♥ TITLE : 1. 파이썬 기반의 머신러닝과 생태계 이해 ♥ DATE : 2022.08.25

1 머신러닝의 개념

- 일반적 > 애플리케이션을 수정하지 않고도 데이터를 기반으로 패턴을 학습하고 결과를 예측하는 알고리즘 기법
- 데이터 의존적
- 머신러닝의 분류
  - 지도학습 : 분류, 회귀
  - 비지도학습
  - 강화학습

2 머신러닝 주요 패키지

① 머신러닝: 사이킷런(Scikit-learn)
② 행렬/선형대수/통계 패키지: 넘파이, 사이파이
③ 데이터 핸들링: 판다스
④ 시각화: 맷플롯립, 시본

3 넘파이(numpy)

- 파이썬의 선형대수 기반의 프로그램을 쉽게 만들 수 있도록 지원하는 대표적 패키지
- 빠른 배열 연산 속도

① ndarray
- 넘파이의 기반 데이터 타입
- array( ) 함수: 데이터 → ndarray
- parameters>
  - i) shape: ndarray 배열의 크기·차원 파악 / (행, 열)
  - ii) ndim: ndarray의 차원 파악

② ndarray의 데이터 타입
- 1종 데이터 타입이 가능함.
- 연산 시 동일 데이터 타입만 가능함. ··· dtype 속성으로 데이터 타입 확인 가능

ex> if > 여러 데이터형이 섞여 있는 경우 더 큰 데이터 타입으로 변환

- astype( ) 메서드: ndarray 내 데이터 타입 변경

③ ndarray 생성
- i) arange(n): 0 이상 n 미만까지의 값을 순차적으로 ndarray 값으로 변환
  - start 값을 부여해 시작점 변경 가능
- ii) zeros(shape): 모든 값을 0으로 채운 해당 shape를 지닌 ndarray 반환
- iii) ones(shape): 〃 1 〃 ex 튜플 형태
- * default > dtype = float64

④ reshape(shape): ndarray를 특정 차원·크기로 변환
- 변경 불가 시 오류 발생
- -1 적용: 원래의 ndarray와 호환되는 새로운 shape로 변환
  - ex> reshape(-1, n): 열은 n개로, 행은 그에 맞춰서 개수 조정

⑤ 인덱싱(Indexing)
- i) 특정 데이터 추출: 원하는 위치의 인덱스 값 지정
- ii) 슬라이싱: 연속된 인덱스 상의 ndarray 추출
  - a:b > a 이상 b 미만 위치의 ndarray 반환
- iii) 팬시 인덱싱(Fancy indexing): 일정한 인덱싱 집합을 리스트 or ndarray로 지정
  - ⇒ 해당 위치의 데이터를 ndarray로!
- iv) Boolean 인덱싱: 특정 조건 해당 여부를 T/F 값으로 인덱싱
  - ⇒ True에 해당하는 인덱스 위치의 데이터의 ndarray 반환

axis = 0: 행(row) 방향의 축 ⇉ ↓ 방향
axis = 1: 열(column) 방향의 축 ↓↓↓ → 방향

set a record

♥ TITLE : 1_ 파이썬 기반의 머신러닝과 생태계 이해 ♥ DATE : 2022.08.26

⑥ 행렬의 정렬

i) 행렬 정렬

- np.sort( ) ┌ numpy에서 sort( ) 호출
  └ 원 행렬은 그대로 유지 → 정렬된 행렬을 새로 만들어 반환
- ndarray.sort( ) ┌ 행렬 자체에서 sort( ) 호출
  └ 원 행렬 자체를 정렬 → 반환 값은 None

* default> 오름차순 정렬 ⇒ 내림차순: [::-1] 적용
  2차원 이상> axis 설정⇒ 축을 기준으로 정렬 가능

ii) 정렬된 행렬의 인덱스 반환

- np.argsort( ): 정렬 행렬의 원본 행렬에서의 인덱스를 ndarray형으로 반환

⑦ 행렬 내적 / 전치 행렬 구하기

i) 행렬 내적: np.dot( )

ii) 전치 행렬(AT): np.transpose( )

④ 판다스(Pandas)

① 파일을 DataFrame으로 로딩

i) read_csv( ) ┌ CSV 파일 ⊕ 필드 구분 문자 기반의 파일 포맷 변환
  └ sep 옵션: 구분 문자 지정 default: 콤마(,)

ii) read_table( ) ┌ read_csv와 비슷
  └ default 델리미터: 탭('\t')

iii) read_fwf( ): 고정 길이 기반의 칼럼 포맷을 DataFrame으로 로딩

- DataFrame.head( ): DataFrame의 맨 앞의 N개의 row 반환
- DataFrame.shape: DataFrame의 행/열을 튜플 형태로 반환
- DataFrame.info( ): 총 데이터 건수, 데이터 타입, Null 건수 파악
- DataFrame.describe( ) ┌ 칼럼별 숫자형 데이터 값의 n-percentile 분포도, 평균, 최대·최소
  └ only 숫자형 칼럼의 분포도만 조사
- Series ┌ Index와 단 하나의 칼럼으로 구성된 데이터 세트
  └ 인덱스는 고유성이 보장된다면 숫자형, 문자형 모두 가능
- Series.value_counts( )
  ↳ DataFrame의 [ ] 내부에 칼럼명 입력 시 Series 형태로 특정 칼럼 데이터 세트가 반환됨.
  ↳ 해당 칼럼값의 유형/건수 확인
  ↳ 많은 건수 순서대로 정렬된다.

♥ TITLE : 1_ 파이썬 기반의 머신러닝과 생태계 이해 ♥ DATE : 2022.08.26

② DataFrame과 리스트·딕셔너리·넘파이 ndarray 상호변환

i) ⬭ → pd.DataFrame
↳ pd.DataFrame(데이터, columns = 컬럼명)

ii) pd.DataFrame → ⬭
a. DataFrame → ndarray : df.values
b. DataFrame → list or dict ┌ df.values.tolist( )
└ df.values.to_dict( )

③ DataFrame의 칼럼 데이터 세트 생성/수정

i) 생성 : pd.DataFrame[새로운 컬럼명] = 값
ii) 수정 : pd.DataFrame[수정하려는 컬럼명] = 값

④ 데이터 삭제

↳ pd.DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')

i) axis ┌ axis=0 : row 방향 축 제거 ⇒ 행 제거
└ axis=1 : column 〃 ⇒ 열 제거
ii) inplace=False : 자기 자신의 df는 삭제하지 않고 삭제된 결과를 반환
iii) labels=[col1, col2, ...] : 지정된 칼럼 삭제

⑤ Index 객체

- DataFrame, Series의 레코드를 고유하게 식별하는 객체
- index 속성을 통해 추출 가능
- 반환된 객체의 실제 값은 1차원 array로 가지고 0 ⇒ numpy 1차원 array로 확인!
- 값의 변경이 제한됨
- 단일 값 반환 / 슬라이싱 가능
- 오직 식별용 ⇒ 연산 불가
- reset_index( ) 메서드 ┌ 새롭게 인덱스를 연속된 숫자형으로 할당
  - 기존 index : 새로운 칼럼형으로 추가
  - cf> Series에 적용 시 DataFrame이 반환됨
  - e 기존의 index가 칼럼으로 추가됨 ⇒ column이 2개
  - drop=True 옵션 > 기존 index 추가 X (삭제)

⑥ 데이터 selection & filtering

i) 데이터프레임의 [ ] 연산자
- 칼럼 지정 연산자
- DataFrame['컬럼명'] ⇐ 인덱싱 자원 X
  ↳ cf> 슬라이싱은 자원 … pandas의 index 형태로 변환 가능해서(사실 바주..)
- Boolean indexing이 자원됨.

<용어 정리>
• 명칭 기반 인덱싱 : 칼럼의 명칭을 기반으로 위치를 지정하는 방식
• 위치 기반 인덱싱 : 0을 출발점으로 하는 가로축/세로축 좌표 기반의 행/열 위치 기반
e DataFrame의 index는 결과적으로 명칭 기반 인덱싱으로 간주됨.

♥ TITLE : 1_ 파이썬 기반의 어신러닝과 생태계 이해 ♥ DATE : 2022.08.26

ii) DataFrame의 iloc[ ] 연산자
- 위치 기반 인덱싱만 허용
  ↳ integer, integer형의 슬라이싱, fancy list 값
- boolean indexing은 지원되지 ×

iii) DataFrame의 loc[ ] 연산자
- 명칭 기반 인덱싱
- 슬라이싱 적용: a 이상 b 이하 ★★★★

iv) Boolean Indexing
- 조건식으로 인덱싱
- 반환되는 객체는 DataFrame ⇒원하는 컬럼만을 별도로 추출할 수 O
- 복합 조건 적용
  - &: and 조건
  - |: or 조건
  - ~: not 조건

⑦ 정렬, Aggregation 함수, GroupBY 적용

i) 정렬
- sort_values( ) 메서드 이용
- 주요 파라미터
  - by: 정렬의 기준 컬럼 지정
  - ascending: True면 오름차순 정렬
  - inplace: False면 기존 DataFrame 유지

ii) Aggregation 함수
- min( ), max( ), sum( ), count( ) 등
- 모든 컬럼에 일괄 적용됨
  ↳특정 컬럼에만 적용하고 싶으면 컬럼 추출 → aggregation 적용하기!

iii) groupby( ) 적용
- by='컬럼명' : 해당 컬럼으로 groupby
- 결과로 또 다른 형태의 DataFrame을 반환 … dtype=DataFrameGroupBy
- 반환된 결과에 aggregation 함수 적용 시 groupby( ) 대상 컬럼 제외 모든 컬럼에 해당 함수 적용
- 여러 개의 다른 aggregation 함수 적용 시 함수명을 agg( ) 내에 인자로 입력해서 사용 ex> agg([max, min])
- 여러 개의 컬럼이 서로 다른 aggregation 함수를 적용 시 agg( ) 내에 딕셔너리 형태로 입력

⑧ 결손 데이터 처리

i) 결손 데이터
- 칼럼에 값이 없는 경우, 즉 NULL인 경우
- numpy의 NaN으로 표시
- 평균, 총합 등의 함수 연산 시 제외

ii) isna( )
- 데이터가 NaN인지 확인 ⇒각 컬럼에 대해 True/False로 알려줌
- 결손 데이터의 개수: isna( ).sum( )

iii) fillna( )
- 결손 데이터를 다른 값으로 대체
- 주의점 반환값을 다시 받거나 inplace=True를 지정해야 데이터 세트가 변경됨

⑨ apply lambda로 데이터 가공
- apply( ) 함수에 lambda 식을 적용
- 복잡한 데이터 처리에 활용