Report
2024140927 김우진

1. Iris 데이터 셋을 활용해 클래스 별 변수 평균 차이 검정

1-1. Iris 데이터셋 불러오기, 구조 산출

```
import seaborn as sns
iris=sns.load_dataset('iris')
print(iris.head())
```

[24]

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

데이터셋은 sepal_length, sepal_width, petal_length, petal_width, 그리고 species 로 이루어져 있음을 알 수 있다.

```
print(iris.info())
```

[25] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

데이터의 구조는 다음과 같다.

1-2. 기술 통계량 산출

1-2 기술 통계량 산출

```
print(iris.describe())  # 기술 통계량 출력력
```

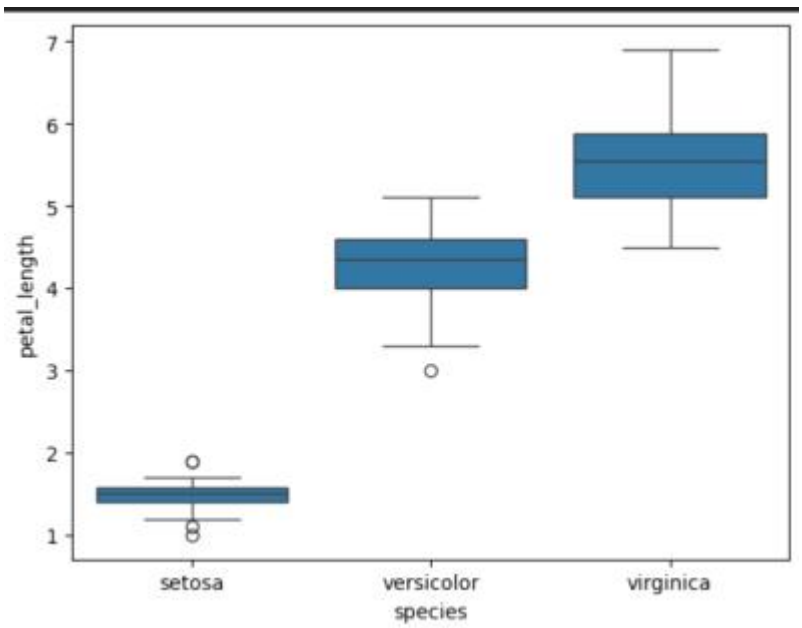
[26] Python

```
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

총 150개의 데이터가 있음을 알 수 있다. 평균(mean), 표준편차(std), 사분위수(25%,50%,75%)는 다음과 같다.

1-3. 시각화

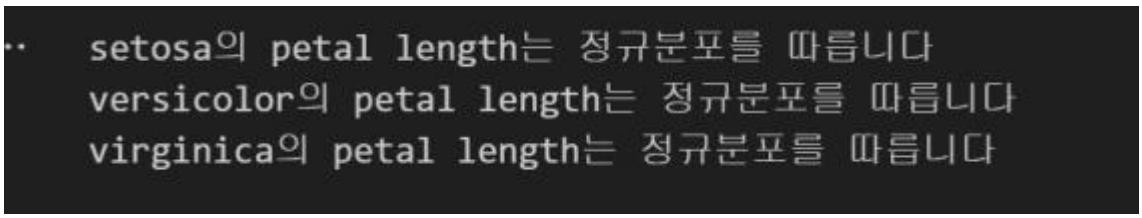
x축을 종, y축을 petal_length로 하는 boxplot은 다음과 같다.



이를 통해 평균 petal_length가 setosa가 가장 작고, versicolor, virginica 순서로 커진다는 것을 알 수 있다. 중앙값, 사분위수 분포도 마찬가지이다.

4. 정규성 검정
각 species 별로 Shapiro-Wilk 검정을 실시한다.
(유의수준=0.05)
귀무가설 H0: species x의 petal_length는 정규분포를 따른다. (p>0.05)
대립가설 H1: species x의 petal_length는 정규분포를 따르지 않는다. (p<=0.05)

검정 결과



setosa, versicolor virginica의 petal length 모두 정규분포를 따른다는 것을 알 수 있따.

5. 등분산성 검정

Levene 검정을 수행한다. (유의수준 0.05)

H0: 집단의 분산이 같다.
H1: 집단의 분산이 다르다.

검정결과, H0을 기각하고 H1을 채택한다. 따라서 세 집합의 분산은 다름을 알 수 있다.

1-6. ANOVA 검정
H0: 모든 그룹의 평균이 동일하다.
H1: 적어도 한 그룹의 평균이 다르다.

ANOVA 검정 결과, 귀무가설을 기각할 수 없다. 따라서, setosa, versicolor, virginica 중 적어도 한 그룹의 평균이 유의미하게 다르다.

1-7. tukey HSD 검정

적어도 한 그룹의 평균이 유의미하게 다르므로, 서로 어떤 그룹의 평균들이 다른지 검정해봐야 한다.

Tukey HSD 검정을 활용한다.

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
setosa	versicolor	2.798	0.0	2.5942	3.0018	True
setosa	virginica	4.09	0.0	3.8862	4.2938	True
versicolor	virginica	1.292	0.0	1.0882	1.4958	True

검정 결과, 세 그룹의 평균이 유의미하게 쌍마다 다름을 알 수 있다.

앞의 검정들의 결과를 바탕으로, 세 그룹의 petal_length의 평균이 유의미 하게 다르고, 그 평균은 setosa, versicolor, virginica 순서대로 커짐을 알 수 있다.

결론: petal_length는 virginica가 제일 크고, 그다음으로 versiclor, setosa 순으로 작아짐을 알 수 있다.

2. 실제 신용카드 사기 데이터셋을 활용해 클래스 불균형 상황에서 분류 모델을 학습

2-1. 데이터 로드 및 기본 탐색

```
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
...
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
```

총 30개의 columns 로 구성된 데이터 셋임을 알 수 있다.

```
Class
0    99.827251
1     0.172749
Name: proportion, dtype: float64
```

클래스 비율상으로 0이 99.83%, 1이 0.17%를 차지하는 클래스 불균형 데이터임을 알 수 있다.

2-2.샘플링

```
2-2. 샘플링

    df_fraud=df[df["Class"]==1]
    df_normal=df[df["Class"]==0].sample(n=10000, random_state=42)
    df_sampled=pd.concat([df_fraud, df_normal],axis=0) # axis:0 의미: 행 방향 새로(아래로 붙임)
    print(df_sampled["Class"].value_counts(normalize=True)*100)

Class
0    95.310713
1     4.689287
Name: proportion, dtype: float64
```

사기 거래는 모두 유지하고, 정상 거래만 10000개 샘플링을 진행하였다.
결과 표본에 정상 거래 95.3%, 사기 거래 4.7%로 구성된 표본을 얻을 수 있었다.

2-3. 데이터 전처리

```
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
df["Amount"]=scaler.fit_transform(df[["Amount"]]) #입력 2차원 요구
X= df.drop("Class", axis=1) #독립변수

y=df["Class"] #종속변수
```

X에는 종속변수 class를 제외한 모든 column들이 포함되었다.
y에는 종속변수 class만 포함되었다.
이를 통해 종속변수와 독립변수들을 분리할 수 있다.
또한 Amount 변수에 대해 표준화를 진행하였다.

2-4. 학습 데이터와 테스트 데이터 분할

```
2-4. 학습 데이터와 테스트 데이터 분할

    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.2, random_state=42, stratify=y)


    print(y_train.value_counts(normalize=True)*100)
    print(y_test.value_counts(normalize=True)*100)

Class
0    99.827075
1     0.172925
Name: proportion, dtype: float64
Class
0    99.827955
1     0.172045
Name: proportion, dtype: float64
```

sklearn의 train_test_split 함수를 통해 학습셋과 테스트셋의 비율을 8:2로 분할하였다.

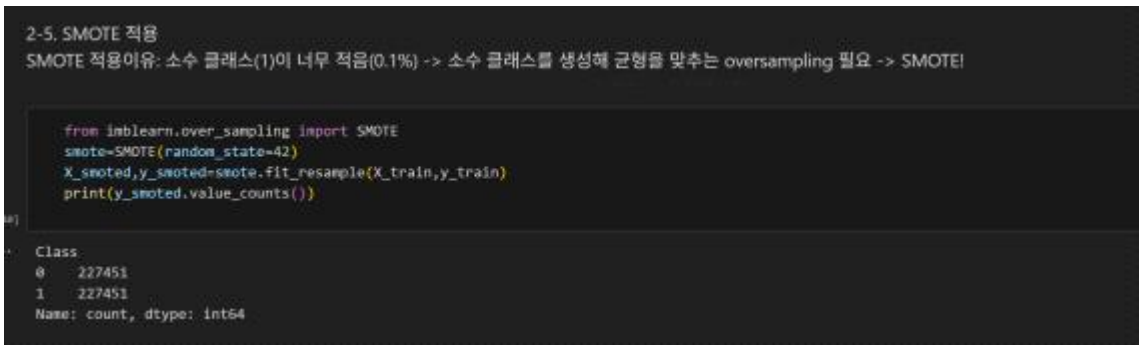
그러나 클래스 비율로 보아 클래스 불균형 상태에 있음을 알 수 있다.

2-5.SMOTE 적용

클래스 불균형을 해결하기 위해 oversampling 또는 undersampling이 필요하다.

이번엔 oversampling을 해보자.

oversampling의 기법 중 하나인 SMOTE를 활용한다.



SMOTE 적용 후 0, 1의 수가 동일해짐을 확인할 수 있다.

2-6. 모델 학습

분류 모델 중 하나인 로지스틱 회귀(logistic regression)을 활용한다.

Precision, recall, f1-score, pr-auc는 다음과 같다.

	precision	recall	f1-score	support
0	0.9994	0.9998	0.9996	56864
1	0.8289	0.6429	0.7241	98
accuracy			0.9992	56962
macro avg	0.9142	0.8213	0.8619	56962
weighted avg	0.9991	0.9992	0.9991	56962

pr_auc: 0.7439799579262683

2-7. 최종 학습 결과

class 1에 대하여, recall, f1, pr-auc 모두 목표치를 충족시키지 못했다.

추가로 시도할 방법으로는 로지스틱 회귀의 임계값(threshold)을 조정하거나, 로지스틱 회귀가 아닌 다른 분류 모델을 활용하는 것을 방법으로 생각해 볼 수 있다. 또는, 하이퍼파라미터 값을 조정할 수 도 있다.

또한, 앙상블 기법을 이용해 다른 모델들과 종합해 분류 모델을 만든다면 더 좋은 성능을 기대할 수 있을 것이다.