# 데이터마이닝이론및응용 HW3

**짱짱조**

## Data Pre-Processing

### 1. 데이터셋 정보

출처:https://public.opendatasoft.com/explore/dataset/airbnb-listings/table/?disjunctive.host_verifications&disjunctive.amenities&disjunctive.features

Opendatasoft 에 공개된 에어비앤비(Airbnb) 숙소 리스트 데이터를 활용하였다. 전 세계 도시에서 실제로 Airbnb 가 어떻게 사용되는지 알 수 있는 숙소 리스트 데이터이다. 공개된 494,954 개의 데이터 중 **미국 뉴욕시에 등록된 10,000 개의 숙소 등록 정보를 API 를 통해 받아와 데이터셋으로 사용하였다**

데이터셋에는 숙소 정보(침대 개수, 구비 물품, 방 개수 등), 호스트 정보(연락 수단, 슈퍼호스트 인증 등), 예약 조건(예약 가능 인원, 가격, 숙박 가능 기간 등), 리뷰(청결도, 의사소통, 위치 등)에 대한 다양한 변수가 포함된 정보가 담겨 있다. 우선 데이터셋에 있는 총 89 개의 Columns 중 분석에 활용할 수 있는 것들만 선별하기로 하였다. 중복되는 항목이거나 숫자로 변환할 수 없는 columns 를 제거하고 남은 30 개의 항목을 선별하였다.

### 2. 데이터 전처리

PCA, FA 를 진행하기 위해 범주형 데이터를 숫자로 변환해야 한다. 변환하려고 하는 Object 타입의 범주형 데이터에 대하여 다음과 같이 처리했다.

**bed_type, cancellation_policy, host_response_time, room_type** : 각 항목의 객체마다 유의미한 차이를 보이므로 Category embedding 을 다음과 같이 진행하였다.

```
def bed_type(row):
  bed = row['bed_type']
  if(bed == 'Real Bed'):
    return 1
  elif(bed=='Futon'):
    return 2
  elif(bed=='Airbed'):
    return 3
  elif(bed=='Couch'):
    return 4
  elif(bed=='Pull-out Sofa'):
    return 5
  else:
    return 0
```

```
def cancellation(row):
  cancel = row['cancellation_policy']
  if(cancel == 'flexible'):
    return 1
  elif(cancel == 'moderate'):
    return 2
  elif(cancel == 'strict'):
    return 3
  elif(cancel.index('super_strict')!=-1) :
    return 4
  else:
    return 0
```

```
def response_time(row):
  r_time = row['host_response_time']
  if(r_time == 'within an hour'):
    return 0
  elif(r_time == 'within a few hours'):
    return 1
  elif(r_time == 'within a day'):
    return 2
  elif(r_time == 'a few days or more'):
    return 3
  else:
    return 4
```

```
def room_type(row):
  r_type = row['room_type']
  if(r_type == 'Shared room'):
    return 0
  elif(r_type == 'Private room'):
    return 1
  elif(r_type == 'Entire home/apt'):
    return 2
  else:
    return 3
```

**amenities**: 숙소에 비치한 amenities 의 수에 따라 변환하였다.

```
df.loc[idx, 'amenities'] = len(row['amenities'].split(',')) if row['amenities']!=float('nan') else 0
```

**features**: 각 숙소의 특징에 대한 항목을 체크해 놓았기에 One-Hot Encoding 으로 처리하였다.

```
df.loc[idx, 'host_has_profile_pic'] = 1 if row['features'].find('Host Has Profile Pic') != -1 else 0
df.loc[idx, 'host_identity_verified'] = 1 if row['features'].find('Host Identity Verified') != -1 else 0
df.loc[idx, 'host_is_superhost'] = 1 if row['features'].find('Host Is Superhost') != -1 else 0
df.loc[idx, 'instant_bookable'] = 1 if row['features'].find('Instant Bookable') != -1 else 0
df.loc[idx, 'is_location_exact'] = 1 if row['features'].find('Is Location Exact') != -1 else 0
df.loc[idx, 'instant_bookable'] = 1 if row['features'].find('Instant Bookable') != -1 else 0
df.loc[idx, 'require_guest_phone_verification'] = 1 if row['features'].find('Require Guest Phone Verification') != -1 else 0
df.loc[idx, 'require_guest_profile_picture'] = 1 if row['features'].find('Require Guest Profile Picture') != -1 else 0
```

**host_verifications**: 호스트와 연락가능한 수단의 수에 따라 변환하였다.

```
df.loc[idx, 'host_verifications_num'] = len(row['host_verifications'].split(',')) if row['host_verifications']!='None' else 0
```

마지막으로 남은 데이터 중 결측값을 모두 0 으로 대체하였다. 따라서 다음과 같이 전처리가 잘 되었음을 확인할 수 있었다.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 35 columns):

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | accommodates | 10000 non-null | float64 |
| 1 | amenities | 10000 non-null | int64 |
| 2 | availability_365 | 10000 non-null | int64 |
| 3 | bathrooms | 10000 non-null | float64 |
| 4 | bed_type | 10000 non-null | int64 |
| 5 | bedrooms | 10000 non-null | float64 |
| 6 | beds | 10000 non-null | float64 |
| 7 | cancellation_policy | 10000 non-null | int64 |
| 8 | cleaning_fee | 10000 non-null | float64 |
| 9 | extra_people | 10000 non-null | int64 |
| 10 | host_has_profile_pic | 10000 non-null | float64 |
| 11 | host_identity_verified | 10000 non-null | float64 |
| 12 | host_is_superhost | 10000 non-null | float64 |
| 13 | host_response_rate | 10000 non-null | float64 |
| 14 | host_response_time | 10000 non-null | int64 |
| 15 | host_total_listings_count | 10000 non-null | float64 |
| 16 | host_verifications_num | 10000 non-null | float64 |
| 17 | instant_bookable | 10000 non-null | float64 |
| 18 | is_location_exact | 10000 non-null | float64 |

```
 19  maximum_nights                      10000 non-null  int64
 20  minimum_nights                      10000 non-null  int64
 21  number_of_reviews                   10000 non-null  int64
 22  price                               10000 non-null  float64
 23  require_guest_phone_verification    10000 non-null  float64
 24  require_guest_profile_picture       10000 non-null  float64
 25  review_scores_accuracy              10000 non-null  float64
 26  review_scores_checkin               10000 non-null  float64
 27  review_scores_cleanliness           10000 non-null  float64
 28  review_scores_communication         10000 non-null  float64
 29  review_scores_location              10000 non-null  float64
 30  review_scores_rating                10000 non-null  float64
 31  review_scores_value                 10000 non-null  float64
 32  reviews_per_month                   10000 non-null  float64
 33  room_type                           10000 non-null  int64
 34  security_deposit                    10000 non-null  float64
dtypes: float64(25), int64(10)
memory usage: 2.7 MB
```

### 3. 기초통계량 확인

전처리 후 분석에 사용될 35 개의 항목에 대한 기초통계량도 확인하였다. (첨부된 코드 참고: df.describe)

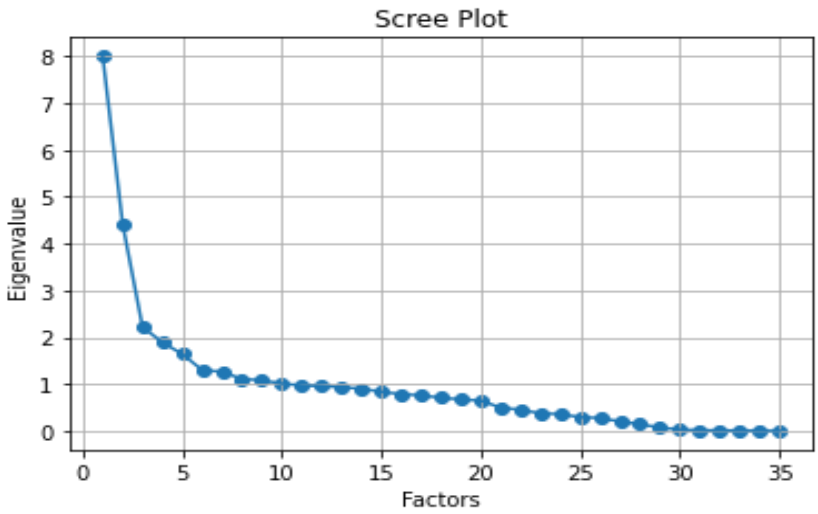
# PCA

### 1. eigenvalue

z-표준화를 통화 데이터를 스케일링 해준 후 PCA 를 적용해준다. 이 때, FactorAnalyzer 모듈을 활용하여 eigenvalue 를 구해보면 다음과 같다.

```
array([7.99893736e+00, 4.42294700e+00, 2.22203762e+00, 1.89416092e+00,
       1.64367928e+00, 1.30282650e+00, 1.26823449e+00, 1.11371936e+00,
       1.08578011e+00, 1.01467399e+00, 9.74454076e-01, 9.63634895e-01,
       9.37751806e-01, 9.06475884e-01, 8.40004230e-01, 7.89270924e-01,
       7.64799517e-01, 7.12153968e-01, 6.78589874e-01, 6.51014090e-01,
       5.01904930e-01, 4.56373889e-01, 3.79535830e-01, 3.63983369e-01,
       2.97437987e-01, 2.81635327e-01, 2.05532188e-01, 1.50124895e-01,
       7.49542763e-02, 3.58890528e-02, 1.99598037e-02, 1.63179120e-02,
       1.32244410e-02, 1.05295742e-02, 7.45063774e-03])
```

이를 시각화하기 위하여 다음과 같은 Scree Plot 을 그릴 수 있다.

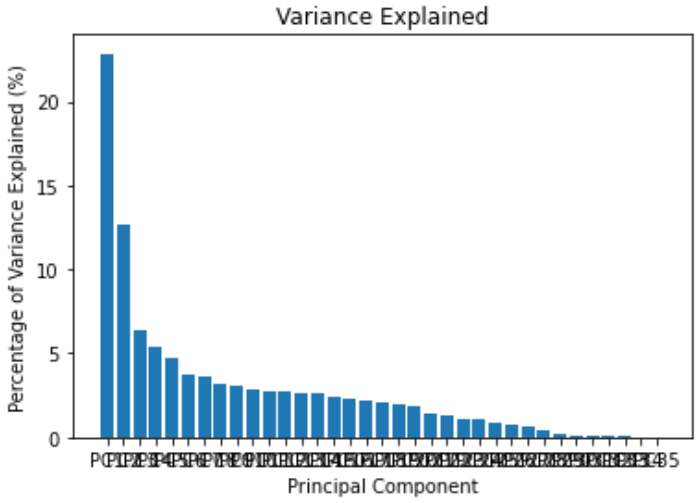



## 2. 분산 설명 비율과 누적 분산 설명 비율

또한 PCA 로 추출된 각 PC 의 분산 설명 비율을 구해보면 다음과 같다.

```
array([0.20537245, 0.11007547, 0.08570808, 0.0740931 , 0.06192651,
       0.0429433 , 0.03358156, 0.03196825, 0.02875957, 0.02752355,
       0.02705493, 0.02607199, 0.02493012, 0.02386251, 0.02252363,
       0.02173221, 0.0205807 , 0.01978013, 0.01930558, 0.01795684,
       0.01693464, 0.01602686, 0.01551218, 0.01527427, 0.01050156])
```

이를 시각화해보면 다음과 같다.

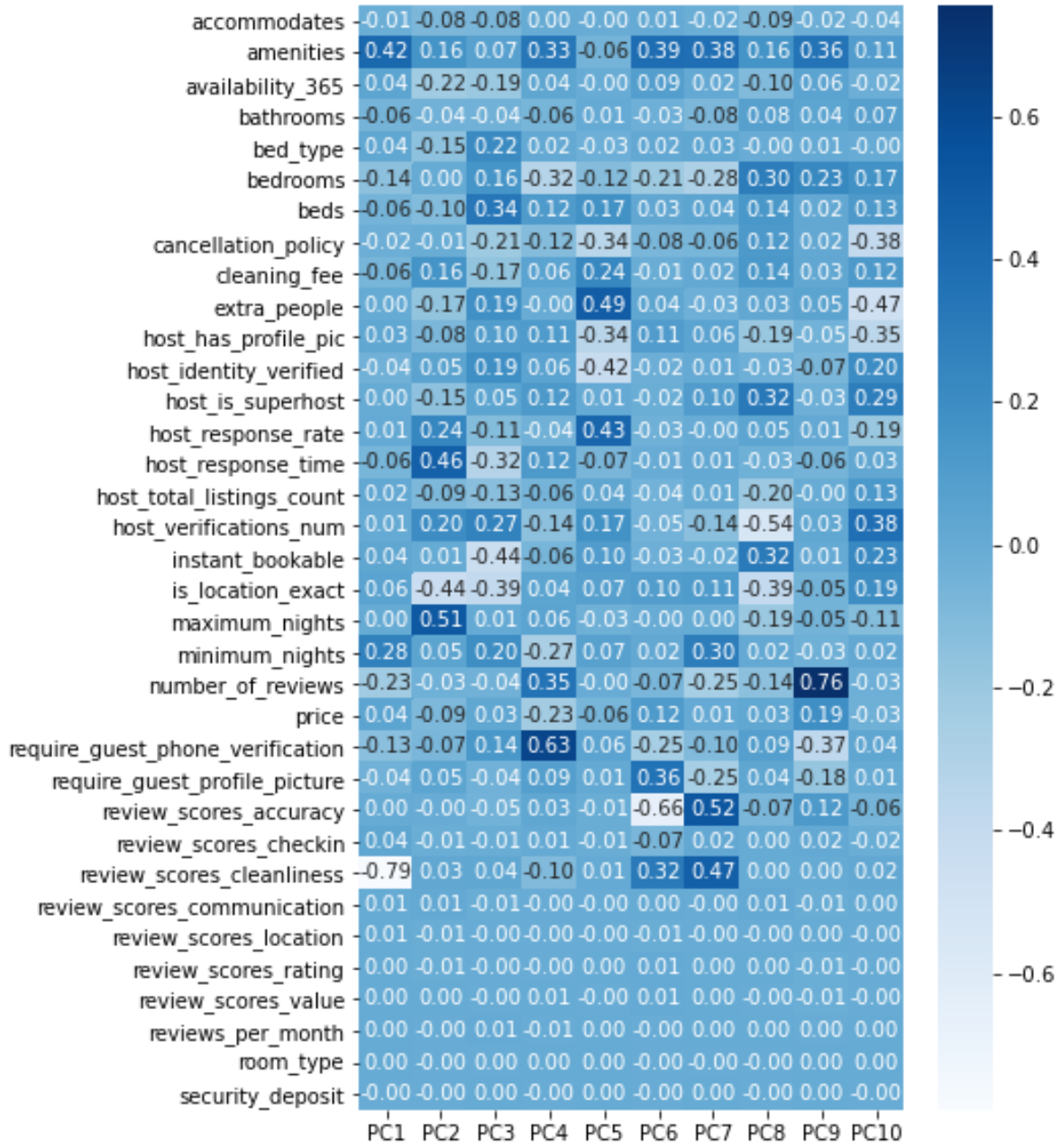


몇 개의 PC 를 도입해야 분산 설명력이 유의미한 수준까지 올라가는지를 판단하기 위하여 누적 분산 설명 비율을 구해보면 다음과 같다.

| | |
|---|---|
| 0 | 0.205372 |
| 1 | 0.315448 |
| 2 | 0.401156 |
| 3 | 0.475249 |
| 4 | 0.537176 |
| 5 | 0.580119 |
| 6 | 0.613700 |
| 7 | 0.645669 |
| 8 | 0.674428 |
| 9 | 0.701952 |
| 10 | 0.729007 |
| 11 | 0.755079 |
| 12 | 0.780009 |
| 13 | 0.803871 |
| 14 | 0.826395 |
| 15 | 0.848127 |
| 16 | 0.868708 |
| 17 | 0.888488 |
| 18 | 0.907794 |
| 19 | 0.925750 |
| 20 | 0.942685 |
| 21 | 0.958712 |
| 22 | 0.974224 |
| 23 | 0.989498 |
| 24 | 1.000000 |

### 3. PC 선택 및 결과 해석

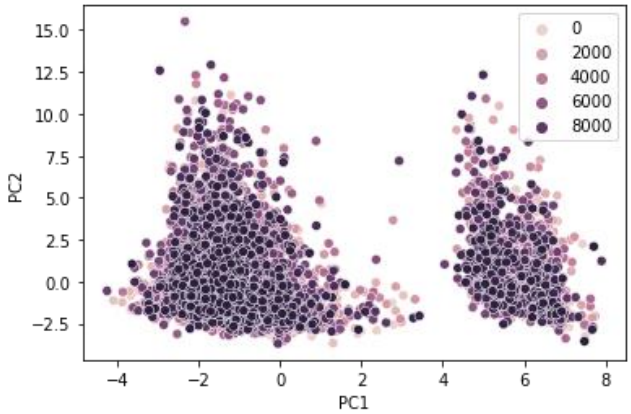
앞에서 구한 eigenvalue 와 분산 설명 비율을 고려했을 때, 우리는 10 개의 PC 를 선택하기로 하였다. 각 PC 의 Eigenvalue 가 1 이상이며 누적 설명 분산 비율 70% 이상인 지점을 기준으로 정하였다. 이에 따른 결과를 시각화해보면 다음과 같다.

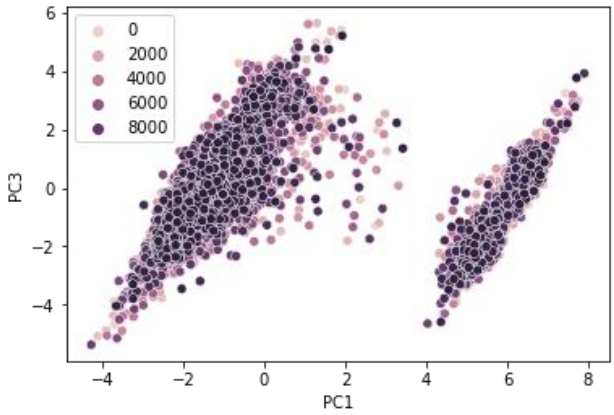


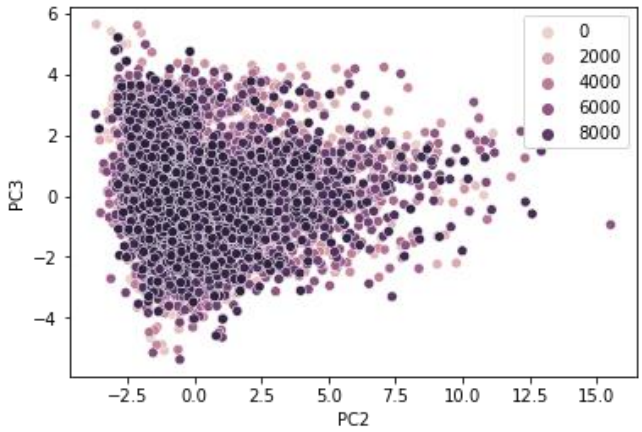

선택한 10 개의 PC 를 PC1~10 으로 명명하였다. Loading score 의 절댓값이 0.5 이상인 (변수, PC) 쌍이 거의 보이지 않으므로 특정 PC 가 어떤 변수들의 영향을 크게 받는 지 알 수 없다. 다시 말해, PCA 를 통해 유용한 결과를 이끌어냈다고 보기 어렵다.

**4. Component pattern plot**

선택된 PC 들 중 2 개를 골라 각각 x, y 축으로 두고 산포도를 그려보면 두 PC 에 대하여 데이터가 어떻게 분포하는지 한눈에 확인할 수 있다. 그러나 10 개의 PC 에 대한 모든 조합을 고려하기는 어려우므로 분산 설명력이 가장 큰 3 개의 PC 를 뽑아 2 개씩 조합하여 산포도를 그려보면 다음과 같다.



그림에서 볼 수 있듯이 두 PC 간 선형 관계는 보이지 않는다. 또한 PC1 이 x 축으로 표현된 첫 번째, 두 번째 산포도를 살펴보면, PC1 은 데이터를 두 개의 군집으로 구분함을 확인할 수 있다. 그러나 PC2, PC3 는 그러한 성질을 보이지 않으며 데이터가 대체로 고르게 분포하는 것으로 보인다.

# Factor Analysis

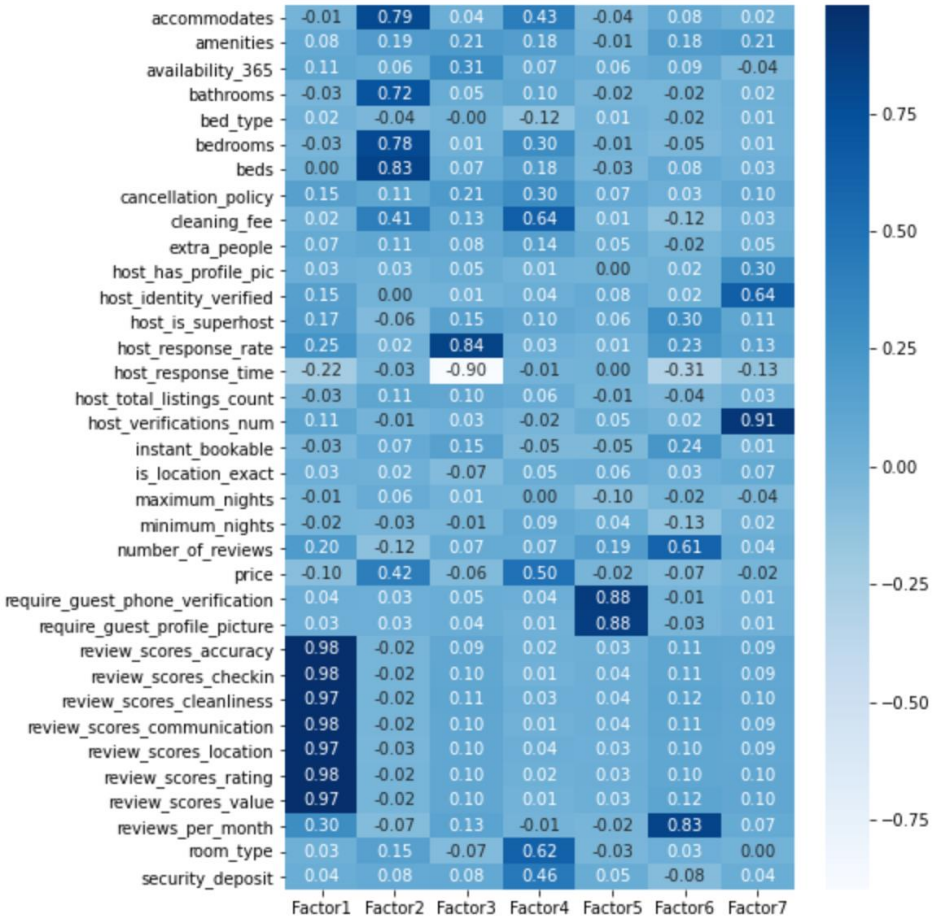
### 1. 적합성 검정

우선 요인 분석에 들어가기 앞서, 우리가 사용하는 데이터 셋이 요인분석 모형에 적합한지 여부를 파악하기 위해 Bartlett Test 와 KMO Test 를 진행하였다. Bartlett Test 결과 p-value = 0.0 으로 상관관계 행렬이 단위행렬이라는 귀무가설을 기각할 수 있었다. KMO Test 결과값 역시 약 0.89 로 굉장히 좋은 값을 가짐을 알 수 있었다.

```
: from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
  chi_square_value, p_value = calculate_bartlett_sphericity(data_scale)
  chi_square_value, p_value # p-value < 0.05 --> 귀무가설 기각

: (329507.223881214, 0.0)

: from factor_analyzer.factor_analyzer import calculate_kmo
  kmo_all, kmo_model =calculate_kmo(data_scale)
  kmo_model # 0.8 이상이므로 꽤 좋음

  C:\Users\CHOI\anaconda3\envs\DM\hw\lib\site-packages\factor_analyzer\utils.py:248: UserWarning: The inverse of the
  variance-covariance matrix was calculated using the Moore-Penrose generalized matrix inversion, due to its determi
  nant being at or very close to zero.
    warnings.warn('The inverse of the variance-covariance matrix '

: 0.8839001947840555
```

### 2. 모델링

PCA 과정과 동일하게 Factor 의 수를 정하는 Threshold 로는 Eigen Value >= 1 로 설정하였다. 그 Factor8 부터 Eigen Value 값이 1 미만으로 떨어지는 것을 확인하였고, 총 7 개의 Factor 를 선택해 FA 모델을 구축하였다. 구축하고 나서는 varimax rotation 을 활용해 변수의 특징을 좀 더 잘 파악할 수 있게 하였다. 이렇게 생성된 모델을 seaborns library 로 시각화한 결과는 다음 그림과 같다. 각 Factor 들마다 feature 을 잘 지니고 있어 추가적인 조작은 하지 않았다.




### 3. 결과 분석

우선 각 Factor 들의 eigenvalue, Proportion Var, Cumulative Var 을 구하면 다음과 같다.

| | Factor1 | Factor2 | Factor3 | Factor4 | Factor5 | Factor6 | Factor7 |
|---|---|---|---|---|---|---|---|
| **SS Loadings** | 7.020209 | 2.937288 | 1.895284 | 1.775511 | 1.642450 | 1.540276 | 1.517662 |
| **Proportion Var** | 0.200577 | 0.083923 | 0.054151 | 0.050729 | 0.046927 | 0.044008 | 0.043362 |
| **Cumulative Var** | 0.200577 | 0.284500 | 0.338651 | 0.389380 | 0.436307 | 0.480315 | 0.523677 |

이때 각 요소들을 다음과 같이 명명할 수 있었다.

- **Factor1: quality of review**

review score 와 관련된 변수들의 계수가 크므로 quality of review 으로 명명하였다.

- **Factor2: quality of room**

주로 방의 상태와 관련된 변수들의 계수가 크므로 quality of room 으로 명명하였다.

- **Factor3: host response**

host response 와 관련된 2 변수의 계수가 크므로 host response 로 명명하였다.

- **Factor4: price**

추가 비용, 선금, 가격 등 돈 관련 변수들의 계수가 크므로 price 로 명명하였다.

- **Factor 5: guest information**

guest information 요구 여부와 관련된 변수들의 계수가 크므로 guest information 으로 명명하였다.

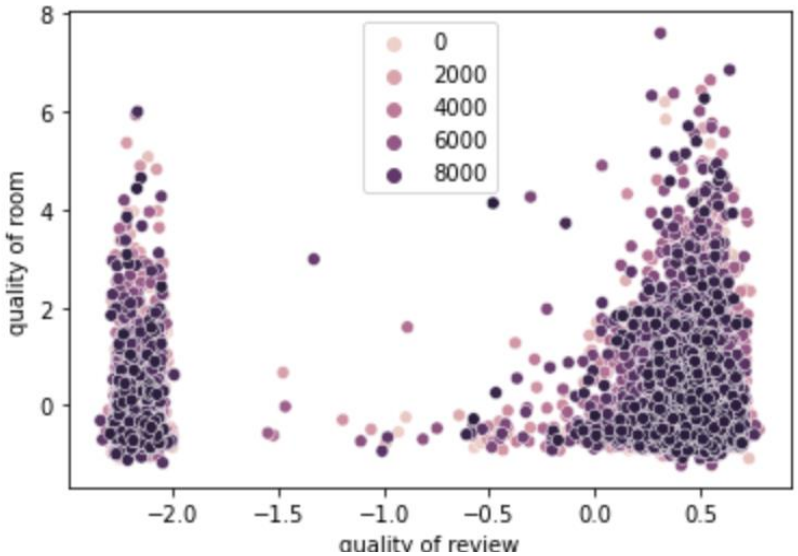
- **Factor 6: quantity of review**

review 수에 관련된 변수들의 계수가 크므로 quantity of review 로 명명하였다.

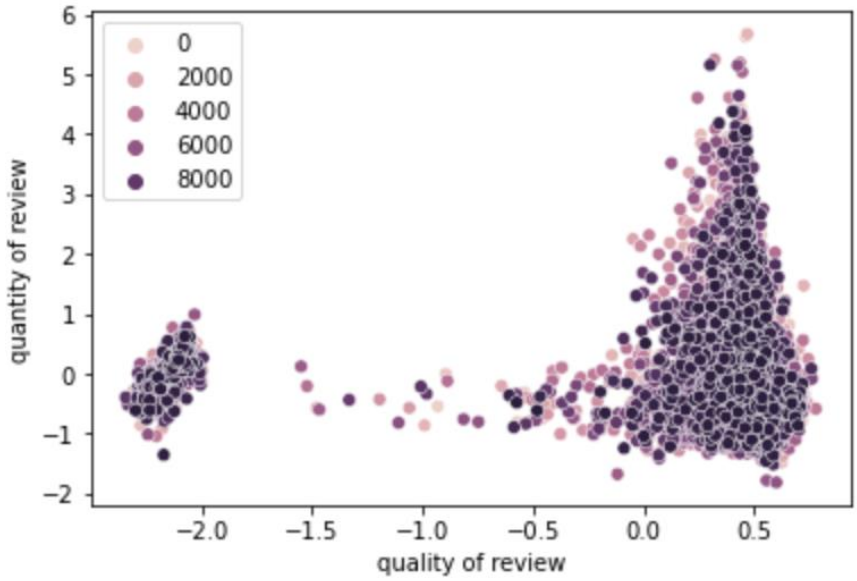
- **Factor 7: host verification**

host verification 과 관련된 변수들의 계수가 크므로 host verification 으로 명명하였다.

그 후, 변수들 간의 상관관계 여부를 조사해보았다. 우선 가장 연관이 있을 법하기도 하고, Eigenvalue 가 가장 크기도 한 두 변수인 'quality of review'와 'quality of room' 간의 scatter plot 을 그려보았다. 그 결과는 아래 그림과 같으며 quality of review 의 양극화가 심하다는 특징 이외에는 이렇다 할 상관관계를 찾지 못하였다.




review 와 관련된 두 변수인 'quality of review'와 'quantity of review'의 scatter plot 을 그려보았다. 아래 그림과 같이 이렇다할 상관관계는 찾지 못했다.

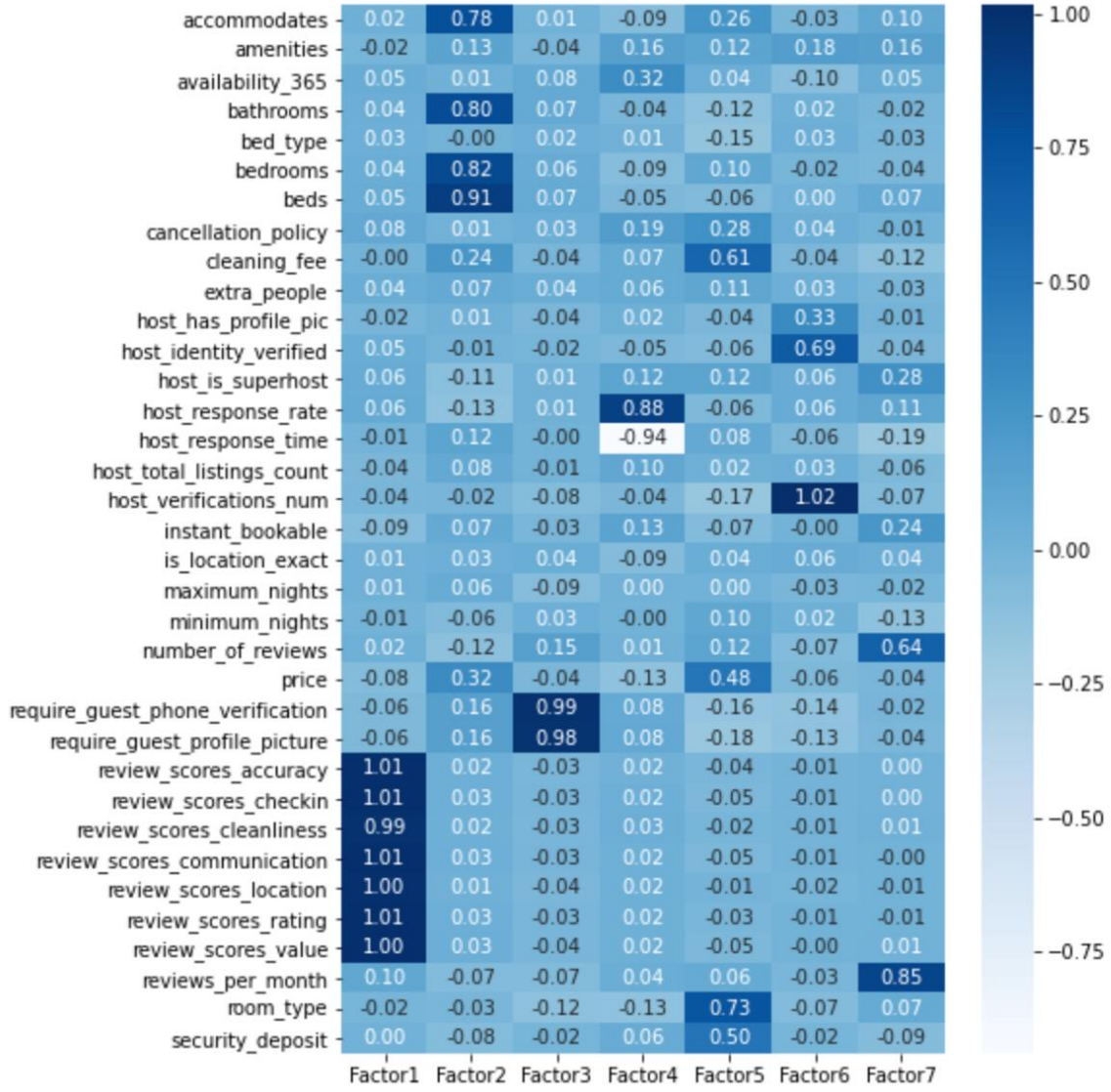



**4. rotation 간의 차이 분석**

해당 실험에는 factor_analyzer package 에서 제공하는 7 가지의 rotation 의 종류 중, varimax, promax, oblimin, oblimax, quartimin, quartimax 6 가지의 rotation 을 사용하였다. 또한 # of Factor 은 위 단락에서 진행했던 것과 같이 7 로 고정 후 진행하였다. (equamax 의 경우, quartimax 와 결과가 동일하게 나와서 제외하였다.)

1) promax

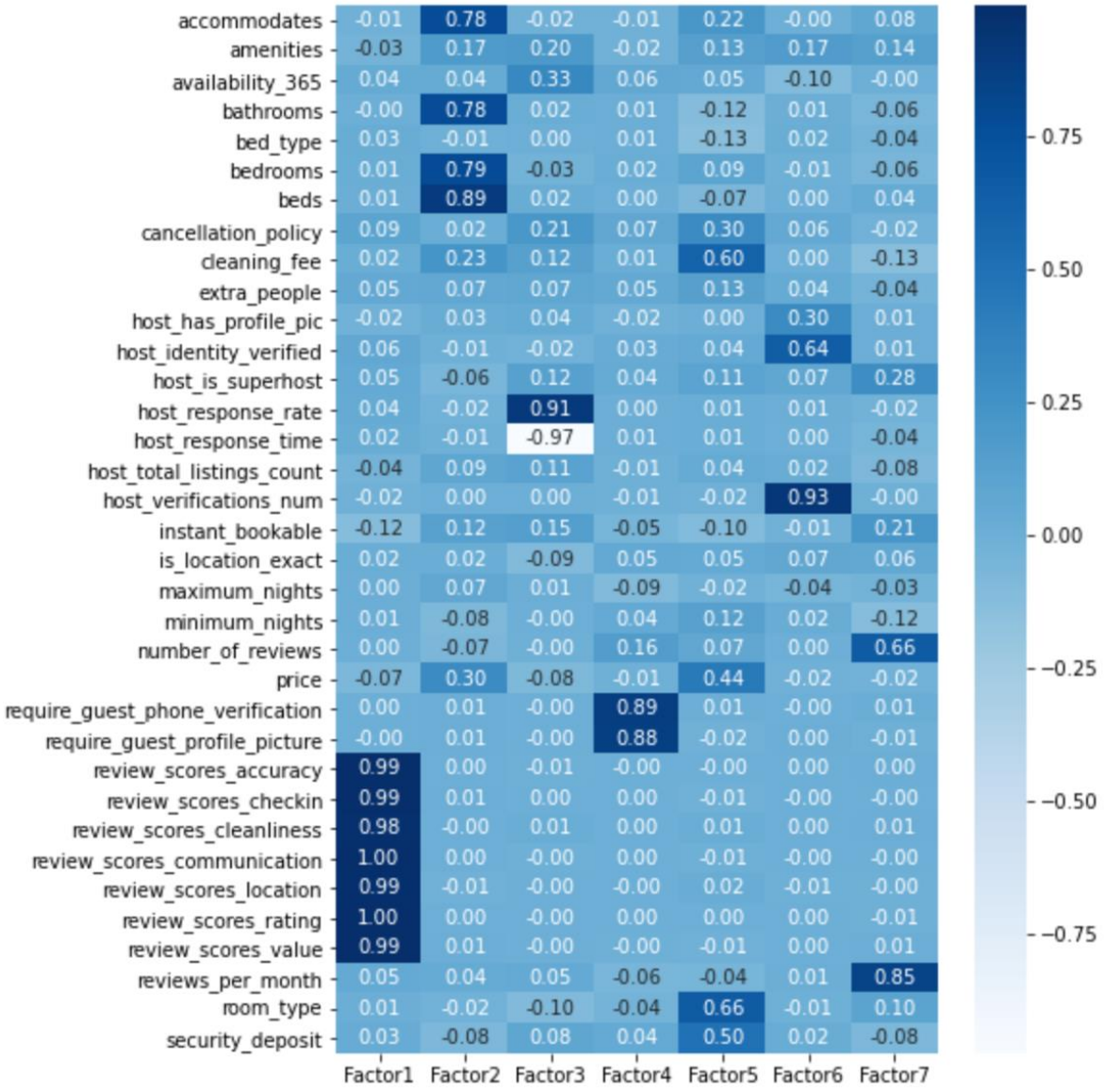
promax rotation 은 사각 회전의 한 종류로 회전에 의한 적재값을 승수로 올려 인자들 사이에 낮은 상관성을 갖도록 하는 방법이고 해당 방법으로 FA 를 실행하였을 때, 계수 히트맵은 다음과 같다.




우선 Factor 들의 순서(eigenvalue)가 Varimax 와는 상이함을 확인할 수 있었고, 각 Factor 들의 feature 가 될 수 있는 변수들의 계수 값이 좀 더 양극화 되어 있음을 알 수 있었다.

2) oblimin

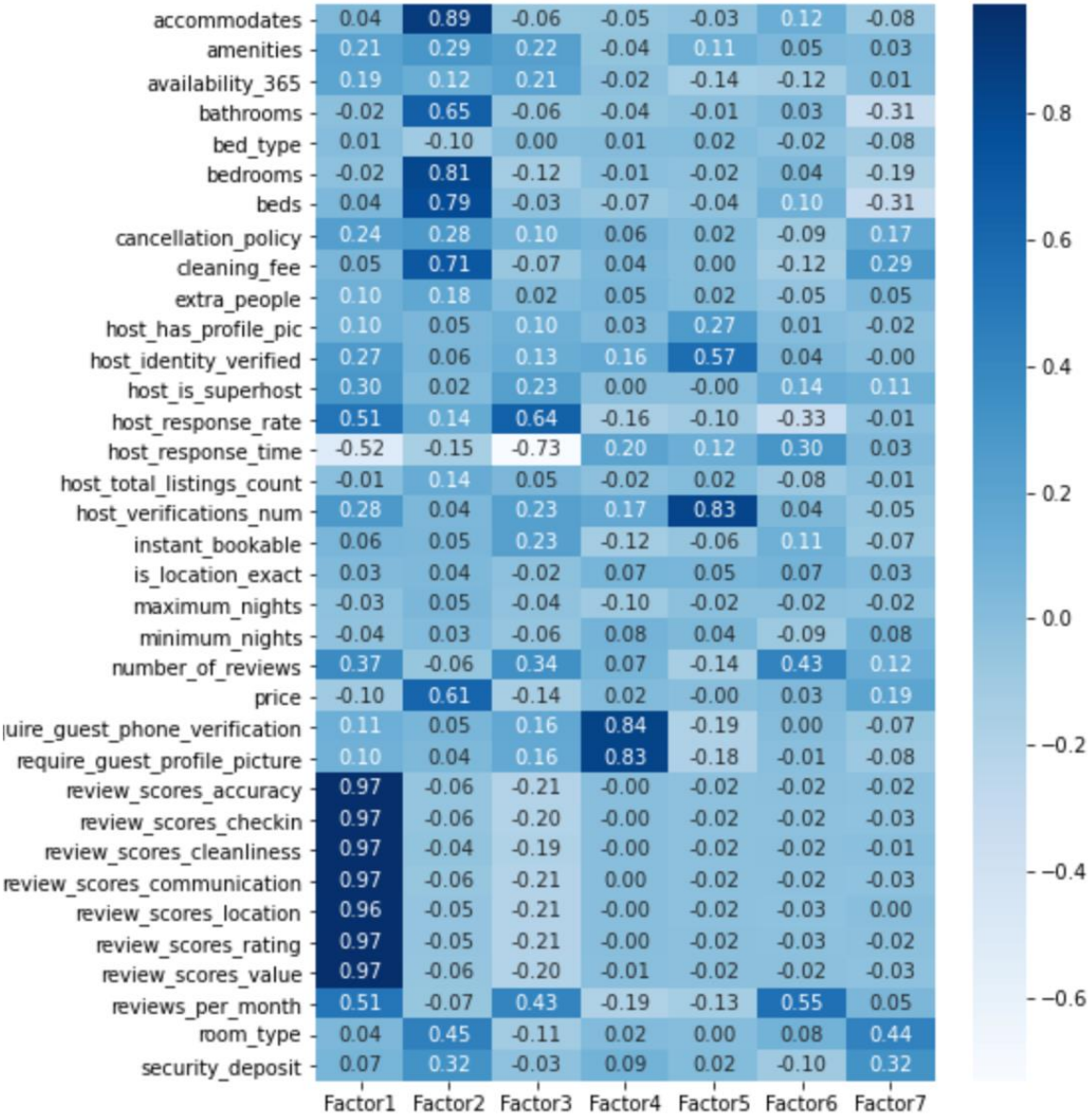
oblimin rotation 은 사각회전의 한 종류로 직교가 아닐지라도 인자부하값들이 0 또는 1 에 가깝도로 회전하게하는 방법이고 해당 방법으로 FA 를 실행하였을 때, 계수 히트맵은 다음과 같다.




oblimin rotation 의 정의처럼 변수들의 계수값이 varimax rotation 보다 0 과 1 에 많이 치중되어 있음을 확인할 수 있었다. 마찬가지로 해당 rotation 의 경우에도 명명했던 factor 들 간의 eigenvalue 순서가 차이가 있음을 확인할 수 있었다.

3) oblimax

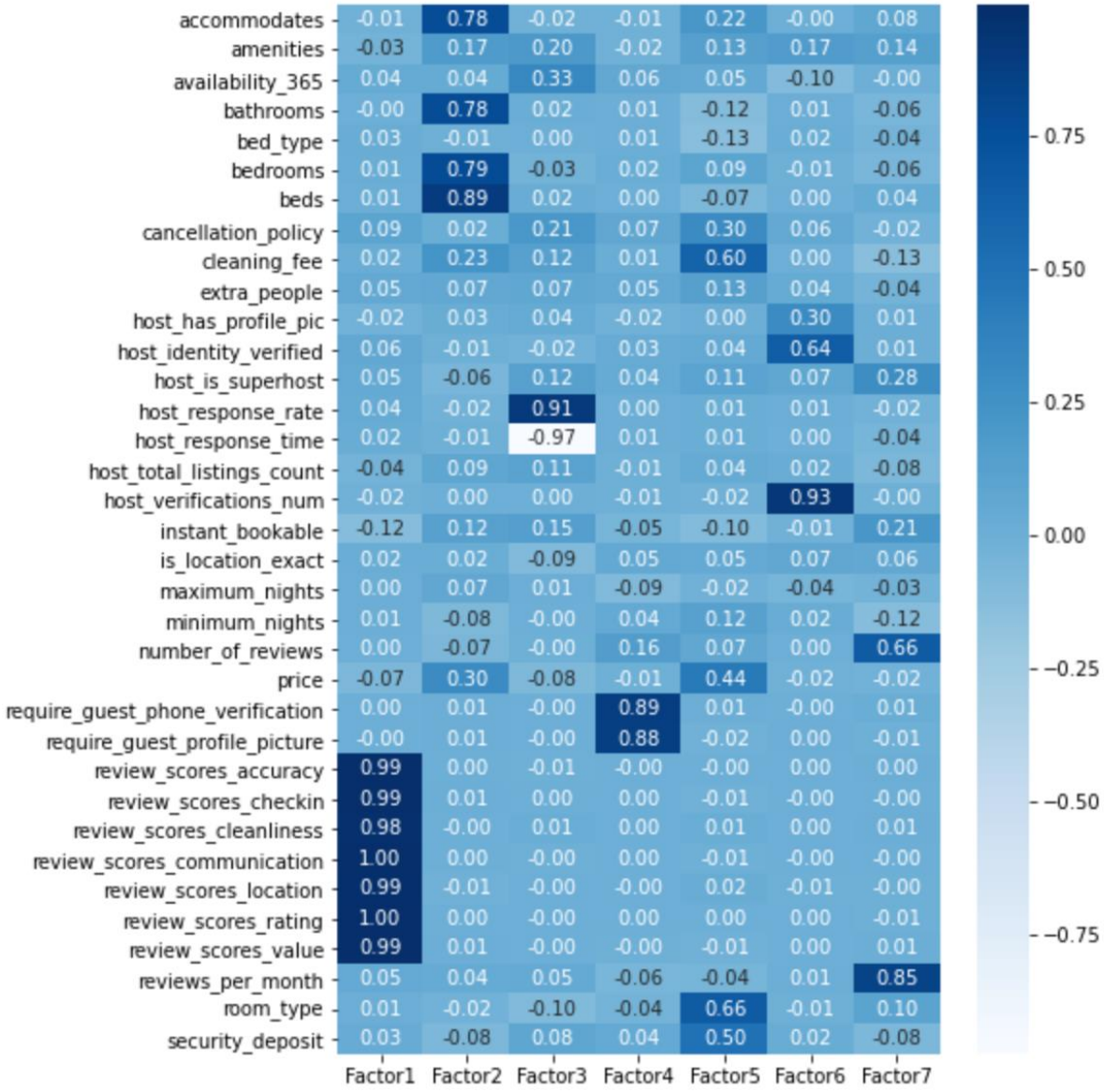
oblimax rotation 은 직교회전의 한 종류로 분석자가 좀 더 의미있는 값을 얻으려할 때 사용하는 방법이다. 해당 방법으로 FA 를 실행하였을 때, 계수 히트맵은 다음과 같다.




Factor6 와 Factor7 을 볼 때, 다른 rotation 들과 비교했을 때 상대적으로 feature 추출 성능이 떨어지는 회전으로 확인된다.

4) quartimin

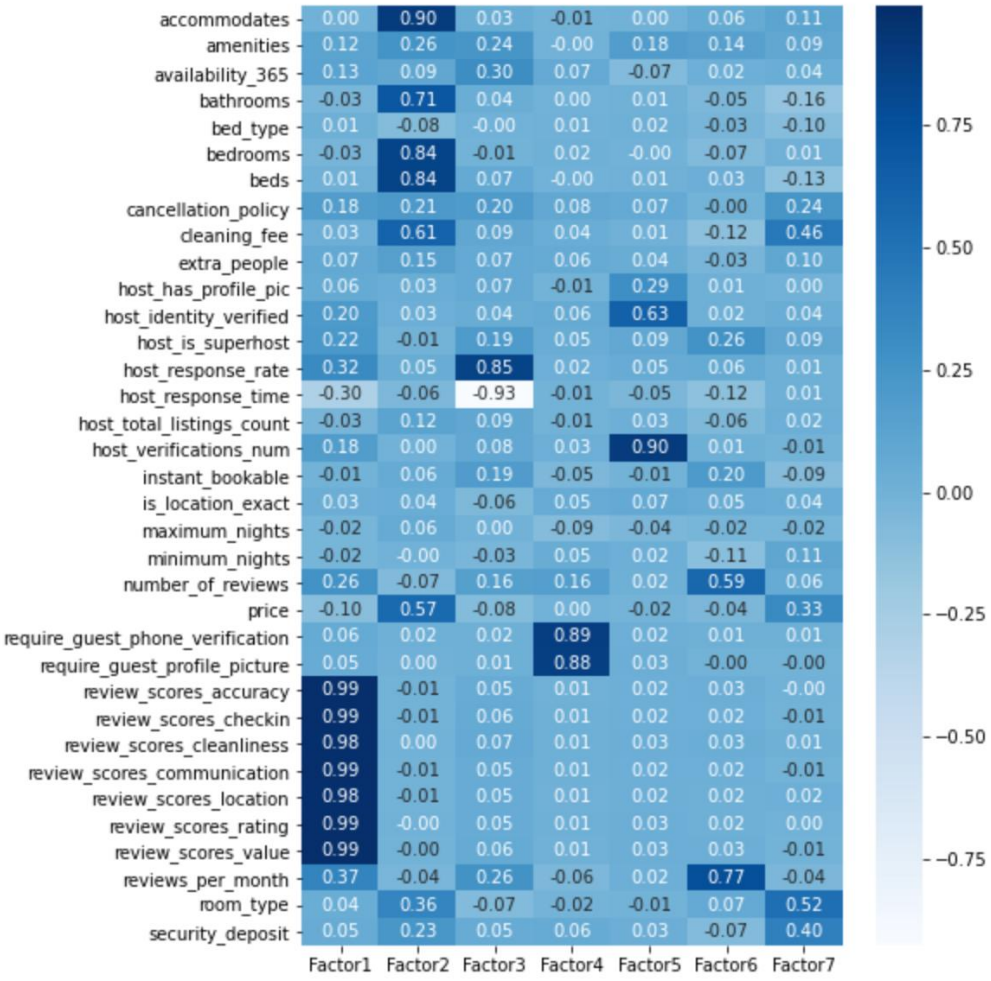
quartimin rotation 은 사각회전의 한 종류로 orthomax family 매개변수를 0 으로 setting 후 oblimin object 를 최소화하는 방법이다. 해당 방법으로 FA 를 실행하였을 때, 계수 히트맵은 다음과 같다.




정의대로 각 Factor 의 순서는 oblimin 과 동일한 형태를 띄고 있다. 하지만 각 Feature 로 볼 수 있는 변수들의 계수값은 상대적으로 덜 추출됨을 확인할 수 있다.

5) quartimax

quartimax rotation 은 직교회전의 한 종류로 행렬의 행을 기준으로 분산을 극대화하는 방법이다. 해당 방법으로 FA 를 실행하였을 때, 계수 히트맵은 다음과 같다.




같은 직교회전인 varimax 와 비교했을 때, 근소하게 feature 들이 좀 더 극화되어 있음을 알 수 있으나 그 차이가 유의미해보이는 않는다. 오히려 더 적은 Factor 들도 존재하고 있다.

## 6. 결과 분석

우선 회전 방식에 따라 우리가 명명했던 Factor 들의 eigenvalue 값이 매번 달라짐을 확인할 수 있었다. 하지만 eigenvalue 값을 조사했을 때, 상대적으로 값이 월등히 컸던 Factor 1 과 2 의 경우는 예상했던 대로 고정되어 있었고, 타 Factor 들만 변동이 있음을 확인할 수 있었다. 또한 상대적으로 사각회전 방식들이 직교회전 방식들보다 Factor 의 feature 변수들의 계수가 양극화되어 있음을 알 수 있었다. 특히 이러한 점은 Factor 2 에서 room_type 과 security_deposit 변수에서 두드러지게 확인할 수 있었는데, 사각회전들의 경우 0 에 근사한 값들을 보였지만 직교회전들의 경우에는 0.3 정도의 어느정도 유의미하다고 볼 수 있는 정도까지의 계수를 보유하고 있었다.