과정명		모듈명		회차명	

◈ 학습열기

데이터를 어떤 방법으로 수집하던 크롤링이던 공공포털에서 가지고 오던지 아니면 실무자들로부터 가져오던지 설문을 통해 가져오던지, 응답과정이나 데이터 수집과정에서 데이터가 누락되거나, 이상치가 발생하곤 합니다. 이상치란 수집과정에서 잘못 입력된 데이터 일수도 있고 예를들면 m단위로 수집해야 하는데 일부는 m단위, 일부는 inch 단위라 단위불일치로 인한 오류뿐만아니라 통계에서는 특정 값이 지나치게 크거나 낮을 경우에도 이를 이상치로 취급합니다. 예를들어서 한 집단의 연봉 평균을 보려고 하는데 대부분 5000~6000 사이인데 그중에 연봉 3억이 추가되어서 평균값을 계산한다면 집단에 대한 평가가 상당히 왜곡되어서 나타납니다. 물론 집단의 구성원이 충분히 많아서 3억을 상쇄하고 남을 정도라면 상관없지만 집단에 구성원이 작아서 결과를 왜곡하게 된다면 연봉 3억도 이상치가 됩니다. 이 장에서는 누락된 값이나 이상치등을 제거하여 데이터 분석 결과의 오차를 방지하는 방법에 대해서 살펴보도록 하겠습니다.

내레이션	학습자가 학습 화면에 구성한 내용을 보고 있다고 가정하고, 학습 설명으로 함께 들어야 할 음성 내용을 자세하게 기입해 주세요. **➔ 이건 나래이션해요? 저도 이런건 나래이션하면 좋겠어요..ㅜㅜ**

과정명		모듈명		회차명	

◈ 학습내용

- 데이터 변환과 정제

◈ 학습목표

- 데이터의 누락치및 이상치를 제거할 수 있다
- 데이터의 타입을 변환할수 있으며 구간나누기등을 통해 문자열 자료형을 범주형자료형으로 변경할 수 있고 원핫인코딩을 할 수 있다

내레이션

(기입하지 않습니다.)

간지 부분

◈ 데이터프레임

누락데이터 처리, 이상데이터 처리

내레이션

(기입하지 않습니다.)

◈ 누락데이터처리

- 머신러닝이나 딥러닝에서 데이터의 누락은 전체 예측결과에 많은 영향을 미칩니다.
- 보다 예측력이 높고 신뢰성 있는 결과를 얻으려면 누락된 데이터, 중복된 데이터, 이상치 데이터에 대한 바른 처리를 해야 합니다.
- 데이터 프레임에는 데이터가 누락될 경우 NaN(not a number)으로 표기됩니다.
- 데이터 분석시 보통 데이터의 크기는 수십만건에서 수백만건이 되기도 합니다. 엑셀등의 프로그램에서 데이터 터를 눈으로 확인해서 오류가 있는 데이터를 검증해내는것은 거의 불가능에 가깝습니다.
- 그래서 파이썬에 제공하는 다양한 라이브러리와 함수등을 이용해 누락되거나 특정 범위를 벗어나는 데이터에 대한 처리방법에 대해서 알아보겠습니다.

내레이션

◈ isnull 함수

- 데이터가 NaN이면 True 아니면 False 를 반환한다.

```
import pandas as pd
import numpy as np

#nan 은 numpy를 사용해 직접 입력 가능하다
s = pd.Series([1,2,3,4,np.nan])
print( s.isnull() )
```

```
0    False
1    False
2    False
3    False
4     True
```

◈ sum함수

- 인자들의 합계를 반환하는데 isnull 함수 사용후 결과값에 sum함수를 사용하면 True개수를
- 반환한다

```
import pandas as pd
import numpy as np

#nan 은 numpy를 사용해 직접 입력 가능하다
s = pd.Series([1,2,3,4,np.nan])
print( s.isnull() )
print( "NaN 값 개수 ", s.isnull().sum() )
```

```
NaN 값 개수  1
```

내레이션

◈ dropna 함수

- DataFrame.dropna(self, axis=0, how='any', thresh=None, subset=None, inplace=False)[source]

- subset에서 지정한 필드들에 NaN값이 있는 행을 삭제한다. inplace=True로 지정하면 자신의 데이터가 수정된다. 필드 지정방법은 list형태이다. subset=[‘필드1’, ‘필드2’]

- thresh : int값 이 필드에서 부여한 개수만큼 NaN이 없는 행이나 열을 삭제

- how : any, all any일 경우 행이나 컬럼값중에 하나라도 있으면 행이나 열을 삭제, all의 경우 행이나 컬럼의 값들이 모두 NaN 이 있을때 행이나 컬럼을 삭제합니다.

- axis : 0은 행 1은 열을 지칭. 삭제할 방향을 지시한다

내레이션

◈ reset_index 함수

- DataFrame 인덱스를 재지정합니다. 새로 필드를 추가하도록 되어 있으므로 drop옵션을 True로 지정해야 합니다.

```
import pandas as pd
import numpy as np

#nan 은 numpy를 사용해 직접 입력 가능하다
s = pd.Series([1,2,3,4,np.nan])

data = s.dropna(how='any', axis=0)
print(data)

data = data.reset_index(drop=True)
```

내레이션

◈ 누락데이터처리

```
#파일명 : exam11_1.py

import pandas as pd

#header가 3번째 줄에 있음
data = pd.read_csv("./data/data.csv")

print("컬럼명 : ", data.columns)
print("인덱스 : ", data.index)


print( data.info() )

#누란된 데이터가 있는지 확인하자.
print( data['height'].value_counts(dropna=False))
print( data['height'].isnull()) #NaN이면 True, 아니면 False반환
print( data['height'].notnull())

#null값인것에 대한 개수 확인
print( data['height'].isnull().sum(axis=0))
```

내레이션

◈ 누락데이터처리

```
# nan값을 thresh개 가진 열 모두 삭제
data_thresh = data.dropna(axis=1, thresh=3)

#합계
print ( data['height'].sum() )

print("데이터 개수")
print(data.shape)
data = data.dropna(subset=['height'], how='any', axis=0)
print("삭제후 데이터개수")
print(data.shape)

data = data.dropna(subset=['weight'], how='any', axis=0)
print("삭제후 데이터개수")
print(data.shape)

#인덱스 다시 부여
data = data.reset_index(drop=True)
print(data)
```

내레이션

◈ 누락데이터처리

- 누락데이터를 삭제하는것도 한 방법이지만, 실제로 데이터를 수집하다보면 누락이 너무 많이 발행해서
- 행을 삭제하거나 열을 삭제할 경우 제대로 분석이 어려운 경우가 발생합니다.
- 이럴때는 행이나 컬럼 전체를 삭제하는것보다는 대체값을 바꾸는게 더 나은 방법입니다.
- 대체값은 보통 평균값을 사용하거나 중간값을 사용합니다.
- 평균값의 경우 최대값과 최소값 편차가 클경우에는 평균값 보다는 중간값이 집단을 대표하는 경우가 더 많습니다.
- 이번엔 삭제가 아니라 값의 대체 예제를 보도록 하겠습니다

내레이션

◈ 누락데이터처리

```
#파일명 : exam11_2.py
#누락데이터 치환하기

import pandas as pd

#header가 3번째 줄에 있음
data = pd.read_csv("./data/data.csv")

#누란된 데이터가 있는지 확인하자.
print( data['height'].value_counts(dropna=False))
print( data['height'].isnull()) #NaN이면 True, 아니면 False반환
print( data['height'].notnull())

#null값인것에 대한 개수 확인
print("height 필드 Nan 개수 : ", data['height'].isnull().sum(axis=0))
print("weight 필드 Nan 개수 : ", data['weight'].isnull().sum(axis=0))


print("누락값 대체 후 ----------------------")
#누락값 대체하기
mean_height = data['height'].mean()
mean_weight = data['weight'].mean()
```

내레이션

◈ 누락데이터처리

```
data['height'].fillna( mean_height, inplace=True)
data['weight'].fillna( mean_weight, inplace=True)

print("height 필드 Nan 개수 : ", data['height'].isnull().sum(axis=0))
print("weight 필드 Nan 개수 : ", data['weight'].isnull().sum(axis=0))

print( data )
```

fillna 함수 NaN 값을 특정 값으로 대체 합니다.
inplace옵션을 True를 해줘야 값이 바뀝니다

```
weight 필드 Nan 개수 :  2
누락값 대체 후 ------------------
height 필드 Nan 개수 :  0
weight 필드 Nan 개수 :  0
   name      height      weight
0    A1  180.000000   92.000000
1    A2  176.000000   70.000000
2    A3  175.000000   65.000000
3    A4  172.000000   64.000000
4    A5  168.000000   67.107143
5    A6  175.000000   74.000000
6    A7  177.000000   68.000000
7    A8  172.000000   65.000000
8    A9  177.888889   77.000000
9   A10  173.000000   59.000000
10  A11  174.000000   66.000000
11  A12  177.000000   63.000000
12  A13  179.000000   65.000000
13  A14  181.000000   72.000000
14  A15  189.000000   79.000000
15  A16  193.000000   84.000000
16  A17  177.888889   34.000000
17  A18  172.000000  160.000000
18  A19  179.000000   84.000000
19  A20  169.000000   64.000000
20  A21  167.000000   56.000000
21  A22  230.000000   12.000000
22  A23  171.000000   15.000000
23  A24  177.888889   80.000000
24  A25  187.000000   78.000000
25  A26  176.000000   67.107143
26  A27  175.000000   59.000000
27  A28  173.000000   62.000000
28  A29  169.000000   53.000000
29  A30  174.000000   59.000000
```

내레이션

◈ 중복 데이터 처리

- 데이터 수집중에 중복된 데이터가 있을 경우 판다스 라이브러리는 쉽게 중복을 제거할 수 있는 함수를 제공합니다.
- dataframeobject['필드명'].duplicated() **중복되지 않았을 경우 False 중복될 경우 True를 반환합니다.**
- 두개 이상의 데이터가 중복되어 있을때 첫번째 데이터는 False 그 뒤에 중복되어 나타나는 데이터 항목은 True로
- 출력됩니다.
- dataframeobject.drop_duplicates()함수를 사용하면 중복된 필드들이 있는 모든 행들이 삭제됩니다.
- dataframeobject.drop_duplicates(subset=[필드명]) subset 필드를 별도로 지정하면 그 필드내의 데이터가 중복될 경우 모든 행들이 삭제됩니다.

내레이션

◈ 누락데이터처리

```
#파일명 : exam11_3.py
#중복데이터 제거하기

import pandas as pd

data = {
    'passenger_code':['A101', 'A102', 'A103', 'A101', 'A104', 'A101', 'A103'],
    'target':['광주', '서울', '부산', '광주', '대구', '광주', '부산'],
    'price':[25000, 27000, 45000, 25000, 35000, 27000, 45000]
}

df = pd.DataFrame(data)

print( df )

print("중복된 데이터 ")
col = df['passenger_code'].duplicated()  #중복된 데이터 확인하기
print( col )
```

내레이션

◈ 누락데이터처리

```
#중복 제거시 모든 데이터가 일치된것만 제거한다.
df2 = df.drop_duplicates()
print( df2 )

print("특정 컬럼값이 중복일때 제거하기 -----")
df3 = df.drop_duplicates(subset=['passenger_code'])
print( df3 )


print("두개의 컬럼값이 중복일때 제거하기 -----")
df3 = df.drop_duplicates(subset=['passenger_code', 'target'])
print( df3 )
```

내레이션

간지 부분

◈ 데이터프레임

정규화, 구간나누기, 원핫인코딩, 컬럼정규화

내레이션

(기입하지 않습니다.)

◈ 컬럼 정규화하기

- 머신러닝이나 딥러닝에서 경우에 따라서 데이터의 단위가 너무 다를 경우 전체 예측 평가에 영향을 미칩니다.
- 그래서 일부 알고리즘에서는 데이터값을 평균을 0, 분산을 1이 되도록 만들어 줘야 합니다. 이를 정규화라고 합니다. 다음은 정규화 수식입니다.

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

(정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터 값들 중 최대값 - 데이터 값들 중 최소값)

물론 sklearn(사이킷런)라이브러리 같은 경우 정규화를 지원하는 클래스도 있습니다만 사이킷런은 이 과정이 범위를 벗어나므로 지원 라이브러리가 있다는 정도만 다루도록 하겠습니다

내레이션

◈ 단위 전환

- 우리나라에서 사용하는 단위와 유럽이나 미국 등 기타 국가들이 사용하는 단위가 달라서 데이터를 보기
- 어려운 경우가 있습니다. 예를 들면 우리나라의 무게는 kg 단위를 사용하는데 일부 국가들은 파운드를 사용합니다.
- 이런 단위들도 일치 시키는게 데이터를 분석할 때 편합니다.

내레이션

◈ 정규화및 단위환산

```
#파일명 : exam11_4.py
#데이터표준화

import pandas as pd

data = pd.read_csv('./data/auto-mpg.csv')
print(data.info())
print(data.head())

#컬럼명 변경하기
data.columns=['mpg', 'cyl', 'disp', 'power', 'weight', 'acce', 'model']
print(data.head())

#정규화
#(정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터 값들 중 최대값 - 데이터 값들 중 최소값)
data['mpg2'] = (data['mpg'] - data['mpg'].min())/(data['mpg'].max()-data['mpg'].min())
print(data)

#단위환산 - 한국 단위로 환산하기
mpg_unit = 1.60934 / 3.78541

data['kpl'] = (data['mpg'] * mpg_unit).round(2)
print( data.head() )
```

정규화는 데이터분석에서 아주 중요한 작업입니다. 정규화 작업을 해줘야만 하는 머신러닝 알고리즘이 많이 있습니다.

내레이션

◈ 타입전환

- 파이썬은 숫자형태의 데이터는 실제 데이터가 어떤 형태로 저장되던(엑셀 같은경우 숫자를 문자열형태로 저장
- 하기도 하고 숫자 형태로 저장하기도 합니다.) 숫자자료로 잘 읽어와서 수치 자료는 크게 문제가 되지 않습니다
- 문제가 되는 경우는 수치자료가 와야 할 열에 수치 자료가 아닌 다른 문자가 가끔씩 끼어 들어올 때 이에 대한 처리를 해줘야 합니다. 문자를 제거하거나 문자가 들어있는 행의 데이터를 버리거나 하고 유형을 전환해줘야 합니다.
- 문자열의 경우에는 분석시에 문자열 자체로 사용을 못합니다. 문자열의 경우에는 범주형 자료 타입으로 전환을 해야 합니다.
- 이번에는 타입 전환에 대한 예에 대해서 살펴보도록 하겠습니다.
- 타입전환은 .astype 함수를 이용해서 합니다

내레이션

◈ 타입전환

```
#파일명 : exam11_5.py
#데이터표준화

import pandas as pd
import numpy as np

data = pd.read_csv('./data/auto-mpg.csv')
print(data.info())
print(data.head())

#타입이 맞지 않을 경우 전환을 해서 사용해야 한다
#현재 사용하는 파이썬 버전은 문자열 데이터라도 수치 형태면 자동으로 수치자료로 처리한다
#파이썬 버전에 따라 다르게 동작할 수 도 있다
data.columns=['mpg', 'cyl', 'disp', 'power', 'weight', 'acce', 'model']
print(data.dtypes)
print(data.head())

print( data['disp'].unique())
```

내레이션

◈ 타입전환

```
#잘못된 데이터를 NaN으로 먼저 바꾼다
data['disp'].replace('?', np.nan, inplace=True)
print(data.head())

data.dropna(subset=['disp'], axis=0, inplace=True)
print(data.head())
print(data.dtypes)
data['disp'] = data['disp'].astype('float')
print(data.dtypes)

#범주형으로 바꾼다
data['model'] = data['model'].astype('category')
print(data.dtypes)
```

데이터를 범주형으로 바꿀때 astype('category')를 사용합니다.
문자열의 형태로 입력된 데이터는 분석시에 꼭 범주형으로
바꾸어서 처리해야 합니다.

내레이션

◈ 구간나누기

- 연속적 데이터라고 하더라도 경우에 따라서는 불연속 데이터로 전환해야 할때가 있습니다.
- 예를들어서 연비의 경우 국가에서 특정 조건을 만족할 경우 구간을 나누어서 등급을 부여할 수있습니다.
- 연비를 수치자료로 듣는 것 보다 이 자동차의 연비는 1등급이야 또는 2등급이야 라고 할때 실제 수치를
- 듣는것보다 정보가 빠르고, 연비 등급별로 각 모델들에 대한 평가나 분석을 해야 하는 경우도 있습니다.
- 이때 구간나누기를 통해 연속 데이터를 불연속 데이터로 변환이 가능합니다.
- 불연속 데이터로 전환 후 실제 분석시에는 특별한 방법을 사용하여 데이터를 저장해야 합니다. 만일 연비를 등급으로 A,B,C,D 4개로 나누었다면, 머신러닝이나 딥러닝 분석에서 이를 그대로 사용하지 못합니다. 머신러닝이나 딥러닝은 수학입니다. 저 등급은 다음과 같은 원핫인코딩 형태로 전환해야 사용이 가능합니다.

내레이션

◈ 원핫인코딩

- 범주형 데이터들을 행렬의 형태로 전환하고, 각 각의 범주가 갖는 값을 다음처럼 표현되도록 저장하는 방식입니다. 4개의 그룹이 있다고 가정하면 분석시에 그룹의 형태를 A,B,C,D등급을 필드의 컬럼으로 만들고 각각 해당하는 요소의 위치는 1 해당하지 않는 요소의 위치는 0으로 해서 행렬구조를 만듭니다

A	B	C	D
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

내레이션

◈ 구간나누기및 원핫인코딩

데이터에 잘못된 데이터가 있는 경우
NaN값으로 먼저 전환후, NaN을 포함하고
있는 행이나 열을 삭제 하는게 일이 빠르다

```
#파일명 : exam11_4.py
#구간분할

import pandas as pd
import numpy as np

data = pd.read_csv('./data/auto-mpg.csv')
print(data.info())
print(data.head())

#타입이 맞지 않을 경우 전환을 해서 사용해야 한다
#현재 사용하는 파이썬 버전은 문자열 데이터라도 수치 형태면 자동으로 수치자료로 처리한다
#파이썬 버전에 따라 다르게 동작할 수 도 있다
data.columns=['mpg', 'cyl', 'disp', 'power', 'weight', 'acce', 'model']
print(data.dtypes)

#잘못된 데이터를 NaN으로 먼저 바꾼다
data['disp'].replace('?', np.nan, inplace=True)
data.dropna(subset=['disp'], axis=0, inplace=True)
data['disp'] = data['disp'].astype('float')
```

내레이션

◈ 구간나누기및 원핫인코딩

```
#범주형으로 바꾼다
data['model'] = data['model'].astype('category')
#연속적인 값을 구간으로 나누어 비연속적인값(범주형)으로 나눈다
#print(data['power'])

print(data['power'].isnull().sum(axis=0))
data.dropna(subset=['power'], inplace=True)
count, bin_dividers = np.histogram(data['power'], bins=4)
print( bin_dividers)

bin_names = ["D", "C", "B", "A"]
data["grade"] = pd.cut( x=data['power'], bins=bin_dividers,
    labels = bin_names, include_lowest=True)

print( data )
```

np.histogram 은 구간을 나누는 함수이다. 데이터의 범위를 알아내서 지정된 구간을 나누고 구간의 경계에 대한 값을 가져온다. 구간을 나누고자 하는 필드와 구간의 개수를 전달하면 구간의 경계, 구간의 데이터개수를 반환한다.

cut함수, x인자에 필드를 정해주고, bins 인자에 np.histogram이 생성한 구간 정보를 전달한다. labels 에 각 구간에 붙일 라벨명을 지정하고 include_lowest=True는 경계값을 포함하라는 의미임

내레이션

◈ 원핫인코딩

```
#onehot encoding
Y_class = np.array(data['grade']).reshape(-1,1)

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(Y_class)

Y_class_onehot = enc.transform(Y_class).toarray()
Y_class_recovery = np.argmax(Y_class_onehot, axis=1).reshape(-1,1)

print(Y_class_onehot)
```

onehot 인코딩을 진행라면 데이터 필드가 1차원인 상태로는 진행이 되지 않습니다. reshape함수를 이용해서 차원을 먼저 바꿔주고 sklearn.preprocessing 의 OneHotEncoder객체를 만든 후 transform함수를 수행하면됩니다. onehot encoding 을 복원하려면 np.argmax 함수를 사용하면 됩니다. no.argmax함수는 전달 받은 함수중에서 가장 큰값을 갖는 인자를 반환하는 함수입니다.

내레이션

◈ 적용하기

data 폴더 아래에 있는 iris.csv 파일을 읽어서 다음 동작들을 수행해 보세요

1) iris 데이터셋에 몇개의 필드가 있고 각 필드의 타입이 무엇인지 확인해 보세요

2) 맨 앞의 데이터를 7개만 출력해보세요

3) iris 데이터셋의 통계량을 확인해보세요(평균, 표준편차, 중간값, 사분위수...)

4) variety 이 Setosa 인 데이터의 통계량을 출력하세요

5) 각각 variety가 Setosa, Virginica Versicolor 의 sepal.length 값의 평균값을 출력하시오

6) 꽃의 종류가 Setosa 이면서 sepal.length 길이가 5cm이상인 것의 개수를 출력하시오

내레이션

(기입하지 않습니다.)

◈ 적용하기

1. data폴더내의 iris.csv파일을 읽어서, 누락된 데이터가 어떤 필드에 몇개 있는지 찾아내세요
2. 누락된 데이터들을 평균치고 대체하세요
3. sepal.length, sepal.width, petal.width 세개의 필드에 정규화를 진행하세요
4. petal.length 필드를 3개의 구간으로 나누어서 A, B, C 이라고 구간 이름을 붙여서
 petal_grade 필드를 추가하세요

원핫 인코딩으로 전환하세요

내레이션

(기입하지 않습니다.)

◈ 적용하기- 풀이

```
import pandas as pd
import numpy as np

data = pd.read_csv('./data/iris.csv')
#NaN값 있는지 체크하기 – 각 필드별로 NaN값 있는 개수 출력
print( data.isnull().sum())

sepal_length_mean = data['sepal.length'].mean()
sepal_width_mean = data['sepal.width'].mean()
petal_length_mean = data['petal.length'].mean()
petal_width_mean = data['petal.width'].mean()

data['sepal.length'].fillna( sepal_length_mean, inplace=True)
data['sepal.width'].fillna( sepal_width_mean, inplace=True)
data['petal.length'].fillna( petal_length_mean, inplace=True)
data['petal.width'].fillna( petal_width_mean, inplace=True)
```

내레이션

(기입하지 않습니다.)

◈ 적용하기 -> 풀이

```
#정규화를 함수로 만들었음
def normalize(columnname):
    max = data[columnname].max()
    min = data[columnname].min()
    return (data[columnname]-min)/(max-min)

data['sepal.length'] = normalize('sepal.length')
data['sepal.width'] = normalize('sepal.width')
data['petal.width'] = normalize('petal.width')

print(data)
```

내레이션

(기입하지 않습니다.)

◈ 적용하기 -> 풀이

```
#구간 나누기
count, bin_dividers = np.histogram(data['petal.length'], bins=3)
print( bin_dividers)

bin_names = ["A", "B", "C"]
data["petal_grade"] = pd.cut( x=data['petal.length'], bins=bin_dividers,
     labels = bin_names, include_lowest=True)

print(data)
```

내레이션

(기입하지 않습니다.)

◈ 적용하기 -> 풀이

```
#onehot 인코딩
Y_class = np.array(data['petal_grade']).reshape(-1,1)

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(Y_class)

Y_class_onehot = enc.transform(Y_class).toarray()
Y_class_onehot = np.argmax(Y_class_onehot, axis=1).reshape(-1,1)

print(Y_class_onehot)
```

내레이션

(기입하지 않습니다.)

◈ 문제풀기

문제 1) 다음 설명중 잘못 된 것은?

① 데이터의 누락치는 데이터 분석에 지대한 영향을 미치므로 모두 삭제해야 한다
② 데이터 누락치는 데이터 분석에 지대한 영향을 미치므로 평균값이나 중간값등으로 대체해야 한다
③ 데이터의 정규화는 분석의 정확도를 위해 필요한 경우가 많다
④ 데이터의 타입이 문자열일 경우 범주형자료로 전환해야 한다

정답) 5
해설) 누락된 데이터가 너무 많을 경우 모두 삭제는 오히려 분석 결과에 안좋은 영향을 미칠 수 있으므로 평균치나 중간값으로 대체 하는 방법을 취하는게 좋습니다.
난이도) 1(1:아주어려움, 2:어려움 3: 보통 4: 쉬움 5: 아주 쉬움)
관련페이지번호) 9

내레이션

(기입하지 않습니다.)

번호	문제	정답	난이도	해설	관련학습보기
2	다음 데이터에서 누락 데이터를 찾고자 합니다. 함수를 바르게 적용한 경우는 ? ① data['height'].isnull().sum() ② data['height'].NaNsum() ③ data['height'].CountNan() ④ data['height'].CntNa()	1	4	isnull() 함수와 sum() 함수를 호출하다	7
3	**데이터중 누락된 데이터(NaN)을 삭제하려고 할 경우 적당한 함수는?** **① isnull** **② dropna** **③ deleterow** **④ removena**	2	5	dropna함수를 사용한다	9
4	**데이터를 삭제하면 인덱스 번호가 뒤죽박죽이 됩니다. 삭제후 인덱스를 재부여 하려면 어떤 함수를 써야 하나요?** **① set_index** **② refresh_index** **③ reset_index** **④ index**	3	4	reset_index 함수를 사용하면 index가 새로 부여된다.	10

번호	문제	정답	난이도	해설	관련학습 보기
5	잘못된 데이터가 들어온경우 처리법으로 가장 적절한 것은 ① 데이터를 메모장에서 직접 수정한다 ② 예외처리를 한다 ③ 오류 데이터를 NaN값으로 모두 전환후 대체값으로 대신하던가 삭제한다 ④ 오류데이터를 엑셀 기능을 이용해 모두 삭제한다.	3	3	데이터 건수가 많을 경우에는 일일이 메모장이나 엑셀에서 작업이 어렵습니다. 우선 데이터를 읽어서 NaN값으로 모두 변환후 한번에 처리하는게 편합니다.	13
6	**다음중 정규화 수식으로 올바른것은?** ① (정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터 값들 중 최대값 - 데이터 값들 중 최소값) ② (정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터평균값) ③ (정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터 분산) ④ (정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터 표준편차)	1	1	(정규화하고자 하는 값 - 데이터 값들 중 최소값) / (데이터 값들 중 최대값 - 데이터 값들 중 최소값)	22
7	**다음 등급을 원핫인코딩 방식으로 변환하려고 합니다. 올바른 변환을 기술하십시요 (원핫인코딩 한 상태의 값을 기술하세요)** **등급 : 좋음, 보통, 나쁨**	해설참조	2	좋음 보통 나쁨 1 0 0 0 1 0 0 0 1	27

번호	문제	정답	난이도	해설	관련학습보기
8	다음 중 구간 나누기를 하는 바른 사유는 ① 연속형 값을 비연속형 값의 형태로 전환하려고 ② 구간별 평균값을 구하려고 ③ 히스토그램 차트를 그리기 위해서 ④ 비연속형 값을 연속형 값으로 전환하려고	1	5	연속형값을 비연속형으로 전환할 필요가 있어서	26
9	dropna 함수를 사용해 데이터를 삭제하고자 할때 inplace=True 옵션이 하는일은 ? 예)data.dropna(subset=['disp'], axis=0, inplace=True)	자신의 데이터를 수장한다	1	inplace속성을 True 로 해야 삭제된 데이터가 자신의 데이터로 적용됩니다.	9
10	**다음 데이터셋중에서 필드 mpg와 cylinders 두 개의 필드에서 데이터가 NaN값이 있으면 삭제하려고 합니다. 다음 코드의 나머지 부분을 완성하세요** data.dropna(, axis=0, inplace=True)	subset=['mpg', 'cyliend ers']	1	subset를 지정해주면 됩니다. subset는 리스트 형태로 전돨됩니다	18

과정명		모듈명		회차명	

◈ 핵심요약

▶데이터의 누락치 제거 절차

- isnull함수등을 이용해 데이터의 누락치가 있는지 먼저 체크한다. isnull함수는 True, False값만 반환하므로 전체적인 누락여부 확인이 어렵다

- isnull() 결과로 sum()함수를 호출하면 NaN 데이터가 존재 하는지 여부를 쉽게 알 수 있습니다.

-누락치 데이터 건수가 작아서 전체 분석에 영향을 미치지 않을거라고 판단되었을 경우에는 삭제를 진행하고, 누락 건수가 많아서 전체 영향을 미친거라고 판단되면 평균값이나 중간값등으로 대체 해서 사용합니다.

▶ 구간나누기, 원핫인코딩

분석시 사용하는 데이터는 크게 두개의 범주로 나뉩니다. 하나는 연속형, 하나는 불연속형입니다.

불연속형은 비연속형, 범주형이라고도 하는데 연속형이 평균값을 중요하게 여기는데 반해 불연속형은 개수를 중요하게 여깁니다.

연속형이 아닌 데이터 타입들은 불연속형으로 다뤄야 하기 대문에 문자열 형은 불연속형으로 전환하야 사용해야 합니다.

연속형 데이터이더라도 불연속형으로 전환해서 분석해야 할 경우에는 구간나누기를 사용합니다.

구간나누기후 실제 분석에서는 데이터를 원핫인코딩 방식으로 전환하여 처리합니다.

내레이션

(기입하지 않습니다.)

◈ 학습맺음

- 이번 회차에서는 DataFrame 의 구조와 DataFrame이 제공하는 API를 이용하셔 외부파일을 일고 쓰는 방법, 데이터 분석 API 등을 살펴보았습니다
  수고 많으셨습니다.

◈ 참고자료

- https://docs.anaconda.com/anaconda/
- https://code.visualstudio.com/docs/getstarted/themes
- https://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html

내레이션

(기입하지 않습니다.)