파머완 2장, 3장.

2-1 사이킷런 소개와 특징

사이킷런: 파이썬 머신러닝 라이브러리 중 가장 많이 사용되는 라이브러리.

☞ 특징: 쉽고 가능 파이썬스러운 API, 다양한 알고리즘과 프레임워크, 성숙한 라이브러리.

2-2 첫 번째 머신러닝 만들기

분류(classification) :대표적인 지도학습(supervised learning) 중 하나의 방법.

지도 학습이란?

다양한 피처와 레이블 데이터로 모델을 학습한 뒤, 별도의 테스트 데이터 세트에서 미지의 레이블을 '예측'하도록 하는 것.⇒ 명확한 정답이 주어진 데이터를 먼저 학습시키고(학습 데이터 세트), 미지의 정답을 예측하는 방식(테스트 데이터 세트) 을 말함.

1. 사이킷런에서 사용할 모듈 임포트 하기.

- sklearn으로 시작하는 명명규칙이 있음. 우리는 의사 결정 트리인 load_iris()를 사용할 것임.
- 학습데이터와 테스트 데이터 분리하기

```
X_train,X_test, y_train, y_test = train_test_split(iris_data,  iris_label,
test_size=0.2, random_state=11)
```

`iris_data:` 피터 데이터 세트

`iris_label:`레이블 데이터 세트

`test_size=0.2 :` 전체 데이터 세트 중 테스트 데이터 세트의 비율

`random_state:`호출할 때마다 같은 학습/테스트 용 데이터 세트를 생성하기 위해 주어지는 난수 발생 값 (이를 지정하지 않으면 수행할 때마다 다른 학습/테스트 용 데이터를 만들 수 있음.동일한 데이터 세트로 분리하기 위해 일정한 숫자 값으로 부여하는 것이 필요함.

2. 분류예측 프로세스

`DecisionTreeClassifier:` ML 알고리즘으로 의사 결정 트리(Decision Tree)

`fit()` : 학습 데이터로 학습 시키기

`predict()`: 예측하기

`accuracy_score()`: 정확도를 평가

2-3 사이킷런의 기반 프레임워크 익히기

1. Estimator(추정기)란?

- 사이킷런에서 분류(`Classifier`) 및 회귀(`Regressor`) 알고리즘을 구현한 모든 클래스를 통칭하는 용어.
- Estimator 클래스는 학습을 위한 `fit()` 메서드와 예측을 위한 `predict()` 메서드를 제공합니다. 비지도학습 클래스는 `fit()`과 `transform()`을 적용하며, 이 둘을 결합한 `fit_transform()`도 제공.

2. 사이킷런의 주요 모듈

모듈명	기능
`sklearn.datasets`	사이킷런에 내장되어 예제로 제공하는 데이터 세트.
`sklearn.preprocessing`	데이터 전처리에 필요한 다양한 가공 기능 제공 (예: 문자열 인코딩, 정규화, 스케일링).
`sklearn.model_selection`	데이터 분리 (학습/테스트), 교차 검증 분할 및 평가, 그리드 서치(Grid Search)로 최적 파라미터 추출 등.
`sklearn.metrics`	분류, 회귀, 클러스터링 등에 대한 다양한 성능 측정 방법 제공 (예: Accuracy, Precision, Recall, ROC-AUC, RMSE).
`sklearn.tree`	의사 결정 트리 알고리즘 제공.
`sklearn.ensemble`	앙상블 알고리즘 제공 (예: 랜덤 포레스트, 그래디언트 부스팅).
`sklearn.linear_model`	주로 선형 회귀, 릿지, 라쏘 및 로지스틱 회귀 등 회귀 관련 알고리즘 지원.
`sklearn.pipeline`	피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공.

2-4 Model Selection (모델 선택) 학습/테스트 데이터 세트 분리

모델이 학습 데이터에만 과도하게 최적화되어 실제 예측 성능이 떨어지는 **과적합 (Overfitting)** 문제를 방지하기 위해 중요.

1. **`train_test_split()`** : 원본 데이터를 학습 및 테스트 데이터 세트로 쉽게 분리합니다.

2.교차 검증(Cross-validation): 고정된 테스트 데이터로 평가 시 발생할 수 있는 편향 문제를 개선하고, 더 다양한 데이터를 기반으로 예측 성능을 평가하기 위해 여러 번 학습과 평가를 수행합니다.

- **K-Fold 교차 검증**: 전체 데이터를 K개의 폴드 세트로 분리하고, K번의 학습과 검증을 반복하여 평균 평가 결과를 반영.

- `KFold` 클래스를 제공

-문제점: 레이블 데이터의 분포를 고려하지 않아, 원본 데이터의 레이블 분포가 불균형할 경우 학습/테스트 세트에 제대로 분배하지 못할 수 있음.

- **Stratified K-Fold 교차 검증**: `KFold`의 문제점을 해결하기 위해 원본 데이터의 레이블 분포를 먼저 고려한 뒤, 이 분포와 동일하게 학습과 검증 데이터 세트를 분배합니다.

- `StratifiedKFold` 클래스를 제공합니다.

-중요성: 분류(Classification) 문제에서는 불균형한 레이블 데이터 세트의 왜곡을 방지하기 위해 일반적으로 Stratified K-Fold를 사용. 회귀(Regression)에서는 지원x

- **cross_val_score() API**: 교차 검증을 편리하게 수행하는 사이킷런 API입니다.

-`estimator`, `X`, `y`, `scoring`, `cv`를 주요 파라미터로 받으며, 내부적으로 분류 시 `Stratified K-Fold`를 사용하여 학습/테스트 세트를 분할

- GridSearchCV - 교차 검증과 최적 하이퍼 파라미터 튜닝:

-개념: 교차 검증을 기반으로 하이퍼 파라미터의 최적 값을 찾아줍니다. `param_grid`에 기술된 모든 파라미터 조합을 순차적으로 적용하여 최적의 파라미터를 탐색합니다.

-주요 파라미터: `estimator`, `param_grid`, `scoring`, `cv`, `refit`.

-주요 속성: `best_params_` (최적 파라미터), `best_score_` (최고 정확도), `best_estimator_` (최적 파라미터로 재학습된 Estimator)

2-5 데이터 전처리

데이터 전처리는 ML 알고리즘만큼 중요하며, 입력 데이터의 품질이 결과에 큰 영향을 미침.

1. 결손값(NaN, Null 값) 처리: ML 알고리즘은 결손값을 허용하지 않으므로, 평균값 대체, 피처 드롭 등 적절한 방법으로 처리
2. 문자열 값 처리: 모든 문자열 값은 인코딩되어 숫자형으로 변환

- 카테고리형 피처는 코드 값으로, 텍스트형 피처는 피처 벡터화(feature vectorization) 기법으로 변환하거나 삭제

3. 피처 스케일링(Feature Scaling)과 정규화(Normalization): 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업

- 표준화(Standardization): 데이터 피처를 평균 0, 분산 1인 가우시안 정규 분포로 변환

- `StandardScaler` 클래스가 이를 지원하며, 가우시안 분포를 가정하는 알고리즘(예: 선형 회귀, 로지스틱 회귀)에서 중요

- 정규화(Normalization) (0~1 범위): 피처의 크기를 0에서 1 사이로 변환

- `MinMaxScaler` 클래스가 이를 지원합니다.

*학습/테스트 데이터 스케일링 시 유의점: 학습 데이터로 fit()을 수행하고, 이 fit()된 Scaler 객체로 학습 데이터와 테스트 데이터 모두에 transform()을 적용해야 함. 테스트 데이터에 `fit()`이나 `fit_transform()`을 직접 적용하면 안 됨.

3-1 정확도(Accuracy)

- **정의**: 실제 데이터에서 예측 데이터가 얼마나 같은지를 판단하는 지표.
- **공식**: `정확도 = (예측 결과가 동일한 데이터 건수) / (전체 예측 데이터 건수)`
- **한계점**: **불균형한 레이블 데이터 세트**에서는 정확도 수치가 모델의 실제 성능을 왜곡할 수 있으므로, 정확도 하나만으로 성능을 평가하지 않아야 함. 예를 들어, 0이 90%, 1이 10%인 데이터에서 모든 예측을 0으로 해도 90%의 정확도를 보일 수 있음.

3-2 오차 행렬(Confusion Matrix)

오차 행렬(confusion matrix)은 이진 분류에서 예측 오류의 양과 유형을 함께 나타내는 지표. 실제 클래스 값과 예측 클래스 값의 매핑에 따라 다음 4분면으로 구성:

- **TN (True Negative)**: 실제 Negative를 Negative로 정확히 예측.
- **FP (False Positive)**: 실제 Negative를 Positive로 잘못 예측 (오탐).
- **FN (False Negative)**: 실제 Positive를 Negative로 잘못 예측 (미탐).
- **TP (True Positive)**: 실제 Positive를 Positive로 정확히 예측.

`confusion_matrix()` API를 사용하여 오차 행렬을 계산ok

3-3정밀도(Precision)와 재현율(Recall)

정밀도와 재현율은 Positive 데이터 세트의 예측 성능에 초점을 맞춘 평가 지표

1.정밀도(Precision):

◦ **의미**: **예측을 Positive로 한 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율.**

◦ **공식**: `정밀도 = TP / (FP + TP)`.

◦ **별칭**: 양성 예측도.

2. **재현율(Recall)**:

◦ **의미**: **실제 값이 Positive인 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율.**

◦ **공식**: `재현율 = TP / (FN + TP)`.

◦ **별칭**: 민감도(Sensitivity), TPR(True Positive Rate).

- **중요성**: 업무 특성에 따라 재현율 또는 정밀도가 더 중요하게 간주될 수 있습니다. 예를 들어, 암 진단과 같이 **실제 Positive를 Negative로 잘못 판단하는 것이 치명적인 경우(FN 감소)에는 재현율이 더 중요**합니다.
- **precision_score() 및 recall_score() API**를 사이킷런에서 제공

3. **정밀도/재현율 트레이드오프(Trade-off)**: 정밀도와 재현율은 상호 보완적이어서, 어느 한쪽을 높이면 다른 쪽은 낮아지기 쉬움

*분류 결정 임곗값(Threshold)을 조정하여 이 관계를 조절

`predict_proba()` 메서드는 각 클래스에 대한 예측 확률을 반환합니다.

`Binarizer` 클래스를 사용하여 임곗값에 따른 예측 클래스 값을 직접 조작할 수 있음. 임곗값을 낮추면 Positive 예측이 많아져 재현율이 증가하고 정밀도는 감소합니다.

`precision_recall_curve()` API: 임곗값 변화에 따른 정밀도와 재현율 값을 배열로 반환하며, 이를 통해 정밀도-재현율 곡선을 시각화할 수 있음.

< 평가 관련 유틸리티 함수 표 >

함수명	기능
`get_clf_eval(y_test, pred, pred_proba=None)`	실제 레이블 `y_test`, 예측 레이블 `pred`, (선택적으로) 예측 확률 `pred_proba`를 입력받아 오차 행렬, 정확도, 정밀도, 재현율, F1 스코어, 그리고 `pred_proba`가 제공되면 ROC AUC 값까지 출력함
`precision_recall_curve_plot(y_test, pred_proba_c1)`	실제 레이블 `y_test`와 Positive 클래스 예측 확률 `pred_proba_c1`을 입력받아 임곗값에 따른 정밀도와 재현율 곡선을 그래프로 시각화함

3-4 F1스코어

F1 스코어(F1 Score) F1 스코어는 정밀도와 재현율을 결합한 지표

정밀도와 재현율이 한쪽으로 치우치지 않고 균형을 이룰 때 상대적으로 높은 값을 가짐.

- **공식**: `F1 스코어 = 2 * (Precision * Recall) / (Precision + Recall)`.
- `f1_score()` API를 제공.
- 불균형한 데이터 세트에서 모델의 전반적인 성능을 평가하는 데 유용.

```
# F1 스코어 핵심 코드 예시
from sklearn.metrics import f1_score
# f1 = f1_score(y_test, pred)
# print(f'F1 스코어: {f1:.4f}')
```

3-5 ROC 곡선(ROC Curve)과 AUC(Area Under Curve)

이진 분류의 예측 성능 측정에서 중요하게 사용되는 지표

1. **ROC 곡선**: **FPR(False Positive Rate)이 변할 때 TPR(True Positive Rate)이 어떻게 변하는지를 나타내는 곡선**
- **FPR**: `FP / (FP + TN)`이며, 1 - 특이성(TNR)과 같습니다.
- **TPR**: 재현율(Recall)과 동일하며 민감도(Sensitivity)라고도 불립니다.
- 분류 결정 **임곗값을 1부터 0까지 변화**시키면서 FPR과 TPR 값을 계산하여 ROC 곡선을 그림.
- ROC 곡선이 가운데 대각선(랜덤 분류, AUC 0.5)에서 멀어져 **왼쪽 상단 모서리(0,1) 쪽으로 가파르게 이동할수록 좋은 성능!**

2. **AUC (Area Under Curve)**:

- **정의**: **ROC 곡선 밑의 면적**을 구한 값.
- **성능 해석**: 일반적으로 **1에 가까울수록 좋은 성능.**
- AUC 수치가 커지려면 FPR이 작은 상태에서 얼마나 큰 TPR을 얻을 수 있느냐가 중요. 랜덤 분류의 AUC는 0.5.
- `roc_curve()` API는 FPR, TPR, 임곗값을 반환하여 ROC 곡선을 그리는 데 사용
- `roc_auc_score()` API는 ROC AUC 값을 계산하는 데 사용.