numpy
pandas
matplotlib

윤 길 배

팔복기술주식회사

2022

numpy

- NumPy(Numerical Python)
- 다차원 데이터 배열을 위한 효율적인 저장과 연산기능 제공.
- 선형 대수학, 푸리에 변환 및 행렬 분야의 기능 탑재.

- Numpy를 사용하는 이유
  - ✓ Python의 list는 처리 속도가 느림.
  - ✓ NumPy는 list보다 최대 50배 빠른 배열 처리속도.

- NumPy가 목록보다 빠른 이유는?
  - ✓ NumPy 배열은 list 와 달리 메모리의 연속적인 한 위치에 저장. 매우 효율적으로 액세스하고 조작할 수 있음.
  - ✓ 최신 CPU 아키텍처에 최적화.
  - ✓ Python으로 작성되지만 빠른 계산이 필요한 부분은 C 또는 C++로 작성.
- 설치
  - ✓ pip install numpy

https://ml-ko.kr/homl2/tools_numpy.html
mumpy 원본 : https://github.com/ageron/handson-ml2/blob/master/tools_numpy.ipynb

numpy – 배열생성, 참조

- numpy 배열 생성방법.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5]) # list → ndarray
print(type(arr), arr.dtype)
np.array((1, 2, 3, 4, 5)) # tuple → ndarray
np.zeros((2,5))
np.ones((3,4))
np.full((2,5), np.pi)
np.empty((2,5))
np.linspace(0, 10, 101)
```

- 배열의 차원

```
arr0 = np.array(42)
arr1 = np.array([1, 2, 3, 4, 5, 6])
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

print(arr0.ndim)
print(arr1.ndim)
print(arr2.ndim)
```

✓ 차원변경

```
arr1.reshape(2,3)
arr1.shape = (2,3)  # arr1 이 변환됨.
arr1 = np.array([1, 2, 3, 4, 5, 6])
arr1.reshape(2,-1)
```

- NumPy 배열 참조

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[0])
```

- 2차원 배열에 접근하기

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])
```

- 3차원 배열에 접근하기

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print(arr[0, 1, 2])
```

- 음수 인덱싱 : 끝에서부터 배열참조.

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Last element from 2nd dim: ', arr[1, -1])
```

numpy – 배열 슬라이싱

- 배열의 부분을 참조하는 기법

- 구문

```
[start:end]
[start:end:step]

# ==========================
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
print(arr[4:])
print(arr[:4])
print(arr[-3:-1])
print(arr[1:5:2])  # step값을 사용
print(arr[::2])  # 전체 대상 step 적용
print(arr[::-1])  # reverse
```

- fancy indexing

  ✓임의의 행, 열 선택 가능

```
b = np.arange(48).reshape(4, 12)
b

b[(0,2), 2:5]  # 0과 2 행,  2 ~ 4 열
b[:, (-1, 2, -1)]  # 마지막, 2, 마지막 열 선택
```

- 축단위 slicing

  ✓boolean indexing(한축에서 선택)

```
# ==========================
b = np.arange(48).reshape(4, 12)
b
rows_on = np.array([True, False, True, False])
b[rows_on, :]  #  == b[(0, 2), :]

cols_on = np.array([False, True, False] * 4)
b[:, cols_on]  # 모든 행, 열 1, 4, 7, 10
```

- 2축 이상에서 선택할 때

  ✓np.ix_() 이용하여 축당 선택 지정

```
np.ix_(rows_on, cols_on)
b[np.ix_(rows_on, cols_on)]
```

numpy – 배열 슬라이싱

✓슬라이싱 - 브로드캐스팅

```
a = np.array([1, 2, 3, 4, 5, 6, 7])
a[2:5] = -1
a
```

✓슬라이스는 원본배열을 참조한다.

```
a_slice = a[2:6]
a_slice[1] = 1000
a  # 원본 배열이 수정됨.

# 원본 배열을 수정하면 ?
a[3] = 2000
a_slice

 # 원본 배열을 보존하려면.
another_slice = a[2:6].copy()
another_slice[1] = 3000
a
```

numpy – 배열 연산

- 원소끼리 연산
  - ✓사칙연산

```
import numpy as np
a = np.array([1,2,3])
b = np.array([4,5,6])

a+b
np.add(a, b)
a*b
a/b
np.divide(a,b)
```

  - ✓벡터연산

```
import numpy as np
a = np.array([[1,2],[3,4]])
b = np.array([[5,6],[7,8]])

np.dot(a, b) # 행렬의 곱
```

- 차원이 다른 배열끼리 연산 (broadcasting)

```
a = np.array([[0,0,0],[10,10,10],[20,20,20],[30,30,30]])
b = np.array([1,2,3])
c = 5

a + b
a * b
a + c
```

- 조건 연산

```
m = np.array([20, -5, 30, 40])
m < [15, 16, 35, 36]
m < 25  # m < [25, 25, 25, 25] 와 동일
m[m < 25]
```

- 통계기능

```
a = np.arange(12).reshape(4, 3)
print(a)
print("평균 =", a.mean())

for func in (a.min, a.max, a.sum, a.prod, a.std, a.var):
    print(func.__name__, "=", func())
for func in (np.sqrt, np.exp, np.log, np.sign, np.ceil, np.cos):
    print("₩n", func.__name__)
    print(func(a))
```

numpy – 데이터 유형

- 모든 Python 데이터 유형 사용가능
- NumPy의 데이터 유형 약자.

```
i – int
B – bool
U – unsigned int
f - float
m – time delta
M - datatime
O - object
S - string
U – Unicode string
```

- 배열의 데이터 유형 확인방법

```
import numpy as np
arr = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
```

- 정의된 데이터 유형으로 배열 생성방법

```
arr = np.array([1, 2, 3, 4], dtype='S')  # 문자열 형식으로 배열 생성
arr = np.array([1, 2, 3, 4], dtype='i4')  # 4바이트 정수 배열
print(arr.dtype)
```

- 기존 배열의 데이터 유형 변환

✓ 매개변수 값 으로 사용하여 데이터 유형을 부동 소수점에서 정수로 변경

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])
newarr = arr.astype(int)
print(newarr)
print(newarr.dtype)
```

✓ 데이터 유형을 정수에서 부울로 변경

```
arr = np.array([1, 0, 3])
newarr = arr.astype(bool)
```

pandas

- 데이터의 조작, 필터링, 그룹핑, 변환기능 제공.
- " Panel Data " + " Python Data Analysis"

- pandas 시작하기

```
> pip install pandas

import pandas

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pandas.DataFrame(mydataset)
print(myvar)
```

- Series

✓ 테이블의 특정 행/열에 해당하는 1차원 배열.

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

- 레이블

✓ 값을 참조하는데 사용됨.

✓ 별도로 지정하지 않으면 인덱스 번호로 레이블이 지정됨.

```
print(myvar[0])
```

- 레이블 만들기

✓ index인수를 사용 하여 고유한 레이블의 이름을 지정.

```
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar)
print(myvar["x"])
```

https://ml-ko.kr/homl2/tools_pandas.html
https://github.com/ageron/handson-ml2/blob/master/tools_pandas.ipynb

pandas - DataFrame

- 2차원 배열 또는 행과 열이 있는 테이블과 같은 2차원 데이터 구조.

- == list로 생성하기 ==

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
```

- == dict 로 생성하기 ==

```
import pandas as pd
data = {'Name':['홍길동', '홍길순', '이철수', '김영희'],
        'Age':[27, 24, 22, 32],
        'Address':['천안', '천안', '아산', '서울'],
        'Qualification':['MS', 'BS', 'PhD', 'BS']}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
```

- 열 작업

```
df['New'] = df['Name'] + df['Age'].astype(str)  # 열추가
df.pop('New')          # 열제거
```

- 행작업

```
df2 = pd.DataFrame([['나', 36]], columns = ['Name','Age'])
df = df.append(df2)# 행추가
df.drop(0)                          # 행제거
```

- 행분석

```
df.head() / df.tail()
df[df.Age > 30]
df.nlargest(2, 'Age')
df.sample(frac=0.5)
```

- query 사용

```
df.query('Age > 30')
df.query("Name.str.startswith('홍')")
df.query("Age > 25 and Name.str.startswith('홍')")
```

- 임의의 행/열 선택

```
df.iloc[0:2]
df.iloc[:, [0,1,2]]
df.loc[:, 'Name':'Address']
df.loc[df['Age'] > 25, ['Age', 'New']]
```

- 기타 기능

```
df.shape
df.info()
df.describe()
df['Name'].value_counts()
```

pandas – 데이터 cleansing

- 데이터 세트에서 잘못된 데이터를 처리하는 작업.

```
df = pd.read_csv('pandas_ex.csv')
```

✓ no data : (22행의 "Date", 18행과 28행의 "Calories")
✓ 잘못된 형식 : (26행의 "Date")
✓ 잘못된 데이터 : (7행의 'Duration')
✓ 중복 : (행 11, 12)

- 빈 데이터 처리

✓ 읽었을 때 Null, NaN 등으로 표현됨.
✓ 빈 셀은 데이터를 분석할 때 모델의 정확성에 악영향 줌.

✓ 행 제거 : 데이터 세트가 매우 크고 빈데이타 개수는 작을 경우.

```
new_df = df.dropna()
```

✓ 새값 정의 : 적절한 새값을 지정할 수 있을 때

```
df["Calories"].fillna(130, inplace = True)  # 원본 DataFrame을 변경
```

- < 실습> 23행 Date 값 채우기.

- 잘못된 데이터 수정

✓ 데이터 세트를 보고 잘못된 데이터를 발견한 경우
✓ duration 은 보통 30 ~ 60(분), 450은 이상 데이타로 의심됨.
✓ 7행에서 "기간" = 45로 설정:

```
df.loc[7, 'Duration'] = 45
```

✓ 일괄수정을 위해 규칙 사용 가능
✓ 값이 120보다 크면 120으로 설정.

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.loc[x, "Duration"] = 120
```

✓ "Duration"이 120보다 크면 삭제

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
```

- 중복 제거

✓ 11행과 12행이 중복의심됨.

```
df.duplicated()
```

< 실습 –중복행만 확인하려면? >

pandas – 데이터 cleansing

- 잘못된 데이터 형식 교정
  - ✓ 22행과 26행 'Date' 열.
  - ✓ to_datetime() 함수 이용.

```
df.loc[22, 'Date'] = '2020-12-22'
df['Date'] = pd.to_datetime(df['Date'])
```

- 데이터 저장
  - ✓ df.to_csv('pandas_ex2.csv', index=False)

pandas – 데이터 조작함수들

FUNCTION	설명	FUNCTION	설명
index()	인덱스(행 레이블) 반환	loc[]	인덱스 레이블을 기반으로 행을 검색
insert()	열을 삽입	iloc[]	인덱스 위치를 기반으로 행을 검색
add()	항목간 더하기	ix[]	인덱스 레이블 또는 인덱스 위치를 기반으로 행을 검색
sub()	항목간 빼기	rename()	인덱스 레이블 또는 열 이름을 변경
mul()	항목간 곱셈	drop()	행이나 열을 삭제
div()	항목간 나누기	pop()	행이나 열을 삭제
unique()	고유한 값 추출	sample()	임의의 행 또는 열 샘플링 반환
nunique()	고유한 값의 수 반환	nsmallest()	열에서 가장 작은 값을 가진 행을 반환
value_counts()	시리즈 내에서 각 고유 값이 발생하는 횟수	nlargest()	열에서 가장 큰 값을 가진 행을 반환
columns()	열 레이블 반환	shape()	차원 반환
axes()	축을 나타내는 목록	ndim()	차원 반환
isnull()	null 값이 있는 행을 추출	dropna()	Null 값이 있는 행/열을 삭제
notnull()	null이 아닌 값이 있는 행을 추출	fillna()	NaN 을 값으로 바꿈
between()	열 값이 미리 정의된 범위 사이에 속하는 행 추출	rank()	순위를 매김
isin()	정의된 컬렉션에 열 값이 있는 행을 추출	query()	하위 집합을 추출하기 위한 문자열 기반 구문
dtypes()	각 열의 데이터 형식이 포함된 Series를 반환	copy()	복사본 생성
astype()	시리즈의 데이터 유형을 변환	duplicated()	중복 값이 있는 행을 추출
values()	Numpy 표현을 반환	drop_duplicates()	중복 행을 식별하고 필터링을 통해 제거
sort_values()- Set1, Set2	오름차순 또는 내림차순으로 정렬	set_index()	인덱스(행 레이블) 설정
sort_index()	값 대신 인덱스 위치 또는 레이블을 기반으로 정렬	reset_index()	인덱스 재설정
		where()	조건을 만족하는 결과 반환

pandas - 상관계수

- corr()함수 : 데이터 세트의 각 열 간의 상관계수 계산.
- "숫자가 아닌" 열은 무시
- -1 ~ 1 의 범위를 가짐.
  - ✓ 1은 완전한 양의 상관관계, -1은 완전한 음의 상관관계를 나타냄.
  - ✓ 0에 가까울 수록 두 변수는 상관성이 없음을 나타냄.

```
df.corr()
```

- < 해석 >
  - ✓ "Duration - Calories"의 상관계수 : 0.922721.
    - 매우 강한 양의 상관관계. 운동 시간이 길수록 더 많은 칼로리를 소모.
  - ✓ "Duration - Maxpulse"의 상관계수 : 0.009403.
    - 매우 약한 상관관계. 운동 시간과 최대맥박은 거의 관계없음.

matplotlib

- 출판 가능한 퀄리티의 그래프와 이미지를 만들기 위한 편리한 인터페이스를 제공
- Matplotlib Pyplot
- matplotlib 대부분의 기능은 pyplot 서브모듈에 위치함.

```
> pip install matplotlib

Import matplotlib
print(matplotlib.__version__)


import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

- 선그래프

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('pandas_ex2.csv')

df.plot()
plt.show()
```

- 산포도

```
df.plot(kind = 'scatter', x = 'Pulse', y = 'Calories')
plt.show()
```

- 가중치 표현한 산포도

```
from numpy.random import rand
x, y, scale = rand(3, 100)
scale = 500 * scale ** 5
plt.scatter(x, y, s=scale)
plt.show()
```

https://jehyunlee.github.io/2020/04/21/Python-DS-10-matplotlib_Tools/
https://ml-ko.kr/homl2/tools_matplotlib.html
https://github.com/ageron/handson-ml2/blob/master/tools_matplotlib.ipynb

matplotlib

- 마커

✓ marker 인수로 지정된 마커로 포인트 강조.

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o')
plt.show()
```

- Matplotlib 라인

✓ 선의 스타일을 변경.

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

- 레이블 및 제목 생성

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.title('Pulse-Calories')
plt.show()
```

- 범례

```
x = np.linspace(-1.4, 1.4, 50)
plt.plot(x, x**2, "r--", label="Square function")
plt.plot(x, x**3, "g-", label="Cube function")
plt.legend(loc="best")
plt.grid(True)
plt.show()
```

matplotlib – subplot

- 여러 개의 그래프를 그림

```
import matplotlib.pyplot as plt
import numpy as np


#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```

- fig, ax 이용한 다중 그래프

```
import matplotlib.pyplot as plt

x = np.linspace(-2, 2, 200)
# 위, 아래 세트로 그래프 그림
Fig1, (ax_top, ax_bottom) = plt.subplots(2, 1, sharex=True)
Fig1.set_size_inches(10,5)
# 위쪽 그래프 정의
Line1, line2 = ax_top.plot(x, np.sin(3*x**2),  " r- " , x, np.cos(5*x**2),  "
b- " )
# 아래쪽 그래프
Line3, = ax_bottom.plot(x, np.sin(3*x),  " r- " )
Ax_top.grid(True)

#두번째 그래프
Fig2, ax = plt.subplots(1, 1)
ax.plot(x, x**2)
plt.show()
```

matplotlib – color

- 색상과 크기로 구분한 산포도

```
for color in ['red', 'green', 'blue']:
    n = 100
    x, y = rand(2, n)
    scale = 500.0 * rand(n) ** 5
    plt.scatter(x, y, s=scale, c=color, alpha=0.3,
edgecolors='blue')

plt.grid(True)

plt.show()
```

- ColorMap을 사용하는 방법

✓ 컬러맵 cmap값과 함께 키워드 인수 로 컬러맵을 지정.

✓ 예제 : 색 배열을 만들고 산점도에 컬러맵을 지정.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='viridis')
plt.colorbar()
plt.show()
```

- < 실습 >

✓ 컬러맵을 다양하게 선택하고 그려보자.

matplotlib

- 막대그래프

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y)
plt.show()
```

- 히스토그램

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x, bins=20, cumulative=True)
plt.hist(x, bins=20, cumulative=False)
plt.show()
```