«به نام خدا»

تکلیف سوم – سوال چهارم – مرضیه علیدادی – 9631983

(کد های مربوط، در دو فرمت py. و ipynb. ضمیمه شده اند. – دیتاست های thyroid_train.csv و thyroid_test.csv نیز ضمیمه شده اند.)

4.

a) سایز دیتاست های حاصل، به این صورت خواهد بود:

```
print(df.shape)
print(df_train.shape)
print(df_test.shape)
```

```
(185, 6)
(148, 6)
(37, 6)
```

دو دیتاست حاصل مربوط به train و test ، ضمیمه شده اند.

b) پارامتر stratify ، نسبتِ target را به همان شکلی که در مجموعه داده های اصلی است، در مجموعه داده های train و test نیز حفظ خواهد کرد.

برای مثال فرض می کنیم target دارای مقادیر 0 و 1 و 2 است، که نسبت آن ها در مجموعه داده های اصلی به ترتیب، 40 و 30 و 30 است. حال وقتی این نسخه اصلی را با استفاده از ()train_test_split و با داشتن پارامتر stratify تقسیم می کنیم، در داخل هر کدام از مجموعه داده های test و train هم، این نسبت 40 و 30 و 30 برای مقادیر 0 و 1 و 2 برای target حفظ می شود.

از این پارامتر استفاده می شود؛ برای اینکه ما غالباً می خواهیم نسبت داده ها را برای پیش بینی بهتر و قابلیت تولید مجدد نتایج حفظ کنیم.

c)

- در دیتاست اصلی، توزیع داده ها به این شکل است:

```
df['Outcome'].value_counts()
```

```
1.0    144
2.0     30
Name: Outcome, dtype: int64
```

یعنی تقریبا 17% داده ها برابر 2.0 است، و 83% داده ها برابر 1.0 است.

- در دیتاست مربوط به train ، توزیع داده ها به این شکل است:

```
df_train['Outcome'].value_counts()
```

```
1.0    115
2.0     25
Name: Outcome, dtype: int64
```

یعنی تقریبا 18% داده ها برابر 2.0 است، و 82% داده ها برابر 1.0 است.

- در دیتاست مربوط به test ، توزیع داده ها به این شکل است:

```
df_test['Outcome'].value_counts()
```

```
1.0    29
2.0     5
Name: Outcome, dtype: int64
```

یعنی تقریبا 15% داده ها برابر 2.0 است، و 85% داده ها برابر 1.0 است.

همانطور که مشخص است، توزیع این داده در این 3 دیتاست به هم نزدیک است؛ ولی همانطور که بالا مشخص شده، با هم متفاوت است.

d)

1. Resampling the training set: دو رویکرد از این دسته برای متعادل کردن دیتاست های imbalanced وجود دارد:

1.1. Under-sampling: در این روش، تمرکز روی کلاسی از متغیر است که دارای فراوانی زیاد است. با کاهش این کلاس، دیتاست را بالانس می کنیم. این متد زمانی استفاده می شود، که کمیت دیتای مورد نظر، کافی است. با حفظ سمپل های کلاسِ با کمیت پایین، و انتخاب تصادفی به همان تعداد، از کلاسِ با کمیت زیاد، دیتاستی بالانس شده را حاصل خواهیم شد.

1.2. Over-sampling: در این روش، تمرکز روی کلاسی از متغیر است که دارای فراوانی کم است. با افزایش تعداد مقادیر این کلاس، دیتاست را بالانس می کنیم. این متد زمانی استفاده می شود، که کمیت دیتای مورد نظر، کافی نیست. با حفظ سمپل های کلاسِ با کمیت بالا، به تعداد مقادیر مربوط به کلاسِ با کمیت پایین می افزاییم، تا دیتاستی بالانس شده حاصل شود.

2. Cluster the abundant class: در این روش، به جای تکیه بر نمونه های تصادفی برای حفظ گوناگونی داده های موجود در دیتاست training، روش دیگری پیشنهاد می شود:
داده های مربوط به کلاسِ با فراوانی بالا را در نظر می گیریم. آن ها را در r گروه (r همان تعداد دلخواه و مدنظری ست که می خواهیم سمپل از کلاسِ با فراوانی بالا داشته باشیم)دسته بندی می کنیم. سپس، برای هر دسته، فقط مرکز آن دسته را در نظر می گیریم و بقیه را حذف می کنیم. درنهایت، مدل را با استفاده از داده های مربوط به کلاسِ با فراوانی پایین و این مرکز دسته های کلاسِ با فراوانی بالا، train می کنیم.