머신러닝 CHAPTER 01~02 객관식 퀴즈

Part 1. 이론 및 실무 (1~40번)

1. 머신러닝에서 데이터를 표현하는 하나의 성질(예: 생선의 길이, 무게)을 무엇이라 하는가?

① 타깃(target)
② 샘플(sample)
③ 특성(feature)
④ 레이블(label)
⑤ 하이퍼파라미터(hyperparameter)

2. 머신러닝 알고리즘이 데이터에서 규칙을 찾는 과정을 무엇이라 하며, 사이킷런에서 이를 수행하는 메서드는?

① 예측 - predict()
② 훈련 - fit()
③ 평가 - score()
④ 변환 - transform()
⑤ 적합 - compile()

3. 사이킷런에서 훈련된 모델로 새로운 데이터의 결과를 예측할 때 사용하는 메서드는?

① fit()
② score()
③ predict()
④ transform()
⑤ evaluate()

4. 사이킷런의 score() 메서드가 반환하는 값의 범위와 의미로 올바른 것은?

① 0~100, 백분율 정확도
② 0~1, 정확히 맞힌 비율
③ -1~1, 상관계수
④ 0~1, 손실값
⑤ 1~5, 등급

5. k-최근접 이웃(KNN) 알고리즘의 예측 방식으로 올바른 것은?

① 데이터에서 수학적 함수를 학습하여 예측한다
② 의사결정 규칙을 트리 형태로 학습한다

③ 전체 데이터를 저장해두고, 새 데이터와 가장 가까운 이웃들의 다수결로 예측한다
④ 데이터의 확률 분포를 학습하여 예측한다
⑤ 가중치를 반복적으로 업데이트하여 최적의 경계를 찾는다

6. KNN에서 n_neighbors 값을 너무 작게(예: 1) 설정했을 때와 너무 크게(예: 전체 데이터 수) 설정했을 때 각각 발생하는 문제의 올바른 조합은?

① 과소적합 / 과대적합
② 과대적합 / 과소적합
③ 데이터 누수 / 샘플링 편향
④ 둘 다 과대적합
⑤ 둘 다 과소적합

7. 도미 35마리, 빙어 14마리(총 49개) 데이터에서 n_neighbors=49로 설정하면 어떤 일이 발생하는가?

① 모든 데이터를 정확히 분류한다
② 모델이 오류를 발생시킨다
③ 모든 입력을 도미(다수 클래스)로만 예측한다
④ 모든 입력을 빙어로만 예측한다
⑤ 랜덤하게 예측한다

8. k-최근접 이웃 알고리즘의 실무적 단점으로 가장 적절한 것은?

① 범주형 데이터를 처리할 수 없다
② 데이터가 많으면 메모리 사용량과 예측 시간이 크게 증가한다
③ 이진 분류만 가능하다
④ 반드시 GPU가 필요하다
⑤ 데이터 전처리가 불가능하다

9. 훈련 데이터에서 정확도가 0.99인데 테스트 데이터에서 정확도가 0.55라면, 이 모델의 문제로 가장 적절한 것은?

① 과소적합 — 모델이 데이터의 패턴을 충분히 학습하지 못했다
② 과대적합 — 모델이 훈련 데이터를 외워버려 새로운 데이터에 일반화하지 못한다
③ 샘플링 편향 — 테스트 세트에 특정 클래스만 포함되어 있다
④ 데이터 전처리 오류 — 스케일이 맞지 않아 발생한 문제이다
⑤ 모델 선택 오류 — KNN이 아닌 다른 알고리즘을 사용해야 한다

10. 사이킷런 모델의 fit() 메서드에 전달하는 입력 데이터의 형태로 올바른 것은?

① 1차원 리스트
② 딕셔너리
③ 2차원 리스트(행: 샘플, 열: 특성)
④ 2차원 리스트(행: 특성, 열: 샘플)
⑤ JSON 문자열

11. 다음 코드에서 fish_target을 올바르게 설정한 것은? (도미 35마리=1, 빙어 14마리=0)

① fish_target = [1, 35] + [0, 14]
② fish_target = [1] * 35 + [0] * 14
③ fish_target = [35] + [14]
④ fish_target = list(range(1, 36)) + list(range(0, 14))
⑤ fish_target = np.array([1, 0]) * 49

12. 파이썬에서 두 리스트의 원소를 하나씩 짝지어 2차원 리스트로 만들 때 사용하는 구문은?

① np.merge(length, weight)
② [[l, w] for l, w in zip(length, weight)]
③ list(length, weight)
④ length.append(weight)
⑤ np.stack(length, weight)

13. 머신러닝 모델을 만들기 전에 산점도(scatter plot)를 그려보는 실무적 이유로 가장 적절한 것은?

① 모델의 정확도를 미리 계산하기 위해
② 데이터의 분포, 이상치, 클래스 간 분리 가능성을 시각적으로 파악하기 위해
③ 사이킷런 fit() 메서드의 필수 입력이기 때문에
④ 데이터를 자동으로 전처리하기 위해
⑤ 훈련 시간을 단축하기 위해

14. 지도 학습에서 모델에 전달하는 두 가지 데이터는?

① 훈련 세트와 테스트 세트
② 입력(특성)과 타깃(정답)
③ 평균과 표준편차
④ 훈련 데이터와 검증 데이터
⑤ 원본 데이터와 전처리 데이터

15. 비지도 학습의 특징으로 올바른 것은?

① 입력 데이터와 정답을 함께 사용한다
② 정답 없이 입력 데이터만으로 데이터의 패턴이나 구조를 파악한다
③ 보상 신호를 통해 최적의 행동을 학습한다
④ 소량의 정답 데이터와 대량의 비정답 데이터를 함께 사용한다
⑤ 사전 훈련된 모델의 가중치를 재사용한다

16. 모델을 훈련한 데이터와 동일한 데이터로 평가하면 안 되는 이유는?

① 계산 시간이 너무 오래 걸리기 때문
② 메모리가 부족해지기 때문
③ 정답을 미리 알고 시험을 보는 것과 같아 실제 성능을 알 수 없기 때문
④ 사이킷런에서 기술적으로 불가능하기 때문
⑤ 과소적합이 발생하기 때문

17. 훈련 세트와 테스트 세트에 샘플이 골고루 섞이지 않아 한쪽으로 치우치는 현상을 무엇이라 하는가?

① 과대적합(overfitting)
② 과소적합(underfitting)
③ 샘플링 편향(sampling bias)
④ 데이터 누수(data leakage)
⑤ 편향-분산 트레이드오프

18. 도미 35마리, 빙어 14마리 데이터를 순서대로 앞 35개를 훈련, 뒤 14개를 테스트로 나누면 테스트 정확도가 0.0이 되는 이유는?

① 데이터가 너무 적어서
② 테스트 세트에 빙어만 있고 훈련 세트에 빙어가 없어서
③ k-최근접 이웃 알고리즘의 버그 때문에
④ 특성의 스케일이 달라서
⑤ n_neighbors 값이 잘못 설정되어서

19. 넘파이 배열의 shape 속성이 (49, 2)를 반환했을 때의 의미는?

① 49개의 특성, 2개의 샘플
② 49개의 샘플, 2개의 특성
③ 49행 2열의 문자열 배열
④ 총 49 x 2 = 98개의 1차원 원소
⑤ 2개의 49차원 벡터

20. 파이썬 리스트 a = [1, 2, 3]과 넘파이 배열 b = np.array([1, 2, 3])에서 a * 2와 b * 2의 결과 차이로 올바른 것은?

① 둘 다 [2, 4, 6]을 반환한다

② a * 2는 [1, 2, 3, 1, 2, 3], b * 2는 [2, 4, 6]을 반환한다

③ a * 2는 오류가 발생하고, b * 2는 [2, 4, 6]을 반환한다

④ 둘 다 [1, 2, 3, 1, 2, 3]을 반환한다

⑤ a * 2는 [2, 4, 6], b * 2는 [1, 2, 3, 1, 2, 3]을 반환한다

21. 넘파이 배열 arr에서 인덱스 배열 [3, 7, 12]를 사용하여 여러 원소를 한 번에 선택하는 방법은?

① arr.select([3, 7, 12])

② arr[[3, 7, 12]]

③ arr.get(3, 7, 12)

④ arr(3, 7, 12)

⑤ arr.pick([3, 7, 12])

22. 파이썬 리스트의 슬라이싱 fish_data[0:5]에서 실제로 선택되는 인덱스 범위는?

① 0, 1, 2, 3, 4, 5

② 1, 2, 3, 4, 5

③ 0, 1, 2, 3, 4

④ 0, 1, 2, 3

⑤ 1, 2, 3, 4

23. 2차원 넘파이 배열 train_input에서 모든 행의 첫 번째 열만 선택하는 방법은?

① train_input[0]

② train_input[:,0]

③ train_input[0,:]

④ train_input[:][0]

⑤ train_input.col(0)

24. 두 개의 넘파이 1차원 배열을 나란히 합쳐 2차원 배열로 만드는 함수는?

① np.concatenate()

② np.column_stack()

③ np.append()

④ np.hstack()

⑤ np.merge()

25. np.concatenate()와 np.column_stack()의 차이로 올바른 것은?

① 기능이 동일하다

② concatenate는 첫 번째 축(행)을 따라 연결하고, column_stack은 두 번째 축(열)을 따라 나란히 연결한다

③ concatenate만 넘파이 배열을 지원한다

④ column_stack은 3차원 이상만 지원한다

⑤ concatenate는 숫자만, column_stack은 문자열만 지원한다

26. 테스트 세트의 정보(평균, 분포 등)가 모델 훈련 과정에 포함되면 발생하는 문제를 무엇이라 하며, 그 결과는?

① 샘플링 편향 — 클래스 비율이 달라진다

② 데이터 누수(data leakage) — 실제보다 높은 성능이 나타나 모델을 신뢰할 수 없다

③ 과소적합 — 모델이 학습을 충분히 하지 못한다

④ 브로드캐스팅 오류 — 배열 연산이 실패한다

⑤ 특성 공학 오류 — 잘못된 특성이 만들어진다

27. train_test_split(fish_data, fish_target, stratify=fish_target)에서 stratify의 역할은?

① 데이터를 시간순으로 정렬한다

② 테스트 세트의 크기를 지정한다

③ 클래스 비율에 맞게 훈련/테스트 세트를 나눈다

④ 랜덤 시드를 고정한다

⑤ 데이터를 표준화한다

28. train_test_split()에 2개의 배열을 전달하면 반환되는 배열의 개수는?

① 2개

② 3개

③ 4개

④ 5개

⑤ 6개

29. train_test_split()의 random_state 매개변수의 역할은?

① 테스트 세트 비율을 지정한다

② 동일한 값을 주면 매번 같은 방식으로 데이터를 나눈다

③ 데이터를 정렬하는 기준을 지정한다

④ k-최근접 이웃의 k값을 지정한다

⑤ 데이터를 표준화하는 기준을 지정한다

30. 길이 25cm, 무게 150g인 도미를 KNN 모델이 빙어로 잘못 예측한 원인은?

① 훈련 데이터가 부족해서
② n_neighbors 값이 너무 작아서
③ 두 특성(길이, 무게)의 스케일이 달라 무게가 거리 계산을 지배해서
④ 도미 데이터에 오류가 있어서
⑤ 모델이 과소적합되어서

31. 데이터 전처리에서 표준점수(z-score)를 구하는 공식으로 올바른 것은?

① (특성값 - 최솟값) / (최댓값 - 최솟값)
② (특성값 - 평균) / 표준편차
③ 특성값 / 최댓값
④ (특성값 - 중앙값) / 사분위 범위
⑤ log(특성값)

32. 테스트 세트를 전처리할 때 반드시 지켜야 하는 원칙은?

① 테스트 세트의 평균과 표준편차를 사용한다
② 전체 데이터의 평균과 표준편차를 사용한다
③ 훈련 세트의 평균과 표준편차를 사용한다
④ 각 샘플마다 개별적으로 정규화한다
⑤ 전처리하지 않는 것이 좋다

33. 넘파이에서 배열의 각 열(특성)별 평균을 구하려면 np.mean()에 어떤 매개변수를 지정해야 하는가?

① axis=1
② axis=0
③ axis=-1
④ axis=None
⑤ dim=0

34. 넘파이의 브로드캐스팅(broadcasting)이란?

① 배열을 네트워크로 전송하는 기능
② 배열을 파일로 저장하는 기능
③ 크기가 다른 배열 간에 자동으로 사칙연산을 확장하여 수행하는 기능
④ 배열을 시각화하는 기능
⑤ 배열을 정렬하는 기능

35. KNeighborsClassifier의 kneighbors() 메서드가 반환하는 것은?

① 예측 결과만
② 정확도만
③ 이웃까지의 거리와 이웃 샘플의 인덱스
④ 훈련 데이터 전체
⑤ 클래스 확률

36. fit() 메서드를 같은 모델 객체에 다시 호출하면 어떤 일이 발생하는가?

① 이전 학습에 추가로 학습된다
② 오류가 발생한다
③ 이전에 학습한 모든 것을 잃고 새로운 데이터로 다시 훈련된다
④ 아무 일도 일어나지 않는다
⑤ 이전 모델과 새 모델이 앙상블된다

37. 다음 중 거리 기반 알고리즘에서 데이터 전처리(스케일 조정)가 특히 중요한 이유는?

① 훈련 속도가 빨라지기 때문
② 메모리 사용량이 줄어들기 때문
③ 특성 간 스케일 차이가 거리 계산 결과를 왜곡하기 때문
④ 모델의 해석력이 높아지기 때문
⑤ 과대적합을 방지하기 때문

38. 머신러닝 실험에서 결과의 재현성(reproducibility)을 확보해야 하는 실무적 이유로 가장 적절한 것은?

① 코드 실행 속도를 높이기 위해
② 실험 결과를 동료와 공유·검증하고, 디버깅 시 동일한 조건을 재현하기 위해
③ 메모리 사용량을 줄이기 위해
④ 모델의 정확도를 높이기 위해
⑤ 데이터 전처리를 자동화하기 위해

39. 전체 데이터를 훈련 세트와 테스트 세트로 나눌 때, 일반적으로 테스트 세트에 할당하는 비율은?

① 50~60%
② 5~10%
③ 20~30%
④ 40~50%
⑤ 1~2%

40. 다음 코드의 실행 결과로 올바른 것은?

```
import numpy as np
arr = np.array([[1, 2], [3, 4], [5, 6]])
print(arr.shape)
```

① (2, 3)

② (3, 2)

③ (6,)

④ (1, 6)

⑤ (3, 3)

Part 2. 과제 기반 문제 (41~60번)

📋 읽기 자료: 과제 설명서

※ 아래는 풀어야 할 문제가 아닙니다. 41~60번 객관식 문제를 풀기 위한 참고 자료입니다. 반드시 읽은 후 문제를 푸세요.

■ 과제 배경

과일 판매점에서 사과(apple)와 귤(mandarin)을 자동 분류하는 시스템을 만드려고 합니다. 각 과일의 무게(g)와 당도(brix)를 측정한 데이터가 아래에 주어져 있습니다.

- 사과: 30개 (무게 150~300g, 당도 11~15 brix)
- 귤: 20개 (무게 50~120g, 당도 8~14 brix)

■ 주어진 데이터

```
# 사과 데이터
apple_weight = [151.0, 153.8, 154.0, 154.5, 163.0, 163.9, 164.5, 173.3, 179.8, 182.8,
                183.1, 183.5, 191.3, 200.5, 201.0, 213.3, 225.8, 231.7, 238.4, 240.6,
                245.9, 247.5, 251.5, 254.7, 260.5, 270.9, 271.4, 277.1, 283.8, 293.6]
apple_sugar  = [14.2, 13.9, 13.1, 14.9, 12.5, 13.2, 14.3, 13.5, 14.4, 13.3,
                13.8, 11.2, 11.9, 12.2, 11.3, 11.9, 11.4, 12.1, 13.5, 12.5,
                12.5, 11.8, 12.1, 14.7, 13.6, 13.4, 11.7, 13.9, 11.7, 12.5]


# 귤 데이터
mandarin_weight = [52.2, 64.8, 66.0, 67.3, 68.5, 68.7, 72.0, 72.1, 77.7, 82.1,
                   89.0, 94.8, 95.9, 97.9, 104.3, 109.0, 111.3, 114.0, 116.0, 119.3]
mandarin_sugar  = [11.4, 9.6, 11.5, 13.4, 10.4, 9.3, 14.0, 11.1, 8.5, 8.3,
                   8.7, 11.8, 12.8, 10.5, 8.4, 10.3, 14.0, 11.2, 13.8, 13.2]
```

■ 과제 요구사항 (A~E)

학생들은 아래 A~E의 빈칸을 채우는 코드를 작성해야 합니다.

A. 데이터 준비 (10점)

```
# weight  : 사과 무게 + 귤 무게를 합친 리스트
# sugar   : 사과 당도 + 귤 당도를 합친 리스트
# fruit_data   : [[무게, 당도], [무게, 당도], ...] 형태의 2차원 리스트
# fruit_target : 사과=1 (30개), 귤=0 (20개)으로 라벨링한 리스트
```

B. KNN 모델 훈련 (20점)

```
from sklearn.neighbors import KNeighborsClassifier
# kn : 기본 매개변수(n_neighbors=5)로 KNeighborsClassifier 객체 생성
# fruit_data와 fruit_target으로 모델을 훈련(fit)
# train_score : 훈련 데이터에 대한 정확도(score) 저장
```

C. 새로운 데이터 예측 (20점)

```
# pred_apple    : 무게 200g, 당도 13인 과일의 예측 결과 (predict 사용)
# pred_mandarin : 무게 80g,  당도 10인 과일의 예측 결과
```

D. n_neighbors 변경 실험 (25점)

전체 데이터 개수(50개)를 n_neighbors로 설정한 KNN 모델을 만들고, 정확도를 확인하세요. 이 경우 항상 다수결(사과 30 > 귤 20)로 사과를 예측하게 됩니다.

```
# kn50 : n_neighbors=50으로 KNeighborsClassifier 객체 생성 → 훈련
# score_50 : 훈련 데이터에 대한 정확도
# expected_score : 전체 50개 중 사과(30개)의 비율 계산 (30/50)
```

E. 최적의 n_neighbors 찾기 (25점)

n_neighbors를 5부터 49까지 변경하며 정확도가 처음으로 1.0 미만으로 떨어지는 값을 찾으세요.

```
kn = KNeighborsClassifier()
kn.fit(fruit_data, fruit_target)
first_drop_n = None       # 처음으로 정확도가 1.0 미만이 되는 n_neighbors
first_drop_score = None   # 그때의 정확도
# 힌트: for 반복문과 kn.n_neighbors 속성을 활용하세요.
```

✔ 여기서부터 풀어야 할 객관식 문제입니다 (41~60번)

위의 과제 설명서를 참고하여 아래 문제의 정답을 고르세요.

41. 위 과제에서 사과와 귤을 분류하기 위해 사용한 두 가지 특성(feature)은?

① 색깔과 냄새

② 무게(g)와 당도(brix)

③ 크기와 가격

④ 산도와 수분함량

⑤ 모양과 원산지

42. 사과 30개(레이블=1)와 귤 20개(레이블=0)로 fruit_target 리스트를 만드는 올바른 코드는?

① fruit_target = [1, 30] + [0, 20]

② fruit_target = [1] * 30 + [0] * 20

③ fruit_target = [30] * 1 + [20] * 0

④ fruit_target = list(range(30)) + list(range(20))

⑤ fruit_target = [1, 0] * 50

43. weight와 sugar 리스트를 사용해 [[무게, 당도], ...] 형태의 fruit_data를 만드는 올바른 코드는?

① fruit_data = [weight, sugar]

② fruit_data = [[w, s] for w, s in zip(weight, sugar)]

③ fruit_data = weight + sugar

④ fruit_data = list(weight, sugar)

⑤ fruit_data = weight.append(sugar)

44. KNeighborsClassifier()를 기본값(n_neighbors=5)으로 생성하고 위 50개 데이터로 훈련했을 때, kn.score(fruit_data, fruit_target)의 예상 결과는?

① 0.5

② 0.6

③ 0.8

④ 1.0

⑤ 0.96

45. n_neighbors=50으로 KNN 모델을 훈련했을 때 훈련 정확도가 0.6이 되는 이유로 가장 적절한 것은?

① 데이터 개수가 너무 적어서

② 전체 50개 이웃을 참고하므로 항상 다수 클래스(사과 30개)로만 예측하기 때문에

③ 귤 데이터의 당도가 잘못 입력되었기 때문에

④ score() 메서드에 버그가 있어서

⑤ n_neighbors를 짝수로 설정했기 때문에

46. n_neighbors를 5부터 49까지 증가시키며 확인할 때, 정확도가 처음으로 1.0 미만으로 떨어지는 n_neighbors 값은?

① 5

② 20

③ 36

④ 42

⑤ 49

47. 훈련된 KNN 모델에서 kn.predict([[200, 13]])의 결과로 올바른 것은?

① [0] — 귤로 예측

② [1] — 사과로 예측

③ [0.5] — 확률 반환

④ ["apple"] — 문자열 반환

⑤ 오류 발생

48. 훈련된 KNN 모델에서 kn.predict([[80, 10]])의 결과로 올바른 것은?

① [1] — 사과로 예측

② [0] — 귤로 예측

③ [0.5] — 확률 반환

④ ["mandarin"] — 문자열 반환

⑤ 오류 발생

49. 이미 생성된 KNN 객체 kn의 n_neighbors 값을 반복문 안에서 직접 변경하는 올바른 코드는?

① kn = KNeighborsClassifier(n_neighbors=n)

② kn.set_neighbors(n)

③ kn.n_neighbors = n

④ kn.update(n_neighbors=n)

⑤ kn.params["n_neighbors"] = n

50. fruit_data와 fruit_target의 길이가 반드시 50이어야 하는 이유로 올바른 것은?

① 사이킷런 KNN 모델이 최소 50개 이상의 데이터를 요구하기 때문에

② 사과 30개와 귤 20개의 데이터를 합쳤기 때문에

③ n_neighbors의 최댓값이 50이어서 데이터도 동일해야 하기 때문에

④ 파이썬 리스트는 50개 단위로 메모리를 할당하기 때문에

⑤ 정확도 0.6을 얻으려면 반드시 50개가 필요하기 때문에

51. 파이썬에서 apple_weight + mandarin_weight를 실행하면 어떤 결과가 나오는가?

① 각 원소끼리 더한 결과 리스트 (벡터 합)

② 두 리스트를 이어붙인 길이 50의 리스트

③ 오류 발생 (리스트끼리 덧셈 불가)

④ 두 리스트의 평균값

⑤ 길이가 긴 리스트만 반환

52. 과제에서 KNeighborsClassifier 객체를 기본 매개변수로 생성하는 올바른 코드는?

① kn = KNeighborsClassifier(k=5)

② kn = KNeighborsClassifier()

③ kn = KNN(n_neighbors=5)

④ kn = sklearn.KNeighborsClassifier(5)

⑤ kn = KNeighborsClassifier.create()

53. kn.score(fruit_data, fruit_target)가 내부적으로 수행하는 작업으로 올바른 것은?

① fruit_data의 통계량(평균, 표준편차)을 계산한다

② fruit_data로 예측한 결과와 fruit_target을 비교하여 정확도를 계산한다

③ 모델을 다시 훈련(fit)한 후 정확도를 반환한다

④ fruit_target의 클래스 비율을 반환한다

⑤ 교차 검증을 수행하여 평균 정확도를 반환한다

54. kn.predict([200, 13]) 대신 kn.predict([[200, 13]])으로 이중 대괄호를 사용해야 하는 이유는?

① 파이썬 문법상 리스트는 항상 이중 대괄호여야 하기 때문에

② predict()는 2차원 배열(샘플 × 특성)을 입력으로 요구하기 때문에

③ 이중 대괄호를 사용하면 예측 속도가 빨라지기 때문에

④ 사이킷런의 버그로 인해 이중 대괄호가 필요하기 때문에

⑤ 단일 대괄호를 사용하면 확률값이 반환되기 때문에

55. expected_score = 30/50의 결과값과 이 값이 의미하는 것은?

① 1.5 — 사과가 귤보다 1.5배 많다는 의미

② 0.6 — n_neighbors=50일 때 모든 샘플을 사과로 예측한 정확도

③ 0.6 — 귤의 비율

④ 30 — 사과의 개수

⑤ 0.6 — 모델의 최대 정확도

56. 정확도가 처음 1.0 미만이 되는 n_neighbors를 찾는 for 반복문의 올바른 코드는?

① for n in range(5, 49):

```
    kn = KNeighborsClassifier(n)
    score = kn.score(fruit_data, fruit_target)
    if score < 1.0:
        first_drop_n = n; break
```

② for n in range(5, 50):

```
    kn.n_neighbors = n
    score = kn.score(fruit_data, fruit_target)
    if score < 1.0:
        first_drop_n = n
        first_drop_score = score; break
```

③ for n in range(1, 50):

```
    kn.fit(fruit_data, fruit_target)
    if kn.predict(fruit_data) != fruit_target:
        first_drop_n = n; break
```

④ while kn.score(fruit_data, fruit_target) == 1.0:

```
    kn.n_neighbors += 1
first_drop_n = kn.n_neighbors
```

⑤ for n in range(5, 50):

```
    kn.n_neighbors = n
```

```
print(kn.score(fruit_data, fruit_target))
```

57. fruit_data를 만든 후 len(fruit_data)의 결과와 fruit_data[0]의 결과는?

① 50과 [151.0]

② 100과 151.0

③ 50과 [151.0, 14.2]

④ 2와 [151.0, 153.8, ...]

⑤ 50과 (151.0, 14.2)

58. kn.predict()와 kn.score()의 차이점으로 올바른 것은?

① predict()는 훈련을, score()는 예측을 수행한다

② predict()는 예측 결과 배열을 반환하고, score()는 정확도(0~1)를 반환한다

③ predict()는 1차원 입력을, score()는 2차원 입력을 받는다

④ predict()는 분류에만, score()는 회귀에만 사용된다

⑤ 둘은 동일한 기능이며 이름만 다르다

59. 만약 귤이 30개, 사과가 20개였다면 n_neighbors=50으로 훈련한 모델의 정확도는?

① 0.4

② 0.5

③ 0.6

④ 0.8

⑤ 1.0

60. 전체 데이터 1000개 중 990개가 정상, 10개가 불량인 제조 검사 데이터에서, 모든 데이터를 "정상"으로 예측하는 모델의 정확도와 이때의 문제점은?

① 정확도 0.01 — 불량만 맞추므로 정확도가 낮다

② 정확도 0.99 — 높은 정확도지만 불량을 하나도 찾지 못하므로 실무에서 쓸모없다

③ 정확도 0.50 — 랜덤 추측과 같은 성능이다

④ 정확도 0.99 — 매우 우수한 모델이므로 바로 배포해도 된다

⑤ 정확도를 계산할 수 없다

정답 및 해설

1번 정답: ③ 특성(feature)

[해설] 특성(feature)은 데이터를 표현하는 하나의 성질이다. 생선 데이터에서 길이와 무게가 각각 하나의 특성에 해당하며, 모델은 이 특성들을 입력으로 받아 예측을 수행한다.

2번 정답: ② 훈련 - fit()

[해설] 훈련(training)은 알고리즘이 데이터에서 규칙을 찾는 과정이며, 사이킷런에서는 fit() 메서드가 이 역할을 수행한다.

3번 정답: ③ predict()

[해설] predict() 메서드는 훈련된 모델에 새로운 데이터를 전달하여 예측값을 반환한다. 입력은 fit()과 마찬가지로 2차원 배열 형태여야 한다.

4번 정답: ② 0~1, 정확히 맞힌 비율

[해설] score() 메서드는 0에서 1 사이의 값을 반환하며, 1은 모든 데이터를 정확히 맞혔다는 것을 의미한다.

5번 정답: ③ 전체 데이터를 저장해두고, 새 데이터와 가장 가까운 이웃들의 다수결로 예측한다

[해설] k-최근접 이웃 알고리즘은 별도의 학습 과정 없이 전체 데이터를 메모리에 저장한 후, 새 데이터가 들어오면 가장 가까운 k개 이웃의 다수결로 예측한다.

6번 정답: ② 과대적합 / 과소적합

[해설] n_neighbors가 너무 작으면 소수의 이웃에 의존하여 노이즈에 민감해지고(과대적합), 너무 크면 데이터의 세밀한 패턴을 무시하고 전체적인 다수결에 따르게 되어(과소적합) 분류 성능이 떨어진다. 적절한 k값을 찾는 것이 모델 성능의 핵심이다.

7번 정답: ③ 모든 입력을 도미(다수 클래스)로만 예측한다

[해설] n_neighbors=49이면 전체 49개 데이터를 모두 참고하므로, 도미 35개가 항상 다수를 차지하여 어떤 데이터를 넣어도 무조건 도미로 예측한다. 정확도는 35/49 ≈ 0.714가 된다.

8번 정답: ② 데이터가 많으면 메모리 사용량과 예측 시간이 크게 증가한다

[해설] KNN은 전체 데이터를 메모리에 저장하므로 데이터가 크면 메모리가 많이 필요하고, 새 데이터마다 모든 훈련 데이터와의 거리를 계산해야 하므로 예측 시간도 크게 증가한다.

9번 정답: ② 과대적합 — 모델이 훈련 데이터를 외워버려 새로운 데이터에 일반화하지 못한다

[해설] 훈련 정확도는 높지만 테스트 정확도가 크게 낮은 것은 과대적합(overfitting)의 전형적인 증상이다. 모델이 훈련 데이터의 노이즈까지 학습하여 새로운 데이터에 대한 일반화 능력이 떨어진 것이며, 실무에서 가장 흔하게 마주치는 문제 중 하나이다.

10번 정답: ③ 2차원 리스트(행: 샘플, 열: 특성)

[해설] 사이킷런은 입력 데이터를 2차원 배열(행: 샘플, 열: 특성)로 기대한다.

11번 정답: ② fish_target = [1] * 35 + [0] * 14

[해설] [1] * 35는 1이 35개인 리스트를, [0] * 14는 0이 14개인 리스트를 생성한다. + 연산자로 합치면 정답 리스트가 만들어진다.

12번 정답: ② [[l, w] for l, w in zip(length, weight)]

[해설] zip() 함수는 여러 리스트에서 원소를 하나씩 꺼내 짝을 만들고, 리스트 내포를 사용하면 2차원 리스트를 손쉽게 생성할 수 있다.

13번 정답: ② 데이터의 분포, 이상치, 클래스 간 분리 가능성을 시각적으로 파악하기 위해

[해설] 모델링 전에 산점도를 그려보면 데이터의 전체적인 분포, 이상치(outlier) 존재 여부, 클래스 간 경계가 명확한지 등을 직관적으로 파악할 수 있다. 이를 통해 적절한 전처리 방법과 알고리즘을 선택하는 데 도움이 된다.

14번 정답: ② 입력(특성)과 타깃(정답)

[해설] 지도 학습은 입력(특성 데이터)과 타깃(정답)을 함께 전달하여 모델을 훈련한다.

15번 정답: ② 정답 없이 입력 데이터만으로 데이터의 패턴이나 구조를 파악한다

[해설] 비지도 학습은 타깃(정답) 없이 입력 데이터만 사용하여 데이터의 패턴이나 구조를 파악한다.

16번 정답: ③ 정답을 미리 알고 시험을 보는 것과 같아 실제 성능을 알 수 없기 때문

[해설] 훈련 데이터로 평가하면 모델이 이미 정답을 외우고 있어 높은 점수가 나오지만, 새로운 데이터에 대한 실제 성능은 알 수 없다.

17번 정답: ③ 샘플링 편향(sampling bias)

[해설] 샘플링 편향(sampling bias)은 훈련/테스트 세트의 클래스 비율이 원본 데이터와 다르게 치우치는 현상이다.

18번 정답: ② 테스트 세트에 빙어만 있고 훈련 세트에 빙어가 없어서

[해설] 데이터를 순서대로 나누면 앞 35개(도미)가 훈련 세트, 뒤 14개(빙어)가 테스트 세트가 된다. 훈련 세트에 빙어가 없으므로 테스트 정확도가 0.0이 된다.

19번 정답: ② 49개의 샘플, 2개의 특성

[해설] shape (49, 2)는 49개의 행(샘플)과 2개의 열(특성)을 의미한다.

20번 정답: ② a * 2는 [1, 2, 3, 1, 2, 3], b * 2는 [2, 4, 6]을 반환한다

[해설] 파이썬 리스트에서 * 연산자는 리스트를 반복(repeat)하고, 넘파이 배열에서는 각 원소에 곱셈을 수행(원소별 연산)한다. 이 차이를 이해하지 못하면 데이터 처리 시 예상과 다른 결과가 나올 수 있으므로, 수치 연산에는 넘파이 배열을 사용하는 것이 일반적이다.

21번 정답: ② arr[[3, 7, 12]]

[해설] 넘파이의 배열 인덱싱을 사용하면 여러 개의 인덱스를 리스트로 전달하여 한 번에 여러 원소를 선택할 수 있다.

22번 정답: ③ 0, 1, 2, 3, 4

[해설] 파이썬 슬라이싱에서 [start:end]는 start부터 end-1까지 선택한다.

23번 정답: ② train_input[:,0]

[해설] train_input[:,0]에서 ':'는 모든 행을 선택하고, 0은 첫 번째 열을 선택한다.

24번 정답: ② np.column_stack()

[해설] np.column_stack()은 전달받은 1차원 배열들을 세로로 세운 다음 나란히 연결하여 2차원 배열을 만든다.

25번 정답: ② concatenate는 첫 번째 축(행)을 따라 연결하고, column_stack은 두 번째 축(열)을 따라 나란히 연결한다

[해설] np.concatenate()는 기본적으로 첫 번째 축(axis=0)을 따라 배열을 이어붙이고, np.column_stack()은 1차원 배열들을 열 방향으로 나란히 쌓아 2차원 배열을 만든다.

26번 정답: ② 데이터 누수(data leakage) — 실제보다 높은 성능이 나타나 모델을 신뢰할 수 없다

[해설] 데이터 누수는 테스트 세트의 정보가 훈련 과정에 유입되는 것을 말한다. 예를 들어 전처리 시 전체 데이터의 평균/표준편차를 사용하면 테스트 세트 정보가 훈련에 반영되어 실제보다 높은 성능이 나타난다. 실무에서 모델 배포 후 성능이 떨어지는 주요 원인 중 하나이다.

27번 정답: ③ 클래스 비율에 맞게 훈련/테스트 세트를 나눈다

[해설] stratify 매개변수에 타깃 배열을 전달하면, 원본 데이터의 클래스 비율이 훈련/테스트 세트에도 동일하게 유지된다.

28번 정답: ③ 4개

[해설] train_test_split()에 2개의 배열(입력, 타깃)을 전달하면 각각 훈련/테스트로 나뉘어 총 4개의 배열이 반환된다.

29번 정답: ② 동일한 값을 주면 매번 같은 방식으로 데이터를 나눈다

[해설] random_state는 난수 시드를 고정하는 매개변수이다. 동일한 값을 지정하면 매번 같은 방식으로 데이터가 분할되어 실험 결과를 재현할 수 있다.

30번 정답: ③ 두 특성(길이, 무게)의 스케일이 달라 무게가 거리 계산을 지배해서

[해설] 길이(10~40 범위)와 무게(0~1000 범위)의 스케일이 크게 다르면, 거리 계산 시 무게 차이만 반영된다.

31번 정답: ② (특성값 - 평균) / 표준편차

[해설] 표준점수는 (특성값 - 평균) / 표준편차로 계산한다.

32번 정답: ③ 훈련 세트의 평균과 표준편차를 사용한다

[해설] 테스트 세트는 반드시 훈련 세트의 평균과 표준편차로 변환해야 한다.

33번 정답: ② axis=0

[해설] axis=0은 행(샘플) 방향으로 연산을 수행하여 각 열(특성)의 통계값을 구한다.

34번 정답: ③ 크기가 다른 배열 간에 자동으로 사칙연산을 확장하여 수행하는 기능

[해설] 브로드캐스팅은 크기가 다른 넘파이 배열에서 자동으로 사칙 연산을 확장하여 수행하는 기능이다.

35번 정답: ③ 이웃까지의 거리와 이웃 샘플의 인덱스

[해설] kneighbors() 메서드는 주어진 샘플에서 가장 가까운 이웃을 찾아, 이웃까지의 거리와 이웃 샘플의 인덱스를 반환한다.

36번 정답: ③ 이전에 학습한 모든 것을 잃고 새로운 데이터로 다시 훈련된다

[해설] fit() 메서드를 다시 실행하면 이전에 학습한 모든 것을 잃어버리고 새로운 데이터로 처음부터 다시 훈련된다.

37번 정답: ③ 특성 간 스케일 차이가 거리 계산 결과를 왜곡하기 때문

[해설] 거리 기반 알고리즘은 유클리디안 거리를 계산하므로, 스케일이 큰 특성이 거리를 지배한다.

38번 정답: ② 실험 결과를 동료와 공유·검증하고, 디버깅 시 동일한 조건을 재현하기 위해

[해설] 재현성은 같은 코드와 데이터로 동일한 결과를 얻을 수 있는 것을 의미한다. 이는 동료 리뷰, 문제 발생 시 디버깅, 모델 비교 실험 등 실무에서 필수적이다. random_state나 seed 값을 고정하는 것은 재현성을 확보하는 대표적인 방법이다.

39번 정답: ③ 20~30%

[해설] 일반적으로 전체 데이터의 20~30%를 테스트 세트로 사용한다. train_test_split()의 test_size 기본값은 0.25(25%)이다.

40번 정답: ② (3, 2)

[해설] arr는 3개의 행과 2개의 열을 가진 2차원 배열이므로 shape는 (3, 2)이다.

41번 정답: ② 무게(g)와 당도(brix)

[해설] 이번 과제에서는 각 과일의 무게(g)와 당도(brix)를 두 개의 특성으로 사용하였다.

42번 정답: ② fruit_target = [1] * 30 + [0] * 20

[해설] [1] * 30은 1이 30개 반복된 리스트를 만들고, [0] * 20은 0이 20개 반복된 리스트를 만든다. 이 둘을 + 연산으로 합치면 길이 50의 타깃 리스트가 완성된다.

43번 정답: ② fruit_data = [[w, s] for w, s in zip(weight, sugar)]

[해설] zip(weight, sugar)은 두 리스트를 짝지어 주고, 리스트 컴프리헨션으로 [w, s] 형태로 묶으면 사이킷런 fit()의 입력 형식인 2차원 리스트가 완성된다.

44번 정답: ④ 1.0

[해설] 사과(무게 150~300g)와 귤(무게 50~120g)은 무게 범위가 겹치지 않아 n_neighbors=5일 때 훈련 데이터를 완벽하게 분류한다. 따라서 정확도는 1.0이다.

45번 정답: ② 전체 50개 이웃을 참고하므로 항상 다수 클래스(사과 30개)로만 예측하기 때문에

[해설] n_neighbors=50이면 전체 50개 샘플을 모두 이웃으로 사용한다. 다수결 투표 시 사과(30표) > 귤

(20표)이므로 모든 샘플을 사과로 예측한다. 실제 사과가 30개이므로 정확도 = 30/50 = 0.6이다.

46번 정답: ③ 36

[해설] n_neighbors=35까지는 정확도 1.0을 유지하지만, 36이 되는 순간 처음으로 1.0 미만으로 떨어진다. 이웃 수가 너무 많아지면 다수결이 왜곡되어 일부 샘플이 잘못 분류되기 시작한다.

47번 정답: ② [1] — 사과로 예측

[해설] 무게 200g은 사과 범위(150~300g), 당도 13도 사과 범위(11~15)에 해당한다. 가장 가까운 5개 이웃이 모두 사과이므로 [1](사과)로 예측한다.

48번 정답: ② [0] — 귤로 예측

[해설] 무게 80g은 귤 범위(50~120g), 당도 10도 귤 범위(8~14)에 해당한다. 가장 가까운 5개 이웃이 모두 귤이므로 [0](귤)으로 예측한다.

49번 정답: ③ kn.n_neighbors = n

[해설] KNeighborsClassifier 객체의 n_neighbors 속성에 직접 값을 대입하여 변경할 수 있다. 새 객체를 생성하지 않고도 하이퍼파라미터를 바꿀 수 있어 for문 실험에 유용하다.

50번 정답: ② 사과 30개와 귤 20개의 데이터를 합쳤기 때문에

[해설] apple_weight(30개) + mandarin_weight(20개) = 50개. sugar도 마찬가지이므로 fruit_data와 fruit_target 모두 길이 50이다.

51번 정답: ② 두 리스트를 이어붙인 길이 50의 리스트

[해설] 파이썬에서 리스트의 + 연산자는 두 리스트를 이어붙인다(concatenate). apple_weight(30개) + mandarin_weight(20개)는 총 50개 원소를 가진 새 리스트를 반환한다. 넘파이 배열과 달리 원소별 덧셈이 아니다.

52번 정답: ② kn = KNeighborsClassifier()

[해설] KNeighborsClassifier()를 매개변수 없이 호출하면 n_neighbors=5가 기본값으로 설정된다. k=5처럼 잘못된 매개변수명을 사용하면 오류가 발생한다.

53번 정답: ② fruit_data로 예측한 결과와 fruit_target을 비교하여 정확도를 계산한다

[해설] score() 메서드는 내부적으로 fruit_data에 대해 predict()를 수행한 뒤, 그 결과를 fruit_target과 비교하여 정확히 맞힌 비율(정확도)을 반환한다.

54번 정답: ② predict()는 2차원 배열(샘플 × 특성)을 입력으로 요구하기 때문에

[해설] predict()는 fit()과 동일하게 2차원 배열을 입력으로 기대한다. [[200, 13]]은 1개 샘플 × 2개 특성의 2차원 형태이고, [200, 13]은 1차원이므로 오류가 발생한다.

55번 정답: ② 0.6 — n_neighbors=50일 때 모든 샘플을 사과로 예측한 정확도

[해설] 30/50 = 0.6은 n_neighbors=50일 때의 예상 정확도이다. 모든 샘플을 사과(다수 클래스)로 예측하면, 실제 사과 30개만 맞추고 귤 20개는 틀리므로 정확도 = 0.6이 된다.

56번 정답: ② for n in range(5, 50):

[해설] range(5, 50)으로 5~49까지 반복하며, kn.n_neighbors = n으로 속성을 직접 변경한다. score()로 정확도를 확인하고, 1.0 미만이면 저장 후 break로 종료한다.

57번 정답: ③ 50과 [151.0, 14.2]

[해설] fruit_data는 50개의 [무게, 당도] 쌍으로 구성된 2차원 리스트이므로 len(fruit_data) = 50이다. fruit_data[0]은 첫 번째 사과의 [무게, 당도] = [151.0, 14.2]를 반환한다.

58번 정답: ② predict()는 예측 결과 배열을 반환하고, score()는 정확도(0~1)를 반환한다

[해설] predict()는 예측된 클래스 레이블 배열(예: [1, 0, 1, ...])을 반환한다. score()는 입력 데이터와 정답을 함께 받아 정확도(0~1)를 반환한다.

59번 정답: ③ 0.6

[해설] n_neighbors=50이면 전체를 참고하므로 다수 클래스인 귤(30개)로 모든 샘플을 예측한다. 실제 귤 30개만 맞추므로 정확도 = 30/50 = 0.6이다. 다수 클래스 비율이 곧 정확도가 되는 원리는 동일하다.

60번 정답: ② 정확도 0.99 — 높은 정확도지만 불량을 하나도 찾지 못하므로 실무에서 쓸모없다

[해설] 990/1000 = 0.99로 높은 정확도이지만, 정작 찾아야 할 불량품을 하나도 탐지하지 못한다. 이처럼 클래스 불균형이 심한 데이터에서는 정확도만으로 모델을 평가하면 위험하며, 정밀도(precision)·재현율(recall) 등 다른 지표도 함께 확인해야 한다.