CH01. 파이썬 기반의 머신러닝과 생태계 이해

01. 머신러닝의 개념

머신러닝의 개념

- 머신러닝 : 어플리케이션을 수정하지 않고도 데이터를 기반으로 패턴을 학습하고 결과를 예측하는 알고리즘 기법

- 데이터 기반으로 숨겨진 규칙(패턴)을 인지 > 수학적 기법 적용하여 데이터 내의 패턴 인지, 예측결과 도출

머신러닝의 분류

- 지도학습(분류, 회귀, 추천 시스템, NLP 등), 비지도학습(클러스터링, 차원 축소, 강화학습), 강화학습

데이터 전쟁

- 머신러닝은 데이터에 의존적 > for improvement of 머신러닝 : 최적의 ML 알고리즘 + 모델 파라미터 + 최적의 데이터(효율적으로 가공, 처리, 추출된)

파이썬 ML vs R ML

- 파이썬 : 개발 전문 프로그램 언어(객체지향, 함수형 프로그래밍을 포괄하는 아키텍쳐, 다양한 라이브러리)

- R : 통계 전용 프로그램 언어

>> 파이썬의 장점 : 딥러닝 프레임워크(텐서플로, 케라스, 파이토치) 등에서 파이썬 우선 지원 정책, 쉽고 뛰어난 개발 생산성으로 높은 활용도, 많은 라이브러리, 다양한 영역에서의 사용 등

02. 파이썬 머신러닝 생태계를 구성하는 주요 패키지

주요 패키지

- 머신러닝 패키지 : 사이킷런(데이터 마이닝 기바의 머신러닝에서 굿), 텐서플로와 케라스(비정형 데이터 분야에서의 딥러닝 라이브러리)

- 행렬/선형대수/통계 패키지 : 넘파이(행렬과 선형대수), 사이파이(자연과학과 통계)

- 데이터 핸들링 : 판다스 > 2차원 데이터 처리에 특화(<>넘파이 : 행렬 기반의 데이터 처리에 특화), 편리함, 맷플롯립 호출하여 시각화 기능 지원 가능

- 시각화 : 맷플롯립(세분화된 API, 투박한 시각적 디자인) > 시본(함축적인 API, 판다스와의 쉬운 연동)

- 주피터 노트북 : 대화형 파이썬 툴, 특정 코드 영역별로 개별 수행 가능

파이썬과 머신러닝을 위한 SW설치

1) Anaconda 설치 : 파이썬 기반의 머신러닝 패키지 일괄적으로 설치 가능

2) Anaconda Prompt를 실행해서 파이썬 설치되었는지 확인

3) MS VStudio 다운로드

03. 넘파이

넘파이

- 머신러닝의 주요 알고리즘 > 선대와 통계 등에 기반 > NumPy : 파이썬에서 선형대수 기반의 프로그램을 지원하는 패키지, 루프 사용X 대량 데이터의 배열 연산을 가능하게

- 저수준 언어(C친구들) 기반의 호환 API를 제공(쉽게 통합 ㄱㄴ)

- 다양한 데이터 핸들링 기능 제공(but 2차원은 판다스)

넘파이 ndarray 개요

- ndarray : 넘파이의 기반 데이터 타입, 다차원 배열을 생성하고 연산

- array() : 파이썬 리스트와 같은 인자를 입력받아 ndarray로 변환하는 기능

- shape 변수 : 행과 열의 수를 튜플형태로 갖고 있어 배열의 차원을 알 수 있다. > ndarray의 차원과 크기를 튜플의 형태로 반환 (row,col)

* 보통 []는 1차원, [[]]는 2차원

ndarray의 데이터 타입

- ndarray내의 데이터값은 숫자, 문자열, 불값 등 모두 가능 but 같은 데이터 타입끼리만 존재 가능

- dtype 속성 : ndarray내의 데이터 타입 확인 ㄱㄴ

? ndarray내에 다른 데이터 타입이 존재할 경우 > 데이터의 크기가 더 큰 데이터 타입으로 형 변환

- astype() : ndarray 내 데이터값의 타입 변경 ㄱㄴ > 인자로 원하는 타입을 문자열로 지정 > 메모리 절약할 수 있을수도..?

ndarray를 편리하게 생성하기-arange, zeros, ones

1) arange() : range()와 유사한 기능 > 0~인자-1 값을 순차적으로 ndarray의 데이터값으로 변환

2) zeros() : np.zeros((a,b), dtype='. ') > 튜플 형태의 shape 값을 입력하면 모든 값을 0으로 채운 ndarray반환, dtype 정하기도 가능

3) ones() : 튜플 형태의 shape 값을 입력하면 모든 값을 1으로 채운 ndarray반환, dtype 정하기도 가능

ndarray의 차원과 크기를 변경하는 reshape()

1) reshape()

- reshape() : 인자값으로 (로우, 칼럼) > ndarray를 특정 차원 및 크기로 변환

- nndarray를 지정된 사이즈로 변경 불가능하면 오류

2) 인자 -1의 활용 > 일부호환

- (-1,5) > 칼럼 인자 5에 맞추어 호환되는 로우의 개수를 조절하기

- (5,-1) > 로우 인자 5에 맞추어 호환되는 칼럼의 개수를 조절하기

3) .reshape(-1,1)의 활용 > 항상 호환

- always 2차원, 여러 개의 로우 but always 1개의 칼럼으로 반드시 변환됨을 보장

넘파이의 ndarray의 데이터 세트 선택하기 - 인덱싱

1) 단일 값 추출 : 특정 데이터만 추출, 원하는 값의 인덱스 값을 지정

- index는 0부터 시작, -는 뒤에서부터 순서

- 다차원 배열에서는 axis 0(로우 방향 축), 1(칼럼 방향 축)등 축을 기준으로 개별 인덱스 사용 가능

2) 슬라이싱 : 연속된 인덱스상의 ndarray를 추출

- (시작 : 종료)

- 다차원 배열에서는 ,로 로우와 칼럼을 구분해서 사용

3) 팬시 인덱싱 : 일정한 인덱싱 집합을 리스트 또는 ndarray 형태로 지정해 해당 위치에 있는 데이터의 ndarray를 반환

- [[a,b],c:d] > a~b-1 로우에 해당하는 c~d-1 칼럼의 값을 인덱싱해서 ndarray 반환

- tolist()

4) 불린 인덱싱 : True에 해당하는 인덱스 위치에 있는 데이터의 ndarray를 반환

- []안에 조건문을 기재 > true에 해당하는 데이터를 반환 ex) array[array1d>5]

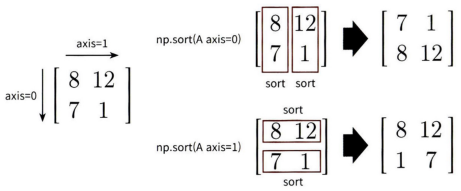
- 불린 ndarray를 array1d[]내에 인덱스로 입력하면 동일한 데이터 세트가 반환됨

행렬의 정렬 - sort()와 argsort()

1) 행렬 정렬(오름차순) > sort()

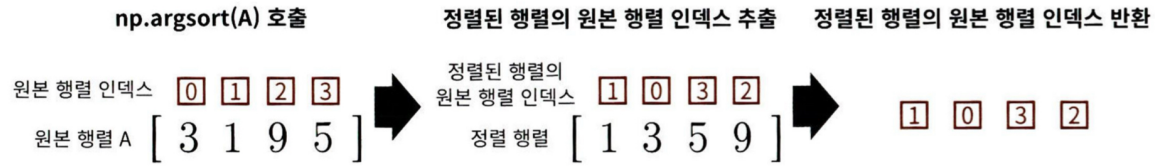
- np.sort() : 원 행렬 유지한 채 원 행렬의 정렬된 행렬 반환 vs ndarray.sort() : 원 행렬 자체를 정렬한 형태로 변환, 반환 값은 None >>기본적으로 오름차순으로 원소 정렬

- 2차원 이상의 경우 : axis 축 값 설정을 통해 로우, 칼럼 방향으로 정렬 수행 가능



2) 정렬된 행렬의 인덱스를 반환하기 > argsort()

- 원본이 정렬되었을 때 원본 행렬 인덱스를 ndarray형으로 반환



- ndarray는 메타 데이터를 못가짐 > 실제값과 그 값을 의미하는 메타 데이터를 별도의 ndarray로 각각 가져야함

선형대수 연산 - 행렬 내적과 전치 행렬 구하기

1) 행렬 내적

- dot()을 활용하여 구현가능

2) 전치 행렬

- 행<>열

- transpose()로 구현가능

04. 데이터 핸들링-판다스

판다스란?

- 판다스 : 행과 열로 이뤄진 2차원 데이터를 효율적으로 가공/처리할 수 있는 기능을 제공하는 라이브러리

- 파이썬의 내부 데이터뿐만 아니라 다양한 유형의 분리 문자로 칼럼을 분리한 파일(csv, tab) 등의 파일을 쉽게 DataFrame으로 변경가능

- DataFrame : Index를 Key값으로, 여러 개의 행과 열로 이뤄진 2차원 데이터를 담는 데이터 구조체 > 칼럼이 여러 개인 데이터 구조체(<>Series는 칼럼이 한개)

판다스 시작 - 파일 DataFrame으로 로딩, 기본 API

1) 판다스

- read_csv(), read_table()> CSV 파일 포맷 변환을 위한 API / 필드 구분 문자는 ,과 ₩t

- read_fwf() : 고정 길이 기반의 칼럼 포맷을 DataFrame으로 로딩하기 위한 API

2) 여러 판다스 메서드

- head() : 맨 앞 인자만큼 로우를 반환

- shape() : DataFrame의 행과 열을 튜플 형태로 반환

- info() : 총 데이터 건수, 데이터 타입, Null 건수 조희가능

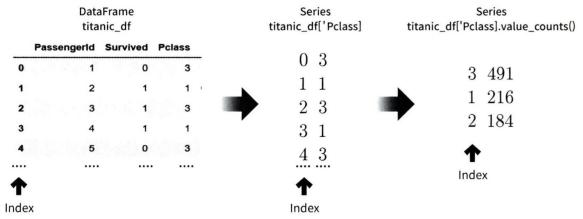
- describe() : 칼럼별 숫자형 데이터값의 n-percentile 분포도, 평균값, 최대최소값 >> 숫자가 숫자가 아닌거에는 의미가 좀..없음

- value_counts() : Series 형태로 특정 칼럼 데이터 세트가 반환 > 지정된 칼럼의 데이터 값 건수 반환 >> Series 객체에서만..!

- []내부의 칼럼명을 입력하면 해당 칼럼에 해당하는 Series 객체를 반환

> Series는 Index와 단 하나의 칼럼으로 구성된 데이터 세트

> Series 객체는 Index에서 고유성이 보장도니 의미있는 데이터값 할당도 가능



DataFrame과 리스트, 딕셔너리, 넘파이 ndarray 상호 변환(2차원이하의 데이터들만)

1) 리스트, 딕셔너리, 넘파이 ndarray를 DataFrame으로 변환하기

- DataFrame : 칼럼명 갖고있음 (<> 리스트, 딕셔너리, 넘파이 ndarray)>> 데이터 핸들링 편함

- 딕셔너리 : Key는 문자열 칼럼명으로, Value는 리스트 형 칼럼 데이터로 매핑

2) DataFrame을 리스트, 딕셔너리, 넘파이 ndarray로 변환하기

- 보통 넘파이 ndarray으로 많이 변환

DataFrame의 칼럼 데이터 세트 생성과 수정

1) 새로운 칼럼 생성

- DataFrame 내에 새로운 칼럼명 입력하고 값 할당 > titanic_df['Age_0'}=0 (일괄 0 값 할당)

 - 기존 칼럼 Series의 데이터를 활용가능

2) 기존 칼럼 일괄적인 업데이트 가능

DataFrame 데이터 삭제

1) drop()

- 원형 : DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')

- axis : 특정 축 방향으로 드롭 수행

- labels : 특정 축의 인덱스나 칼럼명 입력

inplace : False로 설정되면 삭제된 결과만 반환, 실제로는 삭제되지는 않음, True여야 원본이 삭제됨

Index 객체

- DataFrame, Series에서 Index 객체를 추출하려면 .index 속성 통해 가능

- 반환된 Index 객체의 실제 값은 넘파이 1차원 ndarray로 볼 수 있음 > values 속성으로 ndarray 값을 알 수 있음

- Index 객체는 식별성 데이터를 1차원 배열로 갖고 있음 > 단일값 반환 및 슬라이싱 가능

but Index 객체는 변경 불가 & only for 식별용 > 연산에서 제외됨

- reset_index() : 새롭게 인덱스를 연속 숫자 형으로 할당 > index라는 새로운 칼럼명으로

추가

-

데이터 셀렉션 및 필터링

* 판다스의 [] 연산자와 넘파이의 []의 차이점

1) DataFrame의 [] 연산자

- 넘파이 [] 연산자 : 행과 열의 위치, 슬라이싱 범위 등을 지정해 데이터 가져오기 ㄱㄴ

- DataFrame의 [] : 칼럼 명 문자(리스트 객체), 인덱스로 변환 가능한 표현식 > 칼럼 지정 연산자 Only 칼럼만 but 인덱스로 변환 가능한 표현식 such as 슬라이싱, 불린 인덱싱 가능

2) DataFrame의 ix[] 연산자

3) 명칭 기반 인덱싱과 위치 기반 인덱싱의 구분

4) DataFrame의 iloc[] 연산자

- 팬시 리스트 값 입력 필요

5) DataFrame의 loc[] 연산자

- 명칭 기반 인덱싱만 가능

6) 불린 인덱싱

정렬, Aggregation 함수, GroupBy 적용

1) DataFrame, Series의 정렬 - sort_values()

2) Aggregation 함수 적용 : 모든 칼럼에 바로 Aggregation 적용됨>DataFrame에 대상 칼럼들만 추출해서 적용

3) GroupBy 적용 : groupby(by=) > by 파라미터에 칼럼을 입력하면 대상 칼럼이 groupby 됨.

- 또다른 형태의 DataFrameGroupBy가 반환되는 거임

결손 데이터 처리하기

* NaN 값을 처리하거나 다른 값으로 대체해야함.

1) isna()로 결손 데이터 여부 확인

2) fillna()로 결손 데이터 대체하기 : [칼럼명].fillna(대체값)

apply lambda 식으로 데이터 가공

* apply 식에 lambda 식을 결합 > DataFrame이나 Series의 레코드별로 데이터 가공 ㄱㄴ

- lambda x(입력인자) : x**2(입력 인자를 기반으로 한 계산식, 호출 시 계산 결과 반환됨.)