데이터 전처리

개론

개론

- 데이터 전처리는 데이터랭글링에 있어 필수적인 작업
- 데이터 전처리 전문가가 되기 위해서는 연구 및 개발 분야에 대한 체계적인 지식 또한 필요
- 연구 및 개발에 필요한 데이터를 수집하고 전처리할 줄 안다면 같은 분야에 종사하는 다른 사람들로부터 자신을 차별화할 수 있음.

파이썬 활용

- 파이썬은 데이터 전처리에 적절한 도구
- 데이터에 일정한 패턴이 존재한다면 함수를 생성하여 반복적인 작업을 피할 수 있음

필요성

- **필요성**
  - 일반적으로 웹스크래핑 등으로 수집한 데이터는 클리닝이 되어 있다고 하더라도 서식의 비일관성이나 가독성과 같은 문제가 있음
  - 특히 두 개 이상의 데이터세트에 담긴 데이터를 동시에 사용하는 경우
  - 데이터를 서식화하고 표준화하지 않는다면 데이터를 적절하게 결합해 유용하게 쓰기 힘들다.

- **전처리된 데이터 공개**
  - 공개시 전처리된 데이터세트와 더불어 미가공 데이터, 그리고 미가공 데이터를 전처리한 표준화된 방법을 함께 공개하는 것이 좋음
  - (좋은 습관) 데이터 전처리 절차를 기록해 놓으면 데이터세트 자체와 연구에서 데이터세트의 용도를 제대로 변호할 수 있음

전처리 분류

- **Header 전처리**
  - 불필요한 헤더 정리 예) chap2. 데이터랭글링 과정
  - 헤더교체하기
- **데이터 클리닝**
  - 결측값, 중복값 처리
  - 이상치 확인 및 제거
- **데이터 변환 및 통합**
  - 데이터 변환 (Transformation) – 정규화 등
  - 데이터 통합 (Integration) – 표준화 등

헤더 교체하기

- **축약된 두문자 헤더 교체**
  - 헤더를 더 알아보기 쉽게 바꾸는 가장 직관적이고 자명한 방법은 축약된 헤더를 우리가 이해하기 쉬운 긴 단어로 교체하는 것
  - 두문자 헤더나 제대로 매칭되지 않은 헤더와 같은 문제

- **사례 분석**
  - 유니세프 가계 수준 설문조사 자료 중 하나를 수집 -> mn.csv
  - 이 자료에는 축약된 두문자 헤더를 포함하고 있어, 이에 대해 이해할 수 있는 **헤더설명**을 자료 수집 -> mn_headers.csv
  - 위 두 개의 파일을 이용해서 헤더 교체하기 (헤더설명을 포함하도록)

실험 데이터세트1

- 미가공된 유니세프 **MICS** 데이터

  MICS(Multiple Indicator Cluster Surveys) : 다수지표군조사

  - UNICEF : http://mics.unicef.org/surveys
  - 전 세계 여성 및 아동의 생활조건을 조사하기 위한 가계 수준의 설문조사로 유니세프 직원과 자원봉사자들에 의해 실시

- 교재 파일 [1] ref. chap3

  - github : https://github.com/jackiekazil/data-wrangling

    -> data / unicef / mn.csv (짐바브웨의 MICS데이터)

- 실험 데이터 Experiment Data

  - mn.csv (github) - utf-8 encoding="utf-8"

실험 데이터세트2

- **세계은행 MICS 헤더 두문자 설명 데이터**
  - 세계은행 : https://microdata.worldbank.org/index.php/catalog/1794/datafile/F5
  - MICS 데이터를 제공하고 있는 세계은행 웹사이트를 참고하면 두문자 헤더 값들이 무엇을 의미하는지 알아낼 수 있음
  - (웹스크래핑) 세계은행 웹사이트에서 설문조사의 두문자 헤더와 두문자 설명, MICS 데이터를 얻기 위해 쓰인 질문 등을 웹스크래핑한 결과가 csv파일로 github에 있음

- **교재 파일** [1] ref. chap3
  - github : data / unicef / mn_headers.csv

- **실험 데이터** Experiment Data
  - mn_header.csv (github) - utf-8 encoding="utf-8"

실습1

- **100개 행만 추출 -> mn_cut100.csv**
  - 실행시간을 줄이기 위해 mn.csv의 데이터를 100개 행만 추출하시오

```
import pandas as pd

#읽기
df_in = pd.read_csv('mn.csv', low_memory=False)

#행삭제
df_cut = df_in[:100]

#열삭제
df_cut = df_cut.drop(['Unnamed: 0'], axis=1)

#저장
df_cut.to_csv('mn_cut100.csv')
```

헤더 둘러보기

- 두 헤더의 길이 비교
  - 설명의 편의를 위해 mn.csv를 (MICS)데이터 파일이라고 하고, mn_headers.csv를 헤더(설명)파일이라고 하자.
  - (실습2.1) 두 csv파일을 list로 읽어 헤더의 길이를 비교한다. 헤더의 길이가 다르면 기본적인 차이가 존재하므로 그로부터 분석 시작

- 헤더의 길이가 다른 이유 분석
  - 데이터와 헤더의 길이가 다르다. 데이터는 한 행이 159열이 있는데, 헤더 리스트에는 210행에 그 정보가 있다.
  - (실습2.2) 데이터와 헤더가 같은 헤더로 시작하는지 눈으로 확인한다. 헤더 행의 첫 번째 요소가 데이터의 첫 번째 행과 일치하는지 확인한다.

실습2.1

- 헤더 길이 비교
  - 헤더가 각각의 파일의 행과 열 중 어느 것인지 파악하고 비교한다.
  - 두 헤더의 길이가 159와 210으로 다르다.

```
from csv import reader

#읽기
data_rdr = reader(open('mn_cut100.csv', 'rt'))
header_rdr = reader(open('mn_headers.csv', 'rt', encoding='utf8'))

#list comprehension
data_rows = [d for d in data_rdr]
header_rows = [h for h in header_rdr]

#두 헤더의 길이 출력
print( len(data_rows[0]) )          #159
print( len(header_rows) )           #210
```

헤더 매칭 확인

- 불량 헤더목록 만들기
  - 헤더를 통일시키는 작업을 시작한다.
  - (실습2.2) 데이터와 일치하지 않는 헤더 목록을 만든다. (60개)

- **9개의 mismatch발견 (실습2.2)**
  - 리스트의 remove 메소드를 사용해 리스트의 특정 행을 제거한다. (정리된 헤더 목록은 원본 210개에서 불량60개를 제거하여 150개)
  - 데이터세트의 헤더를 순회하여 정리된 헤더목록에 매칭되지 않는 헤더를 찾는다. (159개 데이터 헤더에서 150개 정리된 헤더 목록에 없는 목록 찾기)
  - 9개의 헤더에 대해 차이점을 분석하여 삭제할지 사용할 지 결정한다.

실습2.2

- 불량 헤더목록 만들기
  - 헤더 시작부분 눈으로 확인
  - 데이터와 일치하지 않는 헤더 목록 만들기 (불량 헤더목록 60개)

```
#실습2.1에 이어서

#데이터 헤더 행 (헤더 시작부분 눈으로 확인)
data_rows[0]
#헤더 시작 2행
header_rows[:2]

#bad_rows에 데이터와 일치하지 않는 헤더 목록 만들기
bad_rows = []
for h in header_rows:
    if h[0] not in data_rows[0]:
        bad_rows.append(h)
len(bad_rows)    #60
```

실습2.3

- **9개의 mismatch발견**
  - 정리된 헤더 목록은 원본 210개에서 불량60개를 제거하여 150개
  - 159개 데이터 헤더에서 150개 정리된 헤더 목록에 없는 목록 찾기

```
#실습2.2에 이어서

#헤더(설명)파일에서 일치하지 않는 헤더 제거
for h in bad_rows:
    header_rows.remove(h)
len(header_rows)          #150

#정리된 헤더 목록에 없는 데이터 헤더 목록 찾기
all_short_headers = [h[0] for h in header_rows]
for header in data_rows[0]:
    if header not in all_short_headers:
        print('mismatch', header)          #9개
```

9개의 mismatch

- **9개의 mismatch발견 (실습2.2)**

  데이터에는 있으나 정리된 헤더에는 없는 것

  - mismatch
  - mismatch MDV1F
  - mismatch MTA8E
  - mismatch mwelevel
  - mismatch mnweight
  - mismatch wscoreu
  - mismatch windex5u
  - mismatch wscorer
  - mismatch windex5r

mismatch 분석

- **삭제할 목록** -> 데이터파일에서 삭제
  - 소문자로 쓰인 타이틀은 유니세프 내부 지표 계산법과 관련된 것이며 우리가 현재 관심 있는 질문들과는 관련이 없다.

- **사용해야 할 목록** -> 헤더파일에 추가 필요
  - MDV1F와 MTA8E는 세계은행 웹사이트 웹 스크래퍼를 통해 발견되지 않은 헤더이다. 제거할 수 도 있고 사용할 수 도 있다.
  - 조사를 해보면, MDV1F는 가정 학대와 관련된 질문으로 설문 조사의 다른 항목에 있는 친밀한 사이에서의 학대와 연관성이 있으므로 사용하자.
  - MTA8E는 흡연하는 담배의 종류와 관련한 다른 질문들과 연관성이 있어 역시 사용하자.

실험 데이터세트3

- **MDV1F와 MTA8E를 추가한 헤더**
  - MDV1F와 MTA83 를 추가한 헤더 역시 github에 있음
  - 이와 같은 추가정보를 찾는 것이 어려운 과정

- 교재 파일 [1] ref. chap3
  - github : data / unicef / mn_headers_updated.csv

- 실험 데이터 Experiment Data
  - mn_header_updated.csv (github) - utf-8 encoding="utf-8"

정리된 헤더 점검

- **정리된 헤더 (실습3.1)**
  - 정리 된 헤더 준비 (152개) : header_rows[]
- **mismatch 7개를 데이터 목록에서 삭제 (실습3.2)**
  - mismatch 7개 데이터 index 추출
  - 7개 데이터 삭제 후 정리된 데이터목록 만들기 (152개)
- **어긋나는 위치 확인 (실습3.3)**
  - 정리된 데이터와 정리된 헤더 사이에 개수(152개)와 종류는 같지만 위치가 어긋난 부분이 있는지 확인 필요
  - ('MMC1', 'MTA1') 부터 어긋남

실습3.1

- 정리된 헤더 준비
  - 데이터와 헤더에 모두 있는 라벨만으로 정리된 헤더 리스트를 만든다

```
from csv import reader
#읽기
data_rdr = reader(open('mn_cut100.csv', 'rt'))
header_rdr = reader(open('mn_headers_updated.csv', 'rt',
encoding='utf8'))

#list comprehension
data_rows = [d for d in data_rdr]
#데이터에 없는 헤더는 제거하고 헤더 리스트를 만든다
header_rows = [h for h in header_rdr if h[0] in data_rows[0]]

#두 헤더의 길이 출력
print( len(data_rows[0]) )          #159
print( len(header_rows) )           #152
```

실습3.2

- **mismatch 7개를 데이터 목록에서 삭제**

```
#mismatch 7개 데이터 index 추출
skip_index = []
all_short_headers = [h[0] for h in header_rows]
for header in data_rows[0]:
    if header not in all_short_headers:
        index = data_rows[0].index(header)
        skip_index.append(index)
skip_index

#7 mismatch 삭제 159->152
new_data = []
for row in data_rows:
    new_row = []
    for i, d in enumerate(row):
        if i not in skip_index:
            new_row.append(d)
    new_data.append(new_row)
len(new_data[0])
```

실습3.3

- 어긋나는 위치 확인
  - enumerate는 list와 같이 순서가 있는 자료형을 인덱스 값을 포함하는 enumerate객체로 만든다. 활용예는 for문에서 인덱스와 값을 같이 변수로 처리하고자 하는 경우에 사용

```
#데이터에서 mismatch 7개를 삭제한 data_headers만들기
data_headers = []
for i, header in enumerate(data_rows[0]):
    if i not in skip_index:
        data_headers.append(header)

#어긋나는 위치 확인
# ('MMC1', 'MTA1')부터 어긋남
for i, header in enumerate(data_headers):
    if header not in all_short_headers[i]:
        print(header, all_short_headers[i])
```

헤더 교체 마무리

- **어긋난 부분 재배열 (실습4.1)**
  - mismatch 7개 데이터 index 추출할 때, 데이터의 헤더 순서대로 헤더의 순서를 조정하여 final_header_rows 리스트에 저장

- **데이터에 헤더설명 추가 (실습4.2)**
  - 헤더 설명 list 만들기
  - 데이터에 헤더설명 추가
  - 헤더 교체한 결과를 csv파일로 저장

실습4.1

- 어긋난 부분 재배열
  - 실습3.2에서 mismatch 7개 데이터 index 추출하는 부분만 변경

```
skip_index = []
final_header_rows = []
all_short_headers = [h[0] for h in header_rows]
for header in data_rows[0]:
    if header not in all_short_headers:
        index = data_rows[0].index(header)
        skip_index.append(index)
    #어긋나는 부분 재배열
    else:
        for head in header_rows:
            if head[0]==header:
                final_header_rows.append(head)
                break
skip_index
```

실습4.2

- 데이터에 헤더설명 추가
  - 데이터파일의 1행에 헤더 설명 list 추가

```
#헤더 설명 list 만들기
header1 = []
for row in final_header_rows:
    header1.append(row[1])
header1

#데이터에 헤더설명 추가
new_data.insert(1, header1)

#list -> dataframe csv저장
import pandas as pd
df = pd.DataFrame(new_data)
df.to_csv('mn_clean02.csv')
```

참고도서

- **reference**

[1] 파이썬을 활용한 데이터길들이기

- 프로그래밍인사이트

[2] 모두의 데이터분석

- 길벗

[3] 파이썬머신러닝 판다스데이터분석

- 정보문화사

End