1. 서론
2. 데이터 전처리

본 프로젝트에서 사용하는 데이터는 서울교통공사에서 제공하는 역별 일별 지하철 승하차인원 데이터[[1]](#footnote-1)와 기상청에서 제공하는 기상예보 단기예보 정보이다.[[2]](#footnote-2) 전처리 후 목표로 하는 데이터의 컬럼은 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 날짜 | 기상 예측 정보(4시간 전) | | | 승하차총원 |
| 최고기온 | 최저기온 | 강수량 |
| (YYYY-MM-DD) | (℃) | ℃ | (mm) | (명) |

* 1. 승하차인원 데이터 전처리

|  |  |  |
| --- | --- | --- |
| 열 이름 | 형식 | 설명 |
| 날짜 | YYYY-MM-DD | 년, 월, 일로 구성된 날짜 |
| 호선 | string | 해당 역이 소속된 호선 |
| 역번호 | int | 해당 역의 ID |
| 역명 | string | 해당 역의 이름 |
| 구분 | string | 승차인원 혹은 하차인원 |
| 06:00이전 ~ 24:00 이후 (총 21개 열) | int | 시간대별 승(하)차인원. '06:00이전'과 '24:00 이후' 외에는 1시간 간격으로 집계한 승(하)차인원 데이터. |

서울교통공사에서 제공하는 역별 일별 지하철 승하차인원 데이터는 다음과 같이 구성됐다. 현재 2020년 1월 1일부터 2020년 10월 30일까지 모은 데이터이며 데이터는 총 168,434개 였다. Raw data의 각 열을 설명하면 다음과 같다.

이 데이터를 특정 기간, 특정 역에서 일별 승하차총원 데이터로 변환하기 위해서 Python을 통해서 전처리 과정을 수행하였다. 전처리 과정을 수행하는 subway\_passenger\_date() 함수는 다음과 같은 과정을 거친다. 우선 '날짜' 열의 데이터를 기준으로 YYYY-MM-DD부터 yyyy-mm-dd까지와 같이 특정 기간으로 데이터를 필터링하는 과정을 거쳤다. 두번재로 '역명' 열의 데이터를 기준으로, 사전에 정의한 특정 역들로 데이터를 필터링했다. 필터링의 결과로 특정 기간과 특정 역들의 승하차 데이터를 한정할 수 있다.

이렇게 필터링한 데이터에서 일별 승하차총원 값을 얻기 위해서 시간대에 해당하는 21개의 열의 값들을 모두 더한 '승하차총원' 열을 추가하였다. 그리고 '날짜'와 '승하차총원' 열만을 포함하는 데이터로 출력하였다.

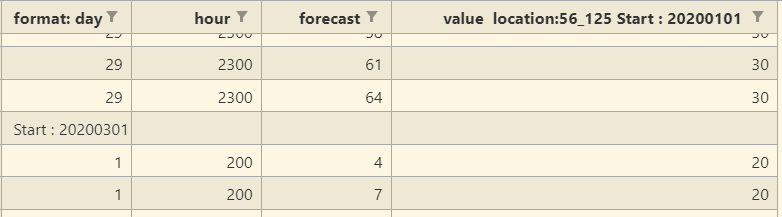
|  |
| --- |
| **def subway\_passenger\_date(csv\_path**  **, start\_date='2020-11-01'**  **, end\_date='2020-11-30'**  **, per\_station=False) :**  **# 파일 읽기 및 불필요 column삭제**  **df = pd.read\_csv(csv\_path, encoding='euc-kr')**  **del df['역번호']**  **# 조회 기간에 해당하는 행 조회**  **df\_d = df[(df['날짜'] >= start\_date) & (df['날짜'] <= end\_date)]**    **# 부천의 역들의 통계만 보도록 한다.**  **station\_in\_bucheon = ('까치울','부천종합운동장','춘의','신중동','부천시청','상동')**  **df\_db = df\_d[ df\_d.역명.isin(station\_in\_bucheon) ]**  **# 날짜별로 승하차 데이터 합치기**  **df\_dbd = df\_db.groupby(by=['날짜'], as\_index=False).sum()**  **df\_dbd['승하차총원'] = df\_dbd.apply(lambda x: sum(x[1:21]), axis=1)**  **df\_dbds = df\_dbd.iloc[:,[0,-1]]**    **return df\_dbds** |
| **>>>**subway\_passenger\_date('서울교통공사승하차인원.csv',  start\_date='2020-10-01',  end\_date='2020-10-31') |
| |  |  |  | | --- | --- | --- | |  | 날짜 | 승하차총원 | | 0 | 2020-10-01 | 28912 | | 1 | 2020-10-02 | 43476 | | 2 | 2020-10-03 | 52262 |   ... |

* 1. 단기예보 데이터 전처리

기상청에서 제공하는 단기예보 데이터에서 동, 읍, 면 관측소마다 측정한 강수확률, 일최고기온, 일최저기온, 풍속 데이터를 수집하였다. 이 4가지 Raw 데이터는 다음의 열(column)을 동일하게 가지고 있었다.

|  |  |  |
| --- | --- | --- |
| 열 이름 | 형식 | 설명 |
| format: day | int | 예보 보도일(日) |
| hour | int | 예보 보도 시간 |
| forecast | int | 예보 시간. +4 ~ +58~67시간  ex) 4 → 4시간 뒤 해당 기상정보 |
| value location:56\_125 Start: 20200101 | int | 해당 데이터의 수치 |

Raw 데이터 열 정보에서 볼 수 있듯이, 날짜를 YYYY-MM-DD의 형식으로 표현하지 않고 일(日)에 해당하는 수치만 존재하였으며, 각각의 월(月) 정보는 다음과 같이 레코스 사이에 행으로 구분되어서 레코드의 정리와 날짜 데이터 변환이 필요했다.



따라서 4가지의 데이터에 각각 전처리를 수행하여 각 데이터의 일별 평균값을 출력하는 highest\_temperature\_date(), lowest\_temperature\_date(), wind\_speed\_date(), rain\_possibility\_date() 함수를 구축하였다. 각 데이터의 구조와 정리가 필요한 오류들은 같은 형태였고, 따라서 함수명과 열 이름을 제외한 다른 과정은 동일하게 진행했다. 다음은 그 중 하나인 highest\_temperature\_date()의 코드이다.

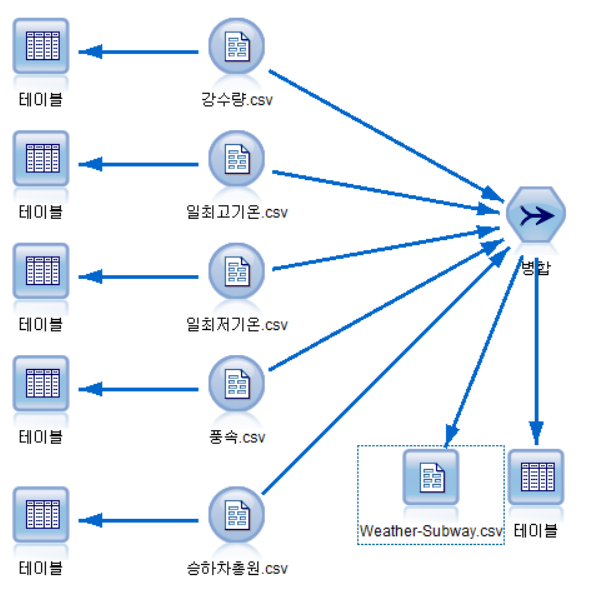
우선 열 이름을 직관적으로 변경하고, 레코드 사이에 있는 월(月) 구분자가 'Start : 20200301'의 String형태였으므로 split()함수 실행후 뒤쪽 숫자만 정수로 형변환하여 취하였다. 그 후 일(日) 정보로 간주되는 31일 보다 작은 정수가 아니라면 필터링되게 하여 제거하였다. 그리고 예보 시간을 4시간인 데이터들을 필터링했고, 일별로 집계연산을 수행하여 일별 최고기온 평균값이 출력되도록하였다.

일최저기온, 강수확률, 풍속 데이터의 전처리 과정은 열 이름 변경 외에 모두 동일하다.

|  |
| --- |
| **def change\_to\_dateformat(arr):**  **global previous\_day**  **global current\_month**  **global current\_year**  **if arr['date'] < previous\_day :**  **current\_month = current\_month + 1**  **previous\_day = arr['date']**  **return str(dt.date(current\_year, current\_month, int(previous\_day)))**  **def highest\_temperature\_date(csv\_path):**  ***# 파일 읽어오기, 월 구분 memo 제거***  ***df = pd.read\_csv(csv\_path)***  ***df = df.rename(columns={df.columns[0]: 'date', str(df.columns[-1]) : 'highest temperature'})***  ***df['date'] = df['date'].map(lambda x : int(x.split()[-1]))***  ***df = df[(df['date'] <= 31) & (df['date'] >= 1)]***    ***# 4시간 전 예보로 축소***  ***df = df[df['forecast'] <= 4]***  ***# 날짜를 date 폼으로 변환***  ***global previous\_day***  ***global current\_month***  ***global current\_year***  ***previous\_day = 1***  ***current\_month = 1***  ***current\_year = 2020***  ***df['date'] = df.apply(change\_to\_dateformat, axis=1)***  ***# 일별 최고기온 데이터 평균 계산***  ***df = df.groupby(by=['date']).mean()***  ***return df.iloc[:,[-1]]*** |
| **>>> highest\_temperature\_date('심곡1동\_일최고기온\_202001\_202010.csv')** |
| |  |  | | --- | --- | |  | **highest temperature** | | **date** |  | | **2020-01-01** | **4.750** | | **2020-01-02** | **5.000** | | **2020-01-03** | **6.000** |   **...** |

* 1. 데이터 병합

IBM SPSS에서 일최고기온, 일최저기온, 풍속, 강수확률, 승하차총원 5가지 데이터를 병합하였다. 레코드작업의 병합을 이용하여 데이터들을 분석할 수 있었다.



1. 데이터분석

1. 서울교통공사, 서울교통공사\_역별\_일별\_시간대별\_승하차인원, url=<https://www.data.go.kr/data/15060424/fileData.do> [↑](#footnote-ref-1)
2. 기상청 기상자료개방포털, 데이터> 기상예보> 동네예보> 단기예보, url=<https://data.kma.go.kr/data/rmt/rmtList.do?code=400&pgmNo=570> [↑](#footnote-ref-2)