|  |  |
| --- | --- |
| 교육 제목 | **데이터 사전처리 & 데이터 프레임 다양한 응용** |
| 교육 일시 | 21.10.01 |
| 교육 장소 | 오프라인 (영우글로벌러닝) |
| **교육 내용** | |
| 오전 | ▣ 데이터 전처리 -------------------------------------------------------------------------------------------  **∇ 전처리 순서**  \* 데이터 불러오기 -> 누락데이터 확인 및 대체하기 -> 행, 열 재정비 하기  -------------------------------------------------------------------------------------------  **∇ 누락데이터 확인하기**  \* isnull() -> NaN이면 True를 반환  \* notnull() -> 값이 존재하면 데이터 값을 반환, 존재하지 않으면 NaN을 반환  -------------------------------------------------------------------------------------------  **∇ 중복데이터 처리하기**  \* df.duplicated() 행의 중복을 check  \* df['c1'].duplicated() 특정 열의 중복을 check  \* df.drop\_duplicates(['c2','c3']) 중복 행 제거  \* df.drop\_duplicates(subset=['c2','c3']) 특정 열을 기준으로 중복 행 제거  -------------------------------------------------------------------------------------------  **∇ 범주형(카테고리) 데이터 처리**  \* 나눠준 범주 명을 컴퓨터가 인식 못함. 더미 변수로 전환  \* 해당 범주에 속해 있던 데이터를 '1'로 인식. 아인 것들은 '0'으로 인식  -------------------------------------------------------------------------------------------  **∇ 더미 변수**  \* horse\_dummies = pd.get\_dummies(df['hp\_bin'])  \* horse\_dummies.head(10)  \* pd.get\_dummies(df['hp\_bin'], prefix = 'hp')  # prefix = '어쩌구'-> 범주 명 앞에 접두사(어쩌구)가 붙음  -------------------------------------------------------------------------------------------  **∇ sklearn 사용하기 -> LabelEncoder(), OnehotEncoder()**  \* from sklearn import preprocessing  label\_encoder = preprocessing.LabelEncoder()  # label encoder 생성  onehot\_encoder = preprocessing.OneEncoder()  # one hot encoder 생성  onehot\_label = label\_encoder.fit\_transform(df['hp\_bin'].head(20))  print(onehot\_labeled) # 1차원 형태로 데이터 변환됨  ------------------------------------------------------------------------------------------- |
| 후 | **▣ Numpy**  -------------------------------------------------------------------------------------------  **∇ numpy**  \* import numpy as np  \* 행렬, 삼각함수 통계 등 수학적 계산 가능  -------------------------------------------------------------------------------------------  \* **∇ array 생성하기**  **Ex))**  import numpy as np  arr = np.arange(24)  print(type(arr))  arr = arr.reshape(3, 4, 2) # 3 by 4 by 2 행렬이 만들어짐 (각각 axis = 0, 1, 2)  arr.ndim # 몇 차원인지 알려줌  arr.sum(axis = 0)  -------------------------------------------------------------------------------------------  **∇ 초기값 부여해서 행렬 차원 만들기**  **Ex))**  a = np.zeros((0, 5))  b = np.zeros((2, 3), dtype='i')  c = np.zeros((0, 5), dtype='u8')  d = np.zeros((0, 5), dtype='i8')  e = np.empty((4, 3)) # 4 by 3에 아무 데이터나 입력 대입 됨. 쓰레기 데이터  f = np.nan() # NaN으로 값 대체  g = np.inf() # infinity로 값 대체  -------------------------------------------------------------------------------------------  **∇ 차원이 달라도 '앞에 것의 행의 개수'와 '뒤에 것의 열의 개수'가 같으면 계산 가능**  -------------------------------------------------------------------------------------------  **▣ 데이터프레임의 다양한 응용**  -------------------------------------------------------------------------------------------  **∇ 매핑 함수를 이용하여 각 원소에 동일한 함수 실행**  **Ex0)**  def add\_10(n):  return n + 10  add\_10(20)  **∇ apply() method를 활용하여 시리즈의 각 원소에 동일한 함수 실행**  **Ex))**  ar = df['age'].apply(add\_10) # 특정 column에 대해서 함수 실행  ar # apply()에는 생성 함수(add\_10)에 따로 변수를 입력 안하고 'age'값이 입력되어 계산됨.  df['ten\_10'] = df['age'].apply(add\_10)  df.head()  -------------------------------------------------------------------------------------------  **∇ lambda도 사용 가능**  **Ex))**  df['age\_lambda'] = df['age'].apply(lambda x: add\_10(x))  df.head() |