빅데이터 기반 AI 응용 솔루션 개발자 전문과정

교과목명 : 분석라이브러리 활용

- 평가일 : 22.7.8
- 성명 : 박혜린
- 점수 : 36.5

2개만 O - Q1. 표준정규분포 기반의 2행 3열 배열을 랜덤하게 을 출력하세요.

In [1]:

```
import pandas as pd
import numpy as np
```

In [466]:

```
a = np.random.randn(2,3)
print(a)


# size , shape
# 크기
```

```
[[ 0.04796212 -0.23041315  1.28843401]
 [ 0.43848518 -0.4196695   1.14855368]]
```

In [448]:

```
# 자료형
a.dtype
```

Out[448]:

```
dtype('float64')
```

In [451]:

```
# 차원
a.ndim
```

Out[451]:

```
3
```

O Q2. arange(), reshape() 이용 1차원 2차원 3차원 배열을 아

[0 1 2 3 4 5 6 7 8 9]

[[0 1 2 3 4]
[5 6 7 8 9]]

```
[[[0 1 2 3 4]
[5 6 7 8 9]]]
```

In [164]:

```
a = np.arange(10).reshape(10)
print(a)
b = a.reshape(2,5)
print(b)
c = np.arange(10).reshape(1,2,5)
print(c)
```

```
[0 1 2 3 4 5 6 7 8 9]
[[0 1 2 3 4]
 [5 6 7 8 9]]
[[[0 1 2 3 4]
  [5 6 7 8 9]]]
```

Q3. 1 ~ 100 까지 배열에서 3과 7의 공배수인 것만을 출력하세

In [453]:

```
np.arange(1,101).map(lambda x : x%3==0 & x%7==0)
```

```
---------------------------------------------------------
----
AttributeError                            Traceback (most
t)
Input In [453], in <cell line: 1>()
----> 1 np.arange(1,101).map(lambda x : x%3==0 & x%7==0)

AttributeError: 'numpy.ndarray' object has no attribute 'm
```

In [ ]:

```

```

O Q4. 아래 3차원 배열을 생성하여 출력한 후 1차원으로 변환
사용)

```
[[[ 0 1 2 3 4]
[ 5 6 7 8 9]]

[[10 11 12 13 14]
[15 16 17 18 19]]

[[20 21 22 23 24]
[25 26 27 28 29]]]
```

Contents

In [63]:

```
a = np.arange(0,30).reshape(3,2,5)
print(a)
print(a.ndim)

b = a.reshape(30)
print(b)
b.ndim
```

```
[[[ 0  1  2  3  4]
  [ 5  6  7  8  9]]

 [[10 11 12 13 14]
  [15 16 17 18 19]]

 [[20 21 22 23 24]
  [25 26 27 28 29]]]
3
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 24 25 26 27 28 29]
```

Out[63]:

```
1
```

O Q5. array2d에서 인덱스를 이용해서 값을 선택하고 리스트로 요.

arr2d = np.arange(1,10).reshape(3,3)

[3, 6]

[[1, 2],
[4, 5]]

[[1, 2, 3]
[4, 5, 6]]

In [76]:

```
arr2d = np.arange(1,10).reshape(3,3)
print(arr2d)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [85]:

```
print(arr2d[:2,2],'\n')
print(arr2d[:2,:2],'\n')
print(arr2d[:2,:])
```

```
[3 6]

[[1 2]
 [4 5]]

[[1 2 3]
 [4 5 6]]
```

O Q6. zeros_like, ones_like, full_like 함수 사용 예를 작성하

In [110]:

```
a = np.random.randint(1,11,(5,5))
print(a,'\n')
b = np.zeros_like((a))
print(b,'\n')
c = np.ones_like((a))
print(c,'\n')
d = np.full_like((a),5)
print(d)
```

```
[[ 3  5  8  2  6]
 [ 7 10  7  8  2]
 [ 8  5  4  6  1]
 [ 2  7 10  4  2]
 [ 9  5  9  5  8]]

[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]

[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]

[[5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]]
```

Q7. 10 ~ 20 사이의 정수 난수로 10행 5열 2차원 배열을 생성하
출력하세요.

In [293]:

```
random_array = np.random.randint(10,20,(2,10,5))
random_array
```

Out[293]:

```
array([[[19, 17, 10, 18, 16],
        [19, 12, 16, 15, 16],
        [15, 18, 16, 17, 13],
        [13, 11, 16, 11, 13],
        [15, 10, 19, 18, 11],
        [10, 19, 14, 19, 19],
        [14, 14, 15, 14, 13],
        [13, 16, 19, 18, 19],
        [18, 12, 16, 15, 11],
        [15, 16, 15, 11, 18]],

       [[12, 11, 15, 16, 13],
        [14, 15, 13, 18, 14],
        [15, 18, 11, 10, 16],
        [16, 10, 14, 15, 13],
        [13, 18, 13, 12, 11],
        [11, 17, 16, 18, 17],
        [16, 16, 17, 19, 15],
        [14, 17, 13, 16, 16],
        [15, 19, 10, 17, 12],
        [15, 16, 19, 18, 13]]])
```

In [455]:

```
random_array = pd.save_csv('dataset/random_array.csv')
```

```
---------------------------------------------------------
----
AttributeError                            Traceback (most
t)
Input In [455], in <cell line: 1>()
----> 1 random_array = pd.save_csv('dataset/random_array.c

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\__i
n __getattr__(name)
    257     from pandas.core.arrays.sparse import SparseA
Array
    259     return _SparseArray
--> 261 raise AttributeError(f"module 'pandas' has no att
")

AttributeError: module 'pandas' has no attribute 'save_csv
```

In [ ]:

Q8. df = sns.load_dataset('titanic')로 불러와서 다음 작업을

- 전체 칼럼중 'survived'외에 모든 칼럼을 포함한 df_x를 산출한 후 dataset/df
- df_x.pkl을 데이터프레임 df_x 이름으로 불러온 후 앞 5개 행을 출력한다.

In [458]:

```
import seaborn as sns

df = sns.load_dataset('titanic')
df.head()
```

Out[458]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	w
2	1	3	female	26.0	0	0	7.9250	S	Third	w
3	1	1	female	35.0	1	0	53.1000	S	First	w
4	0	3	male	35.0	0	0	8.0500	S	Third	

In [457]:

```
df_x = df.drop(columns=['survived'])
df_x.head()
```

Out[457]:

	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adu
0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	female	38.0	1	0	71.2833	C	First	woman	
2	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	female	35.0	1	0	53.1000	S	First	woman	
4	3	male	35.0	0	0	8.0500	S	Third	man	

In [ ]:

```
df_x = pd.
```

In [ ]:

```

```

O Q9. df = sns.load_dataset('titanic')로 불러와서 deck 열에
요.

In [116]:

```
df['deck'].isnull().sum()
```

Out[116]:

```
688
```

O Q10. Q9의 df에서 각 칼럼별 null 개수와 df 전체의 null 개

In [120]:

```
df.isnull().sum()
```

Out[120]:

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [121]:

```
df.isnull().sum().sum()
```

Out[121]:

```
869
```

아래 tdf 데이터프레임에서 Q11 ~ Q12 작업을 수행하세요.

In [379]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
tdf = df[['survived','sex','age','class']]
tdf.head()
```

Out[379]:

	survived	sex	age	class
0	0	male	22.0	Third
1	1	female	38.0	First
2	1	female	26.0	Third
3	1	female	35.0	First
4	0	male	35.0	Third

Q11. age를 7개 카테고리로 구분하는 새로운 칼럼 'cat_age'를 카테고리 구분을 수행하는 사용자 함수를 만들고 그 함수를 ag tdf1에 저장하고 출력하세요.

[카테고리]
age <= 5: cat = 'Baby'
age <= 12: cat = 'Child'
age <= 18: cat = 'Teenager'
age <= 25: cat = 'Student'
age <= 60: cat = 'Adult'
age > 60 : cat = 'Elderly'

Contents

In [336]:

```
tdf['age'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 891 entries, 0 to 890
Series name: age
Non-Null Count  Dtype
--------------  -----
714 non-null    float64
dtypes: float64(1)
memory usage: 7.1 KB
```

In [380]:

```
def catage(x):
    cat=''
    if x <= 5:
        return 'Baby'
    elif x <=12:
        return 'Child'
    elif x <=18:
        return 'Teenager'
    elif x <=25:
        return 'Student'
    elif x <=60:
        return 'Adult'
    elif x > 60:
        return 'Elderly'
    else :
        pass
```

In [386]:

```
tdf1 = tdf.copy()
tdf1['cat_age'] = tdf1['age']
tdf1.head()
```

Out[386]:

	survived	sex	age	class	cat_age
0	0	male	22.0	Third	22.0
1	1	female	38.0	First	38.0
2	1	female	26.0	Third	26.0
3	1	female	35.0	First	35.0
4	0	male	35.0	Third	35.0

In [387]:

```
tdf1 = tdf1.cat_age.apply(catage)
tdf1
```

Out[387]:

```
0      Student
1        Adult
2        Adult
3        Adult
4        Adult
        ...
886      Adult
887    Student
888       None
889      Adult
890      Adult
Name: cat_age, Length: 891, dtype: object
```

Q12. tdf1의 sex, class 칼럼을 '_'으로 연결한 'sc'칼럼을 추가
요.

In [ ]:

```

```

In [ ]:

```

```

2.5 O - Q13. join() 메소드는 두 데이터프레임의 행 인덱스를 기
식데이터를 가져와서 join() 메소드로 아래와 같이 결합한 후 대
요.

- df1과 df2의 교집합만 출력되도록 결합하여 df3에 저장하고 출력
- df3에서 중복된 칼럼을 삭제한 후 블린 인덱싱을 이용하여 eps가 3000 보다
터를 선택하여 데이터프레임을 생성하고 df4 이름으로 저장 및 출력하세요.

In [390]:

```
df1 = pd.read_excel('./dataset/stock price.xlsx', index_co
df2 = pd.read_excel('./dataset/stock valuation.xlsx', inde
```

In [393]:

```
df3 = df1.join(df2, how='inner')
df3
```

Out[393]:

id	stock_name	value	price	name	eps	bps
130960	CJ E&M	58540.666667	98900	CJ E&M	6301.333333	54068
139480	이마트	239230.833333	254500	이마트	18268.166667	295780
145990	삼양사	82750.000000	82000	삼양사	5741.000000	108090
185750	종근당	40293.666667	100500	종근당	3990.333333	40684
204210	모두투어리츠	3093.333333	3475	모두투어리츠	85.166667	5335

In [417]:

```
df4 = df3.drop(columns =['name'])
df4.head()
```

Out[417]:

id	stock_name	value	price	eps	bps	per
130960	CJ E&M	58540.666667	98900	6301.333333	54068	15.695091
139480	이마트	239230.833333	254500	18268.166667	295780	13.931338
145990	삼양사	82750.000000	82000	5741.000000	108090	14.283226
185750	종근당	40293.666667	100500	3990.333333	40684	25.185866
204210	모두투어리츠	3093.333333	3475	85.166667	5335	40.802348

In [ ]:

```
df3에서 중복된 칼럼을 삭제한 후 블린 인덱싱을 이용하여 eps가 3000
데이터프레임을 생성하고 df4 이름으로 저장 및 출력하세요.(단, '<' 외
```

In [468]:

```
df4 = df4(['eps'] < 3000) | (['stock_name']== '이마트')
```

```
-----------------------------------------------------------
----
TypeError                                 Traceback (most
t)
Input In [468], in <cell line: 1>()
----> 1 df4 = df4(['eps'] < 3000) | (['stock_name']== '이마

TypeError: '<' not supported between instances of 'list' a
```

Q14. 배열 a에 대하여 3차원 자리에 2차원을 2차원 자리에 1차 넣어서 변환하여 출력하세요

In [161]:

```
a = np.arange(6).reshape(1,2,3)
print(a,a.shape,'\n')

print(a.ndim)
b = a.reshape(2,3)
print(b,b.shape,'\n')
print(b.ndim)
c = b.reshape(6)
print(c,c.shape,'\n')
print(c.ndim)
d = c.reshape(1,2,3)
print(d,d.shape)
print(d.ndim)
```

```
[[[0 1 2]
  [3 4 5]]] (1, 2, 3)

3
[[0 1 2]
 [3 4 5]] (2, 3)

2
[0 1 2 3 4 5] (6,)

1
[[[0 1 2]
  [3 4 5]]] (1, 2, 3)
3
```

Q15. 'mpg'를 'kpl' 로 환산하여 새로운 열을 생성하고 반올림하지 처음 5개행을 출력하세요.

In [167]:

```
# read_csv() 함수로 df 생성
import pandas as pd
auto_df = pd.read_csv('./dataset/auto-mpg.csv')
# 열 이름을 지정
auto_df.columns = ['mpg','cylinders','displacement','horse
                   'acceleration','model year','origin','name']
print(auto_df.head(3))
```

```
    mpg  cylinders  displacement horsepower  weight  accel
year  \
0  18.0          8         307.0        130    3504
70
1  15.0          8         350.0        165    3693
70
2  18.0          8         318.0        150    3436
70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
```

In [432]:

```
auto_df['kpl'] = auto_df['mpg']/2.352
auto_df.head()
```

Out[432]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	or
0	18.0	8	307.0	130	3504	12.0	70	
1	15.0	8	350.0	165	3693	11.5	70	
2	18.0	8	318.0	150	3436	11.0	70	
3	16.0	8	304.0	150	3433	12.0	70	
4	17.0	8	302.0	140	3449	10.5	70	

In [ ]:

Q16. './dataset/stock-data.csv'를 데이터프레임으로 불러와서 Date 칼럼을 datetime 타입으로 변환한 후에 년, 월, 일로 분리하고 year를 인덱스로 셋팅하여 출

In [180]:

```
df_time = pd.read_csv('./dataset/stock-data.csv')
df_time.head()
```

Out[180]:

	Date	Close	Start	High	Low	Volume
0	2018-07-02	10100	10850	10900	10000	137977
1	2018-06-29	10700	10550	10900	9990	170253
2	2018-06-28	10400	10900	10950	10150	155769
3	2018-06-27	10900	10800	11050	10500	133548
4	2018-06-26	10800	10900	11000	10700	63039

In [440]:

```
from datetime import datetime
df_time['Date'].dtype
```

Out[440]:

```
dtype('O')
```

In [ ]:

Q17. titanic 데이터셋(titanic = sns.load_dataset('titanic'))에 5개 열을 기준으로 그룹화를 수행한 후 아래와 같이 출력하였다. 다

5개 열 : ['age','sex', 'class', 'fare', 'survived']

- 그룹별 평균 출력
- 그룹별 최대값 출력

In [218]:

```
titanic = sns.load_dataset('titanic')
df3 = titanic.groupby(['class'])
df3
```

Out[218]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x
0>
```

In [ ]:

In [ ]:

```
```

Q18. titanic 데이터셋에서 'Third'그룹만을 선택해서 group3 표를 출력하세요.

In [24]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
df.head()
```

Out[24]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	w
2	1	3	female	26.0	0	0	7.9250	S	Third	w
3	1	1	female	35.0	1	0	53.1000	S	First	w
4	0	3	male	35.0	0	0	8.0500	S	Third	

1개만 O - Q19. titanic 데이터셋에서 다음 전처리를 수행하세.

1. df에서 중복 칼럼으로 고려할 수 있는 컬럼들(6개 내외)을 삭제한 후 나머지 df1 이름으로 저장 후 출력하세요.
2. df1에서 null값이 50% 이상인 칼럼을 삭제 후 df2 이름으로 저장하고 출력하
3. df2에서 결측값이 있는 age 칼럼에 대해서 평균값으로 대체 처리를 수행하
4. df2에서 결측값이 있는 embarked 칼럼에 대해서 앞행의 값으로 대체 처리
5. df2 문자로 되어있는 칼럼들을 레이블 인코딩 수행하여 숫자로 변환 후 df2.

In [238]:

```
df.head()
```

Out[238]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	w
2	1	3	female	26.0	0	0	7.9250	S	Third	w
3	1	1	female	35.0	1	0	53.1000	S	First	w
4	0	3	male	35.0	0	0	8.0500	S	Third	

In [232]:

```
df1 = df.drop(columns = ['class','who','adult_male','embar
df1.head()
```

Out[232]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	deck	al
0	0	3	male	22.0	1	0	7.2500	S	NaN	Fa
1	1	1	female	38.0	1	0	71.2833	C	C	Fa
2	1	3	female	26.0	0	0	7.9250	S	NaN	T
3	1	1	female	35.0	1	0	53.1000	S	C	Fa
4	0	3	male	35.0	0	0	8.0500	S	NaN	T

In [234]:

```
df1.isnull().sum()
```

Out[234]:

```
survived      0
pclass        0
sex           0
age         177
sibsp         0
parch         0
fare          0
embarked      2
deck        688
alone         0
dtype: int64
```

In [236]:

```
df2 = df1.drop(columns = ['deck'])
df2.head()
```

Out[236]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	alone
0	0	3	male	22.0	1	0	7.2500	S	False
1	1	1	female	38.0	1	0	71.2833	C	False
2	1	3	female	26.0	0	0	7.9250	S	True
3	1	1	female	35.0	1	0	53.1000	S	False
4	0	3	male	35.0	0	0	8.0500	S	True

In [ ]:

```
df2에서 결측값이 있는 age 칼럼에 대해서 평균값으로 대체 처리를 수행
df2에서 결측값이 있는 embarked 칼럼에 대해서 앞행의 값으로 대체 처
df2 문자로 되어있는 칼럼들을 레이블 인코딩 수행하여 숫자로 변환 후
```

In [244]:

```
df2['age'].mean()
```

Out[244]:

```
29.69911764705882
```

In [251]:

```
df2.age = fillna(df2['age'].mean())
```

```
---------------------------------------------------------
----
NameError                                 Traceback (most
t)
Input In [251], in <cell line: 1>()
----> 1 df2.age = fillna(df2['age'].mean())

NameError: name 'fillna' is not defined
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Q20. 보스톤 주택가격 데이터를 탐색한 후 가장 중요한 독립변
시각화하여 설명하세요.

In [254]:

```
import warnings
warnings.filterwarnings('ignore')

from sklearn.datasets import load_boston
# boston 데이타셋 로드
boston = load_boston()

# boston 데이타셋 DataFrame 변환
bostonDF = pd.DataFrame(boston.data , columns = boston.fea

# boston dataset의 target array는 주택 가격임. 이를 PRICE 컬럼
bostonDF['PRICE'] = boston.target
print('Boston 데이타셋 크기 :',bostonDF.shape)
bostonDF.head()
```

```
Boston 데이타셋 크기 : (506, 14)
```

Out[254]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTR
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

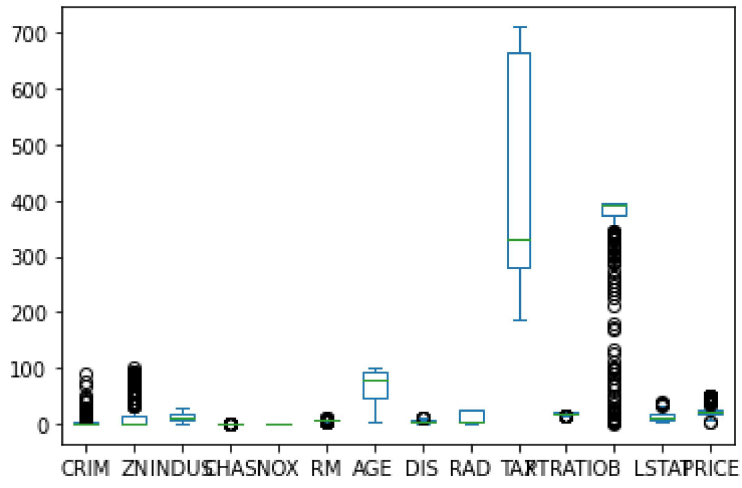
In [465]:

```
import matplotlib.pyplot as plt

bostonDF.plot(kind='box')
```

Out[465]:

```
<AxesSubplot:>
```



In [ ]:

Contents