파이썬 기반의 머신러닝과 생태계 이해

머신러닝의 개념

- 머신러닝이란?
    - 데이터를 기반으로 패턴을 학습하고 결과를 예측하는 알고리즘 기법
    - 복잡한 조건, 규칙이 다양한 형태로 결합하고 시시각각 변화함으로 인해 이들을 관통하는 일정한 패턴을 찾기 힘든 경우 유용하게 사용.

머신러닝의 분류

- 지도학습
    - 분류
    - 회귀
    - 추천 시스템
    - 시간/음성 감지/인지
    - 텍스트 분석, NLP
- 비지도학습
    - 클러스터링
    - 차원 축소
    - 강화학습

데이터 전쟁

- 머신러닝의 가장 큰 단점 : 데이터에 매우 의존적
- 어떤 품질의 데이터를 사용했냐에 따라 회사에 경쟁력이 결정됨.

파이썬과 R 기반의 머신러닝 비요

- 통계 분석에 능한 사용자라면 **R** : 통계를 위해 특화된 언어
- 이제 머신러닝을 시작하는 사람이라면 **파이썬** : 대다수의 프레임워크가 파이썬을 중심으로 발전되는 중.

파이썬 머신러닝 생태계를 구성하는 주요 패키지

- 머신러닝 패키지
    - 사이킷런
- 행렬, 선형대수, 통계 패키지
    - 넘파이
    - 사이파이
- 데이터 핸들링
    - 판다스
- 시각화
    - 맷플롯립
    - 시본
- 주피터 노트북

넘파이

- 파이썬에서 선형대수 기반의 프로그램을 쉽게 만들 수 있도록 지원하는 패키지
- 대량 데이터의 배열 계산을 가능하게 함.

ndarray

- ndarray : 넘파이의 기반 데이터 타입, 다차원 배열 생성 및 연산 수행 가능
- array() : 인자를 입력받은 후 ndarray로 변환
- shape : ndarray의 크기 저장
- ndim : array의 차원을 리턴

ndarray의 데이터 타입

- 숫자, 문자열, 불 모두 가능
- 연산 : 같은 타입만 가능 -> dtype로 확인 가능
- 다른 데이터 유형이 섞여 있을 경우 -> 더 큰 데이터 타입으로 형 변환
- astype(): 데이터값의 타입 변경 메소드

ndarray를 편리하게 생성하기 - arrange, zeros, ones

- arrange(): array를 range()로 표현
- zeros(), ones(): 모든 값을 0(1)으로 채운 해당 shape의 ndarray 반환

reshape()

- reshape(): ndarray()를 특정 차원 및 크기로 변환
- -1 사용 시 호환되는 새로운 hspae로 변환

인덱싱

- 단일 값 추출 : 인덱스를 [] 안에 입력
- 슬라이싱 : ':' 기호를 이용해 연속한 데이터 슬라이싱 가능
- 팬시 인덱싱 : 리스트나 ndarray로 인덱스 집합 지정 시 해당 위치에 해당하는 ndarray 반환
- 불린 인덱싱 : [] 안에 조건문을 작성 시 true인 인덱스값만 저장됨

sort(), argsort()

- np.sort() vs ndarray.sort()
    - np.sort() : 원 행렬 유지, 원 행렬의 정렬된 형태 반환
    - ndarray.sort() : 원 행렬 정렬한 형태로 반환
- 내림차순 정렬 : [::-1]
- 2차원 이상 행렬 -> axis = 0 : row, 1 : column

정렬된 행렬의 인덱스 반환

- np.argsort() : 원본 행렬이 정렬되었을 때 기존 원본 행렬의 인덱스 ndarray형으로 리턴

선형대수 연산

- np.dot(): 내적
- transpose(): 전치

데이터 핸들링 - 판다스

- 판다스란?
    - 파이썬에서 데이터 처리를 위해 존재하는 가장 인기 있는 라이브러리
    - 2차원 데이터를 효율적으로 가공, 처리할 수 있는 기능 제공

기본 API

- head() : 맨 앞에 있는 n개의 row 반환 (디폴트 5개)
- shape: 행과 열 튜플 형태로 반환
- info(): 총 데이터 건수, 데이터 타입, Null 건수 확인
- describe(): 숫자형 칼럼의 분포도 조사
- value_counts(): 지정된 칼럼의 데이터값 건수 반환

DataFrame과 리스트, 딕셔너리, 넘파이 ndarray 상호 변환

넘파이 ndarray, 리스트, 딕셔너리 -> DataFrame

- 칼럼명 필요
- 2차원 이하의 데이터만 변환 가능

DataFrame -> 넘파이 ndarray, 리스트, 딕셔너리

- ndarray: values 사용
- 리스트: ndarray에 tolist()
- 딕셔너리: to_dict() 사용

DataFrame의 칼럼 데이터 세트 생성과 수정

- DataFrame [] 내에 새로운 칼럼명을 입력하고 값 할당.

데이터 삭제

- drop() 메소드
    - labels: 여러 개 칼럼 삭제 시 사용
    - axis: 특정 칼럼 or 행 드롭
    - inplace: 원본 df에 결과 반영할 지 결정

index 객체

- DataFrame, Series의 레코드를 고유하기 식별하는 객체 (PK와 유사)
- index 속성으로 추출 가능
- reset_index() : 인덱스를 새롭게 할당, 기존 인덱스는 칼럼에 추가

데이터 셀렉션 및 필터링

DataFrame의 [] 연산자

- [] : 칼럼만 지정할 수 있는 **칼럼 지정 연산자**

- 슬라이싱, 불린 인덱싱 가능 (슬라이싱은 지양하는 편이 좋음)

ix[] 연산자

- 행, 열 위치 지정으로 원하는 데이터 추출
- 향후 사라질 예정 (loc, iloc 대신 사용 가능)

명칭 기반 인덱싱과 위치 기반 인덱싱의 구분

- 명칭 기반 인덱싱 : 칼럼의 명칭을 기반으로 위치 지정
- 위치 기반 인덱싱 : 0으로부터의 좌표를 기준으로 위치 지정
- 인덱스값: 명칭 기반 인덱싱

iloc[], loc[]

- iloc[]: 위치 기반 인덱싱만 허용
- loc[]: 명칭 기반 인덱싱만 허용

불린 인덱싱

- 주어진 조건에 따라 자동으로 데이터를 필터링
- [], ix[], loc[]에서 사용가능

정렬, Aggregation 함수, GroupBy 적용

sort_values()

- ascending: 오름차순 / 내림차순
- inplace: 결과 원본에 적용할지 말지

Aggregation 함수

- 모든 칼럼에 함수가 적용됨
- 대상 칼럼만 추출해 적용 가능

groupby()

- by에 칼럼 입력 시 해당 칼럼으로 groupby

결손 데이터 처리하기

- isna() : NaN 여부 확인
- fillna() : 결손 데이터 대체

apply lambda 식으로 데이터 가공

- lambda 식 : 함수형 프로그래밍을 지원하기 위해 만들어진 식
- df에 apply 메소드를 덧댄 뒤 lambda 식을 작성하는 방법으로 사용 가능
- if else 지원, 만약 너무 조건문이 길어질 시 함수로 빼는 것도 가능