DataAnalysis Study
4-2조 프로젝트 발표
(5.2 프로젝트 : 치킨 매장 주소의 시각화를 위한 데이터 전처리)

김지율
양지윤
정유정
천승원
황유림

목차

5.2.1 데이터 병합하기
분량상 제외

01. 5.2.2 치킨 매장 데이터 합치기

[1] Pandas (판다스)

파이썬 데이터분석 라이브러리로, 엑셀, 통계 데이터셋, 시계열 형식의 데이터과 같이 행, 열로 이루어진 데이터를 분석한다. 데이터의 집계 및 변환에 사용되며, 다양한 형식의 데이터 입출력이 가능하다는 특징이 있다. Series, Dataframe 2개의 자료구조를 가지고 있다.

[1-1] Series (시리즈) : 축 방향으로 label을 가지고 있는 1차원 형태의 배열

[1-2] DataFrame (데이터프레임) : 표 형식의 2차원 형태의 데이터 구조로 판다스의 가장 중요한 자료구조이다.

01. 5.2.2 치킨 매장 데이터 합치기

[2] UTF-8

UTF-8은 유니코드를 위한 가변 길이 문자 인코딩 방식이다. 파이썬은 문자를 바이트로 변환하는 인코딩 방식을 사용한다. 따라서 문자가 깨지는 문제를 해결하기 위해서 utf-8을 사용한다.

[3] CSV 파일

Comma Separator Value의 줄임말로, 엑셀에서 읽어 들일 수 있는 텍스트 형식의 파일이다.

[3-1] read_csv 함수 : CSV 파일을 읽어와서 데이터프레임을 생성하는 함수

(1) index_col : 해당 컬럼(열)을 색인으로 옮기면서 데이터를 읽기 위한 옵션

[3-2] to_csv 함수 : CSV 파일을 생성하는 함수

01. 5.2.2 치킨 매장 데이터 합치기

[4] concat 함수

두 개 이상의 데이터프레임에 대하여 행을 합칠 수 있는 함수

[4-1] axis

axis: 0이면 새로운 시리즈를 생성한다.

즉 행의 개수를 증가시키며, 만일 이것이 1이면 새로운 데이터프레임을 생성한다.

[4-2] ignore_idex

ignore_index=True면 기존의 row 색인을 무시하고 새로운 색인을 다시 생성한다.

01. 5.2.2 치킨 매장 데이터 합치기

[5] head 함수

앞에서부터 5개의 행을 추출하는 함수

[5-1] (　　=　　) = pd.options.display.max_columns = N

head 함수와 유사하게, 최대 N개의 칼럼의 수를 보여준다.

[6] info 함수

데이터에 대한 정보(행과 열의 크기, 컬럼명, 컬럼을 구성하는 데이터유형 등)을 출력한다.

02. 치킨 매장 데이터 합치기 프로젝트 코드 원본

```
import pandas as pd
from pandas import DataFrame

myencoding='utf-8'

chickenList=['cheogajip', 'goobne', 'nene', 'pelicana']
newframe=DataFrame()
print(newframe)
print('-' * 40)

for onestore in chickenList :
    filename=onestore + '.csv'
    myframe=pd.read_csv(filename, index_col=0, encoding=myencoding)
    newframe=pd.concat([newframe, myframe], axis=0, ignore_index=True)
    print(myframe.head())
    print('-' * 40)

print('a' * 40)
pd.options.display.max_columns = 10
print(newframe.head())
print('-' * 40)
totalfile='allstore.csv'
newframe.to_csv(totalfile, encoding='utf-8')
newframe.info()
print(totalfile + ' 파일 저장 완료')
print('-' * 40)
```

02. 치킨 매장 데이터 합치기 프로젝트 코드 설명(주석 버전)

```
# (1) 모듈세팅
import pandas as pd #pandas 모듈을 pd로 불러옴
from pandas import DataFrame # pandas 모듈의 DataFrame 불러옴

myencoding='utf-8' # 인코딩

#(2) 변수 chickenLst를 각 매장의 파일 이름을 리스트 형식으로 저장
# chickenList=['cheogajip', 'goobne', 'kyochon', 'nene', 'pelicana', 'bbq']
chickenList=['cheogajip', 'goobne', 'nene', 'pelicana']
newframe=DataFrame()
print(newframe)
print('-' * 40)

# (3) 반복문
# chickenList(각 매장의 파일)을 읽어와서 onestore 하나 씩 반복

for onestore in chickenList :
    filename=onestore + '.csv' # filename을 onestore+'.csv'로 CSV 파일 형식으로 저장

                # read_csv 함수 / filename의 CSV파일 중 0번째 열을 색인으로 읽음
    myframe=pd.read_csv(filename, index_col=0, encoding=myencoding)

                # concat 함수 / 새로운 색인을 다시 생성해서 newframe과 myframd의 데이터를 행으로 누적
    newframe=pd.concat([newframe, myframe], axis=0, ignore_index=True)
```

02. 치킨 매장 데이터 합치기 프로젝트 코드 설명(주석 버전)

```
# (4-1) head 함수 / 5개의 행을 출력
    print(myframe.head())
    print('-' * 40)
# (4-2) options.display.max_columns / 10개의 행을 출력
print('a' * 40)
pd.options.display.max_columns = 10
print(newframe.head())
print('-' * 40)

# (5) to_csv함수 / allstore.csv를 totalfile에 저장
totalfile='allstore.csv'
newframe.to_csv(totalfile, encoding='utf-8')

#(6) newframe의 정보 출력
newframe.info()
print(totalfile + ' 파일 저장 완료')
print('-' * 40)
```

03. 5.2.3 치킨 매장 데이터 전처리

데이터 전처리: 데이터를 분석 및 처리에 적합한 형태로 만드는 과정

| 일반적으로 수집한 데이터들은 가공하기 힘든 형태나 잘못된 형식으로
되어있는 경우가 많다.

| 잘못되어 있는 데이터들의 오류를 찾고, 이를 교정하는 과정을 전처리라고 한다.

| 데이터 분석 및 처리 과정에서 중요한 단계이며
데이터 분석, 데이터 마이닝, 머신 러닝 프로젝트에 적용된다.

03. 5.2.3 치킨 매장 데이터 전처리

<클래스 함수>

: def대신 class로 정의하는 객체 생성 함수 -> 호출하면 내부에 정의된 대로의 객체를 리턴함.

-객체: 파이썬에서 모든 데이터의 값(파이썬의 모든 것)

-파이썬의 객체는 속성이나 값을 나타내는 상태(state)와 메서드에
해당하는 행동(behavior)을 가질 수 있음.

-객체를 지향한다는 것 == 클래스를 지향한다는 것 == 새롭게 규정된 것을 만들어내서
그것을 공유하는 틀을 지향한다는 것

03. 5.2.3 치킨 매장 데이터 전처리

-일반 함수 사용

```
def some_function(something):
    print(something)
```

-클래스 함수 사용

```
class SomeClass:
    def __init__(self,something):
        self.something = something

    def some_function(self):
        print(self.something)
```

-호출할 때, 내부 코드를 실행하는 점에 대해서는 일반 함수와 차이 없음.

03. 5.2.3 치킨 매장 데이터 전처리

클래스 함수와 일반 함수의 차이

함수 -> 전역 클래스에 종속

클래스 -> 전역 클래스에 종속X 새로운 틀을 만듦

ex)함수에서는 함수 안에서 새로운 행동을 만들어 “x는 2다.” 라고 선언해도 함수 바깥에 나와서는 이미 전역 클래스에 담겨 있는 “x는 10이다.”라는 상태(사실)는 변하지 않음

클래스에서는 모두가 공유하는 것을 분명하게 정의하고 그 새로운 틀 안에 모든 것이 생김. 그러므로 클래스에서는 상태와 행동이 변하지 않고 지속적

03. 5.2.3 치킨 매장 데이터 전처리

클래스의 특징

1. 꼭 self를 붙여야만 고유한 상태 변수 유지가능
   -> 클래스로 객체가 생성될 때마다 고정된 위치를 기억해줘야 하기 때문

2. 클래스 내에 기재되어 있는 함수 : 메소드
   클래스 내에 여러 개의 메소드 정의 가능

3. self — 클래스 함수를 쓸 때 정해져 있는 형식
   "인스턴스 자신", "그 시점의 자신", "메소드 임의의 인수" 등으로 불림

4. __init__ — 초기화를 위한 메소드이며 컨스트럭터라고 부름

   객체를 생성할 때, 데이터의 초기를 실시하는 함수

   객체를 실행할 때, 처음에 호출되는 특수한 함수

04. 치킨 매장 데이터 전처리 프로젝트 코드

목적: 수집한 불완전한 데이터들을 수정하기 위한 전처리

파일들을 읽어온 후, 딕셔너리 정보로 변환한 다음 보정할 값들을 모두 치환한다. 보정된 데이터들을 출력한다.

```
import pandas as pd

pd.options.display.max_columns = 1000  # 표를 출력할 때 최대 열 수
pd.options.display.max_rows = 100  # 표를 출력할 때 최대 행 수

class ChickenCorrection():  #클래스
    myencoding = 'utf-8'

    def __init__(self, workfile):  #처음 실행되며 초기화하는 메소드
        self.workfile = workfile
        self.myframe = pd.read_csv(self.workfile, encoding = self.myencoding, /
                                   index_col = 0)  # 데이터 로딩하기
        # print(self.myframe.info())
        self.correctionSido()  # 각각의 메소드 호출하기
        self.correctionGungu()
        self.showMergeResult()
        self.correctionAddress()

        self.myframe.to_csv('allStoreModified.csv', encoding='utf-8')  # 데이터 저장하기
```

04. 치킨 매장 데이터 전처리 프로젝트 코드

수정 예시) '강원도'와 '강원'은 표현은 다르지만 같은 것이므로 하나의 정보로 만들어준다.

1. 시도 관련 데이터 수정 '강원'->'강원도' / '경기'->'경기도' / '세종', '세종시'-> '세종특별자치시'
2. 군구 관련 데이터 수정 '강서'->'강서구' / '강남'->'강남구'

```
def correctionAddress(self):  # 주소지 보정하기
    try:
        for idx in range(len(self.myframe)):
            # print(self.myframe.iloc[idx])
            imsiseries = self.myframe.iloc[idx]
            addrSplit = imsiseries['address'].split(' ')[2:]
            imsiAddress = [imsiseries['sido'], imsiseries['gungu']]
            imsiAddress = imsiAddress + addrSplit
            self.myframe.iloc[idx]['address'] = ' '.join(imsiAddress)
    except TypeError as err:
        pass

def showMergeResult(self):
    # 표준화된 행정 구역 데이터와 비교
    district = pd.read_csv('district.csv', encoding='utf-8')
    mergedData = pd.merge(self.myframe, district, on=['sido', 'gungu'], /
                          how='outer', suffixes=['', '_'], indicator = True)
    result = mergedData.query('_merge == "left_only"')
    # print('좌측에만 있는 데이터')
    # print(result[['sido', 'gungu', 'address']])
    # print('[' + result[['gungu']] + ']')
```

'district.csv': 전국의 시도와 군구의 표준화된 행정 구역 이름을 저장하고있는 파일

이 파일 이용하여 비표준 이름들을 표준화된 이름으로 변경하기

군구가 너무 많기 때문에 데이터를 육안으로 찾기 힘들다.

그러므로 merge()함수를 사용하여 2개의 데이터프레임을합친 후 데이터를 보정한다.

merge() 함수로 두 개의 데이터프레임을 병합한 후, sido 컬럼과 gungu 컬럼을 구분하고
그 이후(인덱스 2분터)부터 끝까지 슬라이싱하면 매장의 주소지가 된다.

04. 치킨 매장 데이터 전처리 프로젝트 코드

before : 주소지 보정하기

sido	gungu	address
서울특별시	노원구	서울시 노원구 공릉로58길 176, 상가동101호(하계동, 우방아파트)
서울특별시	광진구	서울시 광진구 광나루로545, 상강 제A동 1층 2호
서울특별시	동대문구	서울시 동대문구 고산자로56길 20

after : 주소지 보정하기

sido	gungu	address
서울특별시	노원구	서울특별시 노원구 공릉로58길 176, 상가동101호(하계동, 우방아파트)
서울특별시	광진구	서울특별시 광진구 광나루로545, 상강 제A동 1층 2호
서울특별시	동대문구	서울특별시 동대문구 고산자로56길 20

```
신규 주소지 = sido + gungu + address.split(' ')[2:]
```

04. 치킨 매장 데이터 전처리 프로젝트 코드

```
def correctionSido(self):  # 시도 보정하기
    # 제거할 데이터
    self.myframe = self.myframe[self.myframe['store'] != 'CNTTEST']  # pelicana 매장
    self.myframe = self.myframe[self.myframe['sido'] != '테스트']  # pelicana 매장

    # print('before sido')
    # print(np.sort(self.myframe['sido'].unique()))
    # print('-' * 40)

    sidofile = open('../05.1 치킨 매장 전처리/sido_correction.txt', 'r', encoding='utf-8')
    linelists = sidofile.readlines()

    #print(linelists)

    sido_dict = {}  # 시도 보정을 위한 사전
    for oneline in linelists :
        mydata = oneline.replace('₩n', '').split(':')
        # print(mydata[1])
        sido_dict[ mydata[0] ] = mydata[1]

    self.myframe.sido = /
        self.myframe.sido.apply(lambda data : sido_dict.get(data, data))

    # print('after sido')
    # print( np.sort(self.myframe['sido'].unique()))
    # print('-' * 40)
```

04. 치킨 매장 데이터 전처리 프로젝트 코드

```
    def correctionGungu(self):
        # print('before gungu')
        # print( self.myframe['gungu'].unique())
        # print('-' * 40)

        gungufile = open('../05.1 치킨 매장 전처리/gungu_correction.txt', 'r', encoding='utf-8')
        linelists = gungufile.readlines()

        # print(linelists)

        gungu_dict = {}
        for oneline in linelists :
            mydata = oneline.replace('₩n', '').split(':')
            # print(mydata[1])
            gungu_dict[mydata[0]] = mydata[1]

        self.myframe.gungu = ₩
            self.myframe.gungu.apply(lambda data : gungu_dict.get(data, data))

        # print('after gungu')
        # print(self.myframe['gungu'].unique())
        # print('-' * 40)
# end class

filename = 'allstore.csv'

chknStore = ChickenCorrection(filename)
```

각각의 데이터들을 분리하여 최종 결과물인

'allStoreModified.csv'를 생성하고 출력한다.

04. 치킨 매장 데이터 전처리 프로젝트 코드

allStoreModified.csv 파일의 일부 내용

	brand	store	saido	gungu	address	phone
0	cheogajip	장성점	전라남도	장성군	전라남도 장성군 장성읍 영천로133-2	061-393-9289
1	cheogajip	신사점	서울특별시	은평구	서울특별시 은평구 신사동 40-6	02-304-7770
2	cheogajip	중곡역점	서울특별시	광진구	서울특별시 광진구 긴고랑로 5, 1층 (중곡동)	02-3409-8292
3	cheogajip	응암점	서울특별시	은평구	서울특별시 은평구 백련산로36, 상가동106호	02-303-8295
4	cheogajip	돈암점	서울특별시	성북구	서울특별시 성북구 아리랑로6길 4, 1층(동선동5가)	02-6489-0101
5	cheogajip	거여점	서울특별시	송파구	서울특별시 송파구 거마로7길 3, 1층 101호(거여동)	02-3402-1511
6	cheogajip	밀양내이점	경상남도	밀양시	경상남도 밀양시 북성로4길 16	055-356-9989
7	cheogajip	남지점	경상남도	창녕군	경상남도 창녕군 남지읍 남지중앙로90	055-536-7333
8	cheogajip	신호동점	부산광역시	강서구	부산광역시 강서구 신호산단3로 66 1층 101호	051-831-0318
9	cheogajip	율하2지구점	경상남도	김해시	경상남도 김해시 율하로479-1, 331동 1층 104호	055-327-1811

Thank you
감사합니다