ML Day8

▼ **Mean subtraction(4)**

```python
math_scores = [40, 60, 80]
english_scores = [30, 40, 50]
n_class = 2
n_student = len(math_scores)

score_sums = []
score_means = []

for _ in range(n_class):
    score_sums.append(0)

for student_idx in range(n_student):
    score_sums[0] += math_scores[student_idx]
    score_sums[1] += english_scores[student_idx]
print("sum of scores:", score_sums)

for class_idx in range(n_class):
    class_mean = score_sums[class_idx] / n_student
    score_means.append(class_mean)
print("mean of scores:", score_means)

for student_idx in range(n_student):
    math_scores[student_idx] -= score_means[0]
    english_scores[student_idx] -= score_means[1]
print("Math scores after mean subtraction:", math_scores)
print("English scores after mean subtraction:", english_scores)
```

▼ **[6]**: 점수의 합, 평균을 집어 넣을 빈 list 생성

▼ **[9]**: n_class 횟수만큼 score_sums에 0을 쌓는다. →[0, 0]

▼ **[12]**: n_student(각 배열 원소의 갯수)만큼 각 과목의 합을 score_sums에 대입, score_sums[0] 위치에는 math_scores의 합산, scores_sums[1] 위치에는 english_scores의 합산

▼ **[17]**: n_class의 횟수만큼 score_sums[class_idx] / n_student 평균을 class_mean 변수에 지정하고 score_means에 쌓아준다.

▼ **[22]**: n_student만큼 각각의 scores에 해당하는 값들에서 평균을 빼준다.

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test.py
sum of scores: [180, 120]
mean of scores: [60.0, 40.0]
Math scores after mean subtraction: [-20.0, 0.0, 20.0]
English scores after mean subtraction: [-10.0, 0.0, 10.0]

Process finished with exit code 0
```

▼ 위 코드를 구현한 결괏값

▼ 분산과 표준편차(3)

```python
scores = [10, 20, 30]
n_student = len(scores)
score_sum, score_square_sum = 0, 0

for score in scores:
    score_sum += score
    score_square_sum += score**2

mean = score_sum / n_student
square_of_mean = mean**2
mean_of_square = score_square_sum / n_student

variance = mean_of_square - square_of_mean
std = variance**0.5

print("variance:", variance)
print("standard deviation:", std)
```

▼ **[3]**: 점수의 합과 합의 제곱을 집어 넣을 빈 list생성

▼ **[5]**: scores list의 원소 값들의 합과 제곱의 합을 for구문으로 계산**[38]**: score 변수가 scores를 for구문으로 돌 동안 score를 계속 합해 score_sum에 대입, score의 제곱을 계속 합한 값을 score_square_sum에 대입

▼ **[11]**: 제곱의 평균 **=** 리스트의 각 원소들의 제곱을 합한 값을 n_student로 나눈다.

▼ **[13]**: 분산 = 제곱의 평균 - 평균의 제곱

▼ **[14]**: 표준편차 = 분산의 0.5제곱 = 분산에 루트를 씌운 값

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test.py
variance: 66.66666666666669
standard deviation: 8.16496580927726

Process finished with exit code 0
```

▼ 위 코드의 결과

▼ Standardization(3)

```python
scores = [10, 20, 30]
n_student = len(scores)
score_sum, score_square_sum = 0, 0

for score in scores:
    score_sum += score
    score_square_sum += score**2

mean = score_sum / n_student
square_of_mean = mean**2
mean_of_square = score_square_sum / n_student
variance = mean_of_square - square_of_mean
std = variance**0.5

for student_idx in range(n_student):
    scores[student_idx] = (scores[student_idx] - mean)/std
print(scores)

score_sum, score_square_sum = 0, 0

for score in scores:
    score_sum += score
    score_square_sum += score**2

mean = score_sum / n_student
square_of_mean = mean**2
mean_of_square = score_square_sum / n_student
variance = mean_of_square - square_of_mean
std = variance**0.5
print("mean:", mean)
print("standard deviation:", std)
```

▼ **[7]**: for 구문으로 scores의 원소들의 합과 원소들의 제곱의 합을 각각 score_sum, score_square_sum에 대입

▼ **[14]**: 분산 : 제곱의 평균 - 평균의 제곱

▼ **[15]**: 표준편차 : 분산의 0.5제곱

▼ **[17]**: standardization과정 - scores에 해당하는 학생의 index들의 값에서 평균을 뺀 값을 표준편차로 나눠준다.(scores[student_idx])

▼ **[21]**: standardization과정 거친 후 다시 평균과 분산을 구해준다.

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
[-1.224744871391589, 0.0, 1.224744871391589]
mean: 0.0
standard deviation: 1.0

Process finished with exit code 0
```

▼ 분산과 표준편차(4)

```python
# 분산과 표준편차(4)
math_scores, english_scores = [50, 60, 70], [30, 40, 50]
n_student = len(math_scores)

math_sum, english_sum = 0, 0
math_square_sum, english_square_sum = 0, 0

for student_idx in range(n_student):
    math_sum += math_scores[student_idx]
    math_square_sum += math_scores[student_idx]**2

    english_sum += english_scores[student_idx]
    english_square_sum += english_scores[student_idx]**2

math_mean = math_sum / n_student
english_mean = english_sum / n_student

math_variance = math_square_sum / n_student - math_mean**2
english_variance = english_square_sum / n_student - english_mean**2

math_std = math_variance**0.5
english_std = english_variance**0.5

print("mean/std of Math:", math_mean, math_std)
print("mean/std of English:", english_mean, english_std)
```

▼ **[8]**: for 구문을 이용해 각 과목의 합과 제곱의 합을 구함

▼ **[15]**: 각 과목의 평균

▼ **[18]**: 각 과목의 분산 = (각 과목의 제곱의 합산) / 학생의 수 - 각 과목의 평균의 제곱

▼ **[21]**: 각 과목의 표준 편차

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
mean/std of Math: 60.0 8.164965809277252
mean/std of English: 40.0 8.164965809277264
mean/std of Math: 60.0 8.164965809277252
mean/std of English: 40.0 8.164965809277264

Process finished with exit code 0
```

▼ **Standardization(4)**

```python
math_scores, english_scores = [50, 60, 70], [30, 40, 50]
n_student = len(math_scores)

math_sum, english_sum = 0, 0
math_square_sum, english_square_sum = 0, 0

for student_idx in range(n_student):
    math_sum += math_scores[student_idx]
    math_square_sum += math_scores[student_idx]**2

    english_sum += english_scores[student_idx]
    english_square_sum += english_scores[student_idx]**2

math_mean = math_sum / n_student
english_mean = english_sum / n_student

math_variance = math_square_sum / n_student - math_mean**2
english_variance = english_square_sum / n_student - english_mean**2

math_std = math_variance**0.5
english_std = english_variance**0.5

for student_idx in range(n_student):
    math_scores[student_idx] = (math_scores[student_idx] - math_mean) / math_std
    english_scores[student_idx] = (english_scores[student_idx] - english_mean) / english_std

print("Math scores after standardization:", math_scores)
print("English scores after standardization:", english_scores)

math_sum, english_sum = 0, 0
math_square_sum, english_square_sum = 0, 0

for student_idx in range(n_student):
    math_sum += math_scores[student_idx]
    math_square_sum += math_scores[student_idx]**2

    english_sum += english_scores[student_idx]
    english_square_sum += english_scores[student_idx]**2

math_mean = math_sum / n_student
english_mean = english_sum / n_student

math_variance = math_square_sum / n_student - math_mean**2
english_variance = english_square_sum / n_student - english_mean**2

math_std = math_variance**0.5
english_std = english_variance**0.5

print("mean/std of Math:", math_mean, math_std)
print("mean/std of English:", english_mean, english_std)
```

▼ **[7:]**: math, english의 합과 제곱의 합을 for구문을 통해 계산

▼ **[14:]**: 각 과목의 평균과 분산, 표준편차를 구함

▼ **[23:]**: 배열 안 원소들에 해당 과목의 평균을 각각 빼준 후 표준편차로 나눔 = Standardization

▼ **[30:]**: standardization된 score값으로 다시 평균과 분산, 표준편차를 구함

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
Math scores after standardization: [-1.2247448713915903, 0.0, 1.2247448713915903]
English scores after standardization: [-1.2247448713915885, 0.0, 1.2247448713915885]
mean/std of Math: 0.0 1.0000000000000009
mean/std of English: 0.0 0.9999999999999996

Process finished with exit code 0
```

▼ **Hadamard Product(4)**

```python
# Hadamard Product(4)

v1 = [1, 2, 3, 4, 5]
v2 = [10, 20, 30, 40, 50]

# method.1
v3 = []
for dim_idx in range(len(v1)):
    v3.append(v1[dim_idx] * v2[dim_idx])
print(v3)

# method.2
v3 = []
for _ in range(len(v1)):
    v3.append(0)

for dim_idx in range(len(v1)):
    v3[dim_idx] = v1[dim_idx] * v2[dim_idx]
print(v3)
```

▼ [7]: v1, v2 각각에 대응하는 원소들의 곱을 집어넣을 v3 list 생성

▼ [8]: v1, v2 같은 index에 위치하는 값을 서로 곱해주고 v3 list에 append

▼ [14]: v1의 원소의 갯수만큼 v3에 0을 append

▼ [17]: v1[0]*v2[0] = v3[0] → 이러한 형식으로 새로운 v3행렬이 생김

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
[10, 40, 90, 160, 250]
[10, 40, 90, 160, 250]

Process finished with exit code 0
```

▼ Vector Norm(3)

```
1  # Vector Norm(3)
2
3  v1 = [1, 2, 3]
4
5  square_sum = 0
6  for dim_val in v1:
7      square_sum += dim_val**2
8  norm = square_sum**0.5
9  print("norm of v1:", norm)
```

▼ [5~7]: square_sum = 0으로 시작값을 주고 v1 list를 for 구문으로 돌 동안 각각의 원소들의 제곱을 합산한 결과를 square_sum에 대입해준다.

▼ [8]: square_sum에 루트를 씌운 값 = norm

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
norm of v1: 3.7416573867739413

Process finished with exit code 0
```

▼ **Making Unit Vectors(3)**

```python
# Making Unit Vectors(3)

v1 = [1, 2, 3]

square_sum = 0
for dim_val in v1:
    square_sum += dim_val**2
norm = square_sum**0.5
print("norm of v1:", norm)

for dim_idx in range(len(v1)):
    v1[dim_idx] /= norm

square_sum = 0
for dim_val in v1:
    square_sum += dim_val**2
norm = square_sum**0.5
print("norm of v1:", norm)
```

▼ [11]: norm값을 구하고 다시 v1 index 순서대로 norm으로 나눠준 값을 다시 v1 list에 대입

▼ [14~17]: 원소의 값들이 바뀐 v1 list의 norm을 다시 구함

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
norm of v1: 3.7416573867739413
norm of v1: 1.0

Process finished with exit code 0
```

▼ **Dot Product(3)**

```python
# Dot Product(3)

v1, v2 = [1, 2, 3], [3, 4, 5]
dot_prod = 0
for dim_idx in range(len(v1)):
    dot_prod += v1[dim_idx] * v2[dim_idx]
print("dot product of v1 and v2:", dot_prod)
```

▼ [5]: v1, v2 각각의 index에 대응되는 값을 곱해준 후, 그 값들을 전부 합산 = dot produc

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
dot product of v1 and v2: 26

Process finished with exit code 0
```

▼ 위 코드를 구현한 결과 = Hadamard product는 각각의 index에 대응되는 값을 곱해준 값이 새로 생긴 행렬의 같은index에 위치하게 됨, 하지만 dot product는 각각의 index에 대응되는 값을 곱해준 후 전부 합산해주기 때문에 결괏값이 한개의 원소로 나오게 됨

▼ **Euclidean Distance(3)**

```python
# Euclidean Distance(3)

v1, v2 = [1, 2, 3], [3, 4, 5]

diff_square_sum = 0
for dim_idx in range(len(v1)):
    diff_square_sum += (v1[dim_idx] - v2[dim_idx])**2
e_distance = diff_square_sum**0.5

print("Euclidean distance between v1 and v2:", e_distance)
```

▼ [6]: v1과 v2의 대응되는 index위치의 원소들의 차에 제곱을 한 값을 diff_square_sum에 대입

▼ [8]: diff_square_sum의 루트를 씌운 값 = e_distance

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
Euclidean distance between v1 and v2: 3.4641016151377544
```

▼ Mean Squared Error(3)

```python
# Mean Squared Error(3)

predictions = [10, 20, 30]
labels = [10, 25, 40]

n_data = len(predictions)
diff_square_sum = 0

for data_idx in range(n_data):
    diff_square_sum += (predictions[data_idx] - labels[data_idx])**2
mse = diff_square_sum / n_data
print("MSE: ", mse)
```

▼ [3]: 예측값

▼ [4]: 결괏값

▼ [9]: 예측값과 결괏값의 각각 대응되는 원소들의 차이에 제곱을 한 값들을 합산하여 diff_square_sum에 대입

▼ [11]: diff_square_sum을 n_data(data의 길이)로 나눠준 값 = mse

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
MSE:  41.666666666666664
```

▼ 숫자 빈도 구하기

```python
# 숫자 빈도 구하기

numbers = [0, 2, 4, 2, 1, 4, 3, 1, 2, 3, 4, 1, 2, 3, 4]
number_cnt = [0, 0, 0, 0, 0]

for num in numbers:
    number_cnt[num] += 1
print(number_cnt)
```

▼ [6]: for구문을 이용하여 각각의 숫자를 만날때마다 number_cnt index에 해당하는 값을 1씩 더해준다.

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
[1, 3, 4, 3, 4]
```

▼ 위 코드를 구현한 결괏값 = 0은 1개, 1은 3개, 2는 4개, 3은 3개, 4는 4개

▼ 초를 분초로 표현하기

```python
# 초를 분초로 표현하기

seconds = 200

if seconds >= 60:
    minutes = seconds // 60
    seconds -= minutes*60
else:
    minutes = 0
print(minutes, "min", seconds, "sec")
```

▼ [6]: seconds를 60으로 나눈 몫을 minutes 변수에 대입

▼ [7]: seconds에서 minutes*60을 뺀 값을 다시 seconds에 대입

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
3 min 20 sec
```

▼ 초를 시분초로 표현하기

```python
# 초를 시분초로 표현하기

seconds = 5000

if seconds >= 3600:
    hours = seconds // 3600
    seconds -= hours*3600
else:
    hours = 0

if seconds >= 60:
    minutes = seconds // 60
    seconds -= minutes*60
else:
    minutes = 0

print(hours, "hr ", minutes, "min ", seconds, "sec")
```

▼ [5]: 3600초 = 1시간, seconds(5000)를 3600으로 나눈 몫을 hours로, seconds에서 hours*3600을 뺀 값을 다시 seconds로 대입

▼ [11]: 60초 = 1분, seconds를 60으로 나눈 후, 몫을 minutes으로, seconds에서 minutes*60을 뺀 값을 다시 seconds로 대입

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
1 hr  23 min  20 sec
```

▼ 두 수 비교하기

```python
# 두 수 비교하기

num1, num2 = 10, 10

if num1 > num2:
    print("first number")
elif num1 == num2:
    print("equal")
else:
    print("second number")
```

▼ [5]: num1 이 num2 보다 크면 ‘first number’출력, 같으면 ‘equal’ 출력

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
equal
```

▼ 성적을 평점으로 바꾸기

```python
# 성적을 평점으로 바꾸기(2)

scores = [20, 50, 10, 60, 90]
grades = []

for score in scores:
    if score > 80:
        grades.append("A")
    elif score > 60:
        grades.append("B")
    elif score > 40:
        grades.append("C")
    else:
        grades.append("F")
print(grades)
```

▼ [6]: 원소들이 scores list를 for구문으로 돌 때 80보다 크면 'A'값을 grades 리스트에 append

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
['F', 'C', 'F', 'C', 'A']
```

▼ 합격/불합격 학생 들의 평균 구하기

```python
# 합격/불합격 학생들의 평균 구하기

scores = [20, 50, 10, 60, 90]
cutoff = 50

p_score_sum, n_p = 0, 0
np_score_sum, n_np = 0, 0

for score in scores:
    if score > cutoff:
        p_score_sum += score
        n_p += 1
    else:
        np_score_sum += score
        n_np += 1

p_score_mean = p_score_sum/n_p
np_score_mean = np_score_sum/n_np
print("mean of passed scores:", p_score_mean)
print("mean of no passed scores:", np_score_mean)
```

▼ [6]: pass한 점수들의 합(p_score_sum)과 pass한 점수의 갯수(n_p)의 시작값을 0으로 지정

▼ [9]: score변수가 scores list를 for구문으로 돌 때, 각 원소가 cutoff 점수보다 높을 경우, p_score_sum에 해당 점수를 더해주고, pass한 점수의 갯수를 1씩 더해준다.

```
C:\Users\SBAUser\anaconda3\python.exe C:\Users\SBAUser\anaconda3\envs\SBA\Jong\test2.py
mean of passed scores: 75.0
mean of no passed scores: 26.666666666666668
```