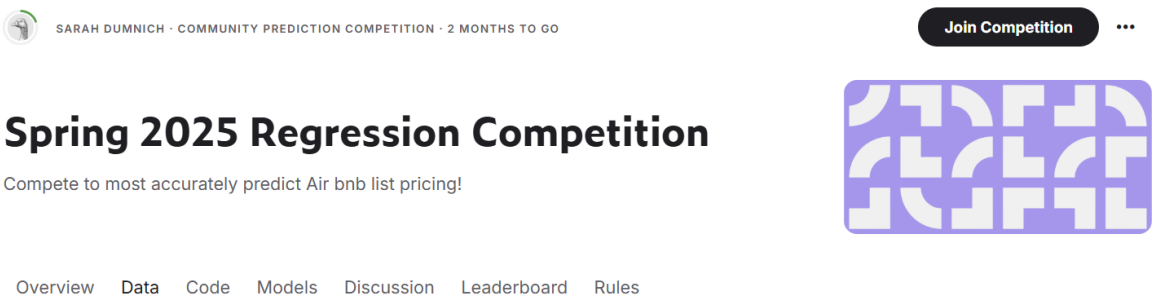
에어비앤비 가격예측 프로젝트



Spring 2025 Regression Competition | Kaggle

요약

- 에어비앤비 가격을 예측하는 프로젝트 (뉴욕시 호텔 데이터, 집값 데이터를 추가 데이터로 활용)

역할

- 프로젝트 전체 인원: 4명
- 나의 역할: 변수를 인원대로 나눠서 전처리, 호텔 데이터 전처리 및 각 에어비앤비로부터 0.8km 반경 이내의 호텔 개수를 출력하는 제작 (0.8km: 뉴욕 시내 지하철 역간 평균 거리)
- 나의 기여도: 프로젝트 초반부에 방향성을 잡는 데 기여했다고 생각한다.

성과

- 리더보드 점수: 80437.84973

시기

- 프로젝트 진행 기간 (ex. 2024.12.27 ~ 2025.01.31) .

📝진행 과정 속 나의 역할 수행

[추가 데이터 병합 과정 중 방향성 잡기]

- 모델의 예측 성능을 높이기 위해 추가로 활용할 수 있는 데이터를 찾았고, 호텔 데이터와 집값 데이터로 정하였다. 처음에 찾은 호텔 데이터는 뉴욕주 데이터인 반면, 에어비앤비 데이터는 뉴욕시 데이터임을 발견하여 기존 데이터와 겹치는 데이터가 적었다. 그래서 뉴욕시 호텔 데이터를 찾아야 한다는 초기 방향성을 잡을 수 있었다.

⇒ 추가 데이터를 사용할 때는 기존 데이터와 호환이 잘 될 수 있는지를 세밀하게 살펴봐야 함을 배움

[에어비앤비 데이터 변수 전처리]

- neighbourhood, price, reviews_per_month에 대한 전처리를 담당했다. 타겟 변수 price의 로그변환을 하지 않은 전처리를 하여 앞으로는 타겟 변수의 분포가 정규분포를 따르지 않을 시에는 로그 변환은 잊으면 안된다는 점을 상기할 수 있었고, reviews_per_month의 경우는 상관계수가 매우 낮아 삭제를 할지 말지 고민해야 하는 상황이다.

⇒ 타겟 변수의 분포를 살펴보고 정규분포를 따르지 않으면 로그 변환 필수

[haversine 함수 이용]

- haversine함수를 사용하여 각 에어비앤비마다 특정 반경 안에 있는 호텔의 개수를 구하는 코드를 알게 됨. 단지 코드 돌아가는 시간이 길어 런타임이 자꾸 끊기는 문제가 발생한다.

⇒ 사양이 좋은 다른 조원의 컴퓨터로 코드를 돌리니 해결. 컴퓨터 사양이 좋아야 하는 것인가...

```python
from haversine import haversine, Unit
import pandas as pd

# 0.8km 이내의 호텔 개수를 저장할 리스트
hotels_within_800m = []

# 데이터 인덱스 재정렬
airbnb_2 = test.reset_index(drop=True)
hotel_2 = hotels.reset_index(drop=True)

# 에어비앤비별로 호텔 데이터를 비교
for i in range(airbnb_2.shape[0]):
    airbnb_location = (airbnb_2["latitude"][i], airbnb_2["longitude"][i])
    count = 0  # 0.8km 이내 호텔 개수 초기화

    for j in range(hotel_2.shape[0]):
        hotel_location = (hotel_2['Latitude'][j], hotel_2['Longitude'][j])
        distance = haversine(airbnb_location, hotel_location, unit=Unit.KILOMETERS)

        # 0.8km 이내인 경우 카운트 증가
        if distance <= 0.4:
            count += 1

    # 에어비앤비별 0.8km 이내 호텔 개수를 리스트에 추가
    hotels_within_800m.append(count)

# 결과 출력
airbnb_2['Hotels_Within_0.8km'] = hotels_within_800m
print(airbnb_2[['latitude', 'longitude', 'Hotels_Within_0.8km']])
```

	latitude	longitude	Hotels_Within_0.8km
0	40.700001	-73.970001	0
1	40.700001	-73.809998	40
2	40.740002	-73.989998	33
3	40.669998	-73.959999	5
4	40.720001	-74.000000	65
...	...	...	...
6842	40.810001	-73.940002	15
6843	40.650002	-73.919998	0
6844	40.750000	-73.970001	1050
6845	40.669998	-73.970001	0
6846	40.759998	-74.000000	17

[OneHotEncoder의 handle_unknown='ignore' 메서드]

- 모델 적용 과정에서 오류가 자꾸 발생했는데 이는 훈련데이터의 neighbourhood와 테스트 데이터의 neighbourhood의 종류가 완전히 같지 않아 생기는 오류였다. 이에 대한 해결책으로OneHotEncoder의 handle_unknown='ignore' 를 통해 훈련 데이터에 없던 새로운 범주(unknown category)가 테스트 데이터에 등장해도 에러를 발생시키지 않고 그 범주를 0으로 처리할 수 있음을 새로 알게 되었다.

⇒ 훈련 데이터에 없는 테스트 데이터가 있을 때 오류 발생: 원핫인코딩의 handle_unknown='ignore' 로 해결