Đề tài: Dự đoán kết quả bóng đá

Nhóm 11:

Các thành viên:

- Nguyễn Hữu Tiến
- Đào Đăng Tài
- Nguyễn Văn Quý
- Doãn Phúc Được

Mục lục

Phần 1: Giới thiệu bài toán:

Trong bài toán này, chúng ta xem xét việc dự đoán kết quả các trận đấu bóng đá dựa trên bộ dữ liệu thực tế.

Dữ liệu cung cấp cho chúng ta thông tin về các trận đấu đã diễn ra tại giải đấu Europe và tỷ lệ cược trên các sàn cá độ tài trợ như Bet365 và Blue Square. Mục tiêu của dự án này là xây dựng mô hình học máy để phân loại các trận đấu thắng và thua trong giải đấu để cung cấp thông tin hợp lý cho các nhà cái dựa trên các thuộc tính đã có sẵn trong bộ dữ liệu.

Các mục

1.Dataset Information

Tập dữ liệu này chứa thông tin về các đội bóng thuộc liên đoàn bóng đá Châu Âu và các trận đấu đã thi đấu cùng với tỉ lệ cược trên các sàn cá độ được diễn ra trong giải đấu Europe.

2.Data Features

Tập dữ liệu bao gồm 18 biến, bao gồm :

ID	ID của từng trận đấu
Country_name	Tên quốc gia của đội bóng tham gia thi đấu
League_name	Tên giải đấu
Season	Mùa giải thi đấu
Date	Ngày diễn ra trận đấu
Home_team	Đội nhà
Away_team	Đội khách
B365H,B365D,B365A	Tỷ lệ cá cược đội nhà thắng, hòa và đội khách thắng trên sàn 365bet
BSH,BSD,BSA	Tỷ lệ cá cược đội nhà thắng, hòa và đội khách thắng trên sàn Blue Square
Diff_goals	Hiệu số bàn thắng
Target	Mục tiêu

3. Data Preprocessing

- **Data cleaning**: Kiểm tra các giá trị còn thiếu và xử lý chúng một cách thích hợp. Đảm bảo tính nhất quán trong các kiểu dữ liệu.
- **Data Exploration:** Thực hiện phân tích dữ liệu khám phá (EDA) để hiểu sự phân bổ dữ liệu và mối quan hệ giữa các tính năng.
- **Feature Engineering:** Create new features if necessary and encode categorical variables (e.g., one-hot encoding for education and marriage).
- **Data Split:** Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra để đánh giá mô hình.

4. Model Building

- **Model Selection:** Chọn một thuật toán học máy thích hợp để phân loại nhị phân. Các lựa chọn phổ biến bao gồm Hồi quy logistic, Rừng ngẫu nhiên hoặc Tăng cường độ dốc.

- **Model Training:** Huấn luyện mô hình đã chọn trên dữ liệu huấn luyện.

- **Model Evaluation:** Evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, F1-score on the testing data

5.Conclusion

- Dự án phân loại các trận đấu bao gồm quá trình tiền xử lý dữ liệu, xây dựng mô hình và triển khai mô hình học máy để phân loại các trận đấu bóng đá.

- Dự án nhằm mục đích hỗ trợ cung cấp thông tin cho các nhà cái nhằm thao tác nhanh hơn và tổng hợp dữ liệu để đưa ra các phương án hợp lý cho các nhà cái.

Phần 2: Tiền xử lý dữ liệu

1.Xử lý dữ liệu thiếu

2.Mã hóa dữ liệu

3.Xử lý dữ liệu nhiễu

4.Xử lý giá trị hỗn hợp

5.Xử lý mất cân bằng dữ liệu

1. Xử lý dữ liệu thiếu

```
data.isnull().mean()
```

country_name	0.000000
league_name	0.000000
season	0.000000
stage	0.000000
date	0.000000
home_team	0.000000
away_team	0.000000
home_team_goal	0.000000
away_team_goal	0.000000
B365H	0.130375
B365D	0.130375
B365A	0.130375
BSH	0.454906
BSD	0.454906
BSA	0.454906
diff_goals	0.000000
target	0.000000

Tỷ lệ thông tin thiếu trong các cột dữ liệu

- B365H: 13%
- B365A: 13%
- B365D: 13%
- BSH: 45%
- BSA:45%
- BSD: 45%

Đối với các cột có B365, tiến hành lấy giá trị trung bình từ các mẫu dữ liệu khác có cùn
cột với nó.

```
mean_B365H = data.B365H.mean()
mean_B365D = data.B365D.mean()
mean_B365A = data.B365A.mean()
```

Đối với các cột còn lại, tiến hành như sau:

- Lấy giá trị trung bình các mẫu dữ liệu cùng cột.
- Lấy giá trị phương sai của các cột tương ứng. (Nhân với số 3).
- Tiến hành cộng 2 giá trị vừa tìm được.

```
eod_value_BSH = data.BSH.mean() + 3*data.BSH.std()
eod_value_BSD = data.BSD.mean() + 3*data.BSD.std()
eod_value_BSA = data.BSA.mean() + 3*data.BSA.std()
```

2. Mã hóa dữ liệu

Tiến hành sử dụng kĩ thuật Label Encoding mã hóa các cột các thông tin có dạng chuỗi chuyển về dạng chữ số.
Tiến hành mã hóa với cột country_name.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

le.fit(data['country_name'])data['country_name'] = le.transform(data['country_name'])
print(data['country_name'].unique())
```

country_name: có 10 nhãn

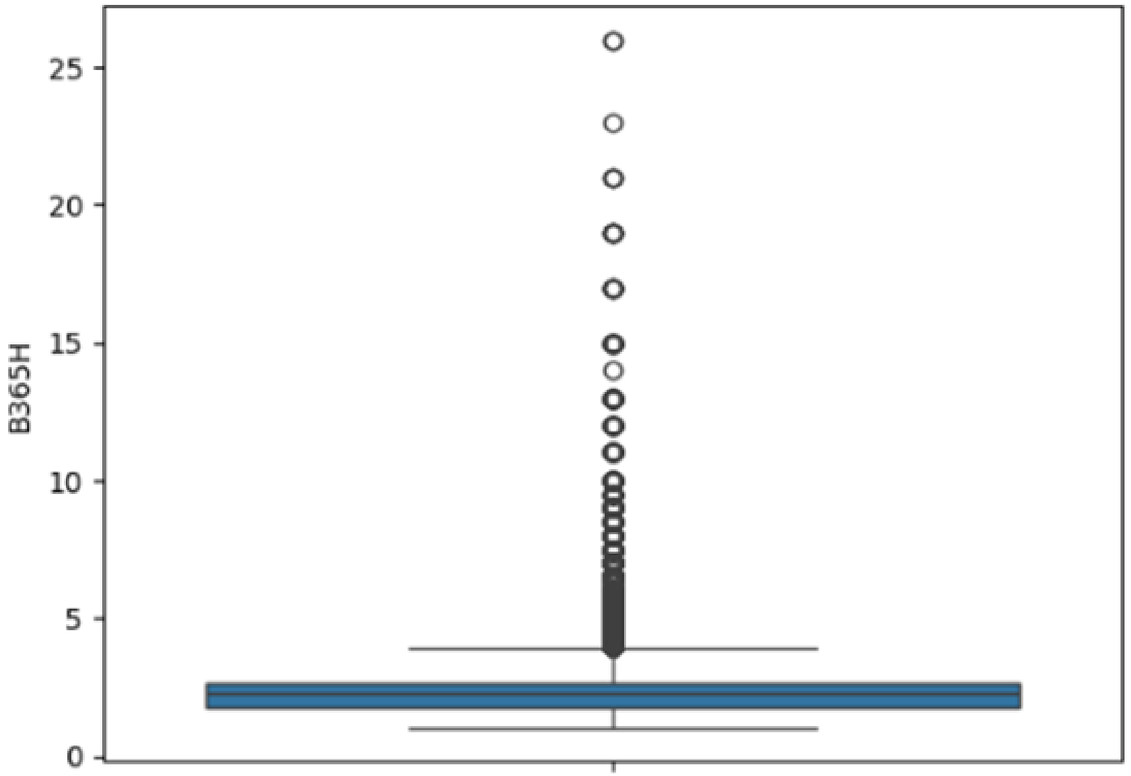
```
['Switzerland' 'Poland' 'France' 'Scotland' 'Germany' 'Belgium' 'England'
 'Portugal' 'Netherlands' 'Italy' 'Spain']
```

Kết quả

```
[10  6  2  8  3  0  1  7  5  4  9]
```

Làm tương tự với các cột league_name ,home_team và away_team.

3. Xử lý dữ liệu nhiễu



Biểu đồ boxplot của cột B365H, ta có thể thấy nhiều dữ liệu nhiễu có giá trị bất thường không tuân theo nguyên tắc dữ liệu, nếu chạy mô hình dẫn đến kết quả không được tốt.

Kỹ thuật IQR (Interquartile Range) là một phương pháp phổ biến được sử dụng để xử lý dữ liệu nhiễu trong thống kê và khoa học dữ liệu. IQR đo độ phân tán của dữ liệu bằng cách tính sự khác biệt giữa phân vị thứ 75 (Q3) và phân vị thứ 25 (Q1). Kỹ thuật này thường được áp dụng trong việc xác định và loại bỏ các điểm dữ liệu nhiễu từ tập dữ liệu.

Các bước cụ thể để sử dụng kỹ thuật IQR để xử lý dữ liệu nhiễu là: Xác định IQR: Tính IQR bằng cách lấy phân vị thứ 75 (Q3) trừ đi phân vị thứ 25 (Q1).

```
IQR1 = data["B365H"].quantile(0.75) - data["B365H"].quantile(0.25)
```

Xác định ngưỡng cắt: Xác định ngưỡng cắt cho dữ liệu nhiễu bằng cách nhân IQR với một hằng số cố định. Một cách phổ biến là sử dụng ngưỡng là k×IQR, trong đó k thường được đặt là 1.5 hoặc 3.

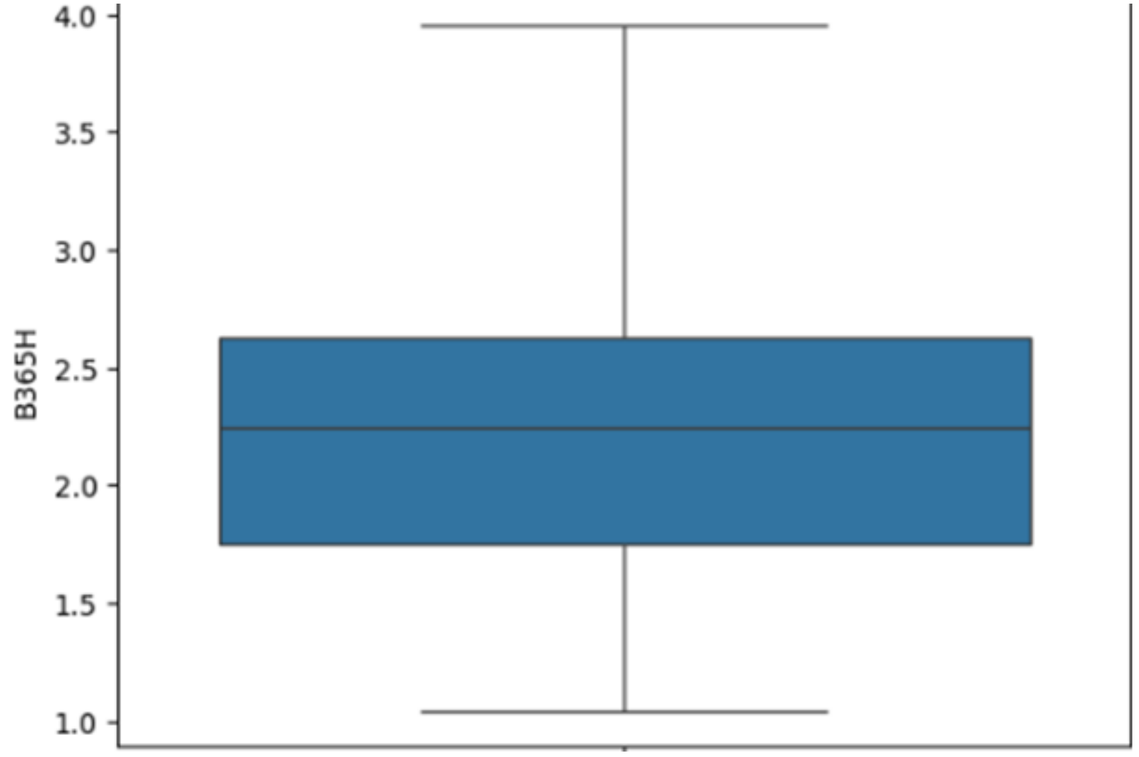
```
lower_B365A_limit = data["B365A"].quantile(0.25) - (IQR2 * 1.5)
upper_B365A_limit = data["B365A"].quantile(0.75) + (IQR2 * 1.5)
```

Loại bỏ dữ liệu nhiễu: Loại bỏ các điểm dữ liệu nằm ngoài phạm vi giới hạn được xác định bởi ngưỡng cắt.

Phân tích hoặc thay thế dữ liệu nhiễu: Sau khi loại bỏ dữ liệu nhiễu, bạn có thể tiến hành phân tích dữ liệu hoặc thực hiện các phương pháp khác như thay thế dữ liệu nhiễu bằng giá trị trung bình, trung vị hoặc dự đoán từ mô hình.

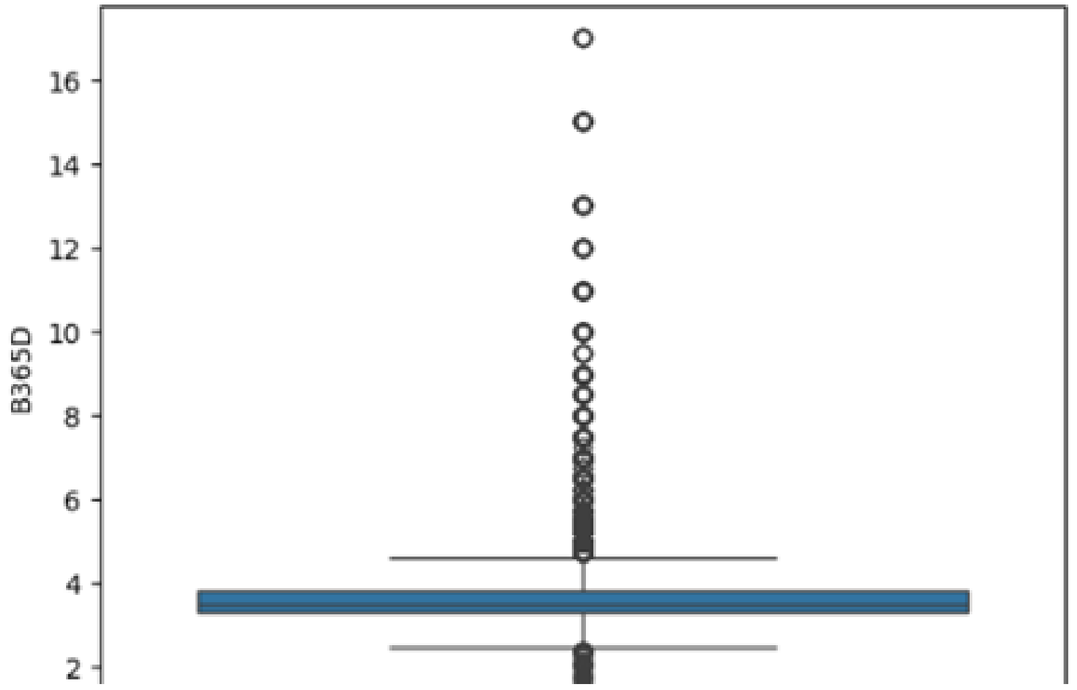
```
data["B365A"]= np.where(data["B365A"] > upper_B365A_limit,
upper_B365A_limit,np.where(data["B365A"] < lower_B365A_limit, lower_B365A_limit,
data["B365A"]))
```

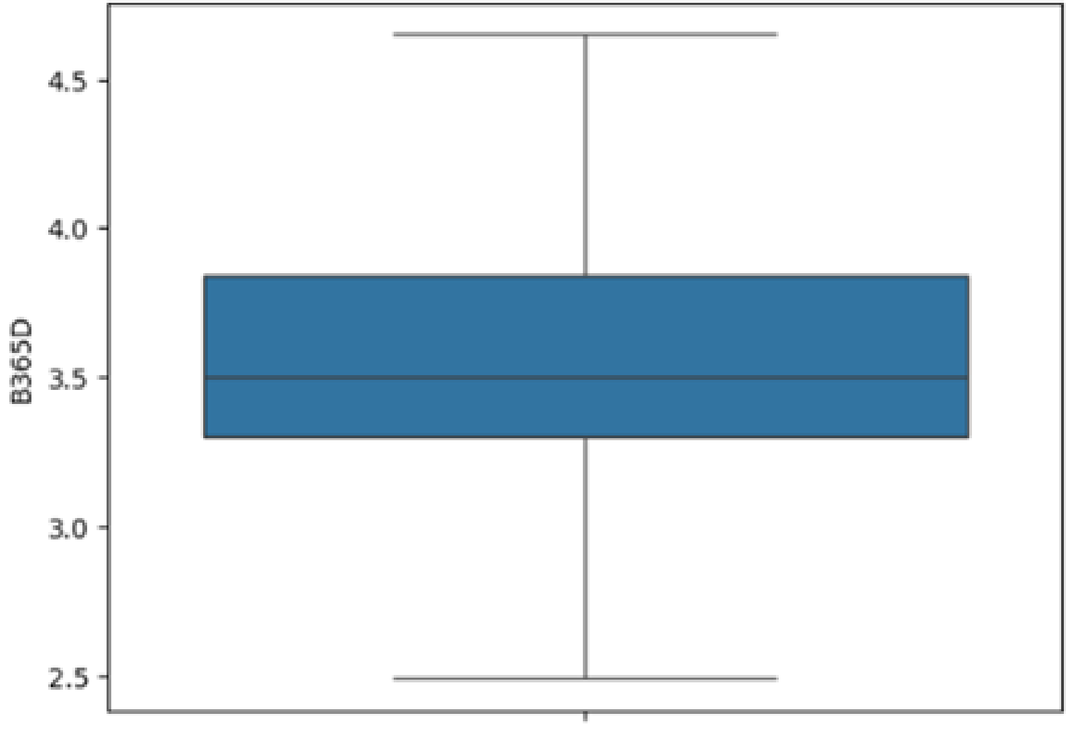
Kết quả



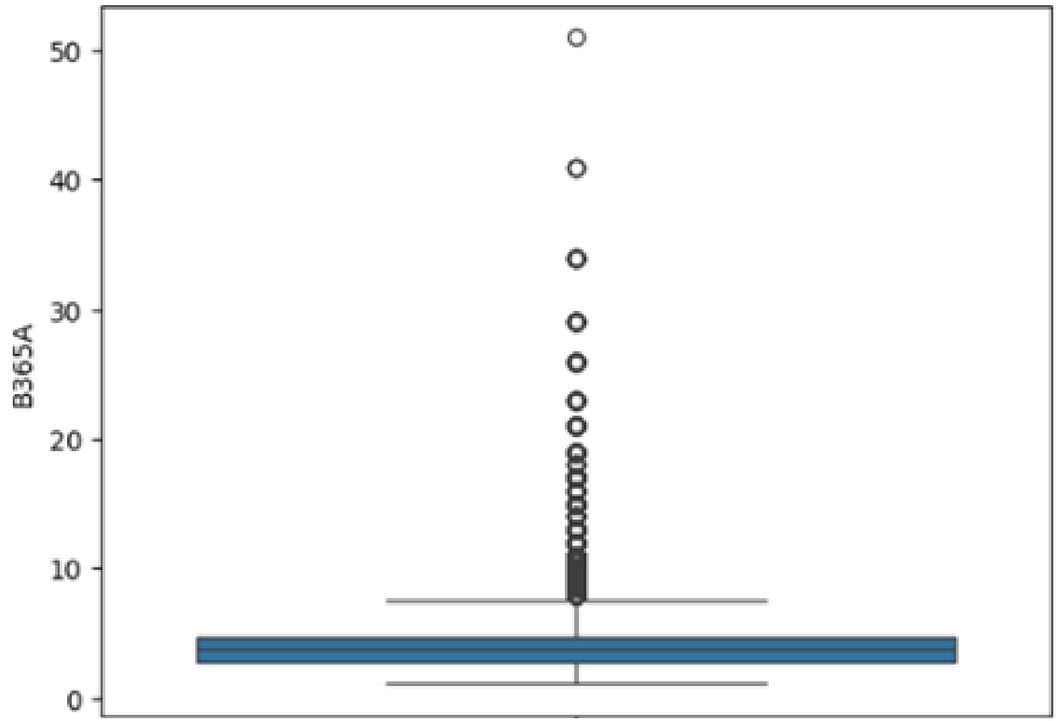
Làm tương tự với các cột còn lại.

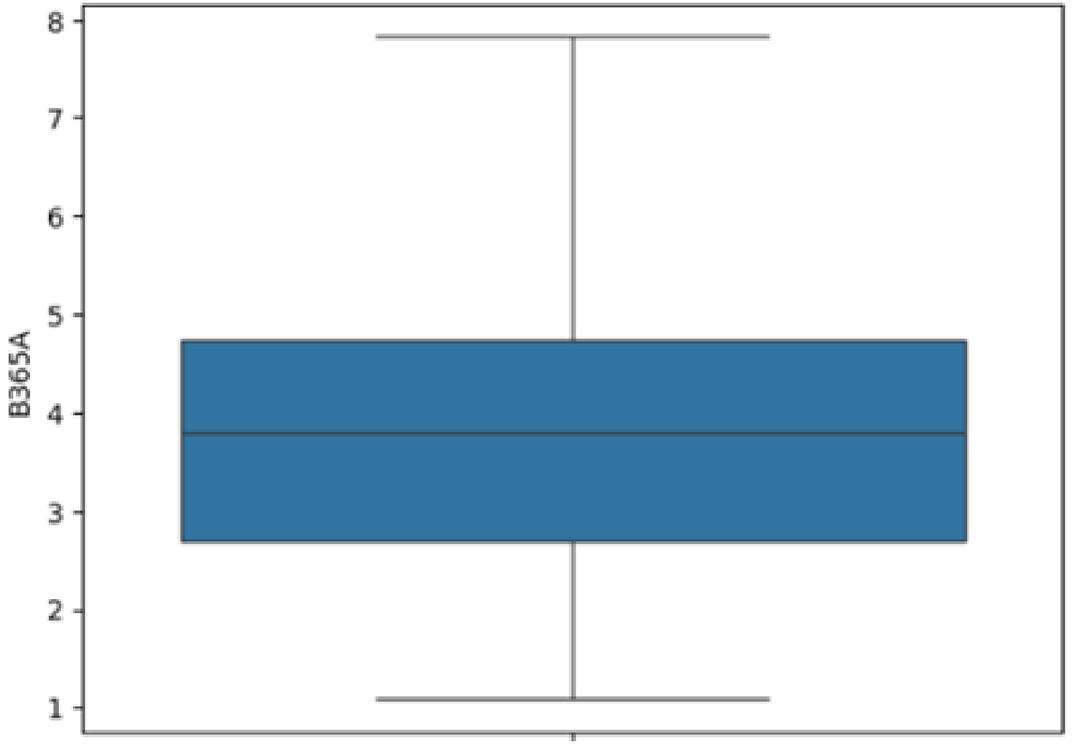
B365D



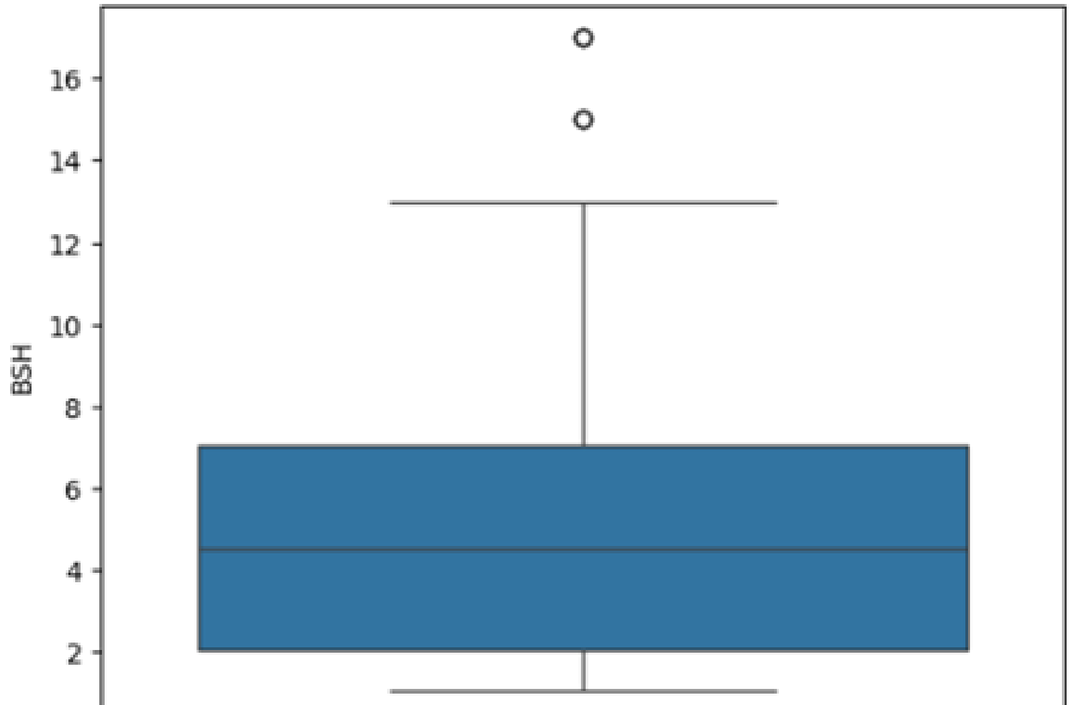


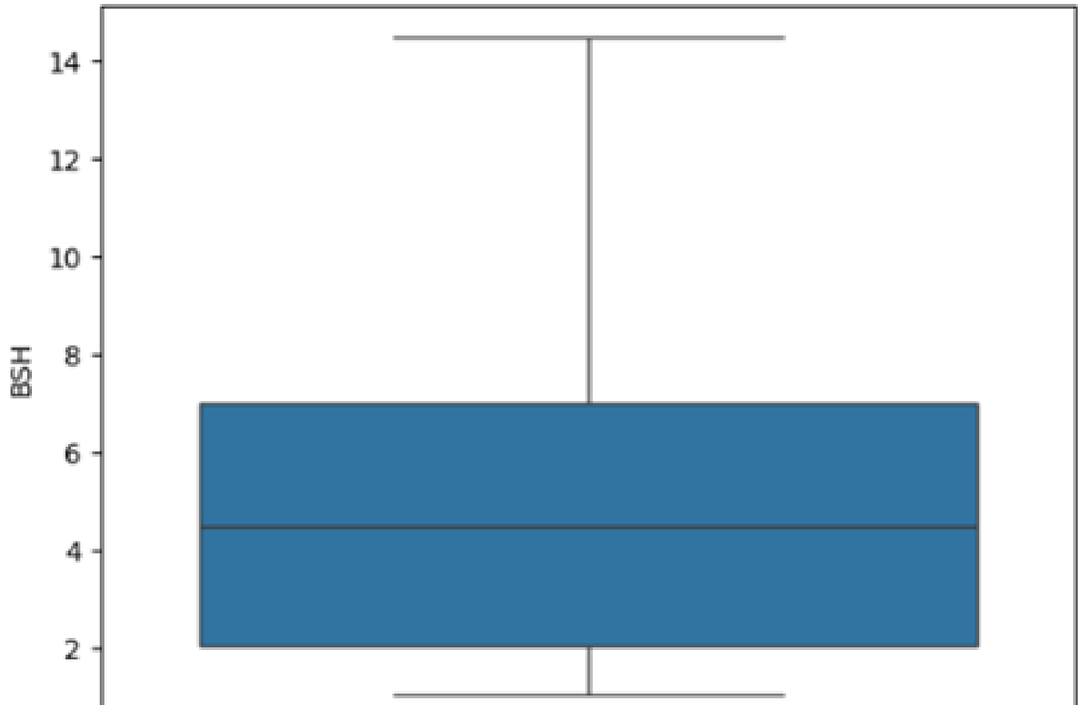
B365A


B365A
0
10
20
30
40
50

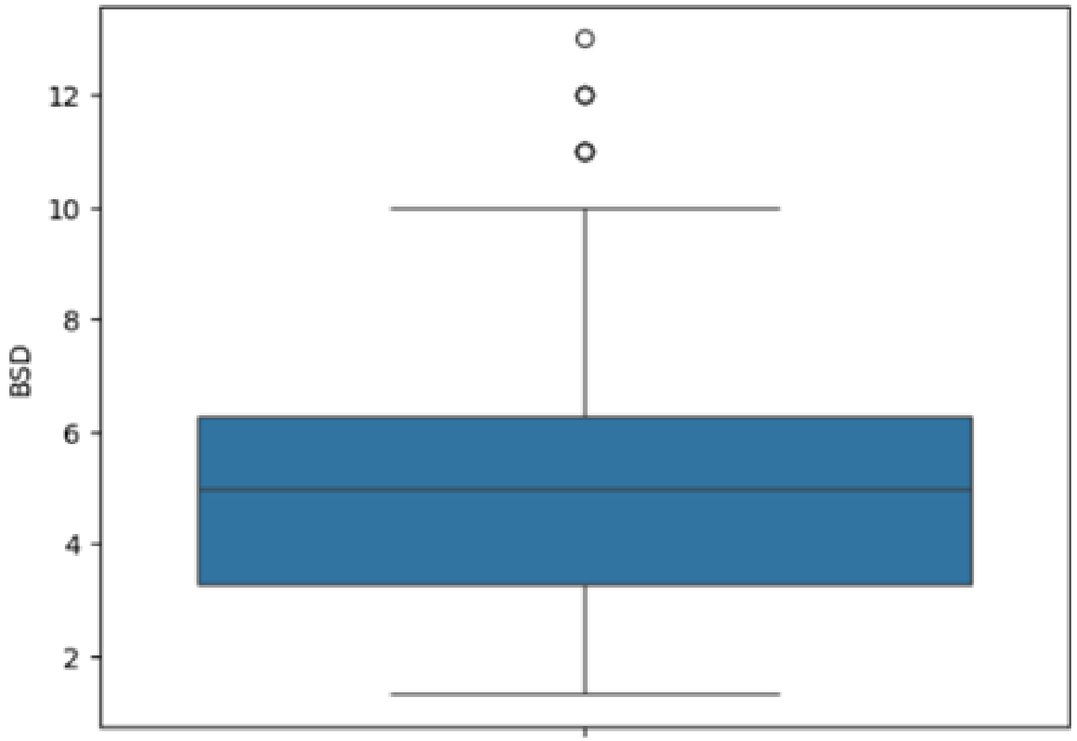

B365A
1
2
3
4
5
6
7
8

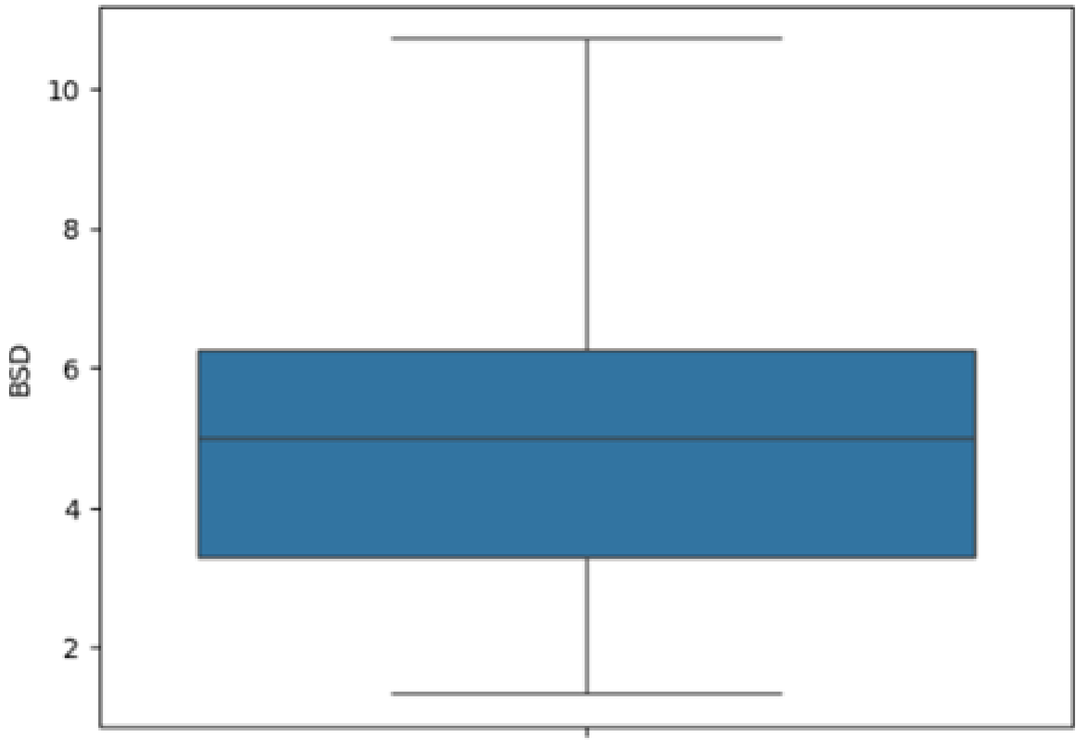
BSH


BSH
2
4
6
8
10
12
14
16

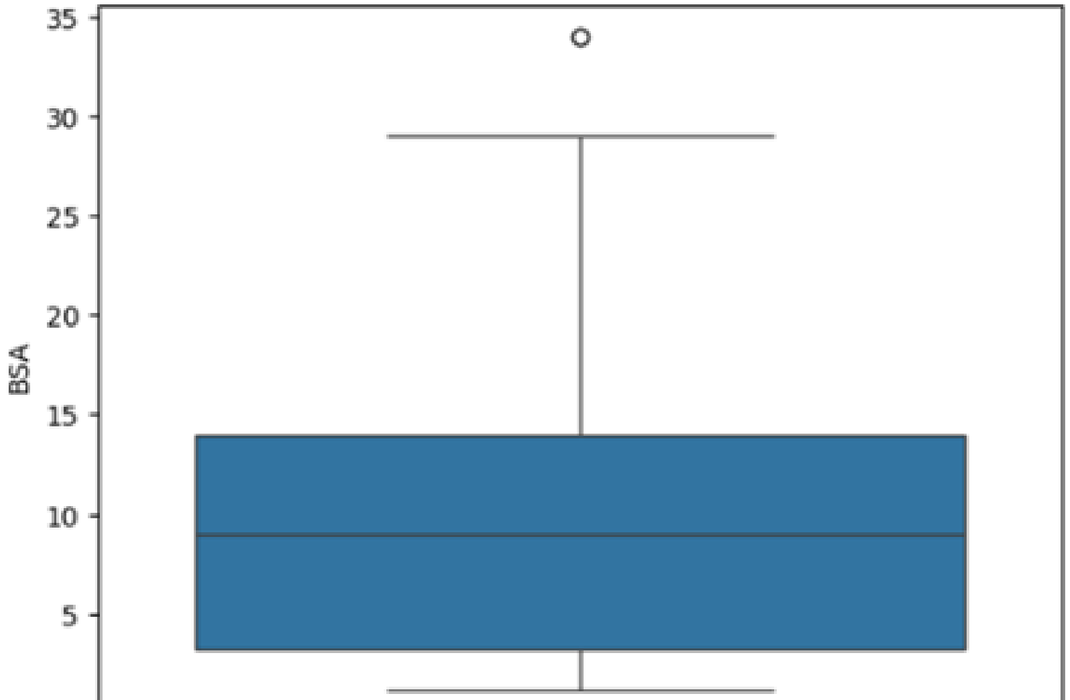

BSH
2
4
6
8
10
12
14

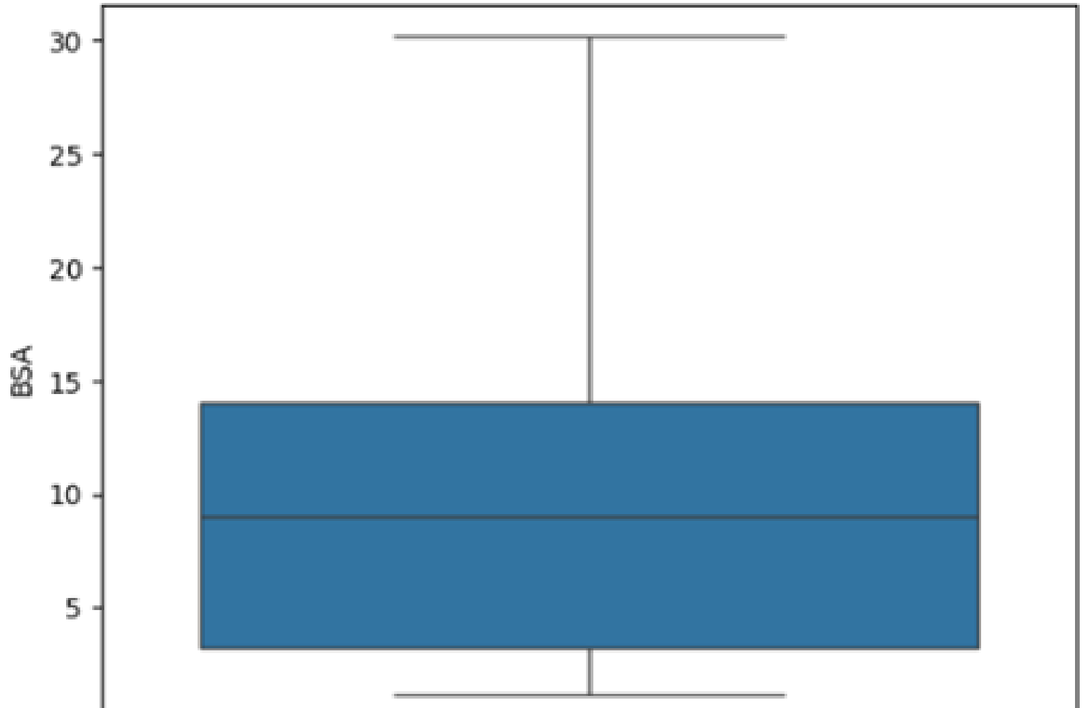
BSD


BSD
12
10
8
6
4
2


BSD
10
8
6
4
2

BSA


BSA
35
30
25
20
15
10
5


BSA
30
25
20
15
10
5

4. Xử lý giá trị hỗn hợp

Cột date có dữ liệu hỗn hợp giữa ngày, giờ và năm.

	country_name	league_name	season	stage	date
0	10	10	2008/2009	1	2008-07-18 00:00:00
1	10	10	2008/2009	1	2008-07-19 00:00:00
2	10	10	2008/2009	1	2008-07-20 00:00:00
3	10	10	2008/2009	1	2008-07-20 00:00:00
4	10	10	2008/2009	2	2008-

Trong các tập dữ liệu thực tế, có thể có trường hợp một cột chứa cả các giá trị số và các chuỗi văn bản. Điều này có thể gây ra khó khăn trong việc phân tích và xử lý dữ liệu, vì các phương pháp thống kê và học máy thường chỉ áp dụng cho một loại dữ liệu cụ thể.

Tiến hành phân tách thành các cột riêng biệt: mỗi cột đại diện cho một loại giá trị cụ thể (văn bản hoặc số). Sau đó, có thể áp dụng các phương pháp xử lý dữ liệu tương ứng cho mỗi loại giá trị.

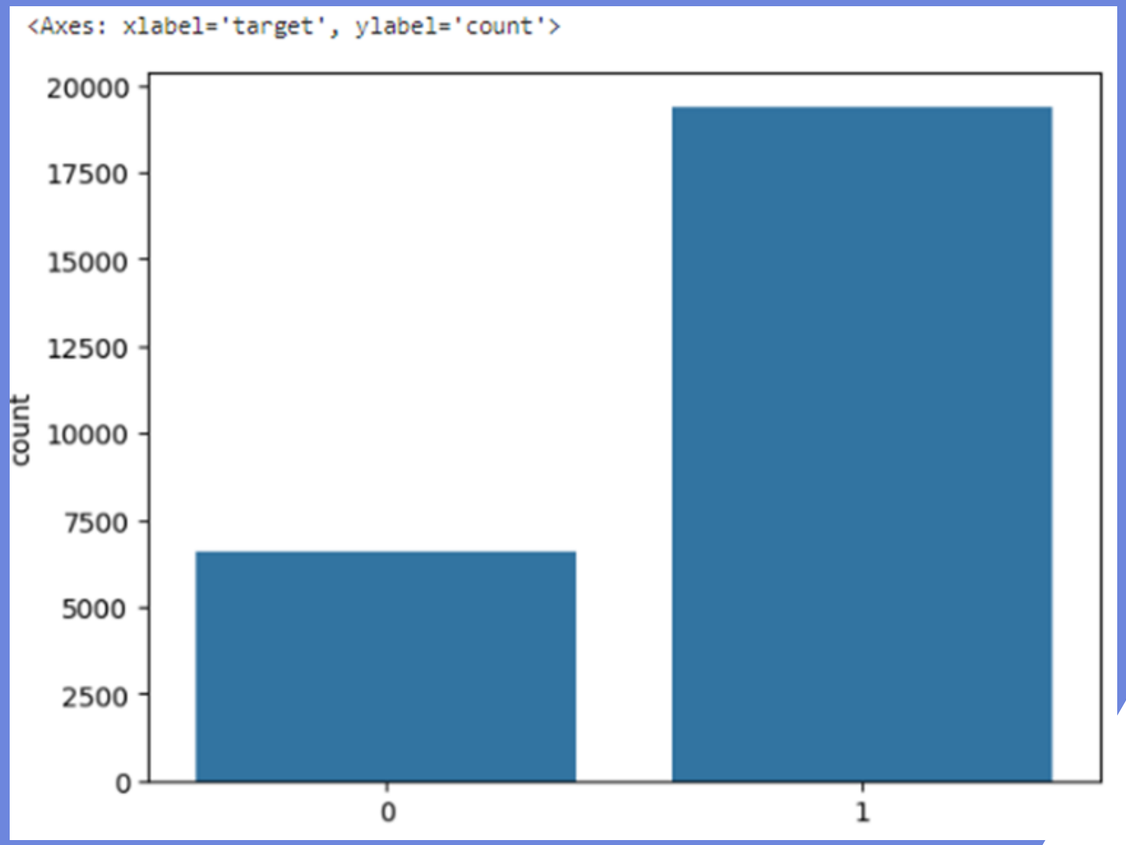
```
data['date'] = pd.to_datetime(data['date'])
data['hour'] = data['date'].dt.hour
data['min'] = data['date'].dt.minute

data['date'].dt.isocalendar()
data['month'] = data['date'].dt.month
data['day_month'] = data['date'].dt.day
data['year']= data['date'].dt.year
data.drop(columns=['date'], inplace=True)
```

Tiến hành chia thời gian hỗn hợp thành nhiều cột riêng biệt về giá trị.
Tương tự, chia thời gian của season thành 2 cột start_season và end_season.

Kết quả

hour	min	month	day_month	year	start	end	start_season	end_season
0	0	7	18	2008	2008	2009	2008	2009
0	0	7	19	2008	2008	2009	2008	2009
0	0	7	20	2008	2008	2009	2008	2009
0	0	7	20	2008	2008	2009	2008	2009
0	0	7	23	2008	2008	2009	2008	2009

5. Xử lý mất cân bằng dữ liệu



Cột target có số lượng không cân bằng giữa các nhãn.

Trong một số bài toán phân loại, có thể xảy ra trường hợp một nhóm lớp có số lượng quan sát ít hơn đáng kể so với các nhóm khác, điều này có thể làm cho mô hình học máy không hoạt động hiệu quả, vì mô hình sẽ có xu hướng học cách dự đoán nhãn của lớp đa số.

Dưới đây là một số phương pháp phổ biến để xử lý mất cân bằng dữ liệu:

Over-sampling (Tăng cường mẫu): Phương pháp này tạo ra thêm mẫu cho các lớp thiểu số bằng cách lặp lại các mẫu hiện có hoặc tạo ra các mẫu mới dựa trên dữ liệu hiện có.

Under-sampling (Giảm mẫu): Phương pháp này loại bỏ một số mẫu từ các lớp đa số để làm cho tỷ lệ giữa các lớp cân bằng hơn. Điều này có thể dẫn đến mất mát thông tin nếu không thực hiện cẩn thận.

Trong bài toán ta tiến hành chọn phương pháp over-sampling.

```
X = data.drop('target', axis=1)
y = data[['target']]
```

Lấy ra các cột thuộc tính và cột nhãn để tiến hành phương pháp.

```
# install imblearn using the following pip command
# pip install imbalanced-learn
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=2)X_us, y_us = sm.fit_resample(X, y)
```

SMOTE là một kỹ thuật phổ biến giúp tăng cường số lượng mẫu trong các lớp thiểu số bằng cách tạo ra các mẫu tổng hợp.

Cách SMOTE hoạt động như sau:

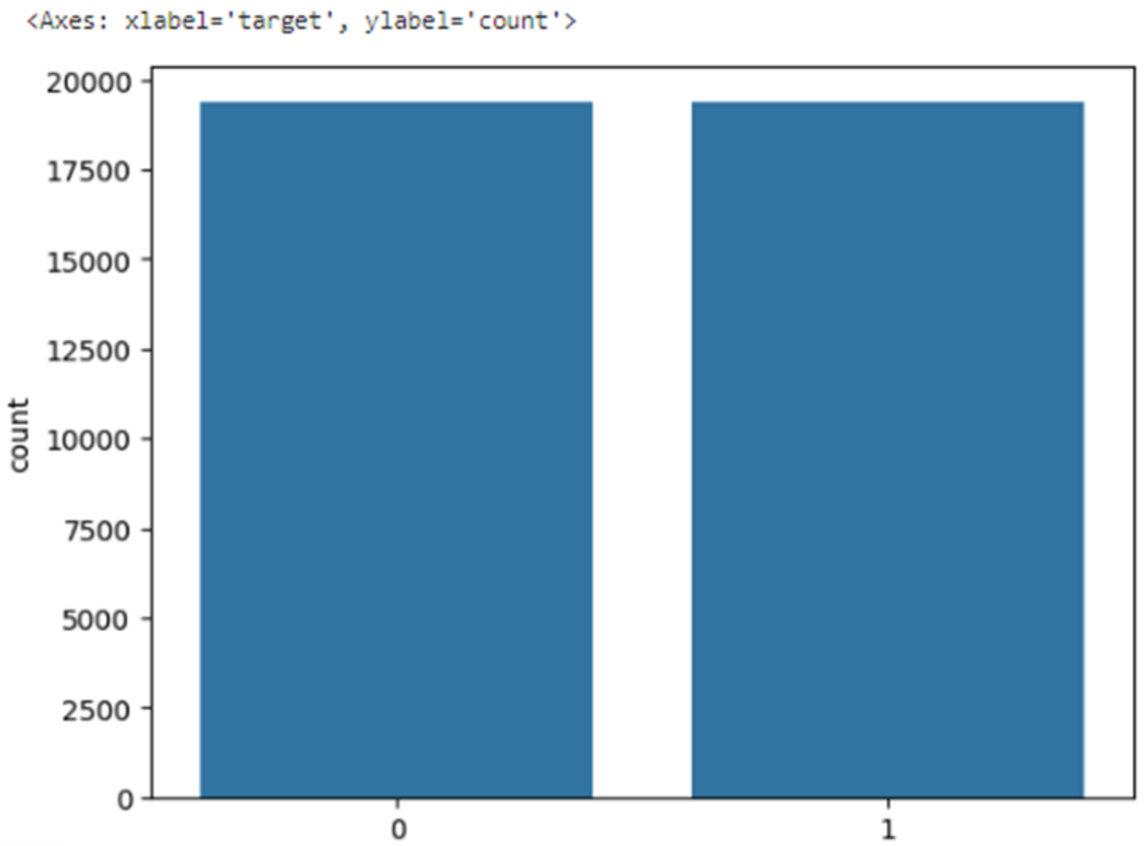
Chọn mẫu cơ sở: Đầu tiên, chúng ta chọn một mẫu từ lớp thiểu số (nhãn hiếm) trong tập dữ liệu.

Tìm hàng xóm gần kề: Sau đó, chúng ta sẽ tìm các hàng xóm gần kề của mẫu đã chọn từ lớp thiểu số, thường dùng k-nearest neighbors (k-NN).

Tạo mẫu tổng hợp: Cho mỗi mẫu cơ sở, chúng ta chọn một trong các hàng xóm gần kề và tạo ra một mẫu tổng hợp mới.

Thêm mẫu mới vào tập dữ liệu: Mẫu tổng hợp mới được thêm vào tập dữ liệu, làm tăng số lượng mẫu trong lớp thiểu số.

Kết quả:



Phần 3: Ứng dụng mô hình học máy

1. Chia dữ liệu

2. Huấn luyện mô hình

3.Kiểm tra độ đo chính xác của mô hình

Sau khi dữ liệu đã được tiền xử lý, ta có thể tiến hành xây dựng mô hình học máy cho bài toán.

- Chia dữ liệu thành tập train và test để huấn luyện mô hình.
- Chạy mô hình bằng tập train.
- Kiểm tra độ chính xác của mô hình bằng tập test và các độ đo như F1,recall,...

1. Chia dữ liệu

train_test_split(X_us, y_us, test_size=0.3, random_stare=42

Sử dụng hàm train_test_split chia tập dữ liệu thành 2 phần trong đó:

- Tập Ttrain chiếm 70%
- Tập Test chiếm 30%

Kết quả

```
print(X_train.shape)    (27136, 23)
print(X_test.shape)     (11630, 23)
print(y_train.shape)    (27136, 1)
print(y_test.shape)     (11630, 1)
```

2. Huấn luyệ mô hình

Chọn mô hình Logistic Regression.

```
model=LogicsticRegression(max_inter=1000)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
```

Hàm fit: huấn luyện mô hình theo tập train.
Hàm predict: đưa ra các giá trị dự đoán dựa trên tập test.

3. kiểm tra độ đo chính xác mô hình

```
accurary = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

Sử dụng 2 độ đo là accuracy và report.

Accuracy (Độ chính xác):
Accuracy là tỷ lệ phần trăm giữa số lượng dự đoán đúng và tổng số lượng mẫu trong tập dữ liệu kiểm tra.
Công thức tính: Accuracy = (Số lượng dự đoán đúng) / (Tổng số lượng mẫu)
Độ chính xác càng cao thì mô hình càng tốt. Tuy nhiên, độ chính xác không phản ánh được hiệu suất của mô hình trong trường hợp dữ liệu mất cân bằng hoặc khi có các lớp có tỷ lệ khác nhau.

Classification Report (Báo cáo phân loại):

Classification report là một báo cáo chi tiết về hiệu suất của mô hình trên các lớp (classes) khác nhau trong bài toán phân loại.

Bao gồm các thông số như precision, recall, F1-score và support cho mỗi lớp.

Precision (độ chính xác): Tỷ lệ giữa số lượng dự đoán đúng của lớp đó và tổng số lượng mẫu được dự đoán là lớp đó.

Recall (độ nhớ lại): Tỷ lệ giữa số lượng dự đoán đúng của lớp đó và tổng số lượng mẫu thuộc lớp đó trong dữ liệu thực tế.

F1-score: Trung bình điều hòa của precision và recall, cung cấp một phép đo tổng thể về hiệu suất của mô hình.

Support: Số lượng mẫu thực tế thuộc lớp đó.

Kết quả

Acuracy:0.76

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.67	0.74	5867
1	0.72	0.85	0.78	5763
accuracy			0.76	11630
macro avg	0.77	0.76	0.76	11630
weighted avg	0.77	0.76	0.76	11630

4.Tổng kết

Trong bài báo cáo này, chúng em đã thực hiện một loạt các bước tiền xử lý dữ liệu và chạy mô hình học máy để giải quyết bài toán. Chúng em đã đạt được một số kết quả quan trọng và nhận định sau đây:

Tiền xử lý dữ liệu:Chúng em đã thu thập và tiền xử lý một tập dữ liệu để chuẩn bị cho quá trình huấn luyện mô hình. Các phương pháp tiền xử lý đã được áp dụng để giải quyết các vấn đề ban đầu của dữ liệu.

Chạy mô hình học máy:Chúng em đã chọn và huấn luyện một số mô hình học máy trên tập dữ liệu đã tiền xử lý. Qua quá trình huấn luyện, mô hình cho kết quả tương đối ổn định trên các thang đo như recall, f1 score,...

Nhận xét và kết luận:Từ quá trình tiền xử lý dữ liệu và chạy mô hình học máy, chúng em nhận thấy như sau:

Ưu điểm:·
Các phương pháp tiền xử lý khá nhanh và hiệu quả.Có thể thay đổi nhiều phương pháp để linh hoạt tùy vào bài toán.

Hạn chế:
Có thể một số phương pháp gây ra mất thông tin đặc trưng về dữ liệu.Nhiều phương pháp có thể tốn nhiều thời gian và không hiệu quả khi dữ liệu không phù hợp.

Chúng em xin có 1 vài ý tưởng:

- Nghiên cứu và thử nghiệm các phương pháp tiền xử lý dữ liệu mới để giảm thiểu mất mát thông tin và tăng cường khả năng tổng quát hóa của mô hình.
- Áp dụng các kỹ thuật tiền xử lý dữ liệu tiên tiến như feature engineering để tạo ra các đặc trưng mới có thể cải thiện hiệu suất của mô hình.

Nhìn chung, quá trình này đã cung cấp cái nhìn sâu rộng về quá trình tiền xử lý dữ liệu và chạy mô hình học máy trong việc giải quyết bài toán thực tế. Những kết quả và nhận định từ bài toán này có thể hữu ích cho các nghiên cứu và ứng dụng trong tương lai.

Nguồn tài liệu tham khảo:

- Lý thuyết:
- Slide bài giảng của TS. Tạ Quang Chiểu, đại học Thủy Lợi.
- Machinelearningcoban.com
- Book data Preprocessing Python master

Dữ liệu: https://wru-my.sharepoint.com/:x:/g/personal/quangchieu_ta_tlu_edu_vn/ESxsTzDJKNdOux-CsWz8lTABao5bPSp3_nABKDOtW7Z_Mw?e=ezrltN