📌 머신러닝(Machine Learning) 개요

1. 머신러닝이란?

머신러닝은 **데이터를 기반으로 패턴을 학습**하고 애플리케이션을 직접 수정하지 않고도 **예측과 의사결정**을 가능하게 하는 알고리즘 기법
즉 데이터를 통해 숨겨진 규칙을 찾아내고 이를 바탕으로 새로운 결과를 예측하는 기술.

2. 머신러닝의 주요 유형

머신러닝은 크게 세 가지로 구분

1) 지도학습

- **라벨(정답)이 있는 데이터**로 학습
- 주요 기법:
  - 분류
  - 회귀
  - 추천시스템
  - 이미지/음성 인식
  - 텍스트 분석, NLP

2) 비지도학습

- **라벨 없는 데이터**에서 패턴을 찾는 방식
- 주요 기법:
  - 클러스터링
  - 차원축소

3) 강화학습

- **행동 → 보상 → 학습** 구조
- 게임, 로보틱스, 자율주행 등에 활용

3. 파이썬 기반 머신러닝 패키지

- 핵심 라이브러리

- **Scikit-Learn** : 가장 널리 쓰이는 머신러닝 라이브러리
- **TensorFlow, Keras** : 딥러닝(영상, 음성, 언어 등 비정형 데이터)에 강력

- 수학/통계 패키지

- **NumPy** : 행렬·선형대수 연산
- **SciPy** : 과학·통계 계산 지원

- 데이터 처리

- **Pandas** : 테이블 형태(2차원) 데이터 처리에 특화

- 시각화

- **Matplotlib** : 파이썬 기본 시각화 라이브러리
- **Seaborn** : Matplotlib 기반, 더 직관적이고 세련된 그래프 제공

📌 넘파이(NumPy) 기초 정리

1. 넘파이란?

- **NumPy (Numerical Python)** : 파이썬의 대표적인 **선형대수·수치연산 라이브러리**
- 특징
  - **빠른 배열 연산** (루프 없이 대량 데이터 처리 가능)
  - C/C++ API 연동 지원 → 다른 프로그램과 쉽게 통합 가능
  - 딥러닝 프레임워크(예: TensorFlow)에서도 핵심 연산에 활용

2. ndarray (다차원 배열)

- 넘파이의 기본 데이터 타입 → **ndarray**
- 생성 : np.array([리스트])
- 크기/차원 확인 : .shape 속성

3. 데이터 타입 (dtype)

- 지원 타입: int, uint, float, complex 등
- 하나의 ndarray 안에는 **같은 데이터 타입만 가능**
- 확인: .dtype
- 변환: .astype(새로운타입)

=> 대용량 데이터 처리 시 **int → float**

또는 그 반대 변환으로 **메모리 절약** 가능

4. 배열 생성 함수

- **arange(start, stop, step)** : range와 유사, 연속값 생성
- **zeros(shape)** : 0으로 채운 배열 생성
- **ones(shape)** : 1로 채운 배열 생성

5. 배열 크기/차원 변경

- **reshape(새로운 shape)** : 원하는 차원/크기로 변경

6. 인덱싱(Indexing) & 슬라이싱(Slicing)

넘파이는 다양한 방식으로 배열 요소를 선택할 수 있음

1. **단일 인덱싱** : arr[2]
2. **슬라이싱** : arr[1:5] (1~4까지 추출)
    - 시작/끝 생략 가능 → arr[:3], arr[2:]

팬시 인덱싱 : arr[[0,2,4]] (특정 인덱스 집합 선택)

불린 인덱싱 : 조건 필터링

7. 정렬 (Sort)

- **np.sort(arr)** : 원본 보존, 정렬된 배열 반환
- **arr.sort()** : 원본 배열 자체를 정렬
- 내림차순: arr[::-1]
- **np.argsort(arr)** : 정렬된 인덱스 반환 (메타데이터 매칭 시 활용)

8. 선형대수 연산

- **행렬 내적 (Matrix Multiplication)**
    - np.dot(A, B)

전치 행렬 (Transpose)

- A.T 또는 np.transpose(A)

📌판다스(Pandas) 데이터 핸들링 기초 정리

1. 판다스란?

- 파이썬에서 가장 널리 쓰이는 **데이터 처리 라이브러리**
- 데이터는 보통 **행(Row) × 열(Column) 구조**
- **핵심 객체**
    - Series: 1차원, 하나의 컬럼만 가짐
    - DataFrame: 2차원, 여러 컬럼으로 구성
    - Index: 각 행을 고유하게 식별하는 키 값

2. 데이터 로딩 & 확인

- CSV → DataFrame
- **기본 확인 메서드**
    - df.head(3) : 앞 3개 행 미리보기 (default는 5개)
    - df.shape : (행 개수, 열 개수)
    - df.info() : 전체 데이터 건수, 컬럼 타입, Null 개수
    - df.describe() : 숫자형 데이터의 통계 요약 (평균, 표준편차, 분포 등)

3. 데이터 선택 (Indexing & Selection)

- **DataFrame[칼럼명]** → 특정 칼럼 선택 (Series 반환)
- **불린 인덱싱**
- **iloc[행, 열]** → 위치 기반 인덱싱 (숫자 좌표)
- **loc[행, 열]** → 명칭 기반 인덱싱 (Index/컬럼 이름)

4. 데이터 가공

- **새로운 칼럼 추가**
- df['Age_0'] = 0
- **값 수정**
- df['Age'] = df['Age'] + 1

- **데이터 삭제**
    - df.drop('Age_0', axis=1) → 컬럼 삭제
    - df.drop(0, axis=0) → 첫 번째 행 삭제
    - inplace=True → 원본에 바로 적용

5. 정렬 & 통계

- **정렬**
- df.sort_values(by='Age', ascending=False)
- **Aggregation 함수**df['Age'].mean(), df['Fare'].sum() 등
- **GroupBy** df.groupby('Pclass')['Age'].mean()

6. 결손 데이터 처리

- **결손값 확인 :** df.isna().sum() → NaN 개수 확인
- **결손값 대체 :** df['Age'].fillna(df['Age'].mean(), inplace=True)

7. value_counts() — 범주형 데이터 분석

- **칼럼 값 분포 확인**
- df['Pclass'].value_counts()
- **기본 옵션**
    - 내림차순 정렬
    - dropna=False 지정 시 NaN 포함

8. 인덱스 다루기

- **확인**: df.index
- **리셋**: df.reset_index(drop=True) → 새 인덱스 생성
- **Series도 동일** (단, drop=False이면 기존 인덱스를 새 칼럼으로 추가)

9. apply & lambda

- **데이터 가공**
- df['Age_group'] = df['Age'].apply(lambda x: 'Child' if x <= 15 else 'Adult')
- **주의**: lambda는 if-else만 지원 (elif는 불가)

✨ 총 정리 ✨

1. NumPy

- NumPy는 머신러닝·과학연산 필수 라이브러리
- 핵심 데이터 구조: ndarray
- 주요 기능
    - 빠른 배열 연산
    - 다양한 생성 함수 (arange, zeros, ones)
    - 강력한 인덱싱 & 슬라이싱 (특히 불린 인덱싱)
    - 정렬 & 선형대수 연산

2. Pandas

- 판다스 핵심: DataFrame
- 자주 쓰는 기능

    1. 데이터 로딩 (read_csv)
    2. 구조 확인 (head, shape, info, describe)
    3. 인덱싱 (iloc, loc, 불린 인덱싱)
    4. 데이터 가공 (추가, 수정, 삭제)
    5. 통계 분석 (sort_values, groupby, agg)
    6. 결손 데이터 처리 (isna, fillna)
    7. 범주형 데이터 분석 (value_counts)
    8. apply + lambda로 데이터 가공