**EURON 1주차 예습과제_정윤지**

# Chapter 01: 파이썬 기반의 머신러닝과 생태계 이해

## 01. 머신러닝의 개념

- 머신러닝: 애플리케이션을 수정하지 않고도 데이터를 기반으로 패턴을 학습하고 결과를 예측하는 알고리즘 기법을 통칭
  - 기존 소프트웨어 코드로 해결하기 어려운 문제 해결이 가능
  - 소프트웨어 코드로 로직을 구성, 이들을 관통하는 일정한 패턴을 찾기 어려운 경우 솔루션 제공
  - 데이터를 기반으로 숨겨진 패턴을 인지하여 해결
  - 통계적 신뢰도 강화, 예측 오류 최소화를 위한 수학적 기법을 적용, 데이터 내의 패턴을 스스로 인지, 신뢰도 있는 예측 결과 도출
  - 데이터를 관통하는 패턴을 학습
  - 난이도와 개발 복잡도가 너무 높아질 수 밖에 없는 분야에서 급속한 발전
  - 데이터에 매우 의존적이라는 단점이 존재, 좋은 품질의 데이터를 갖추지 못한다면 수행 결과가 좋지 못함

- 머신러닝의 분류: 지도학습과 비지도학습, 강화학습
  - 지도학습: 분류, 회귀, 추천 시스템, 시각/음성 감지/인지, 텍스트 분석/NLP
  - 비지도학습: 클러스터링, 차원 축소, 강화학습

- 파이썬과 R 기반의 머신러닝 비교
  - R: 통계 전용 프로그램 언어
  - 파이썬: 개발 전문 프로그램, 직관적인 문법과 객체지향과 함수형 프로그래밍 모두를 포괄하는 유연한 프로그램 아키텍처, 다양한 라이브러리 등, 쉽고 뛰어난 개발 생산성, 오픈 소스 계열의 전폭적 지원, 뛰어난 확장성과 유연성, 호환성, 애플리케이션 개발의 가능

## 02. 파이썬 마신러닝 생태계를 구성하는 주요 패키지

- 머신러닝 패키지: 사이킷런(데이터 마이닝 기반의 머신러닝에서 독보적인 위치)
- 행렬/선형대수/통계 패키지: 넘파이(행렬 기반의 데이터 처리에 특화)
- 데이터 핸들링: 판다스(데이터 처리 패키지, 2차원 데이터 처리에 특화)
- 시각화: 맷플롯립, seborn(보완하여 재출시)
- 서드파티 라이브러리: 파이썬 기반의 머신러닝을 편리하게 지원하기 위함
- 주피터 노트북: 아이파이썬 툴, 대화형 파이썬 툴

- 파이썬 머신러닝을 위한 S/W 설치

**03. 넘파이**

- 선형대수 기반의 프로그램을 쉽게 만들 수 있도록 지원하는 대표적인 패키지
  - 루프를 사용하지 않고 대량 데이터의 배열 연산을 가능하게 하므로 빠른 연산속도를 보장
  - 빠른 계산 능력이 중요한 대량 데이터 기반의 과학, 공학 프로그램은 넘파이에 의존

- C/C++과 같은 저수준 언어 기반의 호환 API를 제공
  - 기존 C/C++ 기반의 타 프로그램과 데이터를 주고 받거나 API를 호출해 쉽게 통합할 수 있는 기능 제공

- 데이터 핸들링 기능을 제공

- 넘파이 ndarray 개요
  - ndarray를 이용해 넘파이에서 다차원 배열을 쉽게 생성, 연산 수행이 가능
  - array() 함수는 인자를 입력받아 ndarray로 변환하는 기능
  - ndarray.shape는 ndarray의 차원과 크기를 튜플 형태로 나타냄

- ndarray의 데이터 타입
  - 데이터 값은 숫자값, 문자열 값, 불 값이 모두 가능
  - 데이터 타입은 그 연산의 특성상 같은 데이터 타입만 가능

- ndarray를 편리하게 생성하기
  - arange(): range()와 유사한 기능, 0부터 함수 인자 값 -1까지의 값을 순차적으로 ndarray의 데이터 값으로 변환
  - zeros(): 함수 인자로 튜플 형태의 shape 값을 입력하면 모든 값을 0으로 채운 해당 shape를 가진 ndarray를 반환
  - ones(): 함수 인자로 튜플 형태의 shape 값을 입력하면 모든 값을 1로 채운 해당 shape를 가진 ndarray를 반환

- ndarray의 차원과 크기를 변경하는 reshape()
  - reshape() 메서드는 ndarray를 특정 차원 및 크기로 변환
  - 지정된 사이즈로 변경이 불가능하면 오류 발생
  - -1을 인자로 사용하면 원래 ndarray와 호환되는 새로운 reshape로 변환

- 넘파이의 ndarray의 데이터 세트 선택하기 - 인덱싱(indexing)
  - 넘파이에서 ndarray 내의 일부 데이터 세트나 특정 데이터만을 선택할 수 있도록 하는 인덱싱

- 특정한 데이터만 추출
- 슬라이싱: 연속된 인덱스 상의 ndarray를 추출하는 방식
- 팬시 인덱싱: 일정한 인덱싱 집합을 리스트 또는 ndarray 형태로 지정해 해당 위치에 있는 데이터의 ndarray를 반환
- 불린 인덱싱: 특정 조건에 해당하는지 여부인 T/F 값 인덱싱 집합을 기준으로 T에 해당하는 인덱스 위티에 있는 데이터의 ndarray를 반환

• 단일 값 추출
- ndarray 객체에 해당하는 위치의 인덱스 값을 [] 안에 입력
- 인덱스는 0부터 시작

• 다차원 ndarray에서 단일 값 추출
- 다차원의 경우 콤마로 분리된 로우와 칼럼 위치의 인덱스를 통해 접근
- axis 0이 로우 방향 축, axis 1이 칼럼 방향 축임(생략 시 axis 0을 의미)

• 슬라이싱
- : 기호를 이용해 연속한 데이터를 슬라이싱, 추출 가능
- : 기호 사이의 시작, 종료 인덱스는 생략이 가능

• 팬시 인덱싱
- 리스트나 ndarray로 인덱스 집합을 지정하며 해당 위치의 인덱스에 해당하는 ndarray를 반환하는 인덱싱 방식

• 불린 인덱싱
- 조건 필터링과 검색을 동시에 수행

• 행렬의 정렬 - sort() 와 argsort()
- 넘파이에서 sort()를 호출하는 방식: 원 행렬은 그대로 유지할 채 원 행렬의 정렬된 행렬을 반환
- 행렬 자체에서 sort()를 호출하는 방식: 원 행렬 자체를 정렬한 형태로 변환하며 반환 값은 None
- 정렬된 행렬의 인덱스를 반환하기: 원본 행렬이 정렬되었을 때 기존 원본 행렬의 원소에 대한 인덱스를 필요로 할 때 np.sort()를 이용

• 선형대수 연산 - 행렬 내적과 전치 행렬 구하기
- 행렬 내적: 행렬 곱, 두 행렬 A와 B의 내적은 np.dot()을 이용해 계산이 가능
- 전치 행렬: 원 행렬에서 행과 열 위치를 교환한 원소로 구성한 행렬

## 04. 데이터 핸들링 - 판다스

- 데이터 처리를 위해 존재하는 가장 인기 있는 라이브러리
- 일반적으로 대부분의 데이터 세트는 2차원 데이터(행 X 열)
- 행과 열로 이루어진 2차원 데이터를 효율적으로 가공/처리가 가능한 기능 제공
- 파이썬의 리스트, 컬렉션, 넘파이 등의 내부 데이터 뿐만 아니라 CSV 등의 파일을 쉽게 DataFrame으로 변경해 데이터의 가공/분석을 편리하게 수행하도록 함
- DataFrame의 로우나 칼럼을 지정하여 데이터를 선택할 수 있는 인덱싱 방식으로 iloc[]와 loc[]를 제공

- DataFrame
  - 판다스의 핵심 객체
  - 여러 개의 행과 열로 이루어진 2차원 데이터를 담는 데이터 구조체
  - index 를 key 값으로 가지며 칼럼이 여러 개인 데이터 구조체임
  - 여러 개의 series 로 이루어짐

- 판다스 시작 - 파일을 DataFrame 으로 로딩, 기본 API
  - 주피터 노트북 생성, 판다스 모듈을 임포트
  - 데이터 파일을 판다스의 DataFrame 으로 로딩하는 편리한 API를 제공
  - read_csv() 는 어떤 필드 구분 문자 기반의 파일 포맷도 DataFrame으로 변환이 가능
  - pd.read_csv() 는 호출 시 파일명 인자로 들어온 파일을 로딩해 DataFrame 객체로 변환
  - DataFrame.head() 는 DataFrame 의 맨 앞에 있는 N개의 로우를 반환
  - Shape 는 DataFrame 의 행과 열을 튜플 형태로 반환
  - info() 메서드를 통해 총 데이터 건수와 데이터 타입, Null 건수 파악 가능
  - describe() 메서드는 칼럼 별 숫자형 데이터 값의 n-percentile 분포도, 평균값, 최댓값, 최솟값을 나타냄
  - value_counts() 메서드를 사용할 때는 Null 값을 무시하고 결괏값을 내놓기 쉬움
  - value_count() 는 Null 값을 포함하여 개별 데이터 값의 건수를 계산할 것인지 dropna 인자로 판단

- DataFrame과 리스트, 딕셔너리, 넘파이 ndarray 상호 변환
  - 넘파이 ndarray, 리스트, 딕셔너리를 DataFrame으로 변환하기: DataFrame으로 변환 시 칼럼명 지정, 판다스 DataFrame 객체의 생성인자 Data는 리스트나 딕셔너리 또는 넘파이 ndarray를 입력받고, 생성 인자 columns는 칼럼명 리스트를 입력받아, DataFrame 생성이 가능
  - DataFrame은 기본적으로 행과 열을 가지는 2차원 데이터이므로 2차원 이하의 데이터들만 DataFrame으로 변환이 가능
  - DataFrame을 넘파이 ndarray, 리스트, 딕셔너리로 변환하기: DataFrame 객체의 values를 이용

- DataFrame의 칼럼 데이터 세트 생성과 수정
  - [] 연산자를 이용, DataFrame[] 내에 새로운 칼럼명을 입력하고 값을 할당

- DataFrame 내의 기존 칼럼 값도 쉽게 일괄적으로 업데이트가 가능
- 업데이트를 원하는 칼럼 series 를 DataFrame[] 내에 칼럼명으로 입력한 후, 값을 할당

• DataFrame 데이터 삭제
- drop() 메서드 이용
- DataFrame.drop(labels=None, axis=0, index=None, colums=None, level=None, inplace=False, errors='raise'
- Axis 값에 따라 특정 칼럼 또는 특정 행을 드롭(axis 0은 로우 방향, axis 1은 칼럼 방향)
- drop() 메서드에 axis=1 을 입력하면 칼럼 축 방향으로 드롭을 수행(칼럼을 드롭)
- 원본 DataFrame은 유지, 드롭된 DataFrame을 새롭게 객체 변수로 받기 위해서는 inplace=False로 설정
- 원본 DataFrame에 드롭된 결과를 적용할 경우 inplace=True를 적용

• Index 객체
- DataFrame, series의 레코드를 고유하게 식별하는 객체
- reset_index()는 인덱스가 연속된 int 숫자형 데이터가 아닐 경우 다시 이를 연속 int 숫자형 데이터로 만들 때 사용
- Series 에 reset_index()를 적용하면 새롭게 연속 숫자형 인덱스가 만들어지고 기존 인덱스는 index 칼럼명으로 추가되면서 DataFrame으로 변환
- drop=True로 성정하면 기존 인덱스는 drop

• 데이터 셀렉션 및 필터링
- DataFram의 [] 연산자: 칼럼명 문자, 인덱스로 변환 가능한 표현식, DataFrame 뒤에 있는 []는 칼럼만 지정할 수 있는 '칼럼 지정 연산자, 넘파이의 []나 series의 []와 다른 연산자

• DataFrame iloc[] 연산자
- 위치 기반 인덱싱
- 0부터 시작하는 행, 열의 위치 좌표에만 의존하는 것
- 행과 열 위치를 0을 출발점으로 하는 세로축, 가로축 좌표 정숫값으로 지정
- 정숫값 또는 정수형의 슬라이싱이나 팬시 리스트를 입력
- 열 위치에 -1을 입력하여 DataFrame의 가장 마지막 열 데이터를 가져오는 데 사용
- 불린 인덱싱은 제공하지 않음

• DataFrame loc[] 연산자
- 명칭 기반 인덱싱
- 인덱스나 칼럼명으로 데이터에 접근하는 것
- 행 위치에는 DataFrame 인덱스 값을, 열 위치에는 칼럼명을 입력
- loc[인덱스값, 칼럼명]

- loc[]에 슬라이싱 기호를 적용하면 종료 값까지 포함하는 것을 의미

• 불린 인덱싱
- 데이터 필터링 방식
- [], loc[]에서 공통으로 지원
- iloc[]는 불린 인덱싱이 지원되지 않음

• 정렬, Aggregation 함수, GroupBy 적용
• DataFrame, series의 정렬: sort_value()
- 주요 입력 파라미터: By, ascending, inplace
- By: 특정 칼럼을 입력하면 해당 칼럼으로 정렬을 수행
- ascending=True로 설정하면 오름차순 정렬, False로 설정하면 내림차순으로 정렬
- inplace=False로 설정하면 sort_value()를 호출한 DataFrame은 그래도 유지하며 정렬된 DataFrame을 결과로 반환, True로 설정하면 호출한 DataFrame의 정렬 결과를 그대로 적용
• Aggregation 함수 적용
- count()를 적용하면 모든 칼럼에 count() 결과를 반환
- Null 값을 반영하지 않은 결과를 반환
- 특정 칼럼에 aggregation 함수를 적용하기 위해서는 DataFrame에 대상 칼럼들만 추출해 적용
• groupby() 적용
- 입력 파라미터 by에 칼럼을 입력하면 대상 칼럼으로 groupby
- DataFrame에 groupby() 를 호출하면 DataFrameGroupBy라는 형태의 DataFrame을 반환
- DataFrame에 groupby()를 호출해 반환된 결과에 aggregation 함수를 호출하면 groupby() 대상 칼럼을 제외한 모든 칼럼에 해당 aggregation 함수를 적용

• 결손 데이터 처리하기
- 판다스는 결손 데이터를 처리하는 편리한 API를 제공
- 결손 데이터는 칼럼에 값이 없는, Null인 경우를 의미, 넘파이의 NaN으로 표시
- 기본적으로 NaN 값을 처리하지 않으므로 이 값을 다른 값으로 대체해야 함(평균, 총합 등의 함수 연산 시 제외)
- isna(): 데이터가 NaN인지 판단(True, Fasle), 결손 데이터의 개수는 sum() 함수를 추가해 구할 수 있음
- fillna(): 결손 데이터를 다른 값으로 대체

• Apply lambda 식으로 데이터 가공
- Apply 함수에 lambda 식을 결합해 DataFrame 이나 series 의 레코드별로 데이터를 가공하는 기능을 제공
- Lambda 식은 if else를 지원하는데, if 절의 경우 if 식보다 반환 값을 먼저 기술해야 함