텍스트 마이닝과 데이터 마이닝

Part 08. 데이터 마이닝 실습

정 정 민

programmers

Chapter 22. 트래픽 신호 시계열 데이터 실습

1. 트래픽 신호 시계열 데이터
2. 시계열 데이터 시각화
3. 주기 분석

트래픽 신호 시계열 데이터

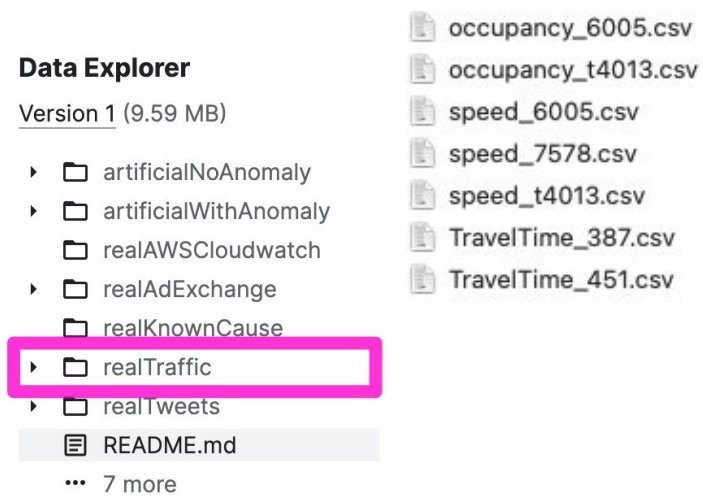
강의 실습 코드 : 링크

트래픽 신호 시계열 데이터

- 미국 미네소타 트윈 시티 메트로 지역의 교통 트래픽을 수집한 데이터
- 교통 관리 기관 (Minnesota Department of Transportation)에서 수집한 데이터
  - 교통 혼잡 상태를 모니터링

- Kaggle에서 공개된 시계열 데이터세트 중 하나 (링크)
- 다운로드 받은 후 realTraffic 폴더의 데이터만 사용
  - 다운로드 받은 후 준비해주세요!

- 2015년 7월 ~ 9월 사이 임의의 시간 동안 교통 센서 데이터 포함
- 교통량(Occupancy), 속도(Speed), 특정 구간 소요 시간 (Travel Time)
- 특정 센서마다의 이름이 뒤에 명시
  - 해당 센서가 설치된 위치를 의미



시계열 데이터 time 처리

- 시계열 데이터에서 중요한 성분은 시간(timestamp)!

- 이 값들은 서로 시간적 계산이 가능
  - 시간차 계산, 시간 범위 세팅, 시간 기준으로 리샘플링 등
- 또한, 이 데이터 형을 활용하면 관련 패키지에서 시간 연산을 수월히 할 수 있음

- 기본적으로 pandas의 to_datetime함수를 활용
  - 이 함수의 입력 인자는 공식 문서 확인 (링크)

- 가능한 시계열 데이터 내에 있는 시간 관련 값을 datetime으로 변경!

```
occupancy_6005 = pd.read_csv("occupancy_6005.csv")
occupancy_6005['timestamp'] = pd.to_datetime(occupancy_6005['timestamp'])
occupancy_6005.head()
```

	timestamp	value
0	2015-09-01 13:45:00	3.06
1	2015-09-01 13:50:00	6.44
2	2015-09-01 13:55:00	5.17
3	2015-09-01 14:00:00	3.83
4	2015-09-01 14:05:00	4.50

시계열 데이터 시각화

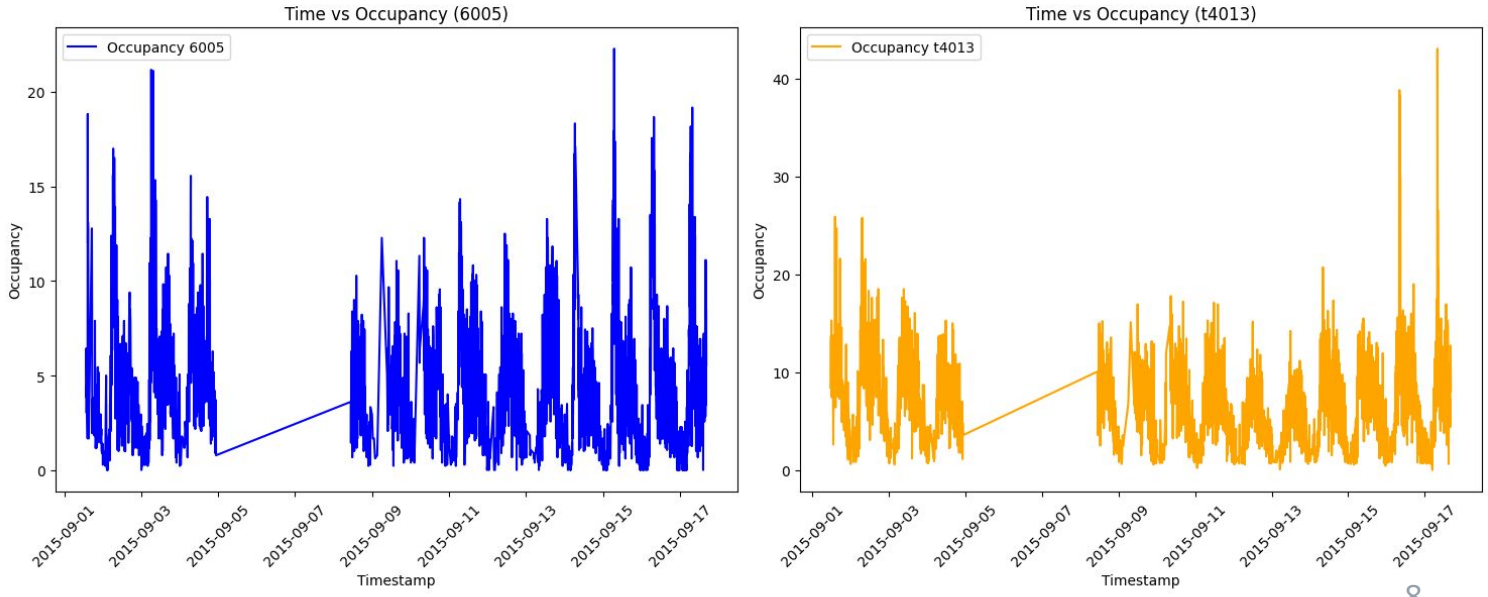
강의 실습 코드 : 링크

matplotlib

- 데이터를 시각화 하기 위한 기본 패키지

- 시간 축을 x축으로, 값을 y축으로 설정
- 서로 다른 데이터를 subplot으로 한번에 시각화 가능

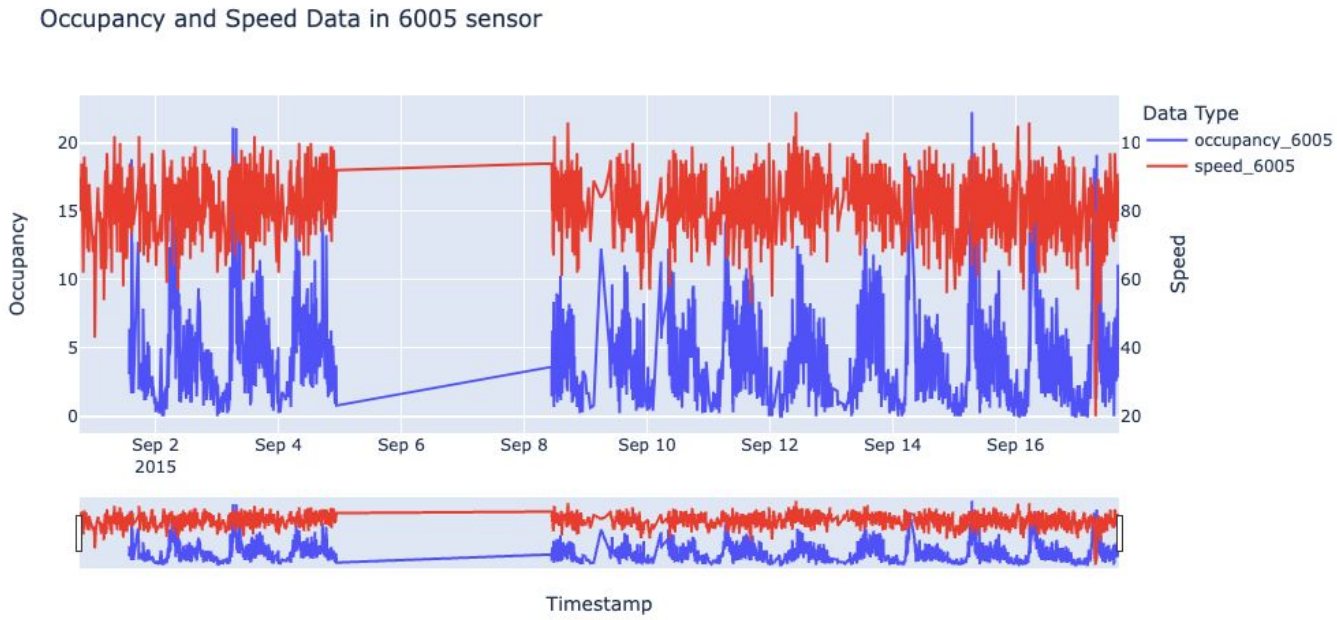
- 하지만 시간적으로 길이가 길다면
- 데이터의 특징을 눈으로 보기 어려움



plotly

- 데이터 시각화를 위해 사용하는 패키지 중 하나
- Python, R, JS 등에서 사용할 수 있는 그래프 생성 라이브러리

- 사용자가 시계열 데이터를 쉽게 탐색할 수 있는 기능 제공
  - 줌 인/아웃 및 슬라이딩
  - 여러 변수 간의 관계 탐색
  - 마우스 오버로 정보 확인
  - 등등



주기 분석

강의 실습 코드 : 링크

시계열 데이터와 주기

- 시계열 데이터의 경우
- 적지 않은 경우로 주기성을 갖는 경우가 있음

- 아래와 같은 반복 예시가 있을 수 있음
  - 매 일을 기준으로 : 출/퇴근 교통량 분석, 일일 웹사이트 트래픽 등
  - 매 주를 기준으로 : 주간 판매량, TV 프로그램 시청 현황 등
  - 매 달을 기준으로 : 월 판매량, 신용카드 소비 패턴
  - 계절을 기준으로 : 농작물 가격, 패션 트랜드, 관광 패턴 등

- 주기를 잘 이용하면 시계열 분석에서 의미 있는 통찰을 확인할 수 있음

Occupancy_6005 데이터로 주기 데이터 변경

- 사용하는 데이터는 교통량 관련 데이터이므로
- 아마도 하루를 기준으로 반복될 것이라 예상

- 원본 데이터의 시간 정보를 바탕으로 시간(Hour) 정보를 추출
- 추출된 시간(Hour) 정보를 활용해 동일한 시간 정보가 같은 데이터끼리 통합
- 결국 0~24시간 데이터끼리 평균을 취해 최종 주기 데이터를 생성

```
occupancy_6005['hour'] = occupancy_6005['timestamp'].dt.hour
avg_occupancy_by_hour = occupancy_6005.groupby('hour')['value'].mean().reset_index()
```

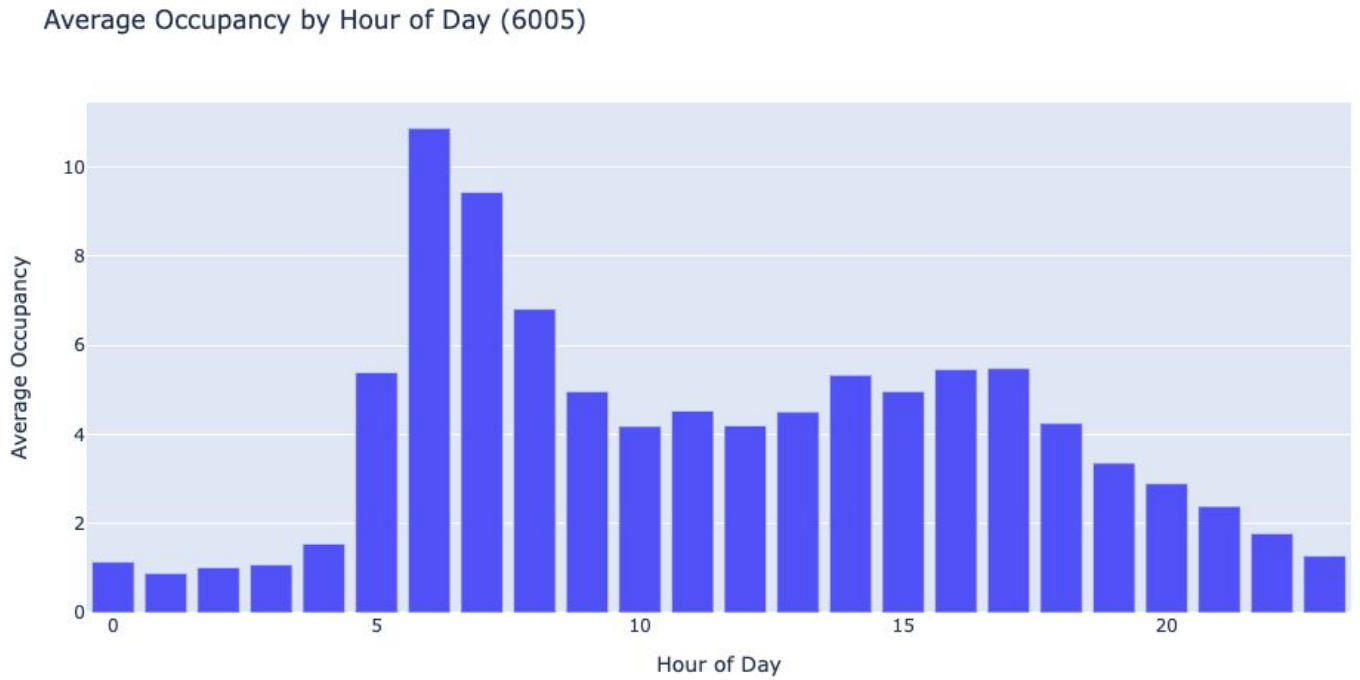
	hour	value
0	0	1.137679
1	1	0.884400
2	2	1.013846
3	3	1.074902
4	4	1.545000
5	5	5.392688
6	6	10.873137
7	7	9.437083
8	8	6.813211
9	9	4.963486
10	10	4.182937
11	11	4.528430
12	12	4.197097
13	13	4.507846
14	14	5.332971
15	15	4.964173
16	16	5.461240
17	17	5.481600
18	18	4.251074
19	19	3.358257
20	20	2.897692
21	21	2.386556
22	22	1.777857
23	23	1.272812

주기 데이터 시각화 (Bar plot)

- Bar Plot의 형태로 시각화 진행

- 교통량의 경우
- 출근과 퇴근 시간이 많을 것으로 판단되며

- 그것이 시각적으로 확인 가능



주기 데이터 시각화 (히트맵 활용)

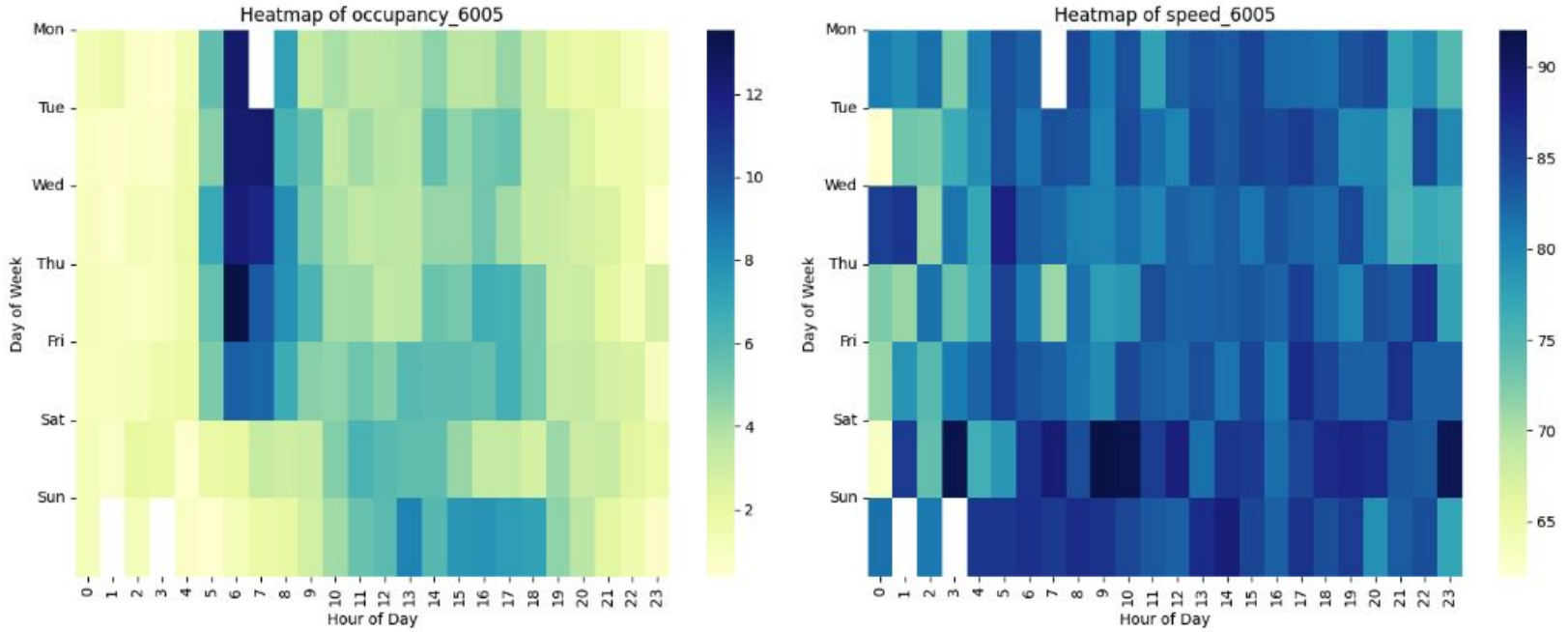
- 조금 더 확장해
- 하루 주기성(시간 별)과 주간 주기성(요일 별)을 기준으로 확인

- 필요한 주기가 2 종류이므로
- 이에 따른 시간대 정리가 필요하며
- 그룹 및 평균 계산 과정이 필요

- 교통량과 차량 속도에
  큰 상관관계가 보이지는 않음

```
data['day_of_week'] = data['timestamp'].dt.dayofweek
data['hour'] = data['timestamp'].dt.hour

heatmap = data.groupby(['day_of_week', 'hour'])['value'].mean().unstack()
```



계절성 분석

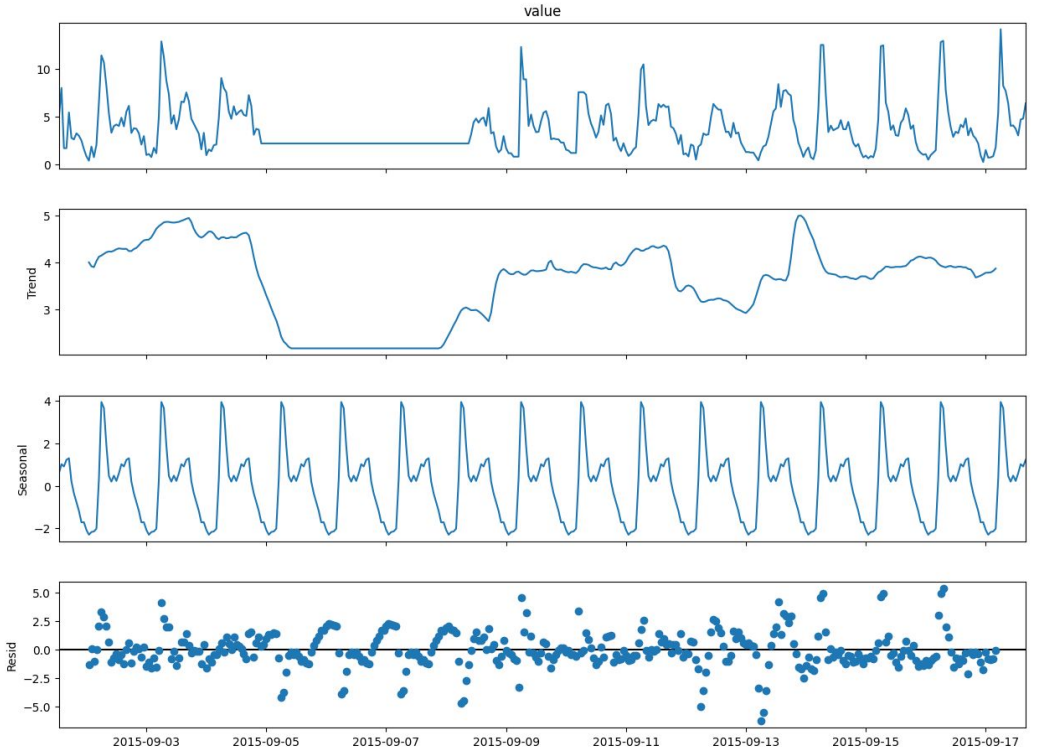
- 특정 주기 갖는 데이터는 계절성 분석을 진행할 수 있음
  - 주, 달, 분기, 계절, 년 등의 어떠한 시간적 주기를 계절성 주기라고 함

- 시계열 데이터는 아래와 같이 3가지 성분으로 구성된다고 가정
  - **Trend**
    - 시간에 따른 데이터 흐름
    - 전반적인 데이터 경향성
  - **Seasonal**
    - 어떠한 주기 때문에 발생되는 데이터 변동
  - **Residual**
    - Trend와 Seasonal로 예측되지 못하는 추가 변동분
    - 노이즈 혹은 비 주기적 정보로 인한 부분
  - 원래 데이터 = Trend + Seasonal + Residual (additive 방식)

- 사용자가 갖고 있는 도메인 지식적 주기 이외의 새로운 주기 혹은 인사이트를 찾아낼 수 있음

Trend, Seasonal, Residual

- **Trend**란, 데이터 전반에 걸쳐 보여지는 일반적인 경향성을 의미하며
  - 시간적 구간(window)을 잡아 해당 구간의 평균값 활용
  - 윈도윙 방식으로 시간의 축 방향으로 연속적으로 계산
- **Seasonal**이란, 데이터에 존재하는 주기적인 성분을 측정
  - 원본 데이터에서 Trend 파트를 제거하고
  - 사용자가 제공한 주기(period) 단위로 값의 평균을 활용
- **Residual**은 Trend와 Seasonal 값으로 설명되지 않는 값으로
  - 원본 데이터 – Trend – Seasonl 과정으로 도출
  - 이 값이 너무 크다면
    - 사용자가 제공한 주기(period)가 터무니 없을 수 있음
    - 계절성 분석으로 분석하기에 데이터가 너무 복잡하거나
    - 노이즈와 이상치에 심한 영향을 받은 데이터일 수 있음



E.O.D