제주도 호텔 가격 회귀 분석

4조 이진서, 김형기

01 프로젝트 목적 및 동기

02 데이터 수집

crawling, API 이용

03 데이터 전처리 (EDA)

성급, 리뷰, 평점 가격 분석

04 회귀 분석

분석 및 모델 평가

05 결론 및 한계점

예측 값 확인 및 한계점

01 프로젝트 목적 및 동기

02 데이터 수집

1. 인터파크 Crawling
2. Booking.com API

인터파크 Crawling

```python
# 함수로 만들기
from selenium import webdriver
from time import sleep
from datetime import datetime
import time

datas = []

def get_hotels(cin):
    url = 'https://travel.interpark.com/checkinnow/search?\
q=%EC%A0%9C%EC%A3%BC,%20%EC%A0%84%EC%B2%B4&disp_q=%EC%A0%9C%EC%A3%BC,\
%20%EC%A0%84%EC%B2%B4&coord=&regiondisp=42090&startdate={}&enddate={}&\
roomOptions=0%5E2%5E0%5E&category1=%EC%88%99%EB%B0%95&type=&value=&index=1\
&npprmaxsort=min'.format(cin, cin+1)
    driver = webdriver.Chrome()
    driver.get(url)

    # 숙소유형 호텔 선택
    driver.find_element_by_xpath('//*[@id="resultContainer"]/div[1]/div/div/dl[2]/dd[1]/label[2]').click()
    time.sleep(3)

    # 1, 2, 3, 4, 5 등급 선택
    driver.find_element_by_xpath('//*[@id="resultContainer"]/div[1]/div/div/dl[3]/dd/label[1]').click()
    driver.find_element_by_xpath('//*[@id="resultContainer"]/div[1]/div/div/dl[3]/dd/label[2]').click()
    driver.find_element_by_xpath('//*[@id="resultContainer"]/div[1]/div/div/dl[3]/dd/label[3]').click()
    driver.find_element_by_xpath('//*[@id="resultContainer"]/div[1]/div/div/dl[3]/dd/label[4]').click()
    driver.find_element_by_xpath('//*[@id="resultContainer"]/div[1]/div/div/dl[3]/dd/label[5]').click()
    time.sleep(3)

    # 총 페이지 갯수
    total_page = (int(driver.find_element_by_css_selector("#sortMenuCount").text)//30)+1

    for page in range(1, total_page+1):
        try:
            print(page)
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") # 스크롤 내리고 로딩 기다리기
            time.sleep(3)

            hlist = driver.find_elements_by_xpath("//div[contains(@class, 'col data')]")

            for i in range(len(hlist)):
                name = hlist[i].find_element_by_css_selector("div:nth-child(1) > div.titArea > a > strong").tex
                price = hlist[i].find_element_by_css_selector("div:nth-child(1) > div.priceArea > strong").text
                star = hlist[i].find_element_by_css_selector('div:nth-child(2) > div > span.inline > span:nth-c
                r_count= hlist[i].find_element_by_css_selector('div:nth-child(3) > div.ratingArea > span.review
                addr = hlist[i].find_element_by_css_selector('div:nth-child(2) > div > span:nth-child(1)').text
                try:
                    r_score = hlist[i].find_element_by_css_selector('div:nth-child(3) > div.ratingArea > span.s
                except:
                    r_score = 0

                datas.append({"cin":cin, "name":name, "price":price, "star":star,
                              "r_count":r_count, "r_score":r_score, "addr":addr})

            page_selector= ".paging > span > a:nth-child({})".format(str(page+1))
            driver.find_element_by_css_selector(page_selector).click() # 다음페이지 클릭
            time.sleep(3)
        except:
            break


    print("crawling : {}".format(cin))
    driver.quit()
```

- *Check_in_date*
- *Name*
- *Price*
- *Star*
- *Rating*
- *Reviews*

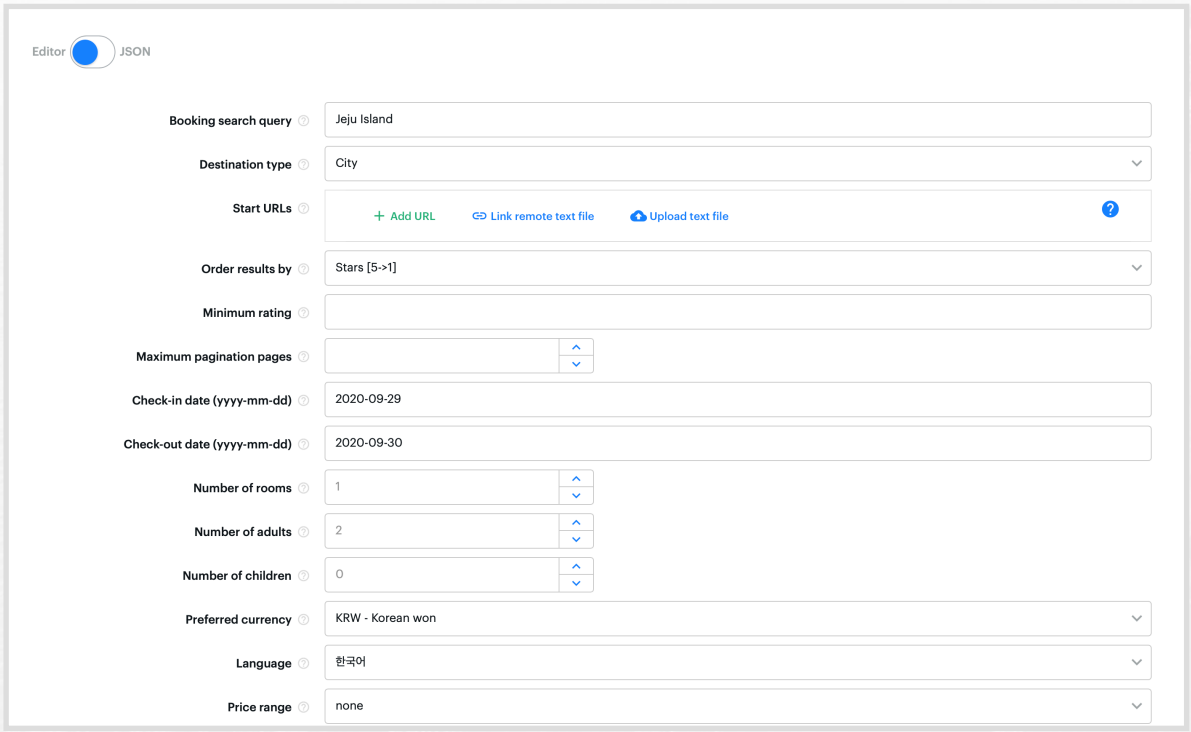
booking.com Crawling

Booking Scraper

dtrungtin/booking-scraper

Extract data from Booking.com where there is no official API. With one click, you can scrape all information about accommodation listings, such as price, rating, reviews, stars, etc. It's easy to use and input is similar to the website.

▶ Try for free



- *Check_in_date*
- *Name*
- *Price*
- *Features*
- *Rating*
- *Reviews*

03 데이터 전처리

1. 인터파크 EDA

2. Booking.com EDA

인터파크 EDA

```python
# 경도 위도 naver API
import numpy as np
import pandas as pd
from urllib.request import urlopen
from urllib import parse
from urllib.request import Request
from urllib.error import HTTPError
from bs4 import BeautifulSoup
import json

# naver api
client_id = '0d9uya4hfz';    # 할당받은 ID
client_pw = 'zB1NWFrXwwySehisDj6ey0lu2EJ33HGtPagEwCaU';    # 할당받은 Secret

api_url = 'https://naveropenapi.apigw.ntruss.com/map-geocode/v2/geocode?query='

# 주소 목록 파일
data = hotels_names
data

# 네이버 지도 API 이용해서 위경도 찾기
geo_coordi = []
for addr in hotels_names['addr']:
    addr_urlenc = parse.quote(addr)
    url = api_url + addr_urlenc
    request = Request(url)
    request.add_header('X-NCP-APIGW-API-KEY-ID', client_id)
    request.add_header('X-NCP-APIGW-API-KEY', client_pw)
    try:
        response = urlopen(request)
    except HTTPError as e:
        print('HTTP Error!')
        latitude = None
        longitude = None
    else:
        rescode = response.getcode()
        if rescode == 200:
            response_body = response.read().decode('utf-8')
            response_body = json.loads(response_body)   # json
            if response_body['addresses'] == [] :
                print("'result' not exist!")
                latitude = None
                longitude = None
            else:
                latitude = response_body['addresses'][0]['y']
                longitude = response_body['addresses'][0]['x']
                print("Success!")
        else:
            print('Response error code : %d' % rescode)
            latitude = None
            longitude = None

    geo_coordi.append([latitude, longitude])


np_geo_coordi = np.array(geo_coordi)
pd_geo_coordi = pd.DataFrame({"name": hotels_names['name'].values,
                              "addr": hotels_names['addr'].values,
                              "latitude": np_geo_coordi[:, 0],
                              "longitude": np_geo_coordi[:, 1]})

pd_geo_coordi.to_csv('/Users/jinseo/Documents/dss/1_programming/Project/2. regress
```

In [119]:
```python
# 제주공항과 거리는? Haversine
!pip install haversine
```

```
Collecting haversine
  Downloading haversine-2.2.0-py2.py3-none-any.whl (4.9 kB)
Installing collected packages: haversine
Successfully installed haversine-2.2.0
```

In [4]:
```python
# 제주공항과 거리는?
from haversine import haversine
```

In [5]:
```python
len(lnglat_df)
```

Out[5]: 75

In [6]:
```python
distance_list = []

data1 = (33.511005, 126.491318) # 제주공항

for i in range(len(lnglat_df)):
    data2 = (lnglat_df["latitude"][i], lnglat_df["longitude"][i]) # 각 호텔들의 위경도
    distance = haversine(data1, data2, unit = 'km') # 제주공항과 호텔의 거리
    distance_list.append(distance)

lnglat_df["distance"] = distance_list
lnglat_df.drop(columns=["name"], axis=1, inplace=True)
lnglat_df
```

Out[6]:

	addr	latitude	longitude	distance
0	제주 서귀포시 안덕면 서광리 산 24번지	33.307187	126.317436	27.823235
1	제주특별자치도 제주시 조천읍 함덕리 1269-9 유탑유블레스호텔	33.542411	126.666212	16.583734
2	제주 제주시 건입동 1443	33.517411	126.526897	3.374514
3	제주 서귀포시 색달동 2812-4	33.248275	126.410407	30.164860
4	제주 제주시 이도이동 1025-8	33.493516	126.534292	4.433829
...	...	...	...	...
70	제주 서귀포시 표선면 표선리 40-69	33.323364	126.844747	38.876091
71	제주 제주시 외도일동 401-8	33.492431	126.428735	6.159437
72	제주 서귀포시 회수동 30	33.286153	126.444396	25.379001
73	제주 서귀포시 색달동 2138	33.259434	126.406306	29.065655
74	제주 서귀포시 색달동 3039-3	33.247167	126.407950	30.341594

75 rows × 4 columns

인터파크 EDA

```
In [8]:  1 hotels_df
```

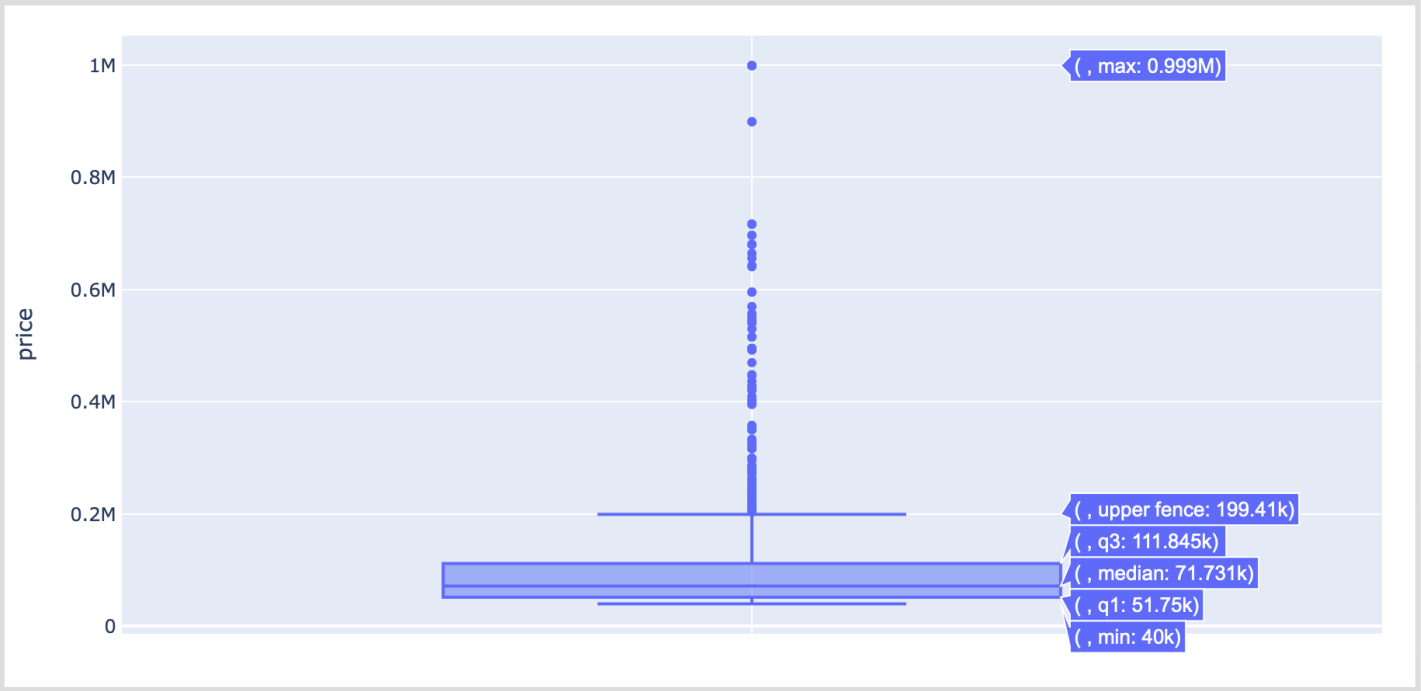
Out[8]:

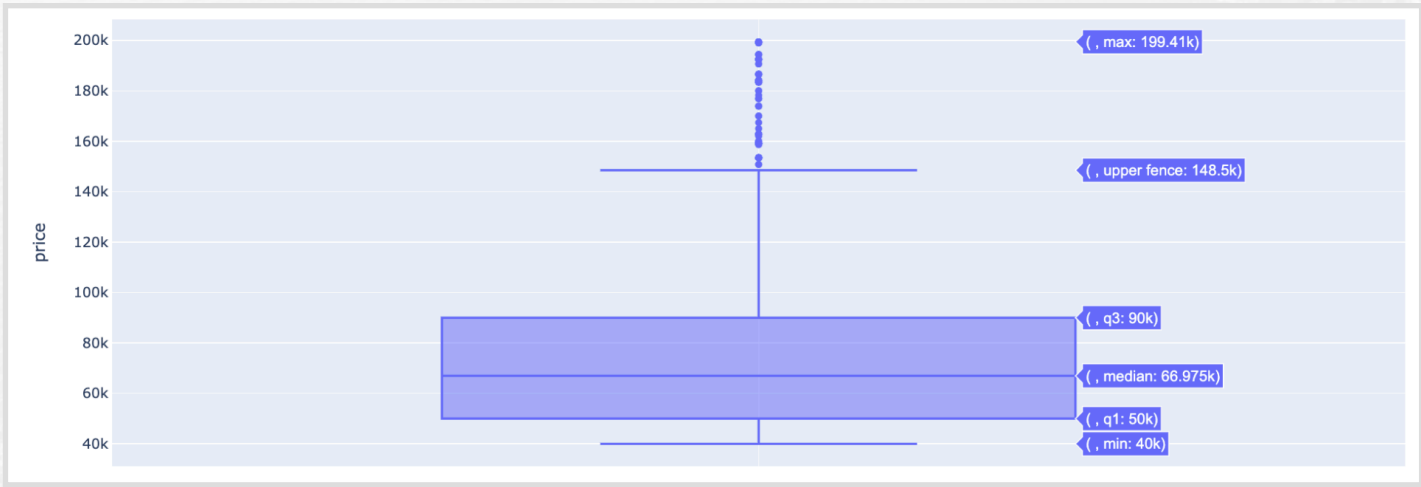
	cin	name	price	star	r_count	r_score	addr	latitude	longitude	distance
0	20200901	메리어트관 제주신화월드 호텔앤리조트	353760	5	224	9.0	제주 서귀포시 안덕면 서광리 산 24번지	33.307187	126.317436	27.823235
1	20200902	메리어트관 제주신화월드 호텔앤리조트	353760	5	224	9.0	제주 서귀포시 안덕면 서광리 산 24번지	33.307187	126.317436	27.823235
2	20200903	메리어트관 제주신화월드 호텔앤리조트	353760	5	224	9.0	제주 서귀포시 안덕면 서광리 산 24번지	33.307187	126.317436	27.823235
3	20200904	메리어트관 제주신화월드 호텔앤리조트	353760	5	224	9.0	제주 서귀포시 안덕면 서광리 산 24번지	33.307187	126.317436	27.823235
4	20200905	메리어트관 제주신화월드 호텔앤리조트	316910	5	224	9.0	제주 서귀포시 안덕면 서광리 산 24번지	33.307187	126.317436	27.823235
...	...	...	...	...	...	...	...	...	...	...
1917	20200925	제주신라호텔	595500	5	4863	9.8	제주 서귀포시 색달동 3039-3	33.247167	126.407950	30.341594
1918	20200926	제주신라호텔	544500	5	4863	9.8	제주 서귀포시 색달동 3039-3	33.247167	126.407950	30.341594
1919	20200927	제주신라호텔	656000	5	4863	9.8	제주 서귀포시 색달동 3039-3	33.247167	126.407950	30.341594
1920	20200928	제주신라호텔	680200	5	4863	9.8	제주 서귀포시 색달동 3039-3	33.247167	126.407950	30.341594
1921	20200929	제주신라호텔	680200	5	4863	9.8	제주 서귀포시 색달동 3039-3	33.247167	126.407950	30.341594

1922 rows × 10 columns

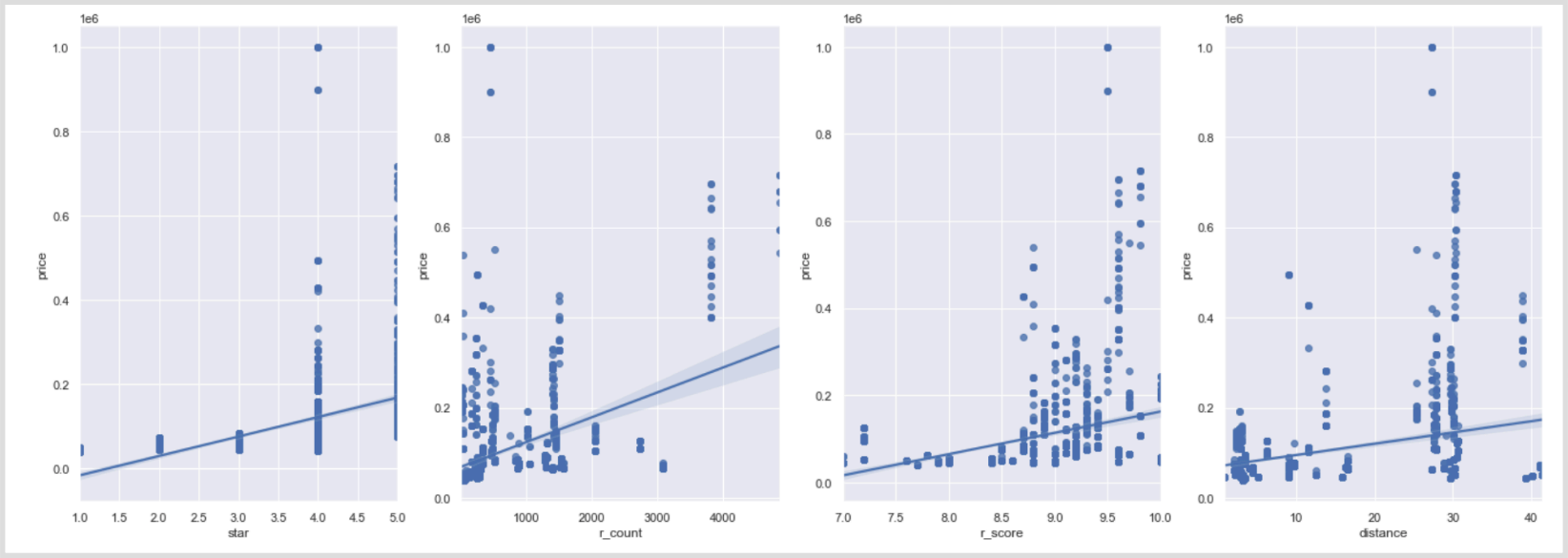
- *Check_in_date(cin)*
- *Name*
- *Price*
- *Star*
- *Rating (r_score)*
- *Reviews (r_count)*
- *Distance*

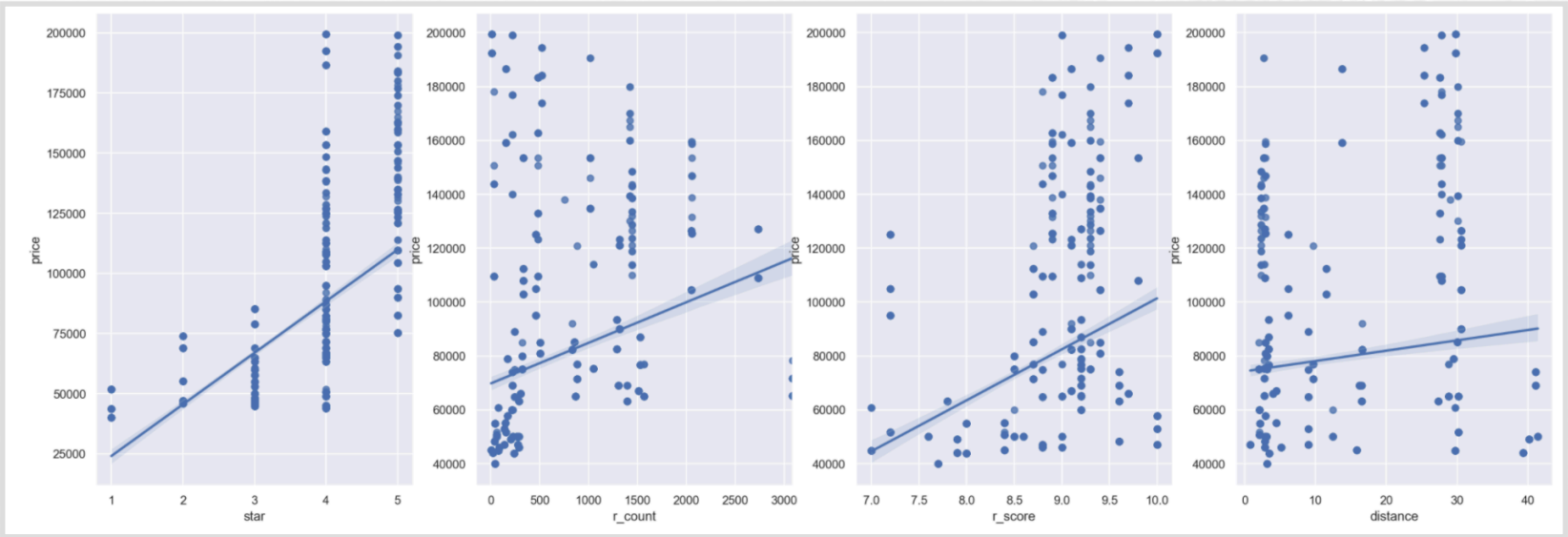
인터파크 EDA


price
1M
0.8M
0.6M
0.4M
0.2M
0
(, max: 0.999M)
(, upper fence: 199.41k)
(, q3: 111.845k)
(, median: 71.731k)
(, q1: 51.75k)
(, min: 40k)


price
200k
180k
160k
140k
120k
100k
80k
60k
40k
(, max: 199.41k)
(, upper fence: 148.5k)
(, q3: 90k)
(, median: 66.975k)
(, q1: 50k)
(, min: 40k)

인터파크 EDA


1e6
price
star
r_count
r_score
distance


price
star
r_count
r_score
distance

Booking.com EDA

In [56]:
```
df = pd.read_csv('/Users/stevenkim/Downloads/booking.com 원본/원본/dataset_CHI_2020_09_01.csv')
df
```

270	대한민국	제주, 281, Hamwa-ro, Jocheon-eup	63334	제주도	281, Hamwa-ro, Jocheon-eup	NaN	NaN	NaN	제주 베스트셀러! 제주에 위치한 함덕 예쁜 집게스트하우스는 무료 Wi-Fi, 에어컨 ...	https://cf.bstatic.com/ir
271	대한민국	제주, 281, Hamwa-ro, Jocheon-eup	63334	제주도	281, Hamwa-ro, Jocheon-eup	NaN	NaN	NaN	제주 베스트셀러! 제주에 위치한 함덕 예쁜 집게스트하우스는 무료 Wi-Fi, 에어컨 ...	https://cf.bstatic.com/ir
272	대한민국	제주, 남성로 131	63172	제주도	남성로 131	유럽식	15:00	20:00	참피온 호텔은 제주 국제공항에서 동쪽으로 2.5km 떨어져 있습니다. 인기 관광지인...	https://cf.bstatic.com/ir

273 rows × 2967 columns

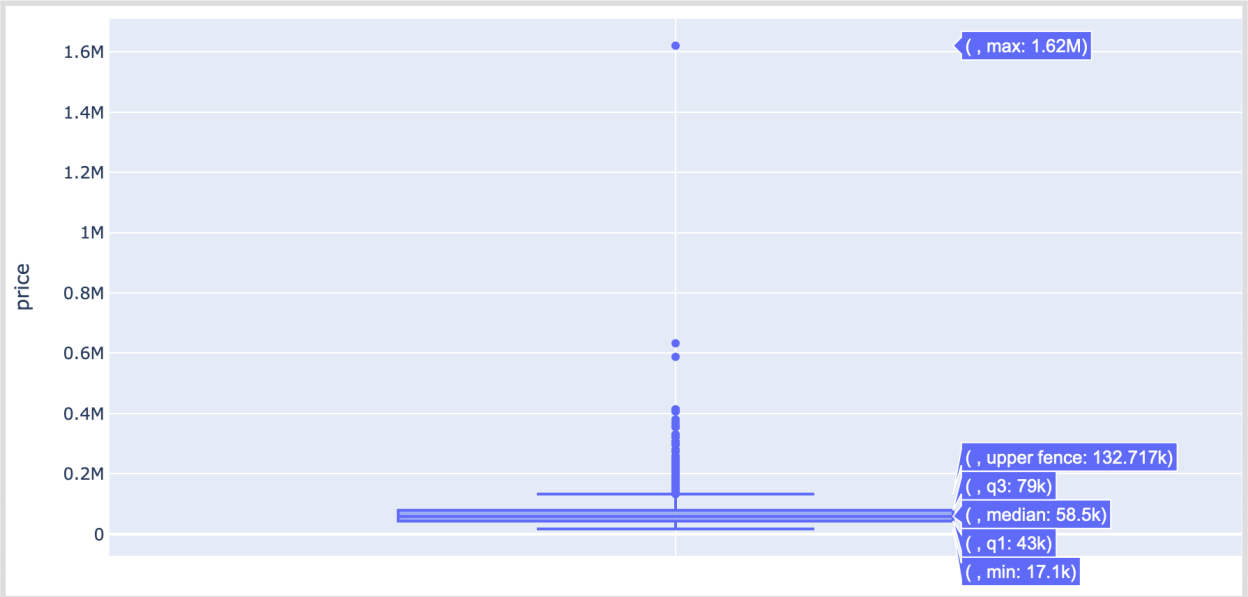
In [3]:
```
hotels_df = pd.concat([df02, df03, df04, df05, df06, df07, df08, df09, df10, df11, df12, df13, df14, df15, df16,
hotels_df = hotels_df[['name', 'price', 'Area', 'rating', 'reviews', 'distance', 'day', 'weekend']]
hotels_df
```

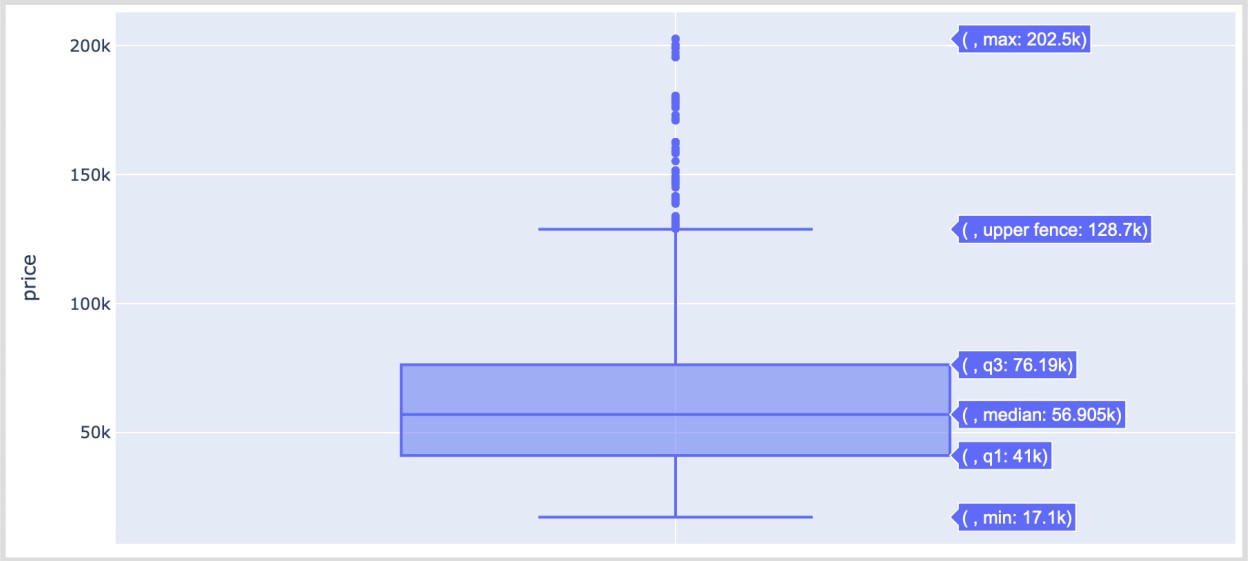
Out[3]:

	name	price	Area	rating	reviews	distance	day	weekend
0	제주오리엔탈호텔 & 카지노	67725	24	7.7	587	2.738362	20200902	0
1	다인 오세아노 호텔	355555	101	8.1	541	12.129263	20200902	0
2	블랙 샌즈 호텔 제주	49600	25	7.7	120	8.986851	20200902	0
3	제주 베스트 힐	79000	59	9.0	20	16.686766	20200902	0
4	제주 하바나 리조트	59000	36	7.7	230	35.060464	20200902	0
...	...	...	...	...	...	...	...	...
4899	제주 삼다 호스텔	46540	45	7.6	9	0.830589	20200928	0
4900	예스준 게스트하우스	25000	20	8.7	12	1.922235	20200928	0
4901	미르 게스트하우스	58500	23	8.5	305	1.809006	20200928	0
4902	예하 게스트하우스	60000	23	8.9	545	2.748919	20200928	0
4903	호텔 메르블루	130784	27	8.7	8	0.657127	20200928	0

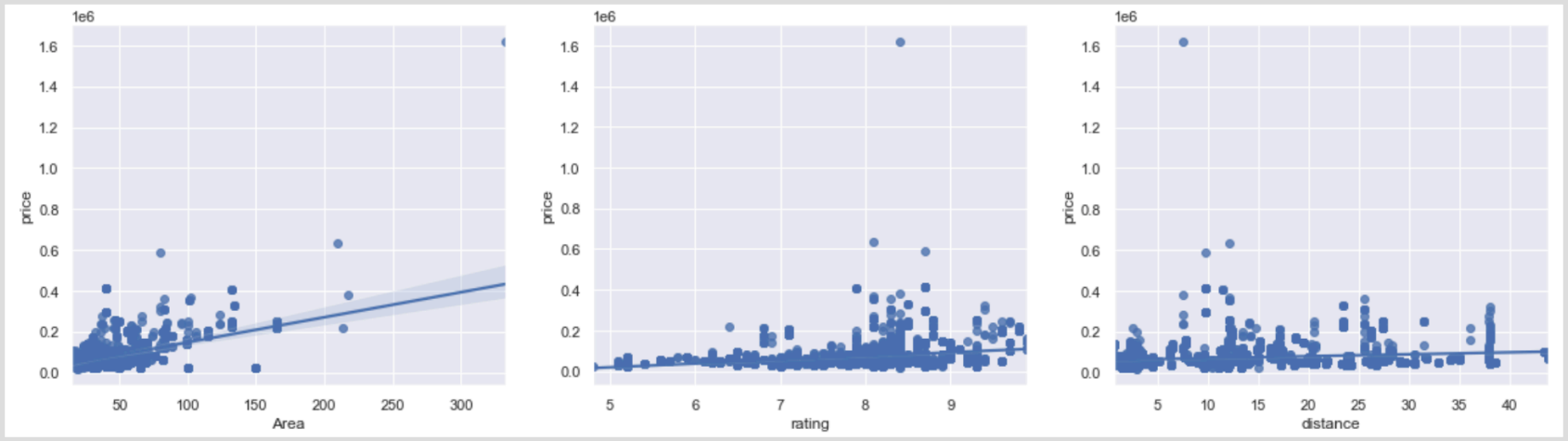
4904 rows × 8 columns

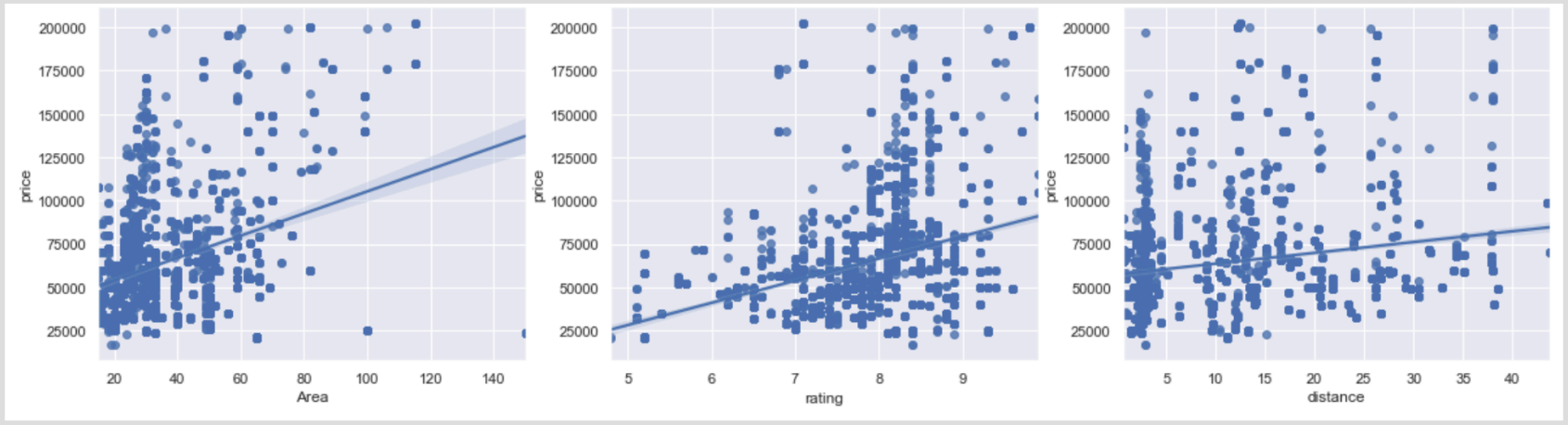
Booking.com EDA





Booking.com EDA



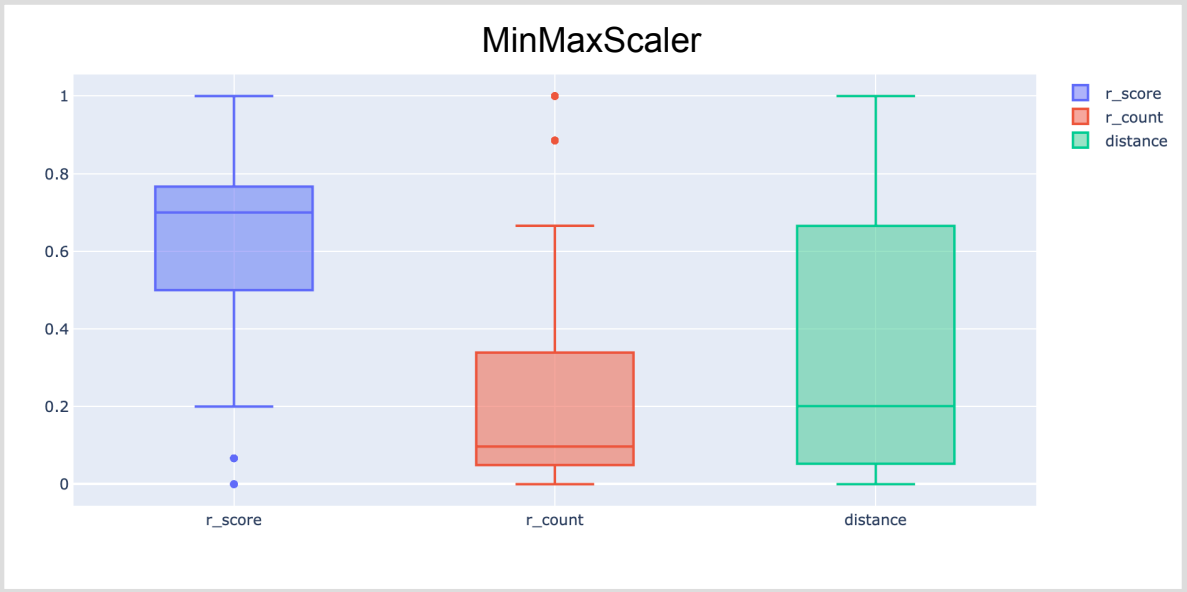


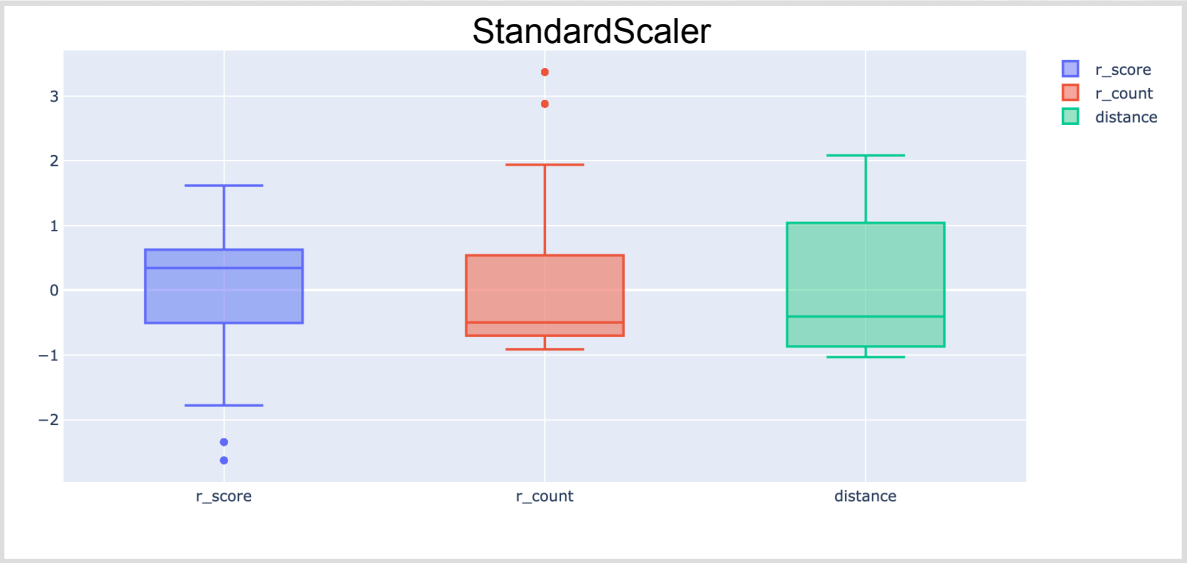
04 회귀 분석

1. 인터파크 회귀 분석

2. Booking.com 회귀 분석

인터파크 회귀 분석


MinMaxScaler
1
0.8
0.6
0.4
0.2
0
r_score
r_count
distance
r_score
r_count
distance


StandardScaler
3
2
1
0
−1
−2
r_score
r_count
distance
r_score
r_count
distance

인터파크 회귀 분석

In [69]:
```
X = hotels_df_copy[["cin", "star", "r_count", "r_score", "distance"]]
y = hotels_df_copy["price"]
```

In [70]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)
```

In [71]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lr = LinearRegression()
lr.fit(X_train, y_train)

pred_tr = lr.predict(X_train)
pred_test = lr.predict(X_test)

rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

print("RMSE of Train Data : ", rmse_tr)
print("RMSE of Test Data : ", rmse_test)

print("r2 score of Train Data : ", r2_score(y_train, pred_tr))
print("r2 score of Test Data : ", r2_score(y_test, pred_test))
```

```
RMSE of Train Data :  28579.599102446613
RMSE of Test Data :  28703.443898098023
r2 score of Train Data :  0.4069110283918662
r2 score of Test Data :  0.39393124632773013
```

인터파크 회귀 분석

In [75]:
```
models = []
models.append(('LinearReg', LinearRegression()))
models.append(('DecisionTree', DecisionTreeRegressor()))
models.append(('RandomForest', RandomForestRegressor()))
models.append(('GradientB', GradientBoostingRegressor()))
models.append(('XGradianB', XGBRegressor()))
models.append(('LightGBM', LGBMRegressor()))
```

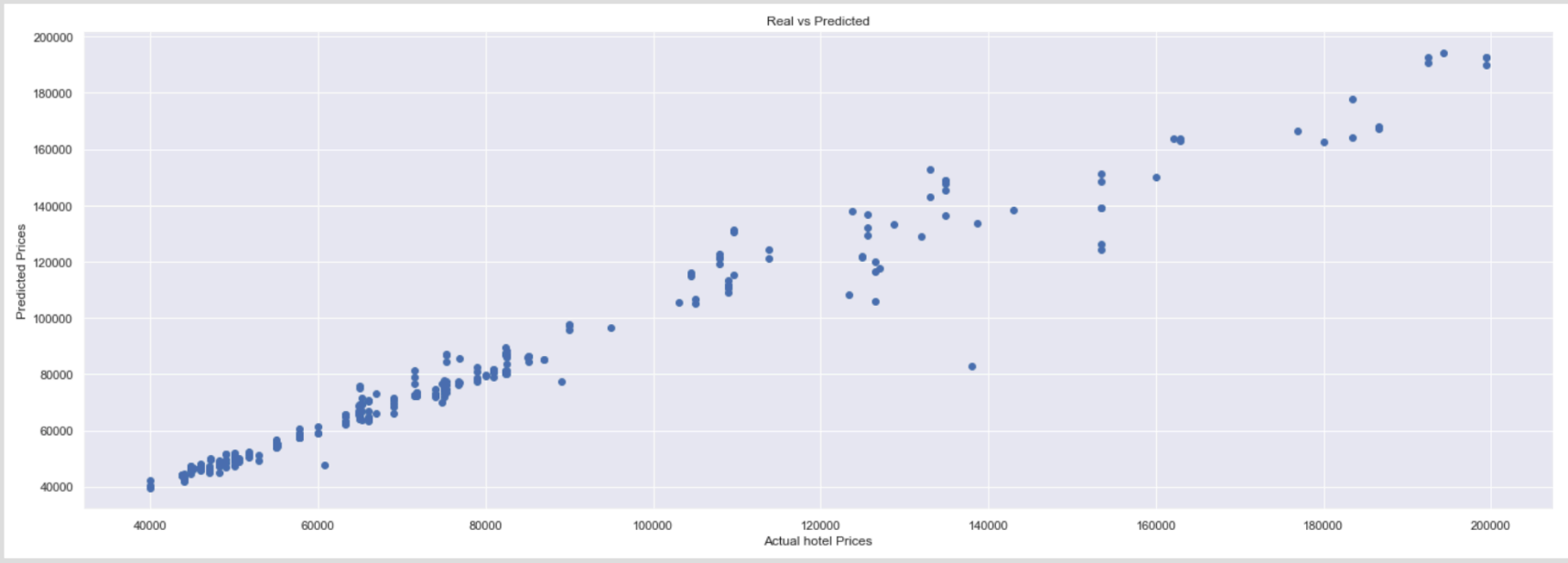
In [76]:
```
names = []
train_rmse = []
test_rmse = []
train_r2 = []
test_r2 = []

for name, model in models:
    model.fit(X_train, y_train)
    pred_test = model.predict(X_test)
    pred_train = model.predict(X_train)

    names.append(name)
    train_rmse.append((np.sqrt(mean_squared_error(y_train, pred_train))))
    test_rmse.append((np.sqrt(mean_squared_error(y_test, pred_test))))
    test_r2.append(r2_score(y_test, pred_test))
    train_r2.append(r2_score(y_train, pred_train))
```

In [77]:
```
result = pd.DataFrame({'model name':names,
                       'train rmse':train_rmse,
                       'test rmse':test_rmse,
                       'train score':train_r2,
                       'test score':test_r2})
result
```

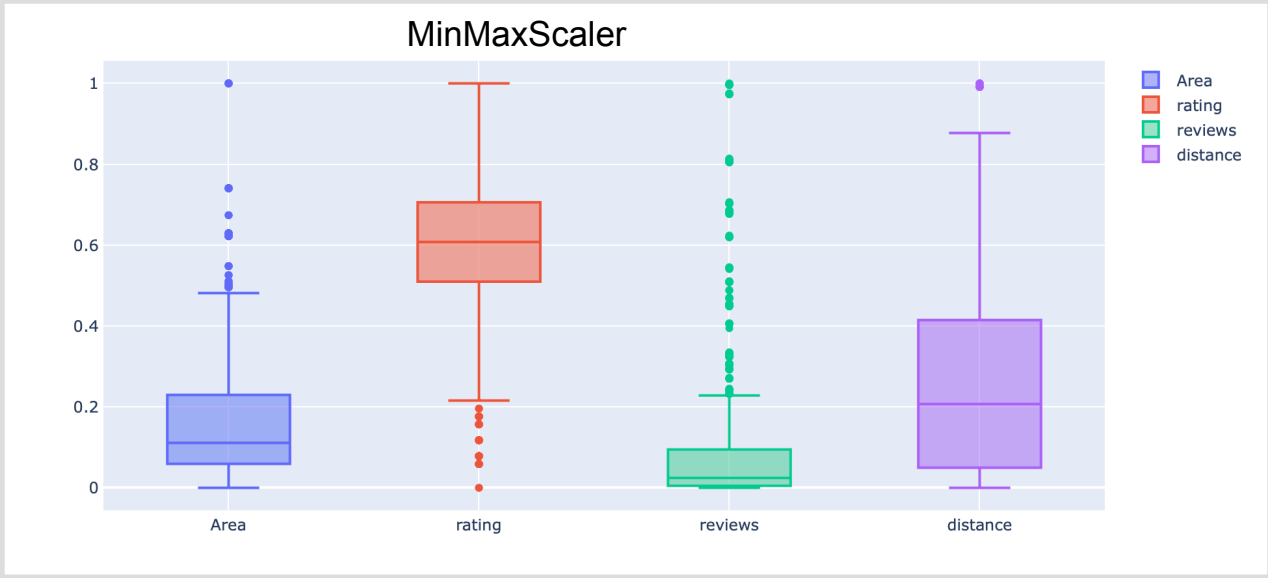
Out[77]:

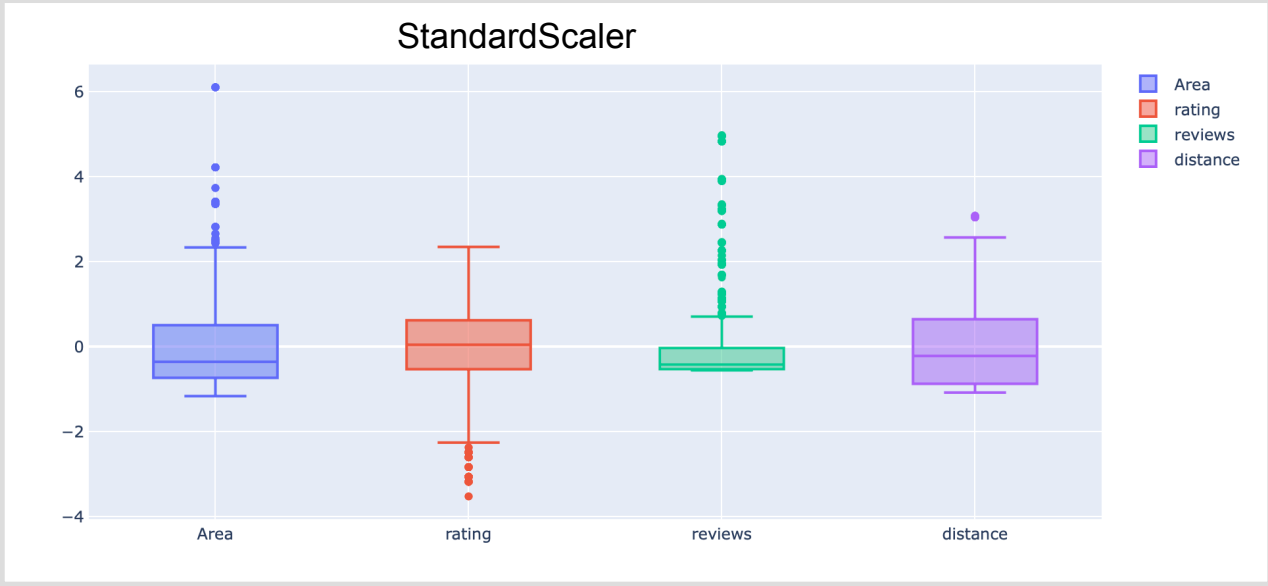
	model name	train rmse	test rmse	train score	test score
0	LinearReg	28579.599102	28703.443898	0.406911	0.393931
1	DecisionTree	4363.608472	8081.181007	0.986174	0.951960
2	RandomForest	4754.385280	7377.912814	0.983587	0.959958
3	GradientB	8100.342462	8627.853661	0.952355	0.945241
4	XGradianB	4382.047461	7868.607918	0.986057	0.954454
5	LightGBM	5043.809811	6861.865932	0.981528	0.965363

인터파크 회귀 분석


Real vs Predicted
Predicted Prices
Actual hotel Prices
200000
180000
160000
140000
120000
100000
80000
60000
40000
40000
60000
80000
100000
120000
140000
160000
180000
200000

Booking.com 회귀 분석





Booking.com 회귀 분석

LinearRegression

In [18]:
```
from sklearn.model_selection import train_test_split

X = hotels_df_copy[['Area', 'rating', 'reviews', 'distance', 'weekend']]
y = hotels_df_copy['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=13)
```

In [26]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

lr = LinearRegression()
lr.fit(X_train, y_train)

pred_tr = lr.predict(X_train)
pred_test = lr.predict(X_test)

rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

print("RMSE of Train Data : ", rmse_tr)
print("RMSE of Test Data : ", rmse_test)

print('Train Acc : ', r2_score(y_train, pred_tr))
print('Test Acc : ', r2_score(y_test, pred_test))
```

```
RMSE of Train Data :  28706.68841444606
RMSE of Test Data :  27549.056886837352
Train Acc :  0.2713355744005723
Test Acc :  0.2962487550499314
```

Booking.com 회귀 분석

In [70]:
```
models = []
models.append(('LinearReg', LinearRegression()))
models.append(('DecisionTree', DecisionTreeRegressor()))
models.append(('RandomForest', RandomForestRegressor()))
models.append(('GradientB', GradientBoostingRegressor()))
models.append(('XGradianB', XGBRegressor()))
models.append(('LightGBM', LGBMRegressor()))
```

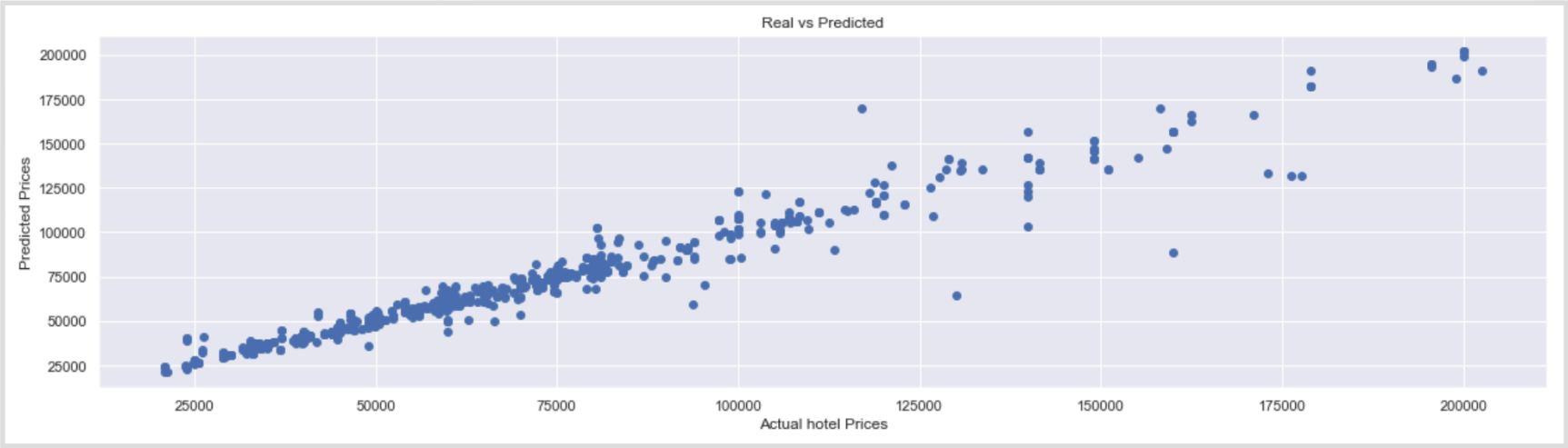
In [71]:
```
names = []
train_rmse = []
test_rmse = []
train_r2 = []
test_r2 = []

for name, model in models:
    model.fit(X_train, y_train)
    pred_test = model.predict(X_test)
    pred_train = model.predict(X_train)

    names.append(name)
    train_rmse.append((np.sqrt(mean_squared_error(y_train, pred_train))))
    test_rmse.append((np.sqrt(mean_squared_error(y_test, pred_test))))
    test_r2.append(r2_score(y_test, pred_test))
    train_r2.append(r2_score(y_train, pred_train))
```

In [72]:
```
result = pd.DataFrame({'model name':names,
                       'train rmse':train_rmse,
                       'test rmse':test_rmse,
                       'train score':train_r2,
                       'test score':test_r2})
result
```

Out[72]:

	model name	train rmse	test rmse	train score	test score
0	LinearReg	28706.688414	27549.056887	0.271336	0.296249
1	DecisionTree	2976.038312	6062.792312	0.992169	0.965916
2	RandomForest	3378.216818	5348.107250	0.989909	0.973478
3	GradientB	13484.785516	14077.261969	0.839214	0.816244
4	XGradianB	3074.690890	5495.817758	0.991641	0.971993
5	LightGBM	4882.735552	6539.204679	0.978919	0.960349

Booking.com 회귀 분석



05 결론 및 한계점

1. 예측 값 확인

2. 회고 (한계점)

예측 값 확인

인터파크

In [59]:
```
jinseo = np.array([[20200925, 4, 1000, 8.5, 5]])
print("Jinseo : ", best_model.best_estimator_.predict(jinseo))
```

```
Jinseo :  [71990.44158191]
```

In [60]:
```
hyunggi = np.array([[20200918, 4, 750, 8, 10]])
print("Hyunggi : ", best_model.best_estimator_.predict(hyunggi))
```

```
Hyunggi :  [94817.71601341]
```

booking.com

In [83]:
```
# Area, Reviews, rating, distance, weekend (0:weekday)
jinseo = np.array([[38, 243, 7.8, 12, 0]])
print("Jinseo : ", best_model_ss.best_estimator_.predict(jinseo))
```

```
Jinseo :  [140252.08570413]
```

In [84]:
```
# Area, Reviews, rating, distance, weekend (1:weekend)
hyunggi = np.array([[38, 243, 7.8, 12, 1]])
print("Hyunggi : ", best_model_ss.best_estimator_.predict(hyunggi))
```

```
Hyunggi :  [141702.38401301]
```

한계점

가격 차이

조회기간 : 2019.07 ~ 2019.07, 여객 : 전체, 화물 : 전체, 공항 : 제주(CJU), 운항 : 전체, 노선 : 전체, 여객화물 : 전체

공항명	공급(석)	운항(편)	여객(명)
제주(CJU)	1,619,236	8,338	1,404,339

조회기간 : 2020.07 ~ 2020.07, 여객 : 전체, 화물 : 전체, 공항 : 제주(CJU), 운항 : 전체, 노선 : 전체, 여객화물 : 전체

공항명	공급(석)	운항(편)	여객(명)
제주(CJU)	1,236,092	6,405	980,222

(source : 항공정보포털시스템 (www.airportal.go.kr)

대략 30 % 감소 → 수요 감소가 주중, 주말 가격 형성에 영향 미쳤는지?

크롤링

데이터 수집에 제약 → 데이터 컬럼들이 많았으면, 다양한 요인들을 바탕으로 가격 분석을 진행할 수 있지 않았을까?

THANK YOU