|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211001 |
| 교육장소 | YGL 학과장 |
| **교육내용** | |
| 오전 | 1. **누락 데이터 처리 : 누락된 데이터(Nan, Not a Number)**   : value\_counts(dropna = False) : 누락 데이터 갯수 확인  # 라이브러리 불러오기  import seaborn as sns  # titanic 데이터셋 가져오기  df = sns.load\_dataset('titanic')  print(df)  # isnull() = NaN 이면 True 반환 / notnull() = NaN 이면 NaN을 반환, 값이 존재하면 데이터의 값을 출력.  #누락된 데이터 처리 - 제거, 치환  print(df.head().isnull())  print(df['deck'].head().isnull())  print('-----------')  print(df.head().notnull()) #값이 존재하면 값을, 없으면 Nan을 출력  print('-----------')  print(df.isnull().sum(axis=0)) # 누락된 데이터가 있는지 확인  #누락된 자료가 500개 이상인 자료를 제거  df\_1 = df.dropna(axis=1, thresh=500) #null값 500개 이상인 행을 제거  'deck' in (df\_1.columns)  print(len(df\_2))  len(df) - df['age'].isnull().sum(axis=0)  #누락 데이터가 많은 컬럼을 제거  df\_1 = df.dropna(axis=1, thresh=500)  #age열에 NaN값이 있으면 행을 제거하는 방법  df\_2 = df.dropna(subset=['age'],how='any',axis=0)  # age 열의 NaN 값을 다른 나이 데이터의 평균으로 변경하기  mean\_age = df['age'].mean(axis=0)  df\_age = df['age'].fillna(mean\_age)  # 앞의 값으로 수정  df\_age\_1 = df['age'].fillna(method='ffill')  # 가장 빈번하게 나오는 값으로 수정  most\_cnt = df['age'].value\_counts(dropna=True).idxmax()  df\_age\_2 = df['age'].fillna(most\_cnt)  print(df\_age.isnull().sum(axis=0))  len(df\_age)   1. **중복 데이터 처리**   import pandas as pd  #중복 데이터를 갖는 데이터 프레임 만들기  df = pd.DataFrame({'c1':['a','a','b','a', 'b'],  'c2':[1,1,1,2,2,],  'c3':[1,1,2,2,2]})  print(df,'\n')  df\_dup = df.duplicated() # 행의 중복을 체크  print(df\_dup)  # 열의 중복 체크  df['c1'].duplicated()  # 데이터프레임에서 중복 행을 제거  df2 = df.drop\_duplicates()  print(df2)  # 특정 열을 기준으로 중복 행을 제거  df2 = df.drop\_duplicates(subset=['c2','c3'])  df2   1. **데이터 표준화**  * 단위 환산 * 자료형 변환   import pandas as pd  # read\_csv()함수로 df 생성  df = pd.read\_csv("./auto-mpg.csv", header=None)  # 열 이름을 지정  df.columns = ['mpg','cylinders','displacement','horsepower','weight','acceleration','model year','origin','name']  df\_head()  print(df.dtypes)  df.horsepower.head(5)  # 자료형, 단위를 변환  # mpg(mile per gallon)를 kpl(kilometer per liter)로 변환 (mpg\_to\_kpl = 0.425)  mpg\_to\_kpl = 1.60934 / 3.78541  # mpg 열에 0.425를 곱한 결과를 새로운 열(kpl)에 추가  df['kpl'] = (df['mpg'] \* mpg\_to\_kpl).round(2)  df.kpl.head(3)  #1. 데이터 타입 확인 , unique() -> '?'를 NaN으로 처리  df.horsepower.unique()  df\_hp = df.copy()  df\_hp.horsepower = df.horsepower.replace('?',np.nan, inplace=True) # '?'를 np.nan으로 변경  #2. NaN데이터 확인 후 -> 처리 -> 0.0값으로 치환  #df\_hp['horsepower'].fillna('0.0', inplace=True)  df\_hp.dropna(subset=['horsepower'], axis=0, inplace=True) #누락데이터 행 삭제  #3. 데이터 타입 변경 -> float64.  df\_hp['horsepower']=df\_hp['horsepower'].astype('float') # 문자형을 실수형으로  # horsepower 칼럼에 대해서 순서대로 처리.  import numpy as np  df.horsepower.unique()  df\_hp = df.copy()  df\_hp.horsepower.replace('?',np.nan, inplace=True) # 쓰레기값 처리  #df\_hp.dropna(subset=['horsepower'], axis=0, inplace=True)  df\_hp['horsepower'].fillna('0.0', inplace=True)  df\_hp.dtypes  # category, 데이터 타입을 범주형 또는 문자형, 숫자형으로 원하는 형태로 변경.  print(df\_hp['origin'].unique()) # 정수형 데이터를 문자로 변경  df\_hp['origin'].replace({1:'USA', 2:'EU',3:'JAPAN'}, inplace=True)  print(df\_hp['origin'].unique())  print(df['origin'].dtypes)  df['origin'] = df['origin'].astype('category')  print(df['origin'].dtypes)  df['origin'] = df['origin'].astype('object')  print(df['origin'].dtypes)   1. 범주형(카테고리) 데이터 처리  * 구간 분할 * 더미 변수   import numpy as np  import pandas as pd  df = pd.read\_csv("./auto-mpg.csv", header=None)  df.columns = ['mpg','cylinders','displacement','horsepower','weight','acceleration','model year','origin','name']  # horsepower 열의 누락 데이터 처리 ('?' 를 삭제 후 실수형으로 변경.)  df['horsepower'].replace('?',np.nan, inplace=True) # '?'를 'np.nan'으로 바꿈  df.dropna(subset=['horsepower'], axis=0, inplace=True)  df['horsepower'] = df['horsepower'].astype('float')  # 3개의 구간( bin ) 으로 나눠서 범위로 처리를 하려고 함  count, bin\_div = np.histogram(df['horsepower'],bins=3) # 경계 리스트 np.histogram은 리턴값이 2개.  print(bin\_div)  bin\_names = ['저출력','보통출력','고출력']  horse\_dummies = pd.get\_dummies(df['hp\_bin'])  horse\_dummies.head(10)  pd.get\_dummies(df['hp\_bin'],prefix='hp')  문)  age 칼럼으로 데이터를 처리, bin을 4로 [유아기, 청소년, 청년, 장년]  # horsepower 열의 누락 데이터 처리 ('?' 를 삭제 후 실수형으로 변경.)  import seaborn as sns  df = sns.load\_dataset('titanic')  df['age'].isnull().sum(axis=0) # NaN 데이터를 처리 평균 나이로 대체  age\_avg = df['age'].round().sum()/df['age'].count().round()  df['age'].fillna(value =age\_avg, axis=0, inplace=True)  # df['age'] = df['age'].round().astype('int') # float을 int로  bin\_names = ['유아기','청소년','청년','장년']  bin\_values = [0, 20, 40, 60, 100]  df['age\_bin'] = pd.cut(x = df['age'], # 데이터 배열  bins = bin\_values, # 경계 리스트  labels = bin\_names, # bin 이름  include\_lowest = True  )  df[['age','age\_bin']]  pd.get\_dummies(df['age\_bin'],prefix="대")   1. **더미변수 get\_dummies()**   컴퓨터가 인식 가능한 입력값으로 변환하기 위해 0 또는 1로 표현되는 더미변수 사용.  0과 1은 크고 작음이 아닌 어떤 특성(Feature)이 있는지 없는지 여부만을 표시.   1. **원핫인코딩 : onehot\_**   label\_encoder로 문자열 범주를 숫자형 범주로 변환 : 머신너링, 회귀분석   * 데이터 분석을 위해서는 narray 2차원 배열, 1차원 배열, n차원 배열 * 2차원 행렬로 형태 변경 reshape(데이터 행수,n)  1. 정규화   : 숫자 데이터의 상대적인 크기 차이를 제거하기 위해 하는 것. 정규화 과정을 거친 데이터의 범위는 0~1 또는 -1~1이 됨. |
| 오후 | 1. **Numpy**   : 다차원 배열이나 행렬과 수학 함수 지원  import numpy as np  arr = np.arange(24)  #print(arr,type(arr),'\n',arr.shape,'n\, arr.size ":", arr.dtype)  arr = arr.reshape(3,4,2) # (0축 3개,1축 4개, 2축 2개)  #int(arr,type(arr),"\n", arr\_shape,'\n', arr.size ':', arr.dtype)  print(arr)  arr.sum(axis=2) # axis 값에따라 n행n열 모양이 바뀌뮤  print('-------------','\n',arr.dtype.name, arr.dtype.itemsize)  a = np.arange(-5,5,0.5)  a  a\_list = [1.0 , 2.0 , 3.0]  print(type(a\_list))  arr\_list = np.array(a\_list)  print('array != list',type(arr\_list))  a1 = np.array([1.0 , 2.0 , 3.0])  a2 = np.array([[1.0,2.0],[3.0,4.0]])  print(a1, type(a1))  print(a2, type(a2))  print(a1.ndim, a1.shape, a1.size, a1.dtype)  print(a2.ndim, a2.shape, a2.size, a2.dtype)  a = np.zeros(5)  b = np.zeros((2,3))  c = np.zeros((5,2), dtype='i')  d = np.zeros(5, dtype='U4')  e = np.ones((2,3,4), dtype='i8')  f = np.ones\_like(b, dtype='f')  g = np.empty((4,3))  x1 = np.array([1.0, 2.0, 3.0])  y1 = np.array([5.0, 10.0, 15.0])  x2 = np.array([[1.0, 2.0],[ 3.0, 4.0]])  y2 = np.array([[5.0,10.0],[15.0,20.0]])  z1 = np.array([-1.0, -2.0])  z2 = np.array([[5.0],[10.0],[15.0]])  # ndarray basic operation  print(x1 + y1) # 차원이 같은거끼리 연산해서 문제없음  print(x1 - y1)  print(x1 \* y1)  print(x1 / y1)  print(x2 + y2)  print(x2 \* y2)  # ndarray broadcast  print(x2)  print(z1)  print(x2 + z1) # 행렬 기초연산, 계산하는 행과 열은 같아야함.  print(x2 \* z1)  print(x1 + z2)  print(x1\*\*2)  print(x1>=2)  # shape manipulation  print(x2.flatten())  print(x2.reshape(2,2))  문) x = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]) 일때  1] 3의 배수  print("3의 배수 :", x[x%3==0])  2] 4로 나누면 1이 남는 수  print('4로 나누면 1남는 수 :', x[x%4==1])  3] 3으로 나누면 나눠지고 4로 나누면 1이 남는 수  a\_arr = list(np.array([x%3==0]) & np.array([x%4==1]))  print("3의 배수이면서 4로 나누면 1 남는 수 : " , x[a\_arr])   1. **함수 매핑**   시리즈 원소에 함수 매핑 : 시리즈 객체에 apply() 메소드를 적용하면 인자로 전달하는 매핑 함수에 시리즈의 모든 원소를 하나씩 입력하고 함수의 리턴값을 돌려받음.  시리즈의 원소에 함수 매핑 : Series객체.apply(매핑함수)  import pandas as pd  import seaborn as sns  titanic = sns.load\_dataset('titanic')  titanic.columns  df = titanic.loc[ : , ['age','fare']] #모든 인덱스에서 'age','fare' 출력  df['ten'] = 10 # 'ten'이라는 column 추가, 원소값은 10.  df  # 매핑 함수를 이용하여 각 원소에 동일한 함수 실행.  def add\_10(n):  return n + 10  # apply() 메서드를 활용하여 시리즈의 각 원소에 동일한 함수 실행.  ar = df['age'].apply(add\_10) # df['age']의 각 원소에 add\_10, 10씩 더해주는것.  # lambda 함수 적용해보기  df['age\_lambda'] = df['age'].apply(lambda x: add\_10(x))  df  df['ten\_10'] = df['age'].apply(add\_10) # ten\_10 이라는 column 생성, 원소값은 df['age']의 각 원소에 10씩 더해준것.  df.head()  df\_map = df.applymap(add\_10) # applymap() : '모든 원소'에 대해 함수 적용  df\_map.head() |