Week 1

Chapter 01: 머신러닝의 개념

머신러닝 (Machine Learning)이란?

- 애플리케이션을 수정하지 않고도 데이터를 기반으로 패턴을 학습하고 결과를 예측하는 알고리즘 기법
- 예: 금융 사기 거래를 적발하는 프로그램
- 따라서 데이터 분석 영역은 머신러닝 기반의 예측 분석 (Predictive Analysis)으로 재편되고 있음

머신러닝의 분류

일반적으로 3개로 나뉜다:

> 💡 1. 지도학습 (Supervised Learning)

- 분류 (대표!)
- 회귀 (대표!)
- 추천 시스템
- 시각/음성 감지/인지
- 텍스트 분석, NLP

> 💡 2. 비지도학습 (Un-supervised Learning)

- 클러스터링
- 차원 축소
- 강화학습

💡 3. 강화학습 (Reinforcement Learning)

머신러닝의 단점

- 데이터에 매우 의존적임
- 가비지 인(Garbage In), 가비지 아웃(Garbage out) = “좋은 품질의 데이터를 갖추지 못한다면 머신러닝의 수행 결과도 좋을 수 없다”

02 파이썬 머신러닝 생태계를 구성하는 주요 패키지

- 머신러닝 패키지 → 사이킷런(Scikit-Learn)
- 행렬/선형대수/통계 패키지 → 넘파이(NumPy)
- 데이터 핸들링 → 판다스 (데이터 처리 패키지, 특히 2차원)
- 시각화: 맷플롯립 & 시본(Seaborn)
- 아이파이썬(대화형 파이썬 툴) → 주피터 노트북

03 넘파이

- 파이썬에서 선형대수 기반의 프로그램을 지원하는 대표 패키지
- 빠른 배열 연산 속도를 보장

💡 넘파이 ndarray 개요

- 넘파이의 기반 데이터 타입은 ndarray임
- ndarray를 이용해 넘파이에서 다파원 배열을 생성하고 연산을 수행
  - 1차원 배열, 2차원 배열, 3차원 배열 등
- 넘파이 array() 함수는 리스트와 같은 인자를 입력 받아서 ndarray로 변환
- shape 변수는 ndarray의 크기

```
array1 = np.array([1,2,3])
array3 = np.array([[1,2,3]])
```

- array1과 array3의 차이점은:

    - array1 = 1차원 array로 3개의 데이터 가지고 있음
    - array1.shape = (3,)
    - array3 = 2차원 데이터
    - array3.shape = (1,3)
- 데이터값으로는 서로 동일하나 차원이 달라서 오류 발생 할 수 있음
    - 해결법: 차원의 차수를 변환

ndarray의 데이터 타입

- 데이터값은 숫자 값, 문자열 값, 불 값 모두 가능
- 숫자형의 경우 int형, unsigned int형, float형이 있음
    - 데이터 타입은 연산의 특성상 같은 데이터 타입만 가능
    - 예: ndarray 객체에 int와float 같이 x
- 만약 다른 데이터 유형이 섞여 있는 리스트를 ndarray로 변경하면 더 큰 데이터 타입으로 변환됨
- astype() 통해 데이터값의 타입 변경 가능
    - 메모리 절약할 때 자주 사용 (예: float → int)

ndarray를 편리하게 생성하기 - arrange, zeros, ones

- 주로 테스트용으로 데이터를 만들거나 대규모 데이터 초기화 할 경우 사용
- arrange(): 파이썬 range()와 유사한 기능으로 0부터 함수 인자 값 -1까지의 값을 순차적으로 ndarray의 데이터값으로 변환
- zeros(): 튜플 형태의 shape 값을 입력하면 모든 값을 0으로 채운 shape ndarray를 반환
- ones(): 위와 같이 모든 값을 1로 채운 ndarray 반환

ndarray의 차원과 크기를 변경하는 reshape()

- reshape(): ndarray를 특정 차원 및 크기 반환

```
array1 = np.arange(10)
array2 = array1.reshape(2,5)
```

- reshape()는 지정된 사이즈로 변경이 불가능하면 오류가 발생함

- reshape()을 효율적으로 사용하려면 -1를 적용해야 함
  - -1을 인자로 사용하면 원래 ndarray와 호환되는 새로운 shape으로 변환해줌
  - -1 인자는 reshape(-1,1)와 같은 형태로 자주 사용됨
  - reshape(-1, 1)은 원본 ndarray가 어떤 형태라도 2차원이고, 여러 개의 로우를 가지되 1개의 칼럼을 가진 ndarray로 변환됨을 보장함

넘파이의 ndarray의 데이터 세트 선택하기 - 인덱싱(Indexing)

- **특정한 데이터만 추출:** 원하는 위치의 인덱스 값 지정해서 반환
- 슬라이싱(Slicing_: 연속된 인덱스상의 ndarray를 추출하는 방식
- **팬시 인덱싱(Fancy Indexing):** 일정한 인덱싱 집합을 리스트 또는 ndarray 형태로 지정해 해당 위치에 있는 데이터의 ndarray를 반환
- **불린 인덱싱(Boolean Indexing):** 특정 조건에 해당하는지 여부인 T/F 값 인덱싱 집합을 기잔으로 True에 해당하는 인덱스 위치에 있는 데이터의 ndarray를 반환

단일 값 추출 (1차원 데이터)

- 1개의 데이터값을 선택하려면 ndarray 객체에 해당하는 위치의 인덱스 값을 []에 입력하면 됨
- 인덱스는 0부터 시작하므로 '데이터이름'[2]는 3번째 위치의 데이터값임
- 인덱스 -1은 맨 뒤의 데이터값을 의미
  - -2는 맨 뒤에서 두번째 값을 의미

단일 값 추출 (다차원 데이터)

- 여기서는 콤마(,)로 분리된 로우와 칼럼 위치의 인덱스를 통해 접근
- 1차원 array1d를 2차원의 3x3로 변환후 [row, col]을 이용해 추출

```
array2d = array1d.reshape(3,3)
```

- 주목할 점: axis 0과 axis 1
  - axis 0은 로우 방향의 축을 의미함
  - axis 1은 칼럼 방향의 축을 의미
  - 즉 ndarray에서는 [row=0, col=1] 인덱싱이 아닌 [axis0=0, axis1=1]이 더 정확한 표현임

- 3차원의 ndarray의 경우는 axis 0, axis1, axis2로 3개의 축을 가지게 됨 (행, 열, 높이)
- 요약: 넘파이의 다차원 ndarray는 axis 구분을 가짐

슬라이싱 (1차원)

- ':' 기호를 이용해 연속한 데이터 슬라이싱 추출 가능
- 단일 데이턱밧 추출 제외하고 슬라이싱, 팬시 인덱싱, 불린 인덱싱으로 추출된 데이터 세트는 모두 ndarray 타입임
- [시작 인덱스: 종료 인덱스-1]
- 1) ":" 기호 앞에 시작 인덱스 생략하면 처음 인덱스 0으로 간주
- 2) ":" 기호 뒤에 종료 인덱스 생략하면 맨 마지막 인덱스로 간주
- 3) ":" 기호 앞/뒤에 시작/종료 인덱스 생략하면 맨 처음/맨 마지막 인덱스로 간주

슬라이싱 (다차원)

- 1차원과 차이점: 콤마(,)로 로우와 칼럼 인덱스를 지칭한다
- [시작 인덱스:종료 인덱스-1, 시작 인덱스:종료 인덱스-1]
- 2차원 ndarray에서 뒤에 오는 인덱스를 없애면 1차원 ndarray를 반환한다.

팬시 인덱싱 (Fancy Indexing)

- 리스트나 ndarray로 인덱스 집합을 지정하면 해당 위치의 인덱스에 해당하는 ndarray를 반환하는 인덱싱 방식
- 예시:

```
array2d[[0,1],2]
```

불린 인덱싱 (Boolean indexing)

- 조건 필터링과 검색을 동시에 할 수 있기 때문에 매우 자주 사용되는 인덱싱 방식
- 예시:

```
#[] 안에 array1d > 5 Boolean indexing을 적용
array3 = array1d[array1d > 5]
```

- 즉 array1d 안에 True 값 (5보다 큰)을 반환한다

행렬의 정렬 - sort()와 argsort()

행렬 정렬

- 종류

np.sort()	ndarray.sort()
넘파이에서 sort()를 호출하는 방식	행렬 자체에서 sort()를 호출하는 방식
원 행렬은 그대로 유지한 채 행렬의 정렬된 행렬을 반환	원 행렬 자체를 정렬한 형태로 변환하며 반환

- 내림차순으로 정렬하기 위해서는 [::-1]을 적용
  - np.sort()[::-1]
- 행렬이 2차원 이상일 경우 axis 축 값 설정을 통해 로우 방향, 또는 칼럼 방향으로 정렬 수행 가능

정렬된 행렬의 인덱스를 반환하기

- np.argsort(): 원본 행렬이 정렬되었을 때 기존 원본 행렬의 원소에 대한 인덱스를 필요로 할 때 이용
- np.argsort()는 정렬 행렬의 원본 행렬 인덱스를 ndarray 형으로 반환

선형대수 연산 = 행렬 내적과 전치 행렬 구하기

행렬 내적(행렬 곱)

- np.dot() 이용해 계산

전치 행렬

- transpose() 이용해 계산

데이터 핸들링 - 판다스

- 판다스는 파이썬에서 데이터 처리를 위해 존재
- 대부분의 데이터 세트는 2차원 데이터
- 판다스 핵심 객체: DataFrame
  - Series 와 DataFrame 차이점: Series는 칼럼이 하나뿐인 데이터 구조체이고, DataFrame은 칼럼이 여러 개인 데이터 구조체

판다스 시작 - 파일을 DataFrame으로 로딩, 기본 API

read_csv()	read_table()	read_fwf()
csv 파일 포맷		Fixed Width
필드 구분 문자 콤마(',')	필드 구분 문자 탭('\t\)	고정 길이 기반의 칼럼 포맷 DF으로 로딩

- read_csv()에서 필드 구분 문자 변환 가능하다
  - ex) sep='\t'
- shape()을 이용해 행과 열 크기를 알아본다
- info()를 통해 총 데이터 건수와 데이터 타입, Null 건수를 알아본다
- describe()를 통해 칼럼별 숫자형 데이터값의 n-percentile 분포도, 평균값, 최댓값, 치솟값을 알아본다
- DataFrame의 []에 칼럼명을 입력하면 Series 형태로 특정 칼럼 데이터 세트가 반환된다
  - value_counts()를 통해 해당 칼럼값의 유형과 건수를 확인

DataFrame과 리스트, 딕셔너리, 넘파이 ndarray 상호 변환

넘파이 ndarray 리스트, 딕셔너리를 DataFrame으로 변환하기

- DF은 리스트와 넘파이 ndarray와 다르게 칼럼명을 가지고 있다.
- 따라서 리스트-ndarray-데이터프레임(칼럼 지정)으로 변환
- 딕셔너리의 키는 칼럼명으로, 값은 키에 해당하는 칼럼 데이터로 변환된다
- 따라서 키는 문자열, 값은 리스트(또는 ndarray) 형태로 딕셔너리를 구성함

DataFrame을 넘파이 ndarray, 리스트, 딕셔너리로 변환하기

- Value를 이용해 DataFrame을 넘파이 ndarray로 변환
- tolist()을 이용해 DataFrame을 리스트로 변환
- to_dict()를 이용해 DataFrame을 딕셔너리로 변환

DataFrame의 칼럼 데이터 세트 생성과 수정

- 세트 생성과 수정 역시 [] 연산자 이용

Aggregation 함수 적용

- min(), max(), sum(), count() 등을 포함

Groupby() 적용

- groupby() 사용 시 입력 파라미터 by에 칼럼을 입력하면 대상 칼럼으로 groupby됨
- 데이터 프레임에서 groupby() 호출하면 DataFrameGroupBy라는 또 다른 형태의 DataFrame을 반환

```
titanic_groupby = titanic_df.groupby(by='Pclass')
```

- 여기서 추가로 aggregation 함수를 호출하면 groupby() 대상 칼럼을 제외한 모든 칼럼에 해당 aggregation 함수가 적용됨
  - 원리: 해당 칼럼을 필터링한 뒤 aggregation 함수를 적용함

  ```
  titanic_groupby = titanic_df.groupby('Pclass')[['Passenger
  ```

  -여러개의 함수명을 DataFrameGroupBy 객체에 입력 가능

  ```
  titanic_df.groupby('Pclass')['Age'].agg([max, min])
  ```

  - 하지만 여러 개의 칼럼이 서로 다른 aggregation 함수를 groupby에서 호출하려면 복잡한 처리가 필요함
  - 약간 딕셔너리 형태와 비슷

```
agg_format={'Age': 'max', 'SibSp': 'sum', 'Fare': 'mean'}
```

결손 데이터 처리하기 (Missing Data)

- NULL인 경우를 의미
- 넘파이의 NaN으로 표시
- 머신러닝 알고리즘은 이 NaN 값을 처리하지 않고 다른 값으로 대체
- NaN 값은 평균, 총합 등의 함수 연산 시 제외됨
- isna()로 NaN 여부 확인
  - 뒤에 sum()을 붙이면 True(있다)는 숫자 1, False(없다)는 숫자 0으로 변환해줌
- fillna()로 NaN 값 다른 값으로 대체
  - 반환 값을 지정하거나 inplace=True 파라미터를 fillna()에 추가해야 실제 데이트 세트 값이 변경됨

apply lambda 식으로 데이터 가공

- apply 함수에 lambda 식을 결합해 레코드별로 데이터 가공하는 기능
- lambda 식은 보통 map() 함수와 결합해 사용
- lambda 식에서 if else 절을 사용해 더 복잡한 가공 가능
  - 주의할 점: if 식보다 반환 값을 먼저 기술해야 함
  - ex) lambda x: if x≤15 'Child' else 'Adult'가 아니라 lambda x: 'Child' if x<15 else 'Adult
  - 또 한가지 if, else만 지원하고 else if는 지원하지 x
- else if가 많이 나와야 하는 경우는 그냥 별도의 함수를 만든 게 더 낫다