기초통계 및 ML 과제 – 이재열

1. Iris 데이터셋을 활용해 클래스별 변수 평균 차이를 검정

1. 먼저

```
   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

     sepal_length  sepal_width  petal_length  petal_width    species
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

<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
```

Iris 데이터셋을 "iris" 라는 이름으로 불러오고, .head(), .tail(), .info() 를 각각 출력한 결과. Dtype이라고 되어 있는 자료형에 맞게 데이터가 제대로 들어가 있는 것까지 확인할 수 있습니다.

2. 기술통계량 산출

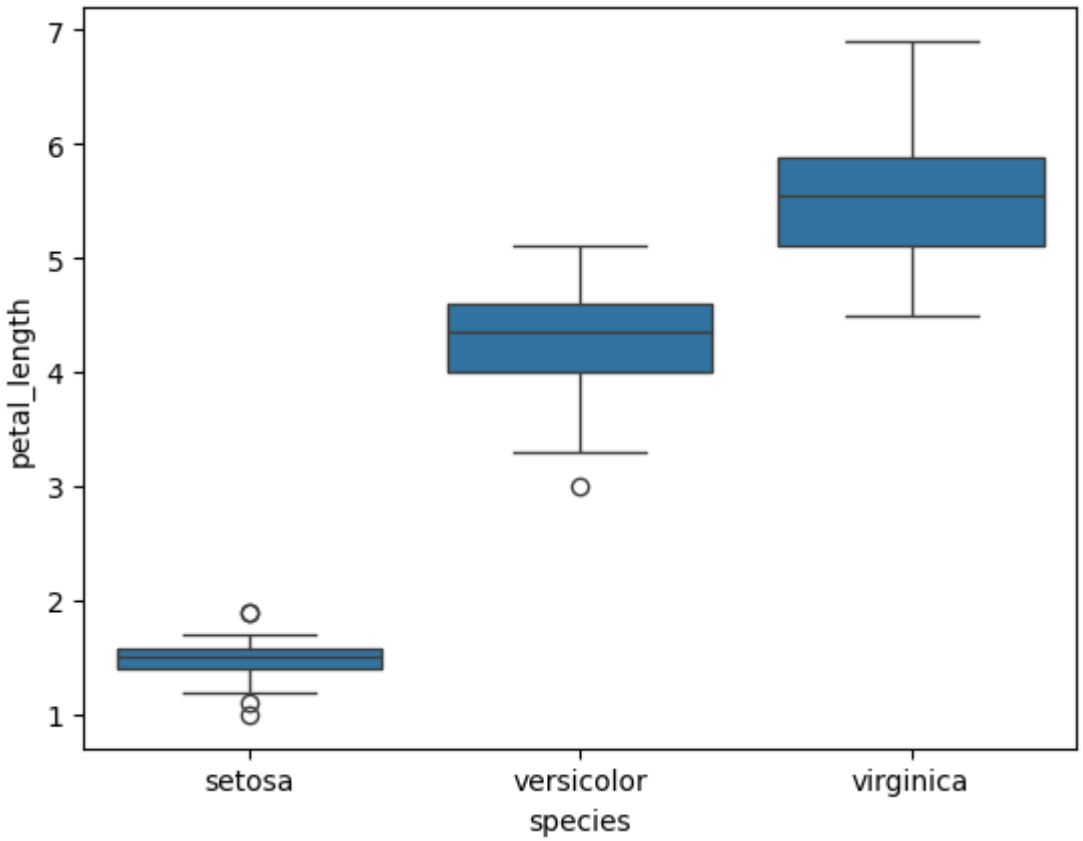
다음 페이지에 나와 있는 사진처럼, 최댓값, 최솟값, 평균, 표준편차, 그리고 사분위수를 각각 출력했습니다. Pandas를 쓰는 건 이번이 처음인데 SQL과 닮은 점이 여러모로 있는 것 같아요.

```
Maximum
species
setosa        1.9
versicolor    5.1
virginica     6.9
Name: petal_length, dtype: float64

Minimum
species
setosa        1.0
versicolor    3.0
virginica     4.5
Name: petal_length, dtype: float64

Mean
species
setosa        1.462
versicolor    4.260
virginica     5.552
Name: petal_length, dtype: float64

Sigma
species
setosa        0.173664
versicolor    0.469911
virginica     0.551895
Name: petal_length, dtype: float64

Quartile
              Q1    Q2     Q3
species
setosa       1.4  1.50  1.575
versicolor   4.0  4.35  4.600
virginica    5.1  5.55  5.875
```

3. 시각화

matplotlib.pyplot의 Boxplot 기능을 이용해 시각화한 결과입니다.

한눈에 봐도 setosa, versicolor, virginica 로 갈수록 petal_length 값이 커지는 것을 알 수 있습니다.



4. 정규성 검정

Shapiro-Wilk 검정은 데이터가 주어졌을 때 그 데이터가 정규분포를 따르는지 안 따르는지를 확인하는 검정 방법입니다.

우선 귀무가설은 "데이터가 정규분포를 따른다" 로 설정하고, 이에 따라 대립가설은 "데이터가 정규분포를 따르지 않는다" 로 설정합니다. 그 후 scipy를 이용해 검정을 실시합니다.

```
p-value of setosa: 0.054811
p-value of versicolor: 0.158478
p-value of virginica: 0.109775
```

모든 species에 대해 p-value가 0.05를 넘는 모습을 볼 수 있었습니다. 이는 귀무가설을 채택해야 함을, 즉 데이터가 정규분포를 어느 정도 따름을 의미합니다.

5. 등분산성 검정

정규성 검정은 완료되었고, 등분산성 검정을 시행합니다. Levene 검정 또한 scipy를 이용해 할 수 있습니다.

귀무가설은 "데이터가 등분산성을 만족한다", 이에 따라 대립가설은 "데이터가 등분산성을 만족하지 않는다" 로 설정합니다. 그 후 scipy를 이용해 검정을 실시합니다.

```
p-value == 0.000000
```

P == 0이 나온 것으로 보아, 이번에는 귀무가설을 기각해야 함을, 즉 데이터가 등분산성을 만족하지 않음을 의미합니다.

6. 가설 수립

다음에 ANOVA 검정을 할 차례이고 ANOVA 검정은 평균을 비교하는 검정 방법이므로, 평균과 관련된 가설을 수립해야 합니다.

H0: 3개의 species에 대해 petal_length의 평균이 모두 같다.

H1: 3개 중 적어도 하나는 petal_length의 평균이 다르다.

이렇게 가설을 수립하고 ANOVA 검정을 진행합니다.

7. ANOVA 실행

One-way ANOVA를 하려면 등분산성이 있어야 한다고 알고 있지만, 우선 이후 분석은 등분산성을 만족한다고 가정하고 한다고 5번 항목에서 명시해 두었기 때문에 등분산성을 만족한다고 치고 ANOVA를 실행합니다.

```
p-value == 0.000000
```

P가 이번에도 0이 나왔습니다. 귀무가설을 기각하고, 적어도 하나는 petal_length가 다르다고 결론지을 수 있겠습니다.

이 결과는 3번 (시각화) 항목에서도 확인할 수 있는데, box plot에서도 분명하게 평균이 차이나는 것을 볼 수 있습니다.

8. 사후검정 (Tukey HSD)

Tukey HSD를 이용해 사후검정을 실행합니다. statsmodels 이라는 라이브러리를 불러오면 파이썬으로 쉽게 할 수 있습니다.

```
   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
-----------------------------------------------------
```

모든 세 쌍에 대해 reject가 True로 나왔습니다. 이 말은 어느 쌍을 골라도 유의미하게 평균이 차이난다는 말이 됩니다.

9. 결과 요약

이제 Boxplot, ANOVA, 사후검정 결과를 조합해 petal_length 값에 대한 결과를 요약합니다.

Boxplot에서 setosa, versicolor, verginica로 갈수록 petal_length 값이 전반적으로 커짐을 알 수 있었습니다.

등분산성을 만족하지는 않는 것으로 드러났지만 그래도 ANOVA를 돌렸을 때 p==0이 나온 것을 보면 통계적인 방법을 사용했을 때도 평균이 최소한 모두 같지는 않다는 결론을 얻을 수 있었습니다. 마지막으로 Tukey HSD 사후검정을 돌렸을 때도 모든 쌍에 대해 평균이 많이 차이난다는 결론을 얻을 수 있었습니다. 세 가지 방법 모두에서 저는 세 species간에 petal_length가 유의미한 차이가 난다는 결론으로 마무리지을 수 있겠습니다.

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

1. 데이터 로드 및 기본 탐색

Drive에 올라와 있던 creditcard.csv 파일을 불러와서 head(), info(), describe()를 실행해보고, 정상거래와(Class == 0) 사기거래(Class == 1)이 각각 몇 개나 되는지 알아봤습니다.

```
Class
0    284315
1       492
Name: count, dtype: int64
```

Column 개수가 많아서 그런지 output 길이가 정말 길어져서 마지막에 Class별 value_counts() 실행결과만 첨부합니다.

2. 샘플링

명세에 써 있는 대로 샘플링을 진행했고, 그에 따라 만들어진 새로운 데이터프레임에서 value_counts() 실행결과가 다음과 같이 바뀌는 것을 확인했습니다.

```
Class
0    10000
1      492
Name: count, dtype: int64
```

샘플링할 때 정상 데이터는 10000개만 뽑았기 때문에 자연스럽게 샘플링된 데이터에서 정상 데이터가 10000개임을 확인했습니다.

3. 데이터 전처리

Amount_Scaled라는 새로운 변수를 만들고 원래 있던 Amount는 제거했습니다.

Dataframe.drop() 함수에 inplace라는 인자를 주면 리턴값 없이 준 데이터프레임을 바꿀지 아니면 새로운 데이터프레임을 리턴해줄지를 결정할 수 있어서 inplace 인자를 적극적으로 활용해보았습니다.

4. 학습 데이터와 테스트 데이터 분할

명세에 나와 있는 대로 sklearn.model_selection.train_test.split을 이용해 테스트 데이터가 20%가 되도록 나누고, 클래스 비율을 출력했습니다. normalize=True라는 인자가 있더군요...

```
train data ratio: Class
0    0.953056
1    0.046944
Name: proportion, dtype: float64
 test data ratio: Class
0    0.953311
1    0.046689
Name: proportion, dtype: float64
```

5. SMOTE 적용

지금까지 만든 데이터는 사기 거래 (Class == 1)인 데이터의 양이 현저히 적어서 사기 거래 패턴 학습이 어렵다는 단점이 있었습니다. 그렇기 때문에 SMOTE를 사용해서 사기 거래 데이터를 새로 생성해야 합니다.

```
Before SMOTE Counter({0: 7999, 1: 394})
After SMOTE Counter({0: 7999, 1: 7999})
```

다음 사진에서 볼 수 있듯, SMOTE 적용 전후 사기 거래 데이터 수는 394에서 7999로 수십 배 늘어난 것을 확인할 수 있습니다.

6. 모델 학습

Logistic Regression 모델을 선정했습니다.

처음부터 짰으면 굉장히 복잡했을 거 같은데 라이브러리로 이렇게까지 쉽게 된다는 게 신기하네요.

처음에 max_iter=1000으로 설정했다가 ConvergenceWarning: lbfgs failed to converge (status=1) 이런 경고를 만나서 max_iter을 10000으로 조절했더니 같은 경고가 뜨지 않았습니다.

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      2001
           1       0.82      0.93      0.87        98

    accuracy                           0.99      2099
   macro avg       0.91      0.96      0.93      2099
weighted avg       0.99      0.99      0.99      2099

0.9550235754867127
```

7. 최종 성능 평가

저번 페이지에 첨부한 사진에서 볼 수 있듯, Recall, F1 값은 Class 1 (사기 거래)에서 F1=0.87이 뜬 점을 제외한다면 기준을 달성했다고 볼 수 있겠습니다.

PR-AUC같은 경우 Class1은 위 사진에서 볼 수 있듯 0.955 정도, Class0은 ipynb에 있는 마지막 코드에서 알 수 있듯 0.998로 매우 높게 나왔고, 기준을 달성했다고 볼 수 있겠습니다.

Class1의 경우 F1값을 올릴 방법은... 아직 제 부족한 지식으로는 잘 모르겠습니다.