[24] from imblearn.over_sampling import SMOTE
     smote = SMOTE(sampling_strategy=0.9)
     X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
[25] plt.title('Biểu đồ thể hiện phần trăm của các giá trị target sau khi balanced'.upper(), fontsize = 14)
     plt.pie(y_resampled.value_counts(),labels=y_resampled.unique(), autopct = '%.f%%', colors = {'lightblue', 'pink'})
     plt.show()
     #target sau khi đã balanced
```

H23: Xử lý giá trị bị Imbalance

Sau khi xử lý Imbalance nhóm sẽ thu được kết quả như sau:

H24: Kết quả xử lý data Imbalance

CHƯƠNG 3. MÔ HÌNH HỌC MÁY VÀ HỌC SÂU

3.1. Giới Thiệu Về 2 Mô Hình Học Máy Và Học Sâu

3.1.1. Mô Hình Máy Học

3.1.1.1. K-Nearest Neighbors

- K-nearest Neighbors (KNN) là một thuật toán học máy được sử dụng để giải quyết các vấn đề phân loại và hồi quy. Ý tưởng cơ bản của thuật toán KNN là dựa trên việc so sánh độ tương đồng giữa một mẫu dữ liệu mới với các mẫu dữ liệu đã được huấn luyện trước đó. KNN lưu trữ toàn bộ dữ liệu huấn luyện và sử dụng chúng để đưa ra quyết định cho một dữ liệu mới dựa trên những điểm dữ liệu gần nhất.

- Thuật toán KNN được triển khai như sau:

- Tính toán khoảng cách giữa dữ liệu mới và các điểm dữ liệu đã biết trong tập huấn luyện. Các metric tính khoảng cách phổ biến của KNN như:

  - Khoảng cách Euclidean: là khoảng cách đường thẳng giữa hai điểm trong không gian Euclide. Cho hai điểm p=(p1, p2, ..., pn) và q=(q1, q2, ..., qn), khoảng cách Euclidean được tính bằng công thức:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

H25: Công thức tính khoảng cách Euclidean.

  - Khoảng cách Manhattan: là khoảng cách theo trục tọa độ khi di chuyển giữa hai điểm. Cho hai điểm p=(p1, p2, ..., pn) và q=(q1, q2, ..., qn), khoảng cách Manhattan được tính bằng công thức:

$$d(p,q) = \sum_{i=1}^{n}|p_i - q_i|$$

H26: Công thức tính khoảng cách Manhattan.

  - Khoảng cách Mahalanobis: tính khoảng cách dựa trên sự khác biệt giữa các vector dữ liệu và các vector trung tâm của chúng. Cho hai vector dữ liệu x và y, khoảng cách Mahalanobis được tính bằng công thức:

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

H27: Công thức tính khoảng cách Mahalanobis.

Trong đó, S là ma trận hiệp phương sai của toàn bộ các vector dữ liệu.

- Khoảng cách Cosine: là khoảng cách giữa hai vector trong không gian đa chiều dựa trên góc giữa chúng. Cho hai vector a và b, khoảng cách Cosine được tính bằng công thức:

$$d(a,b) = 1 - \frac{a \cdot b}{\|a\|\|b\|}$$

H28: Công thức tính khoảng cách Cosine.

Trong đó, a.b là tích vô hướng của hai vector, và |a| và |b| là độ dài của chúng.

- Chọn k điểm dữ liệu gần nhất với dữ liệu mới.
- Đối với bài toán phân loại, lớp của dữ liệu mới được phân loại bằng cách chọn lớp phổ biến nhất trong k điểm gần nhất. Đối với bài toán hồi quy, giá trị đầu ra của dữ liệu mới được tính bằng cách lấy trung bình giá trị đầu ra của k điểm gần nhất.

KNN là thuật toán phi tham số, điều đó có nghĩa là không có giả định nào về phân phối của dữ liệu. Thuật toán này đơn giản và dễ cài đặt, nó có thể hoạt động tốt trên các tập dữ liệu nhỏ có biên giới phân lớp đơn giản, và không yêu cầu nhiều thời gian để huấn luyện. Độ phức tạp tính toán nhỏ

Tuy nhiên, KNN cũng có một số hạn chế, bao gồm:

- KNN yêu cầu lưu trữ toàn bộ tập dữ liệu huấn luyện trong bộ nhớ, điều này có thể gây ra vấn đề về tài nguyên bộ nhớ khi tập dữ liệu quá lớn.
- KNN có thể không hoạt động tốt với các tập dữ liệu có số chiều lớn.
- KNN không thể xác định được quan hệ giữa các đặc trưng.
- Dễ bị ảnh hưởng bởi outliers khi k nhỏ.

Để cải thiện hiệu suất của thuật toán KNN, ta có thể thực hiện các bước sau:

- Chuẩn hóa dữ liệu để giảm thiểu ảnh hưởng của các biến về quy mô.
- Tiền xử lí dữ liệu, loại bỏ outliers, giảm chiều dữ liệu. Sử dụng phương pháp chọn đặc trưng để loại bỏ các đặc trưng không quan trọng.

3.1.1.1. Support Vector Machine

Support Vector Machine (SVM) là một thuật toán học có giám sát dùng để giải quyết các bài toán phân loại và hồi quy. Ý tưởng của thuật toán SVM là tìm ra một siêu mặt phẳng trong không gian dữ liệu để phân loại các điểm dữ liệu vào các lớp tương ứng.

Thuật toán SVM có ưu điểm như sau:

- Hiệu quả cao trong việc xử lý dữ liệu lớn: SVM có thể giải quyết các bài toán phân loại với dữ liệu lớn mà không gây ra vấn đề về bộ nhớ.
- Độ chính xác cao: SVM cung cấp độ chính xác cao khi xử lý các bài toán phân loại phức tạp.
- Linh hoạt trong việc chọn hàm kernel: SVM cho phép chọn nhiều loại hàm kernel khác nhau để ánh xạ các điểm dữ liệu vào không gian cao chiều hơn, từ đó giúp tăng độ chính xác của thuật toán.

Tuy nhiên, thuật toán SVM cũng có những hạn chế sau:

- Đòi hỏi nhiều thời gian và công sức để tối ưu hoá các tham số: Tham số của SVM như độ rộng của vùng an toàn, độ phân cách giữa các lớp, độ dốc của hàm kernel phụ thuộc vào dữ liệu và cần được tối ưu hoá để đạt được kết quả tốt nhất. Việc tìm kiếm các giá trị tối ưu của các tham số này có thể rất khó khăn và tốn nhiều thời gian.
- Nhạy cảm với nhiễu: SVM có thể bị ảnh hưởng bởi các điểm dữ liệu nhiễu, vì vậy phải cẩn thận trong việc xử lý các điểm nhiễu.

Để cải thiện hiệu quả của thuật toán SVM, ta có thể áp dụng một số kỹ thuật như:

- Tối ưu hoá các tham số: Tối ưu hoá các tham số của SVM như hàm kernel, độ rộng của vùng an toàn và độ phân cách giữa các lớp có thể giúp tăng độ chính xác của thuật toán.
- Tiền xử lí outliers, xử lí loại bỏ nhiễu hoặc xử lí cho phù hợp với bài toán đặt ra trước khi đưa vào mô hình.

3.1.1. Mô Hình Học Sâu – Artificial Neutral Network

Artificial Neural Network (ANN), hay còn gọi là mạng neuron nhân tạo, là một loại thuật toán học máy được lấy cảm hứng từ cách thức hoạt động của hệ thần kinh sinh học. ANN được xây dựng bằng cách kết nối các đơn vị xử lý đơn giản (neuron) lại với nhau để tạo thành một mạng lưới neuron tương tác và thực hiện các phép tính toán phức tạp.

Ý tưởng của ANN là học từ dữ liệu thông qua việc tinh chỉnh các trọng số kết nối giữa các neuron để tối ưu hóa hàm mất mát. Một mô hình ANN thường gồm có các lớp neuron đầu vào, các lớp neuron ẩn và lớp neuron đầu ra. Dữ liệu được đưa vào qua lớp neuron đầu vào, sau đó được truyền qua các lớp neuron ẩn để cuối cùng đưa ra kết quả qua lớp neuron đầu ra. Các neuron trong một lớp neuron sẽ nhận đầu vào từ các neuron của lớp trước đó và tính toán đầu ra của mình bằng một hàm kích hoạt nhất định.

Ưu điểm của ANN:

- Khả năng học và khả năng tự điều chỉnh: ANN có khả năng tự điều chỉnh các trọng số kết nối giữa các neuron để tối ưu hóa một hàm mất mát, do đó nó có khả năng học từ dữ liệu và cải thiện hiệu suất dự đoán.
- Khả năng xử lý dữ liệu phi tuyến tính: ANN có khả năng xử lý dữ liệu phi tuyến tính, cho phép nó học được các mối quan hệ phức tạp giữa các đặc trưng.
- Khả năng tìm ra mối quan hệ giữa các đặc trưng: ANN có khả năng tìm ra mối quan hệ giữa các đặc trưng, giúp cải thiện khả năng dự đoán.

Hạn chế của ANN:

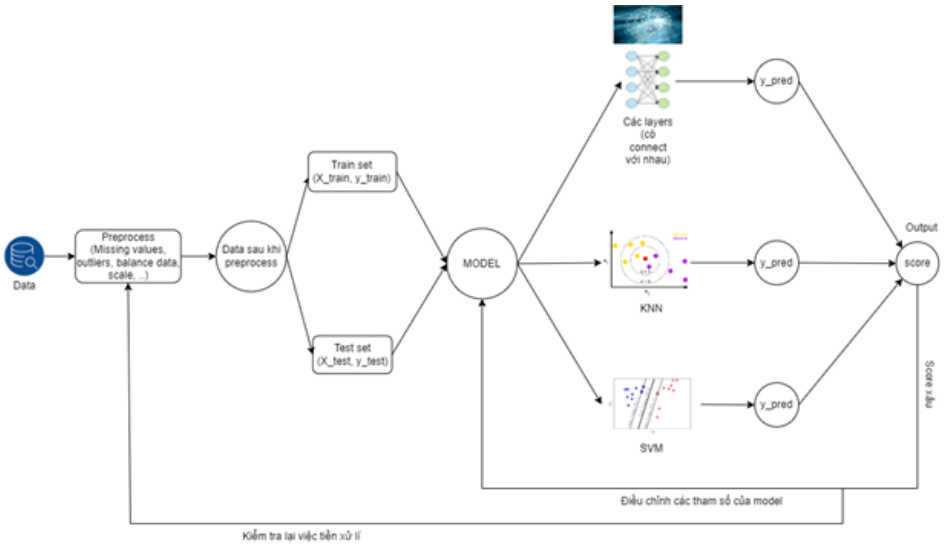
- Cần một lượng lớn dữ liệu huấn luyện: ANN cần một lượng lớn dữ liệu huấn luyện để học được mô hình và đưa ra dự đoán chính xác.
- Độ phức tạp tính toán cao: ANN có độ phức tạp tính toán cao, đòi hỏi một số lượng lớn các phép tính để đưa ra dự đoán, đặc biệt là khi số lượng neuron và lớp neuron lớn.
- Overfitting: Trong quá trình huấn luyện, ANN có thể mất đi hiệu quả dự báo trên tập dữ liệu mới do quá trình học tập quá nhiều các chi tiết cục bộ và không khái quát được những đặc trưng chung của dữ liệu. Điều này được gọi là overfitting.

- Khó khăn trong lựa chọn kiến trúc: Kiến trúc của ANN phải được chọn sao cho phù hợp với đặc tính của dữ liệu. Tuy nhiên, không có cách nào đơn giản để xác định kiến trúc tối ưu cho một bài toán cụ thể.
- Khó khăn trong việc diễn giải kết quả: ANN được coi là một "hộp đen" do tính chất phi tuyến của nó, điều này khiến cho việc diễn giải kết quả rất khó khăn. Điều này khiến cho ANN ít phù hợp trong các bài toán mà diễn giải kết quả là yếu tố quan trọng.

Cách để cải thiện ANN:

- Tăng cường số lượng dữ liệu: ANN có thể học tốt hơn khi có nhiều dữ liệu hơn để học. Vì vậy, một cách để cải thiện hiệu suất của ANN là tăng cường số lượng dữ liệu huấn luyện.
- Tinh chỉnh tham số: ANN có nhiều tham số như số lượng hidden layers, số lượng nơ-ron trong mỗi lớp, batch_size, và epochs. Tinh chỉnh tham số của mô hình là cách để cải thiện hiệu suất của nó.
- Sử dụng hàm kích hoạt phù hợp: Hàm kích hoạt là một yếu tố quan trọng trong ANN. Sử dụng hàm kích hoạt phù hợp sẽ giúp cho mô hình học tốt hơn. Ví dụ, hàm kích hoạt ReLU (Rectified Linear Unit) thường được sử dụng vì nó đơn giản và hiệu quả.
- Thêm các kỹ thuật chính quy hóa (Regularization): Các kỹ thuật chính quy hóa như Dropout và L2 regularization có thể giúp tránh overfitting trong ANN.
- Sử dụng kiến trúc mạng phù hợp: Chọn kiến trúc mạng phù hợp là cách để cải thiện hiệu suất của ANN. Kiến trúc mạng phổ biến bao gồm: mạng nơ-ron truyền thẳng, mạng nơ-ron sâu, và mạng học sâu (deep learning).
- Tiền xử lý dữ liệu: Tiền xử lý dữ liệu có thể cải thiện hiệu suất của ANN. Ví dụ, chuẩn hóa dữ liệu, loại bỏ nhiễu, và chọn các tính năng quan trọng có thể giúp cho ANN học tốt hơn.

3.2. Ứng Dụng 2 Mô Hình Học Máy Và Học Sâu Cho Bài Toán



H29: Framework xử lí dữ liệu và xây dựng mô hình.

3.2.1. Mô Hình Máy Học

```
[36] def score_model(m, y_test):
       use_model = m
       use_model.fit(X_train, y_train)
       y_pred = use_model.predict(X_test)


       accuracy = accuracy_score(y_test, y_pred)
       precision = precision_score(y_test, y_pred, average='weighted')
       recall = recall_score(y_test, y_pred, average='weighted')
       f1 = f1_score(y_test, y_pred, average='weighted')
       auc = roc_auc_score(y_test, y_pred, multi_class='ovr')
       print('Accuracy:', accuracy)
       print('Precision:', precision)
       print('Recall:', recall)
       print('F1 score:', f1)
       print('AUC:', auc)
```

H30: Xây dựng hàm đánh giá mô hình bằng các điểm số (accuracy, precision, recall, F1, AUC)

Tạo hàm **score_model** có chức năng đánh giá cho mô hình học máy **m** dựa trên tập dữ liệu kiểm tra **y_test** , với đầu vào là mô hình máy học **m** và tập kiểm tra **y_test** sau khi train_split. Sau đó hàm sẽ huấn luyện mô hình ***m*** dựa trên tập dữ liệu huấn luyện **X_train** và **y_train** rồi tiến hành dự đoán trên tập dữ liệu kiểm tra **X_test** tạo thành tập **y_pred**.

Sau đó tính toán các chỉ số đánh giá như accuracy_score, precision_score, recall_score, f1_score. Với tham số average = ' **weighted**' có nghĩa là tính trung bình trọng số của các chỉ số đánh giá cho từng lớp, với trọng số được tính bằng số lượng mẫu trong mỗi lớp. Với tham số multi_class= '**ovr**' sử dụng phương pháp one-vs-rest.

3.2.1.1. K-Nearest Neighbors

```
] score_model(KNeighborsClassifier(n_neighbors = 2), y_test)
```

H31: Mô hình KNN với k = 2, các tham số khác được chọn là mặc định.

Đối với mô hình KNN, ta sẽ chọn k = 2 với với tham số metric được mặc đỉnh sẵn là minkowski và các tham số khác cũng được để mặc định.

Ta gọi hàm **score_model** với mô hình **KNeighborsClassifier** với tham số **n_neighbors = 2** và tập dữ liệu kiểm tra **y_test**

3.2.1.2. Support Vector Machine

Đối với mô hình SVM ta sẽ chọn kernel là linear với các tham số khác được để mặc định.

```
score_model(svm.SVC(kernel='linear'), y_test)
```

H32: Mô hình SVM với kernel là linear và các tham số khác được chọn là mặc định.

Ta gọi hàm ***score_model*** với mô hình ***SVC*** với tham số ***kernel = linear*** và tập dữ liệu kiểm tra ***y_test***

3.2.2. Mô Hình Học Sâu – Artificial Neutral Network.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import regularizers



model = Sequential()
model.add(Dense(128, kernel_initializer='he_uniform', activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, kernel_initializer='he_uniform', activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the model
dl_train_model = model.fit(X_train, y_train, batch_size=32, epochs=100, validation_data=(X_test, y_test))
```

H33: Tạo kiến trúc cho mô hình học sâu.

Ta sẽ tạo một mô hình với 3 Dense layers với mỗi layer chứa số neurons khác nhau và activation khác nhau. Dense layer đầu tiên có 128 neuron và dùng activation relu với **kernel_initializer** là **he_uniform**, đối với dense layer thứ 2 cũng tương tự như dense layer thứ nhất nhưng ta giảm số neurons xuống còn 64. Và trong dense layer cuối cùng thì ta chỉ dùng 1 neuron và chọn activation là sigmoid để cho ra output là các xác suất nằm trong khoảng 0 đến 1. Ngoài ra, ta còn thêm các **drop layer** để có thể giảm bớt tình trạng model bị overfitting.

Sau khi đã định nghĩa cấu trúc của mô hình bằng cách thêm các layer vào trong mô hình thì ta sẽ tiến hành compile nó lại với loss là **binary_crossentropy** (vì dữ liệu target của ta là binary), optimizer là **adam** và metrics là **accuracy** để model được đánh giá bằng **accuracy** là chính trong quá trình huấn luyện mô hình.

Cuối cùng, ta tiến hành huấn luyện mô hình bằng việc fit model với tập dữ liệu đã chia sẵn là **X_train** và **y_train**, với **batch_size** là 32, **epochs** là 100 và **validation_data** là **X_test**, **y_test** để ta có thể thấy được điểm số của loss và accuracy trên tập train và tập validation trong quá trình huấn luyện.

CHƯƠNG 4. CÁC KẾT QUẢ THỰC NGHIỆM

4.1. Các Tình Huống

4.1.1. Đối với mô hình máy học

Mô hình học máy KNN: Vì khi cho k = 2 và metric được mặc định sẵn là minkowski thì mô hình KNN cho ra các điểm số rất cao nên ta sẽ không xét các trường hợp khác nữa.

Mô hình học máy SVM: Đối với mô hình SVM thì khi ta chọn kernel là linear thì mô hình cũng cho ra kết quả là các điểm số cao tuy nhiên còn thấp hơn model knn rất nhiều nên ta sẽ thử dùng các kernel khác (như rbf, poly,..) để có thể tăng hiệu quả của mô hình và sau đó chọn ra kernel tốt nhất.

4.1.2. Đối với mô hình học sâu

```
def define_model(dropout_rate=0.0, weight_constraint=0, neurons= (256,), reg_strength=0.01):
    model = Sequential()
    if len(neurons) == 1:
        model.add(Dense(neurons[0], activation='relu', input_dim=x_grid.shape[1],
                        kernel_constraint=maxnorm(weight_constraint), kernel_regularizer=regularizers.l2(reg_strength)))
        model.add(Dropout(dropout_rate))
        model.add(Dense(neurons[0], activation='relu', kernel_regularizer=regularizers.l2(reg_strength)))
        model.add(Dropout(dropout_rate))
        model.add(Dense(neurons[0], activation='relu', kernel_regularizer=regularizers.l2(reg_strength)))
        model.add(Dropout(dropout_rate))
        model.add(Dense(1, activation='sigmoid'))
    else:
        model.add(Dense(neurons[0], activation='relu', input_dim=x_grid.shape[1],
                        kernel_constraint=maxnorm(weight_constraint), kernel_regularizer=regularizers.l2(reg_strength)))
        model.add(Dropout(dropout_rate))
        for n in neurons[1:]:
            model.add(Dense(n, activation='relu', kernel_regularizer=regularizers.l2(reg_strength)))
            model.add(Dropout(dropout_rate))
        model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='Adam', metrics=['accuracy'])
    return model

from keras.wrappers.scikit_learn import KerasClassifier

model = KerasClassifier(build_fn=define_model,
                        verbose=0)
```

H34: Hàm định nghĩa kiến trúc của mô hình học sâu.

Ta sẽ tạo một hàm **define_model** để có thể tạo ra các mô hình có thể có các layers có cùng neuron với nhau hoặc các layer sẽ có số neurons khác nhau. Ở đây, ta sẽ tạo ra các mô hình có 2, 3 hoặc 4 lớp layer trở lên. Và nhóm quyết định chọn các mô hình đều có 4 lớp layer với 3 Dense layer có activation là relu, còn Dense layer cuối cùng sẽ là activation sigmoid. Các mô hình này được tạo ra để nhằm mục đích ta có thể tìm ra được các siêu tham số tối ưu nhất cho mô hình để có thể cho ra điểm số cao và đúng với mong muốn.

```
dropout_rate = [0.2, 0.3, 0.4]
weight_constraint = [1, 2, 3]
neurons = [(256,), (256, 192, 256), (512, 256, 256), (512,)]
reg_strength = [0.01, 0.001, 0.0001]
batch_size = [64, 96, 128]
epochs = [5, 10]
```

ɔdel 1

```
param_grid = dict(dropout_rate=dropout_rate,
                  weight_constraint=weight_constraint,
                  neurons=neurons, reg_strength=reg_strength,
                  batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs = 3, cv=3)
```

```
grid_result = grid.fit(x_grid, y_grid)
```

H35: Tạo param_grid chứa các giá trị cần chọn lọc của các siêu tham số và tiến hành huấn luyện các mô hình dựa trên các giá trị của từng siêu tham số.

Các siêu tham số được điều chỉnh để chọn ra được siêu tham số tối ưu nhất ứng với mô hình ở đây có 5 siêu tham số là dropout_rate, weight_constraint, neurons, reg_strength, batch_size. Đối với epochs thì ta chỉ đặt epochs nhỏ để cho quá trình huấn luyện các dữ liệu diễn ra nhanh hơn để ta có thể chọn ra các siêu tham số một cách nhanh chóng. Vì số các giá trị trong các siêu tham số có khá nhiều và ta sẽ phải chọn từng giá trị trong từng siêu tham số để huấn luyện mô hình nên sẽ có tất cả 648 mô hình ứng với 648 bộ giá trị từ 6 siêu tham số trên.

Sau đó, ta tiến hành tạo 1 dictionary chứa 6 siêu tham số này và các giá trị của chúng. Ở đây, ta sẽ dùng GridSearchCV của sklearn để tiến hành chọn ra từng giá trị của từng siêu tham số mà khi cho vào mô hình thì mô hình sẽ đạt được điểm cao nhất. Ở đây, ta sẽ chọn cv = 3 để chia ra 3 folds và sử dụng phương pháp đánh giá chéo (cross-validation). Trong 3 folds thì ta sẽ dùng 2 folds để huấn luyện và dùng folds còn lại để kiểm định. Như vậy, ta có thể đánh giá được khả năng dự báo của mô hình đối với các dữ liệu unseen.

Tiếp theo, ta sẽ tiến hành train các bộ giá trị từ các siêu tham số để tiến hành chọn ra bộ giá trị cho ra mô hình có điểm số cao nhất. Ví dụ, ta sẽ có các bộ giá trị như sau:

- Bộ siêu tham số 1 :
  + batch_size: 128,
  + dropout_rate: 0.2

+ epochs: 10
+ neurons: (512, 256, 256)
+ reg_strength: 0.0001
+ weight_constraint: 1

– Bộ siêu tham số 2 :

+ batch_size: 96,
+ dropout_rate: 0.2
+ epochs: 10
+ neurons: (512,)
+ reg_strength: 0.0001
+ weight_constraint: 1

– Bộ siêu tham số 3 :

+ batch_size: 128,
+ dropout_rate: 0.2
+ epochs: 1000
+ neurons: (512, 256 , 256)
+ reg_strength: 0.0001
+ weight_constraint: 1

...

4.2. Phân Tích và Đánh Giá

4.2.1. Mô hình K-Nearest Neighbors (K-NN)

Vì biến target y chỉ có 2 giá trị nên ta chọn chỉ số k =2 là lựa chọn tối ưu nhất

```
score_model(KNeighborsClassifier(n_neighbors = 2), y_test)
```

```
Accuracy: 0.9579241286347006
Precision: 0.9582150615217825
Recall: 0.9579241286347006
F1 score: 0.9579635101283803
AUC: 0.9585568923523464
```

H36: Các điểm số của mô hình máy học KNN.

Với các chỉ số rất cao, cụ thể:

- Tính chính xác (CA) là 95.97%
- Giá trị trung bình điều hoà (F1) là 95.97%
- Độ chính xác (Precision) là 96%
- Độ phủ (Recall) là 95.97%

4.2.2. Mô hình Support Vector Machine (SVM)

```
[78] score_model(svm.SVC(kernel='linear'), y_test)

     Accuracy: 0.8549008280377431
     Precision: 0.8567616028743354
     Recall: 0.8549008280377431
     F1 score: 0.855187133938328
     AUC: 0.8562081833863716

[79] score_model(svm.SVC(kernel='rbf'), y_test)

     Accuracy: 0.9224918159060274
     Precision: 0.9244472408979023
     Recall: 0.9224918159060274
     F1 score: 0.9226522233987065
     AUC: 0.9246988911717323

[80] score_model(svm.SVC(kernel='poly', degree=2), y_test)

     Accuracy: 0.8552859618717504
     Precision: 0.8565696826332095
     Recall: 0.8552859618717504
     F1 score: 0.8544305742643498
     AUC: 0.8497253291599279
```

H37: Các điểm số của mô hình SVM theo từng kernel.

Ta chọn 3 kernel đặc trưng để xử lý dữ liệu trong mô hình SVM : linear , rbf (Gaussian) và poly.

Qua đó ta thấy mô hình SVM với kernel RBF đạt chỉ số cao hơn với:

- Tính chính xác (CA) là 92.25%
- Giá trị trung bình điều hoà (F1) là 92.25%
- Độ chính xác (Precision) là 92.24%

- Độ phủ (Recall) là 92.25%

4.2.3. Mô hình Artificial neuron network (ANN)

Sau khi đã chọn ra được bộ siêu tham số batch_size: 128, dropout_rate: 0.2 , epochs: 1000, neurons: (512, 256 , 256) , reg_strength: 0.0001 ta có được mô hình với tính chính xác cao nhất là 96.53%

```
[ ] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
    y_test = y_test.astype(int)
    y_pred = (y_pred > 0.5).astype(int)
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    print(acc)
    print(prec)
    print(rec)
    print(f1)

    0.9652883666696565
    0.9459302325581396
    0.9783523752254961
    0.9618681643511676
```

H38: Dự đoán kết quả của mô hình học sâu và đánh giá mô hình qua các điểm số (accuracy, F1,...).

Nhận xét:

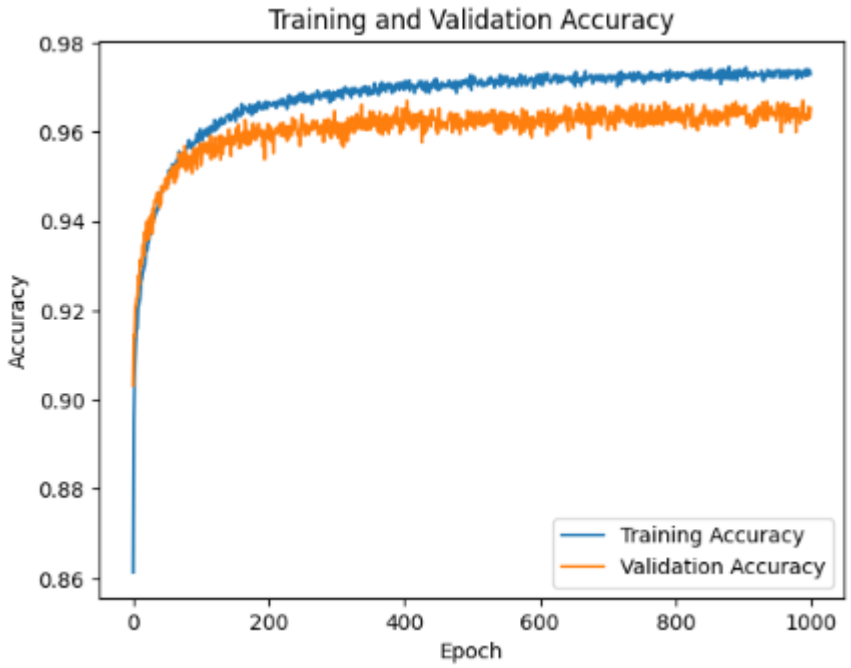
Ta đánh giá chất lượng các mô hình thông qua các chỉ số tính chính xác (accuracy), độ chính xác (precision), độ phủ (recall), giá trị trung bình điều hoà (f1 score).

Qua đó ta thấy mô hình ANN mang lại số liệu tốt nhất, cụ thể:

- Tính chính xác (CA) là 96.53%
- Giá trị trung bình điều hoà (F1) là 94.59%
- Độ chính xác (Precision) là 97.83%
- Độ phủ (Recall) là 96.18%

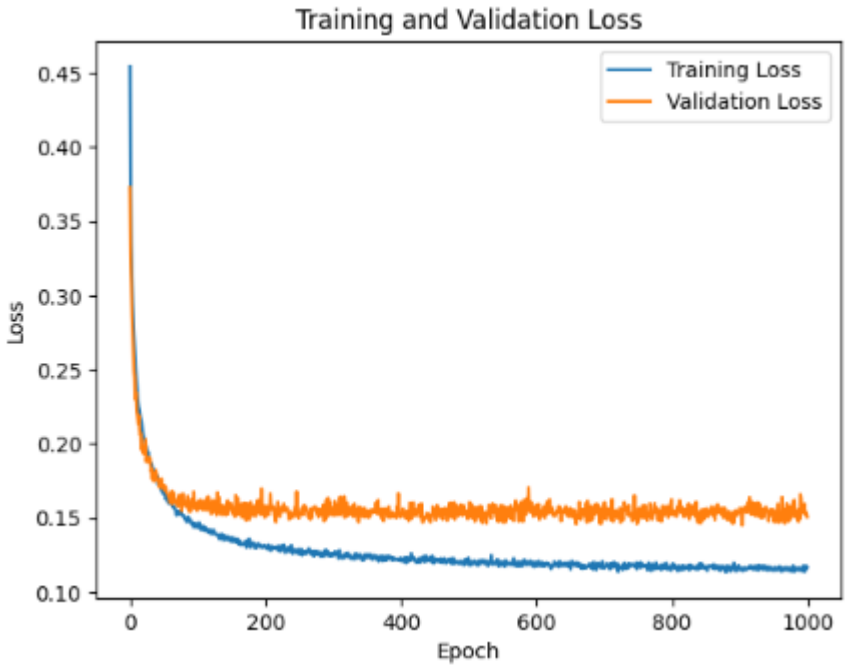
Mặc dù có thể điều chỉnh các thông số để mô hình học sâu ANN trở nên tốt hơn so với 2 mô hình học máy nhưng thời gian để huấn luyện của mô hình học sâu nhìn chung khá chậm so với 2 mô hình còn lại.

4.3. Biểu Đồ



H39: Biểu đồ thể hiện giá trị của accuracy của tập train và tập validation theo epochs.

Biểu đồ thể hiện xu hướng độ chính xác qua các epoch cho cả dữ liệu huấn luyện và kiểm định. Trục y biểu thị độ chính xác, trong khi trục x biểu thị số epoch. Đường màu xanh biểu thị độ chính xác huấn luyện, trong khi đường màu cam biểu thị độ chính xác kiểm định . Ta thấy kết quả train và test gần nhau chứng tỏ mô hình không overfitting . Kết quả đạt được rất tốt với điểm accuracy của tập train ở epoch thứ 1000 đạt đến 0.9730 , đối với tập test điểm accuracy ở epoch thứ 1000 đạt đến 0.9653 .



H40: Biểu đồ thể hiện giá trị của loss của tập train và tập validation theo epochs.

Biểu đồ thể hiện xu hướng mất mát qua các epoch cho cả dữ liệu huấn luyện và kiểm định. Trục y biểu thị mất mát, trong khi trục x biểu thị số epoch. Đường màu xanh biểu thị mất mát huấn luyện, trong khi đường màu cam biểu thị mất mát kiểm định. Thông qua biểu đồ ta thấy được có sự cải thiện trong quá trình mô hình học tập khi giá trị loss qua từng epoch giảm dần từ 0.35 xuống 0.12 , tại epoch thứ 1000 gia trị loss giảm còn 0.1167 ; Kết quả giá trị loss của test cũng tương tự khi giảm từ 0.3 xuống còn 0.15, tại epoch thứ 1000 giảm còn 0.1506.

CHƯƠNG 5. KẾT LUẬN

5.1. Các Kết Quả Đạt Được

Sau các bước tiền xử lý dữ liệu và đưa vào huấn luyện trong hai mô hình học máy, một mô hình học sâu. Kết quả cho thấy, mô hình học sâu ANN cho kết quả dự đoán chính xác nhất với tỉ lệ độ chính xác lên đến 96.53%. Như vậy việc sử dụng mô hình học sâu ANN có khả năng dự đoán khách hàng tiềm năng mở tài khoản tiết kiệm với độ chính xác rất cao. Việc áp dụng học máy để dự đoán khách hàng tiềm năng sẽ giúp cho các tổ chức tài chính tiết kiệm được thời gian và nỗ lực trong việc tìm kiếm khách hàng tiềm năng, từ đó tăng trưởng kinh doanh và tăng thu nhập. Tuy nhiên, cần lưu ý rằng việc huấn luyện mô hình học sâu ANN tốn khá nhiều thời gian và công sức, và cần áp dụng các phương pháp tối ưu hóa tham số để đạt được kết quả tốt nhất.

5.2. Những Hạn Chế và Hướng Phát Triển

Những hạn chế:

Tuy nhiên quá trình huấn luyện mô hình học sâu ANN tốn khá nhiều thời gian và yêu cầu sự hiểu biết về toán học và lập trình. Đồng thời, trong quá trình điều chỉnh tham số cho mô hình học sâu ANN, việc đạt được kết quả cao dễ dẫn đến overfitting - khi mô hình học tập quá nhiều từ dữ liệu huấn luyện và không thể tổng quát hóa với dữ liệu mới. Điều này cần phải được kiểm soát để đảm bảo tính đúng đắn của mô hình.

Hơn nữa, để đạt được kết quả mong muốn, tốn nhiều thời gian trong việc chọn các tham số của các mô hình. Đối với mô hình KNN, việc đánh giá mô hình trở nên khó khăn vì mô hình này cho ra điểm số quá cao. Do đó, việc xây dựng mô hình học sâu trở nên phức tạp hơn vì mô hình này bắt buộc điểm số phải cao hơn mô hình máy học.

Ngoài ra, tốc độ huấn luyện của mô hình có thể bị ảnh hưởng bởi cấu hình phần cứng và phần mềm của hệ thống, và đòi hỏi một nguồn tài nguyên máy tính đủ mạnh để có thể xử lý các tác vụ huấn luyện phức tạp.

Hướng phát triển:

Trong quá trình phát triển và ứng dụng các mô hình máy học, chúng tôi đã gặp phải một số hạn chế trong việc tối ưu hóa hiệu suất và độ chính xác của mô hình như đã đề cập ở trên. Để cải thiện hiệu suất và độ chính xác của mô hình, chúng tôi đề xuất một số hướng phát triển sau đây.

Đầu tiên, để giải quyết vấn đề mất cân bằng dữ liệu, chúng tôi đề xuất tích hợp thêm dữ liệu mới vào bộ dữ liệu đã có để cân bằng lại dữ liệu thô ban đầu. Kết quả là mô hình sẽ có khả năng dự đoán chính xác hơn trên toàn bộ bộ dữ liệu.

Thứ hai, việc tối ưu hóa mô hình là một trong những vấn đề quan trọng trong lĩnh vực học máy. Chúng tôi đề xuất sử dụng các thuật toán tối ưu hóa để cải thiện hiệu suất và độ chính xác của mô hình. Bằng cách tối ưu hóa tham số của mô hình, ta có thể giảm thiểu sai số và cải thiện độ chính xác của mô hình.

Thứ ba, khi mô hình học máy cho ra điểm số cao, ta không cần sử dụng mô hình học sâu và chỉ ứng dụng mô hình học sâu vào các dữ liệu khó học hay phức tạp. Điều này giúp tiết kiệm thời gian và tài nguyên khi chỉ sử dụng các mô hình đơn giản để giải quyết các bài toán dễ hơn.

Thứ tư, để xây dựng mô hình đa nhiệm, chúng tôi đề xuất tích hợp bài toán dự đoán xác suất khách hàng mở tài khoản tiết kiệm với bài toán dự đoán khách hàng sẽ thực hiện giao dịch nào tiếp theo. Kết quả là mô hình sẽ cung cấp thông tin chi tiết hơn về hành vi của khách hàng, từ đó giúp ngân hàng phát triển các chiến lược tiếp thị và quản lý khách hàng hiệu quả hơn.

Ngoài ra, để tối ưu hóa quá trình huấn luyện mô hình, có thể sử dụng các kỹ thuật tăng tốc học (learning rate schedule) để giảm thời gian và chi phí cho quá trình huấn luyện mô hình.

Các kỹ thuật học tăng cường RL có thể được áp dụng để cải thiện khả năng tương tác với khách hàng và tăng cường trải nghiệm của khách hàng. Việc sử dụng RL cũng có thể giúp tối ưu hóa quá trình quyết định và hành động của mô hình.

Hơn nữa, để thu thập dữ liệu khách hàng một cách tự động, có thể sử dụng AI để tự động gom dữ liệu từ các nguồn khác nhau và phân tích dữ liệu để đưa ra quyết định. Điều

này giúp giảm thiểu công sức và thời gian cho quá trình thu thập dữ liệu, đồng thời cải thiện độ chính xác và tính đồng nhất của dữ liệu.

TÀI LIỆU THAM KHẢO

Slide bài giảng LMS

Source of the dataset : UCI Machine Learning Repository: Bank Marketing Data Set

SVM : 1.4. Support Vector Machines — scikit-learn 1.2.2 documentation

KNN : sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.2.2 documentation

ANN : Artificial neural network - Wikipedia

https://arxiv.org/abs/2304.10191

https://www.researchgate.net/publication/360816402_Variational_Bayesian_dropout_with_a_Gaussian_prior_for_recurrent_neural_networks_application_in_rainfall-runoff_modeling

https://www.researchgate.net/publication/359045992_Analysis_of_Dropout_in_ANN_using_MNIST_Dataset

PHỤ LỤC

Link GitHub: https://github.com/sonbao0901/bank-marketing