[Chapter 01. 파이썬 기반의 머신러닝과 생태계 이해]

1. 머신러닝의 개념

1) 머신러닝(Machine Learning): 애플리케이션을 수정하지 않고도 데이터를 기반으로 패턴을 학습하고 결과를 예측하는 알고리즘 기법
데이터 분석 영역 → 머신러닝 기반의 예측 분석(Predictive Analysis)

2) 머신러닝의 분류
(1) 지도학습 (Supervised Learning)
분류(Classification), 회귀(Regression), 추천 시스템, 시각/음성 감지/인지, 텍스트 분석, NLP
(2) 비지도학습 (Un-supervised Learning)
클러스터링, 차원 축소, 강화학습
(3) 강화학습 (Reinforcement Learning)

3) 데이터 전쟁
- 데이터의 중요성이 무척 커짐
- 머신러닝의 가장 큰 단점: 매우 의존적임 (데이터 품질이 중요하다는 말)

4) 파이썬과 R 기반의 머신러닝 비교
- R: 통계 전용
- 파이썬: 다양한 영역

2. 파이썬 머신러닝 생태계를 구성하는 주요 패키지

1) 일반적 패키지
(1) 머신러닝 패키지: 사이킷런(Scikit-Learn) 기반 머신러닝 구현에 초점 맞출 것
(2) 행렬/선형대수/통계 패키지: 머신러닝의 이론적 백그라운드는 선형대수와 통계, 넘파이(NumPy)-파이썬의 대표적인 행렬과 선형대수 다룸, 사이파이(SciPy)-자연과학과 통계, 사이킷런에 도움
(3) 데이터 핸들링: 판다스는 파이썬에서 대표적인 데이터 처리 패키지임, 넘파이는 행렬 기반에 특화되어 부족한 부분 많음, 판다스는 2차원 데이터 처리에 특화(맷플롯립(Matplotlib) 호출해 시각화 가능)
(4) 시각화: 파이썬의 대표적인 시각화 패키지는 맷플롯립 → 시본

(5) 아이파이썬 툴 – 주피터 노트북

3. 넘파이
   - NumPy = Numerical Python
   - 선형대수, 통계 기반
   - 대량 데이터의 배열 연산 가능케 함
   - C/C++과 같은 저수준 언어 기반 호환 API 제공 ex) 텐서플로

1) 넘파이 ndarray 개요

넘파이의 기반 데이터 타입: ndarray

array() 함수: 다양한 인자 입력받아 ndarray로 변환

생성된 ndarray 배열의 shape 변수는 ndarray의 크기, 즉 행과 열의 수를 튜플 형태로 가짐, 차원도 알 수 있음

array() 함수의 인자로는 파이썬의 리스트 객체를 주로 사용

- Array의 차원은 ndarray.ndim 으로 확인
- 차원과 크기를 ndarray.shape로 확인

2) ndarray의 데이터 타입
   - 숫자 값, 문자열 값, 물 값 등이 모두 가능
   - 숫자형의 경우 int형(8bit, 16bit, 32bit), unsigned int형(8bit, 16bit, 32bit), float형(16bit, 32bit, 64bit, 128bit), complex 타입
   - 연산의 특성상 같은 데이터 타입만 가능
   - 데이터 타입은 dtype 속성으로 확인
   - ndarray 내의 데이터값은 모두 int32형
   - 서로 다른 데이터 타입이 섞여 있을 경우 데이터 타입이 더 큰 데이터 타입으로 변환되어 int형이 유니코드 문자열 값으로 변환함
   - ndarray 내 데이터값의 타입 변경은 astype() 메서드 이용해서 할 수 있음
3) ndarray를 편리하게 생성하기 – arange, zeros, ones
   - arange()함수는 range()와 유사한 기능
   - 초기화
   - zeros()는 함수 인자로 튜플 형태의 shape 값 입력하면 모든 값을 0으로 채운 해당 shape를 가진 ndarray를 반환함
   - ones()는 함수 인자로 튜플 형태의 shape 값 입력하면 모든 값을 1로 채운 해당 shape를 가진 ndarray를 반환함 → 함수 인자로 dtype을 정해주지 않으면 default로 float64형의 데이터로 ndarray를 채움
4) ndarray의 차원과 크기를 변경하는 reshape()

- -1을 인자로 사용하면 원래 ndarray와 호환되는 새로운 shape로 변화해줌
  자동으로 알아서 새롭게 생성하라는 의미
- ndarray는 tolist()메서드를 이용해 리스트 자료형으로 변환 가능

5) 넘파이의 ndarray의 데이터 세트 선택하기 – 인덱싱(Indexing)
넘파이에서 ndarray 내의 일부 데이터 세트나 특정 데이터만을 선택할 수 있도록 하는 인덱싱

(1) - 특정한 데이터만 추출: 원하는 위치의 인덱스 값을 지정하면 해당 위치의 데이터가 반환됨
- 슬라이싱(Slicing): 슬라이싱은 연속된 인덱스상의 ndarray를 추출하는 방식임
  ':' 시호 사이에 시작 인덱스와 종료 인덱스를 표시하면 시작 인덱스에서 종료 인덱스-1 위치에 있는 데이터의 ndarray를 반환함
- 팬시 인덱싱(Fancy Indexing): 일정한 인덱싱 집합을 리스트 또는 ndarray 형태로 지정해 해당 위치에 있는 데이터의 ndarray를 반환합니다.
- 불린 인덱싱(Boolean Indexing): 특정 조건에 해당하는지 여부인 True/False 값 인덱싱 집합을 기반으로 True에 해당하는 인덱스 위치에 있는 데이터의 ndarray를 반환함

(2) 단일 값 추출
1개의 데이터값을 선택하려면 ndarray 객체에 해당하는 위치의 인덱스 값을 [] 안에 입력하면 됩니다.

(3) 다차원 ndarray에서 단일 값 추출(3차원이나 2차원이나 큰 차이가 없음)
1차원과 2차원 ndarray에서의 데이터 접근의 차이는 2차원의 경우 콤마(,)로 분리된 로우와 칼럼 위치의 인덱스를 통해 접근하는 것임
axis 0: 로우 방향의 축 (행)
axis 1: 칼럼 방향의 축 (열)
axis 2: 높이
axis 생략되면 axis 0 의미

(4) 슬라이싱
':' 기호를 이용
데이터값 추출을 제외하고 슬라이싱, 팬시 인덱싱, 불린 인데싱으로 추출된 데이터 세트는 모두 ndaray 타입임
':' 사이에 시작 인덱스와 종료 인덱스를 표시하면 시작 인덱스에서 종료 인덱스-1의 위치에 있는 데이터의 ndarray를 반환

- 시작 인덱스를 생략 → 자동으로 맨 처음 인덱스인 0으로 간주
- 종료 인덱스를 생략 → 자동으로 맨 마지막 인덱스로 간주
- 기호 앞/뒤에 시작/종료 인덱스를 생략하면 자동으로 맨 처음/맨 마지막 인덱스

로 간주

- 2차원 슬라이싱도 1차원과 방법은 동일하며, 단지 콤마(,)로 로우와 칼럼 인덱스를 지칭하는 부분만 다름

(5) 불린 인덱싱

- 조건 필터링과 검색을 동시에 할 수 있음
- for loop/if else 문보다 훨씬 간단
- Step1: array1d>5와 같이 ndarray의 필터링 조건을 [] 안에 기재
- Step2: False 값은 무시하고 True 값에 해당하는 인덱스값만 저장(True 값 자체인 1을 저장하는 것이 아닌 True 값을 가진 인덱스를 저장하는 것)
- Step3: 저장된 인덱스 데이터 세트로 ndarray 조회

(6) 행렬의 정렬 – sort()와 argsort()

행렬을 정렬하는 대표적인 방법인 np.sort(), ndarray.sort()

정렬된 행렬의 인덱스를 반환하는 argsort()

- np.sort()는 원본 행렬을 변경하지 않고 정렬된 형태로 반환
- ndarray.sort()는 원본 행렬 자체를 정렬한 값으로 반환함
- 모두 기본적으로 오름차순으로 정렬
- 내림차순으로 적용하려면 [::-1] 사용 ex) np.sort()[::-1]
- 행렬이 2차원 이상일 경우에 axis 축 값 설정을 통해 로우 방향, 또는 칼럼 방향으로 정렬을 수행할 수 있음

(7) 정렬된 행렬의 인덱스를 반환하기

기존 원본 행렬의 원소에 대한 인덱스를 필요로 할 때 np.argsort()를 이용

정렬 행렬의 원본 행렬 인덱스를 ndarray형으로 반환됨

- 내림차순으로 정렬 시 원본 행렬의 인덱스를 구하는 것도 np.argsort()[::-1]과 같이 적용하면 됨
- 넘파이의 ndarray는 메타 데이터를 가질 수 없으므로 실제 값과 그 값이 뜻하는 메타 데이터를 별도의 ndarray로 각각 가져야만 한다

6) 선형대수 연산 – 행렬 내적과 전치 행렬 구하기

(1) 행렬 내적(행렬 곱)

행렬 내적 = 행렬 곱

np.dot를 이용해 계산 가능

왼쪽 행렬의 열 개수 = 오른쪽 행렬의 행 개수 여야 내적 연산이 가능함

(2) 전치 행렬

원 행렬에서 행과 열 위치를 교환한 원소로 구성한 행렬을 일컫음

A^T

Transpose()를 이용해 전치 행렬을 쉽게 구할 수 있습니다.

4. 데이터 핸들링 – 판다스

판다스는 이처럼 행과 열로 이뤄진 2차원 데이터를 효율적으로 가공/처리할 수 있는 다양하고 훌륭한 기능을 제공함

여러 개의 행과 열로 이뤄진 2차원 데이터 다룸

즉, DataFramed은 여러 개의 Series로 이루어져 있음

Index: 개별 데이터를 고유하게 식별하는 Key값

1) 판다스 시작 – 파일을 DataFrame으로 로딩, 기본 API

read_csv(): 칼럼을 콤마',로 구분한 파일 포맷 변환을 위한 API, 어떤 필드 구분 문자 기바의 파일 포맷도 DataFrame으로 변환 가능(인자인 sep에 해당 구분 문자를 입력하면 됨), 가령 탭으로 필드가 구분되어 있다면 rea_csv('파일명', sep='₩t')처럼 쓰면 됨, 만약 인자를 생략하면 자동으로 콤마로 할당함(sep = ',')

read_table(): 필드 구분 문자(Delimeter)가 탭'₩t'

read_csv()와 read_table() 기능상 큰 차이 없음

read_fwf(): Fixed Width, 즉 고정 길이 기반의 칼럼 포맷을 DataFrame으로 로딩하기 위한 API

- read_csv(filepath_or_buffer, sep=', ', ...) 함수에서 가장 중요한 인자는 filepath
- 나머지 인자는 지정하지 않으면 디폴트 값으로 할당됨
- FIlepath에는 로드하려는 데이터 파일 경로 포함 파일명 입력하면 됨
- df.head() defaultfh 맨 앞 5개 로우 반환
  숫자 입력 시 n개의 로우 반환
- 행과 열 크기는 shape 변수 이용
- 칼럼 타입, Null 데이터 개수, 데이터 분포도 등의 메타 데이터 등 조회는 info(), describe()
- 이렇게 반환된 Series 객체에 value_counts() 메서드 호출하면 해당 칼럼값의 유형과 건수를 확인할 수 있음
- 왼쪽은 Seriesd의 Index, 오른쪽은 Series의 데이터 값

2) DataFrame과 리스트, 딕셔너리, 넘파이 ndarray 상호 변환

(1) 넘파이 ndarray, 리스트, 딕셔너리를 DataFrame으로 변환하기

- 2차원 이하의 데이터들만 DataFrame으로 변환 가능
- Key 값은 칼럼명, 각 Value는 각 칼럼 데이터로 매핑됨

(2) DataFrame을 넘파이 ndarray, 리스, 딕셔너리로 변환하기

-values를 이용

-리스트로의 변환은 values로 얻은 ndarray에 tolist()를 호출하면 됩니다.

- 딕셔너리로의 변환은 DataFrame 객체의 to_dict() 메서드를 호출, 인자로 'list'를 입력하면 딕셔너리 값이 리스트형으로 변환됨

(3) DataFrame의 칼럼 데이터 세트 생성과 수정

- [] 연산자 이용, DataFrame [] 내에 새로운 칼럼명 입력 후 값을 할당해주기만 하면 됨

(4) DataFrame데이터 삭제

- drop() 매서드 이용
- 중요한 파라미터는 labels, axis, inplace
- axis값에 따라 특정 칼럼 또는 특정 행 드롭, 판다스의 DataFrame은 2차원 데이터만 다룸 so, axis 0, axis 1로망 구성되어 있음
- inplace = False로 설정해두면 자기 자신의 DataFrame 데이터는 삭제하지 않고 객체 변수만 삭제됨
- inplace = True로 설정하면 자신의 Dataframe 삭제함
- 여러 개의 칼럼 삭제시 리스트 형태로 삭제하고자 하는 칼럼 명을 임력해 labels 파라미터로 입력하면 됨
- drop() 시 inplace=True로 설정한 채로 반환 값을 다시 자신의 DataFrame 객체로 할당하면 안됨 왜냐면 객체 변수를 아예 None으로 만들어버리기 때문이다

(5) Index 객체

- 추출하려면 DataFrame.index 또는 Series.index 속성 이용

Index 객체는 식별성 데이터를 1차원 array로 가지고 있어 단일 값 반환 및 슬라이싱 가능

- 한 번 만들어진 DataFrame 및 Series의 Index 객체는 함부로 변경 불가
- Series 객체는 Index 객체를 포함하지만 Series 객체에 연산 함수 적용시 Index는 연산에서 제외됨, 오직 식별용임
- DataFrame 및 Series 에 reset_index() 매서드 수행하면 새롭세 인덱스를 연속 숫자 형으로 할당, 기존 인덱스는 'index'라는 새로운 칼럼 명으로 추가함
- Reset_index()는 인덱스가 연속된 int 숫자형 데이터 아닐 때 다시 연속 데이터로 만들 대 주로 사용
- Drop = True 로 설정하면 기존 인덱스는 삭제됨

3) 데이터 셀렉션 및 필터링

판다스의 경우 ix[], iloc[], loc[] 연산자를 통해 수행

- 넘파이에서 []는 행의 위치, 열의 위치, 슬라이싱 범위 등을 지정해 데이터 가져올 수 있음
- DataFrame 바로 뒤에 '[]' 안에는 칼럼 명 문자(또는 칼럼 명의 리스트 객체), 또는 인덱스로 변환 가능한 표현식 → 칼럼 지정 연산자

(1) DataFrame ix[] 연산자

Ix[] 사라짐 그대신 칼럼 명칭 기반 인덱싱 연산자인 loc[]과 칼럼 위치 기반 인덱싱 연산자인 iloc[] 이 새롭게 만들어짐

Ix[]는 ndarray의 [] 연산자와 동일하게 단일 지정, 슬라이싱, 불린 인덱싱, 팬시 인덱싱 모두 가능

(2) 명칭 기반 인덱싱과 위치 기반 인덱싱의 구분

결과적으로 DataFrame 의 인덱스값은 명칭 기반 인덱싱

(3) DataFrame iloc[] 연산자

불린 인덱싱 제공 안함

(4) DataFrame loc[] 연산자

슬라이싱 기호 적용하면 종료값까지 포함함

(5) 불린 인덱싱

Iloc[]은 정수형 값이 아닌 불린 값 지원 안함

4) 정렬, Aggregation 함수, GroupBy 적용

(1) DataFrame, Series의 정렬 – sort_values()

- 주요 입력 파라미터는 by, ascending, inplace
- Ascending = True 오름차순
- Inplace = False 그대로 유지하며 정렬된 것 결과로 반환

(2) Aggregation 함수 적용

- Min(), mac(), sum(), count()

(3) Groupby() 적용

칼럼 기준으로 GroupBy 된 DataFrame Groupby 객체 반환

복잡한 처리 때문에 agg() 이용

5) 결손 데이터 처리하기

NULL 인 경우, 즉 칼럼에 값이 없는 경우 의미

넘파이의 NaN

- NaN 여부를 확인하는 API는 isna()
- NaN 값을 다른 값으로 대체하는 API는 fillna()

(1) Isna()로 결손 데이터 여부 확인

(2) 데이터의 개수는 isna() 결과에 sum() 함수 추가해 구하기

(3) Fillna()로 결손 데이터 대체하기

- 반환값을 다시 받거나 inplace = True 파라미터를 추가해야 실제 데이터 세트 값이 변경된다

6) Apply lambda 식으로 데이터 가공

파이썬에서 함수령 프로그래밍 지원하기 위해 어쩔 수 없이 사용

여러 개의 값을 입력 인자로 사용할 때는 보통 map() 함수 결합해서 사용

- Lambda 식은 if else를 지원하는데, if 절의 경우 if 식보다 반환 값을 먼저 기술해야 함
- 식 ':' 오른편에 반환 값이 있어야 함

- Else는 식이 먼저 오고 반환 값이 나중에 오면 됨
- If, else는 지원하지만 if, else if, else 는 지원 안함
- Else if 이용하려면 else 절을 ()로 내포해서 () 내에서 다시 적용해 사용함