airbnb

Analysis of European AirBnBs using Machine Learning

Fri 8-10 Group B

Lance Nillo
Danny Weng
Nancy Wu
Daniel Chua
Winson Liang

Introduction

Winson Liang
Business Implications and Overview

Nancy Wu
Slide design and dataset summary

Lance Nillo
Data analysis and ML model

Daniel Chua
Model analysis and results

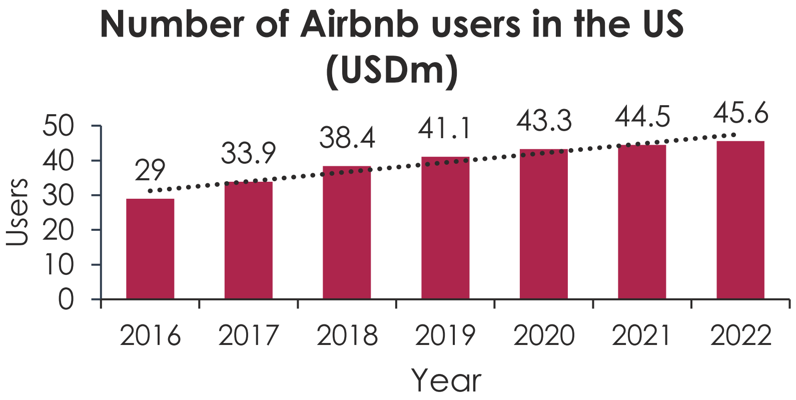
Danny Weng
Discussion and model analysis

Agenda

1. Airbnb business overview and problem statement
2. Dataset analysis
3. Data preprocessing and exploration
4. Machine learning model
5. Results
6. Business implications and conclusion

Overview

Airbnb business overview



Source: Statista (https://www.statista.com/statistics/346589/number-of-us-airbnb-users/)

Source: Airbnb

Problem statement

How can we effectively assess the appropriate rental rates for AirBnB listings in European cities while considering the significance of various factors influencing pricing?

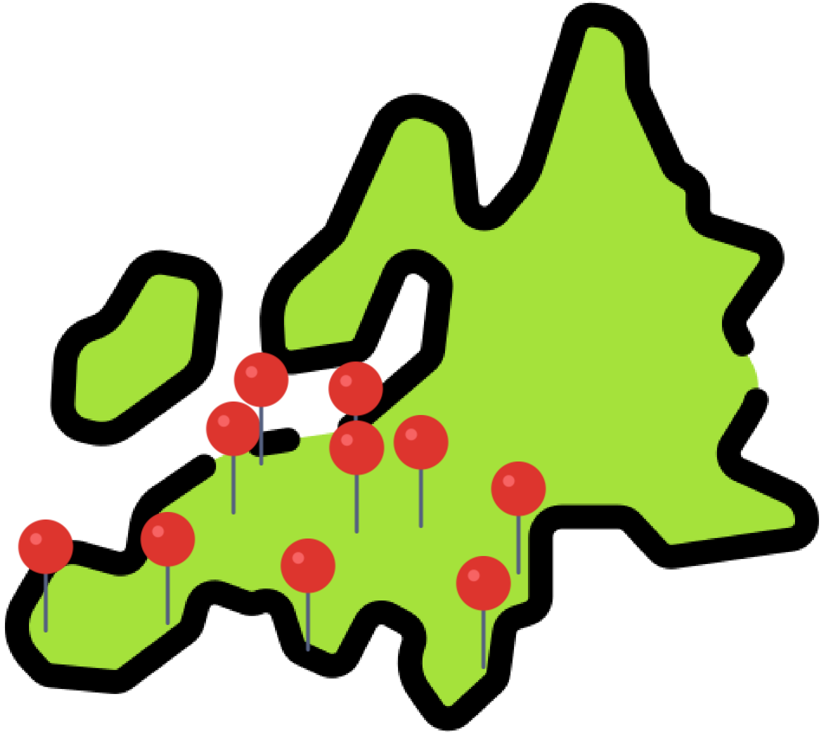
Hypothesis

We hypothesise that City Center, Bedrooms, and Attraction Index are significant determinants to European Airbnb pricing.

- As proximity to the city center increases, pricing increases.
- As the number of bedrooms increases, pricing increases.
- As attraction index increases, pricing increases.

Dataset Analysis

What does our AirBnB dataset look like?



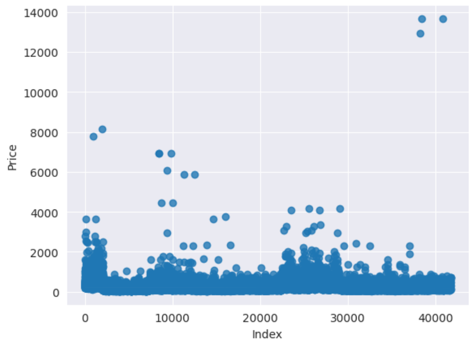
Accommodation from different cities across Europe, including Amsterdam, Athens, Barcelona, Berlin, Budapest, Lisbon, Paris, Rome and Vienna

41,714 rows ⇒ 41,714 Airbnbs

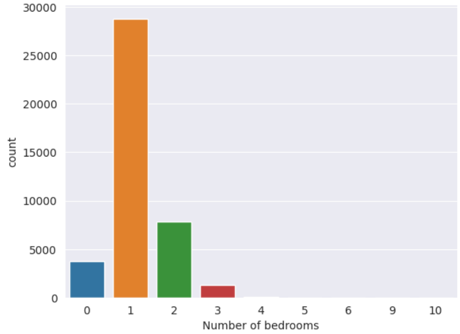
16 columns ⇒ 16 different feature variables

Eight columns of interest to focus on

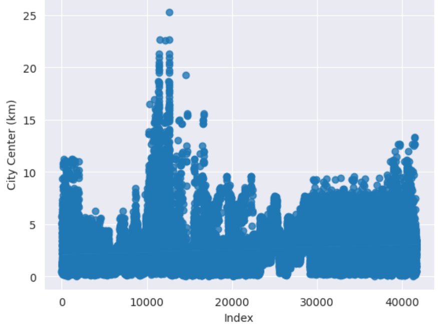
Price



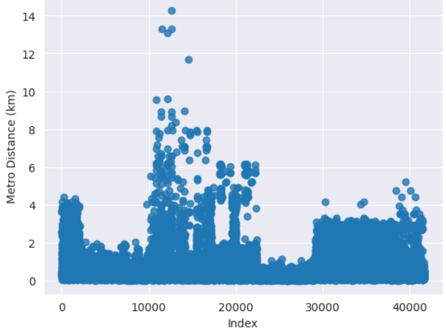
Bedrooms



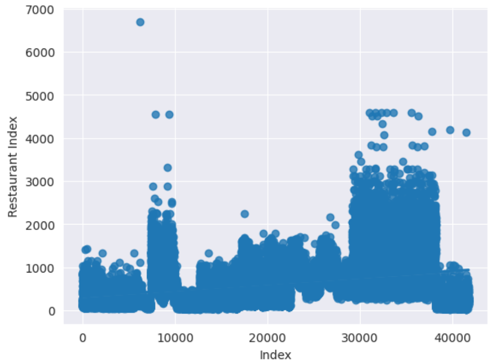
City Center (km)



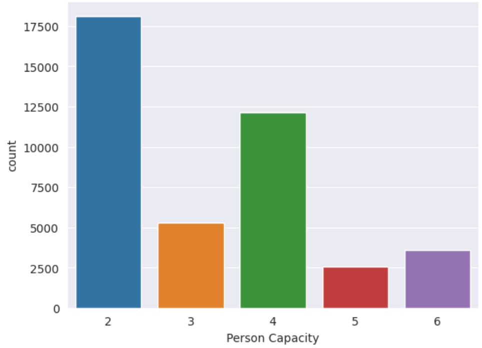
Metro Distance (km)



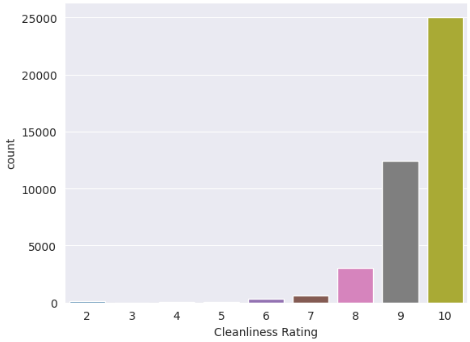
Restaurant Index



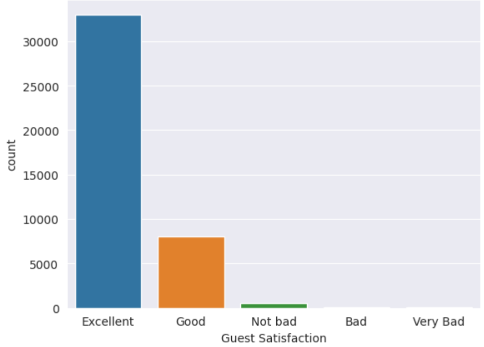
Person Capacity



Cleanliness Rating



Guest Satisfaction



Data Preprocessing

3 key steps we undertook for data preprocessing

1 → 2 → 3

1. Drop Price null values
2. Clear remaining null values using mean imputation
3. Use Label encoding for Guest Satisfaction variable

Price has too many null values

1

Drop Price null values

```
df = bnbdf.dropna(subset=['Price'] )
```

Price
41,714 rows of data
- 33,372 non-null values
- 8,342 null values

Price
33,372 rows of data
- 33,372 non-null values

City Center and Metro Distance also have null values

2

Clear remaining null values using mean imputation

```
df['City Center (km)'].fillna((df['City Center (km)'].mean()), inplace=True)

df['Metro Distance (km)'].fillna((df['Metro Distance (km)'].mean()), inplace=True)
```

City Center (km)
33,372 rows of data
- 32,695 non-null values
- 677 null values

Metro Distance (km)
33,372 rows of data
- 30,006 non-null values
- 3,366 null values

City Center (km)
33,372 rows of data
- 33,372 non-null values

Metro Distance (km)
33,372 rows of data
- 33,372 non-null values

Guest Satisfaction is currently a categorical variable

3

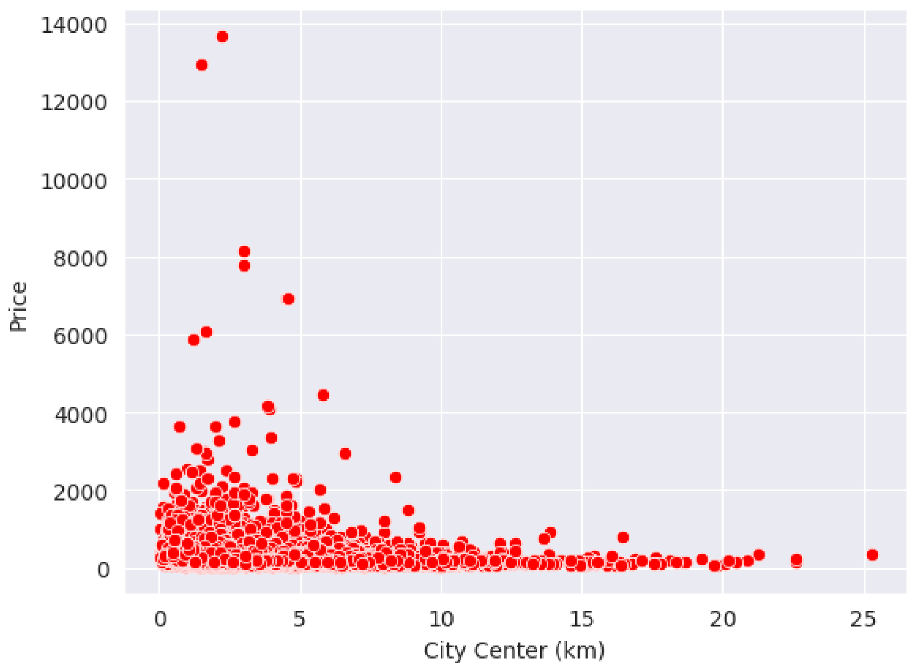
Use Label encoding for Guest Satisfaction variable

```
satisfaction_map = {
    'Very Bad': 0,
    'Bad': 1,
    'Not bad': 2,
    'Good': 3,
    'Excellent': 4
}

df['Guest Satisfaction'] = df['Guest Satisfaction'].map(satisfaction_map)

df['Guest Satisfaction'].value_counts()
```

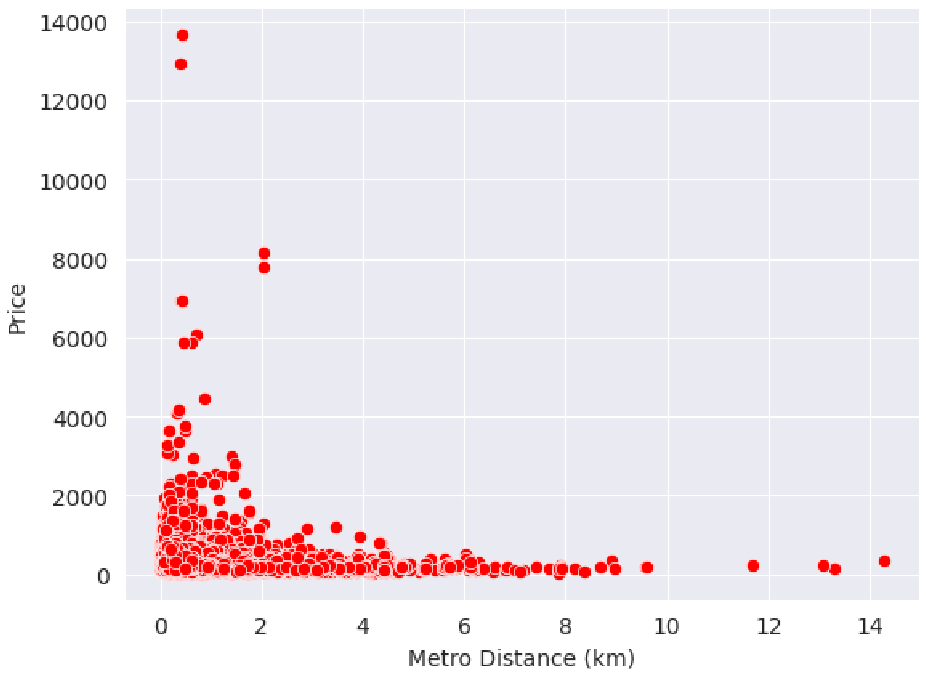
Data Exploration

City Center and Metro distance are both right skewed

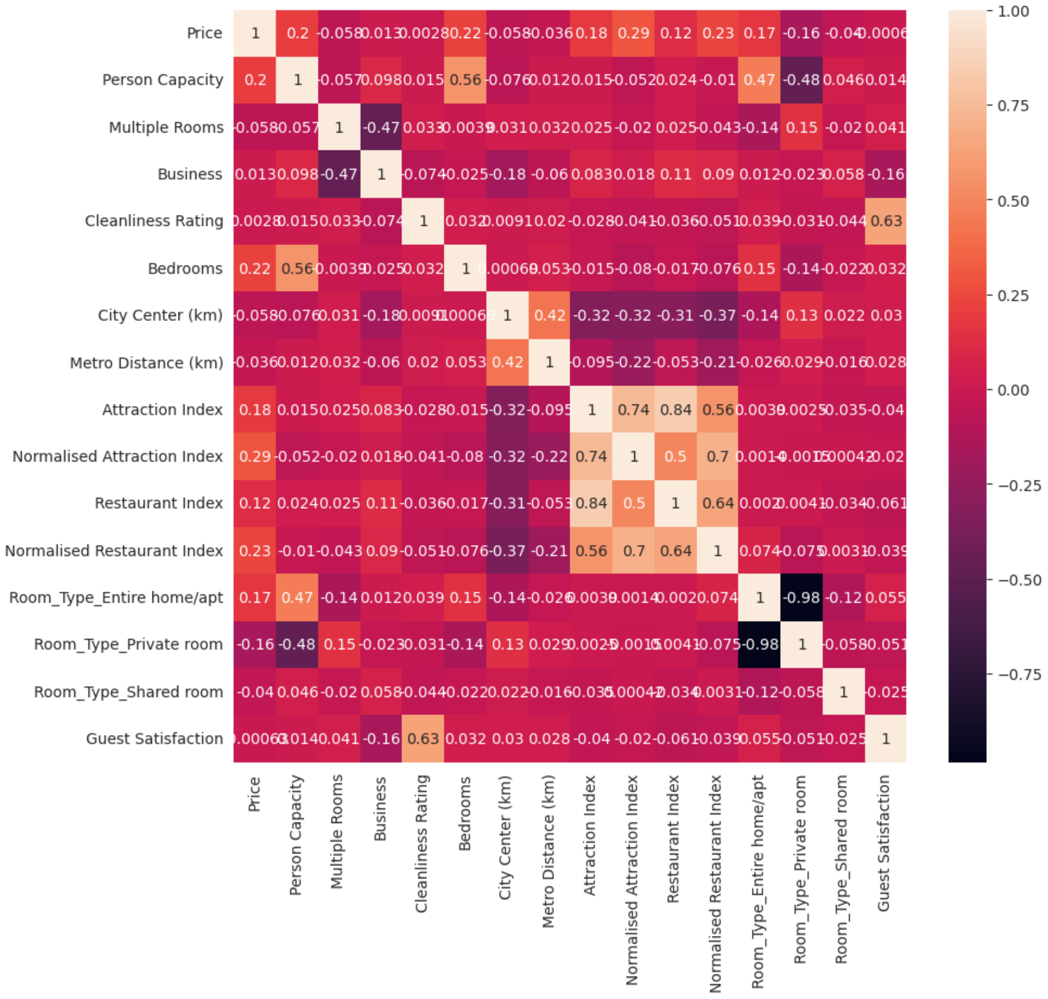
City Center vs Price



Metro Distance vs Price



An initial correlation table appears to show no correlation at all



City Center (km): -0.058

Metro Distance (km): -0.036

Normalised Attraction Index: 0.29

Normalised Restaurant Index: 0.23

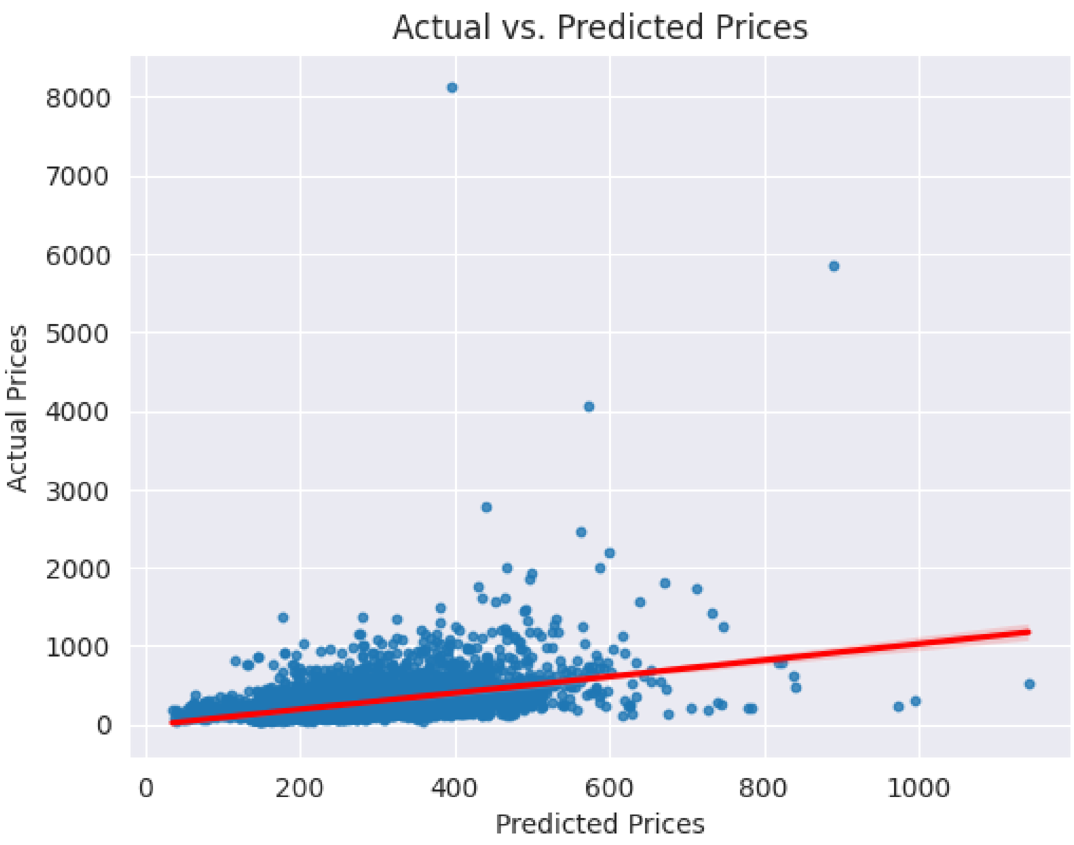
airbnb

Machine Learning Model

Training and testing the data with Linear Regression

```
x_values = df_dropped.drop(['Price'], axis = "columns")
y_values = df_dropped['Price’]


x_train, x_test, y_train, y_test = train_test_split(x_values, y_values, test_size=0.2,
random_state=42)
model = LinearRegression().fit(x_train, y_train)
preds = model.predict(x_test)


#Plotting
sns.regplot(x=preds, y=y_test, scatter_kws={'s':10}, line_kws={'color':'red'})
```

A look into our first Machine Learning Model



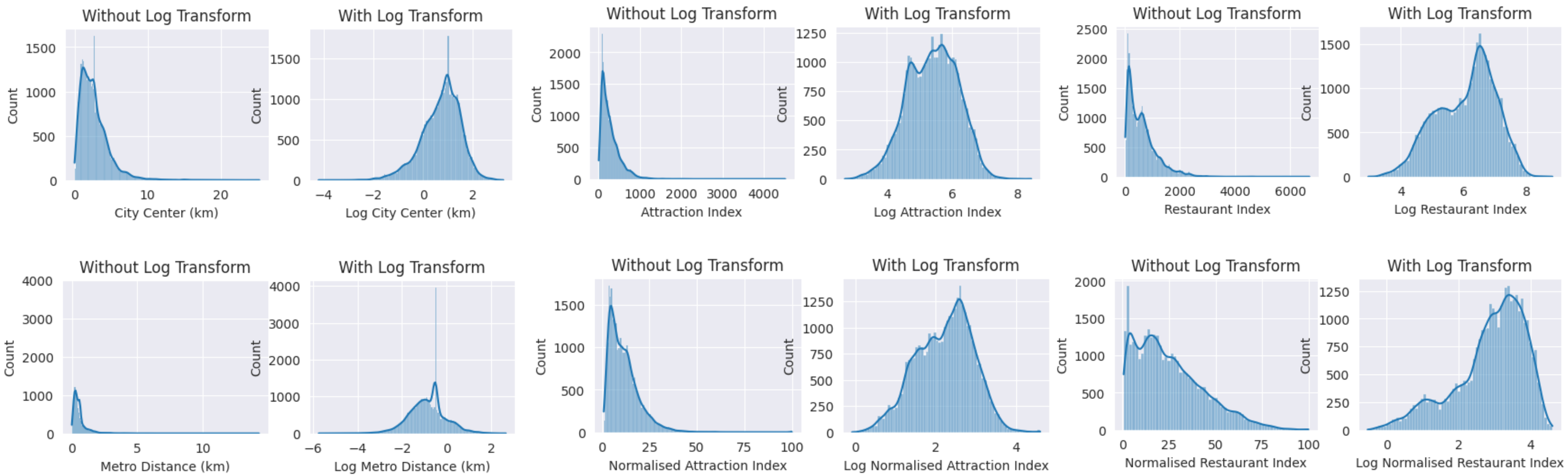
MAE = 106.13

MSE = 40,668.64

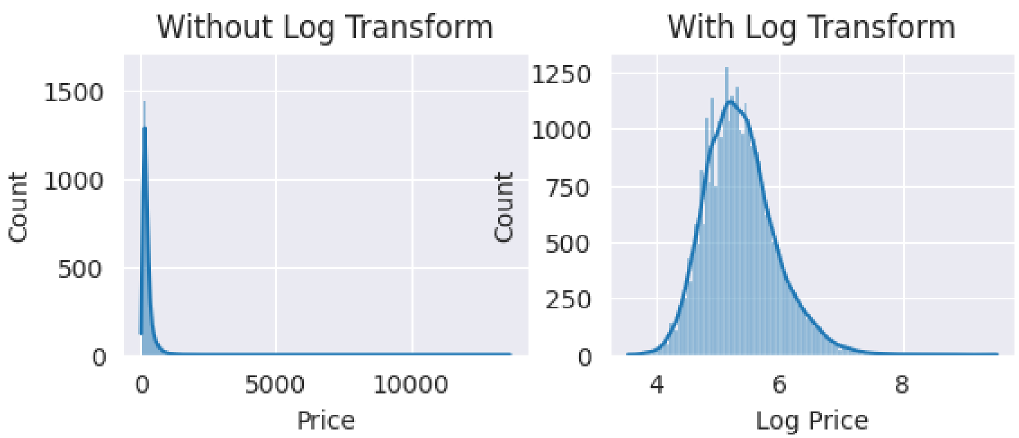
RSME = 201.66

$r^2 = 0.21$

A skew analysis of independent variables



A skew analysis of our target variable, Price

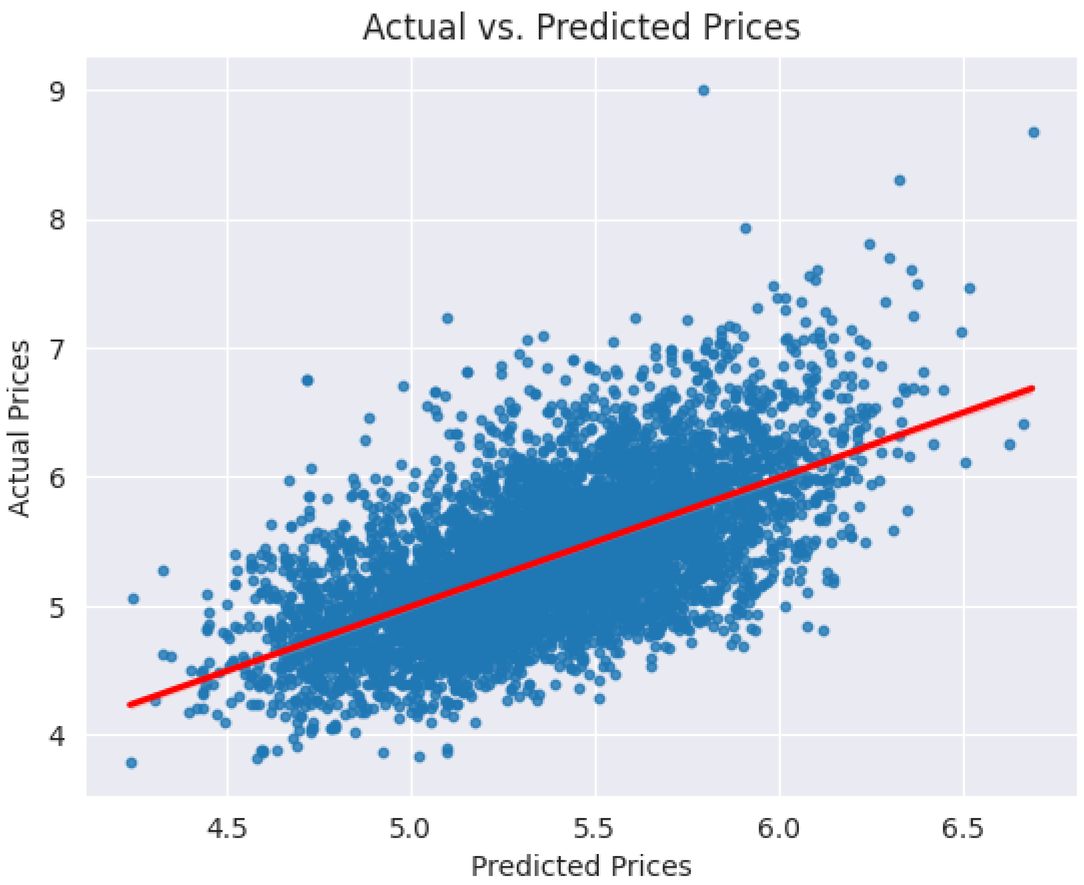



```
#Log transforming features and creating updated dataframe

logged_df_improved = df_improved.copy()

continuous_cols.remove('Normalised Restaurant Index') #Removing Normalised
Restaurant Index as no improvement for skew after log transformation

for col in continuous_cols:
    logged_df_improved[col] = np.log(logged_df_improved[col])

logged_df_improved['Price'] = np.log(logged_df_improved['Price'])
```

Back-transforming for our final model

```
x_values = logged_df_improved.drop(['Price'], axis = "columns")
y_values = logged_df_improved['Price']

x_train, x_test, y_train, y_test = train_test_split(x_values, y_values, test_size=0.2, random_state=42)

model = LinearRegression().fit(x_train, y_train)

preds = model.predict(x_test)

#Back-transform predictions and test values
preds_exp = np.exp(preds)
y_test_exp = np.exp(y_test)

#Plotting the log-transformed model
sns.regplot(x=preds, y=y_test, scatter_kws={'s':10}, line_kws={'color':'red'})

#Plotting back-transformed model on original scale
sns.regplot(x=preds_exp, y=y_test_exp, scatter_kws={'s':10}, line_kws={'color':'red'})
```

Log-transformed Machine Learning model



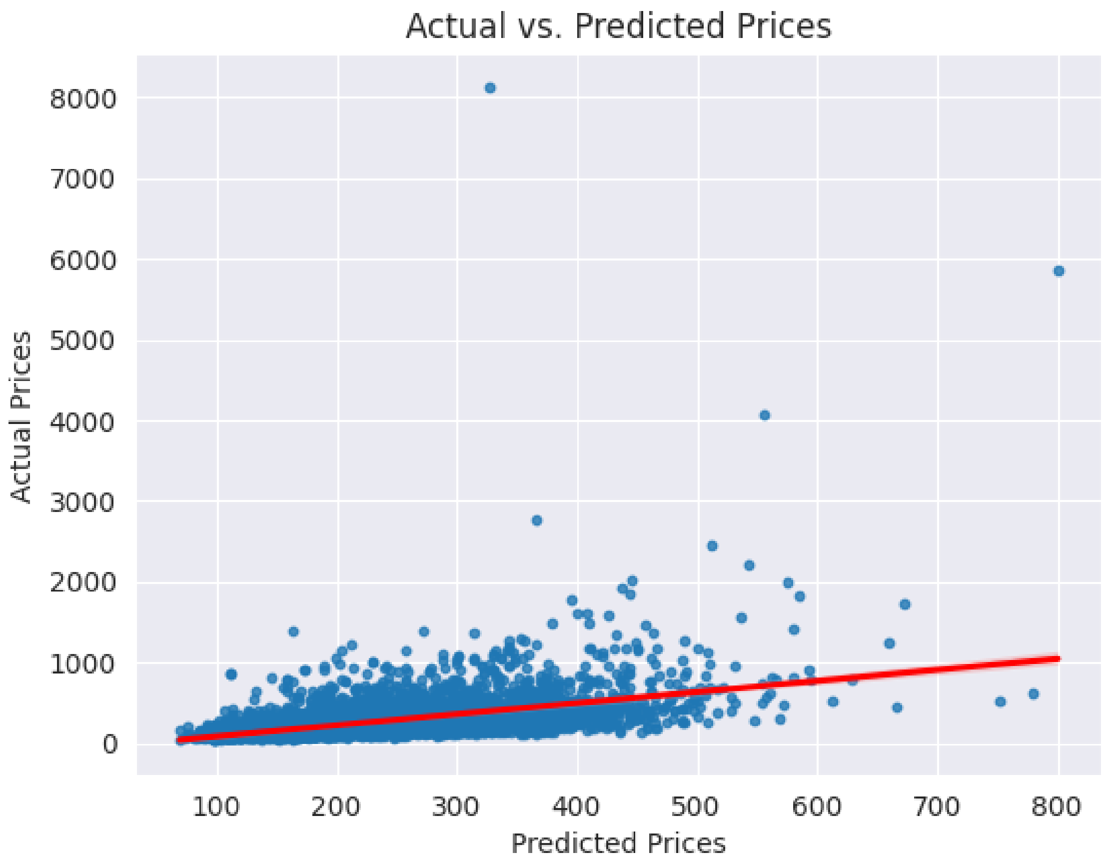
MAE = 0.34

MSE = 0.20

RSME = 0.44

r^2 = 0.39

Final back-transformed model



MAE = 92.96

MSE = 40,016.90

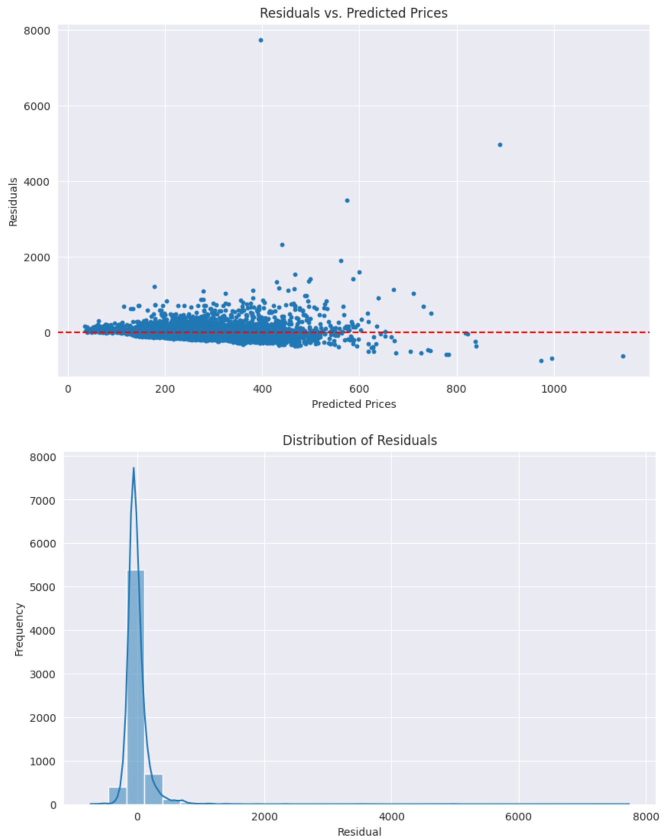
RSME = 200.04

$r^2 = 0.39$

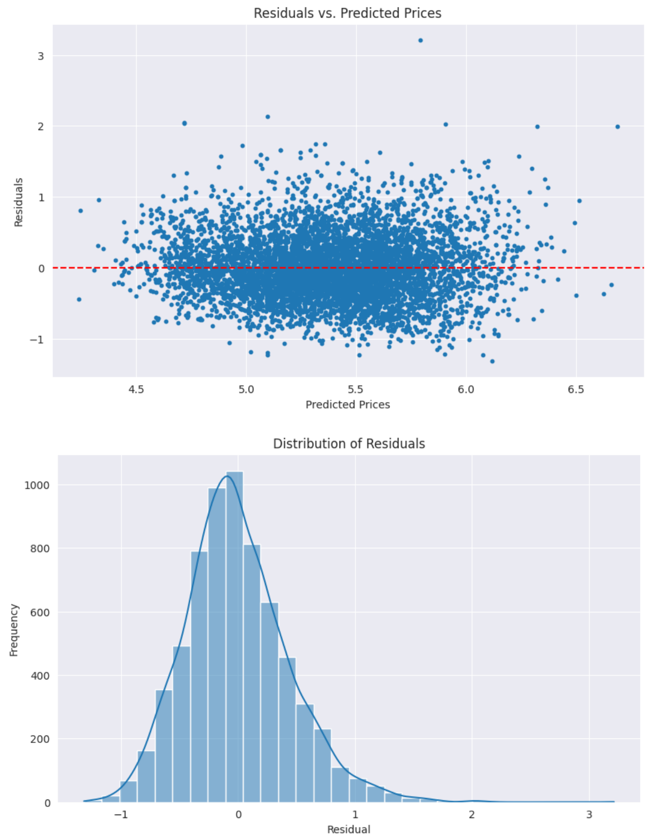
Results

Comparison of residuals

Initial model

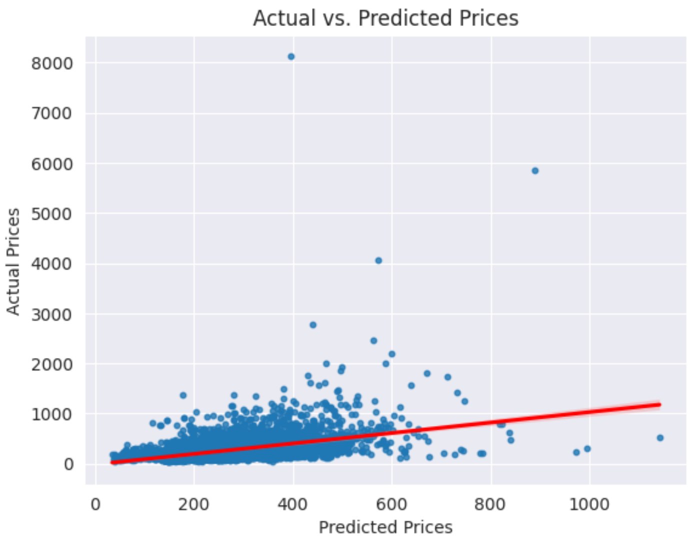


Log transformed model



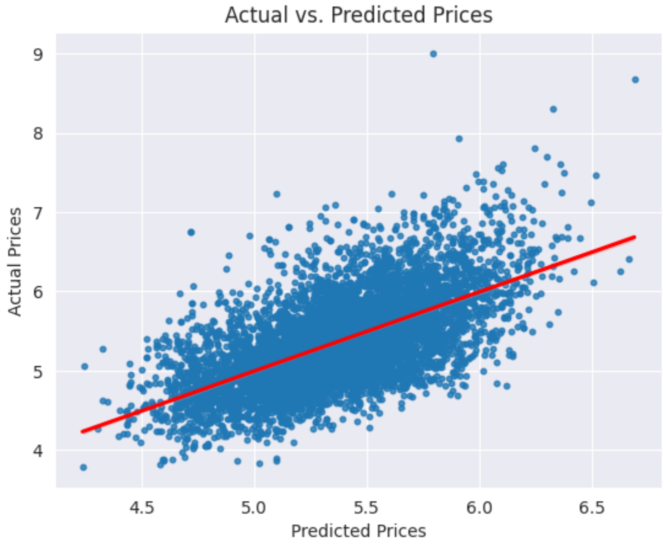
Comparison of models

Initial model



- MAE = 106.13
- MSE = 40,668.64
- RSME = 201.66
- $r^2 = 0.21$

Log transformed model



- MAE = 92.96
- MSE = 40,016.90
- RSME = 200.04
- $r^2 = 0.39$

What does our final model mean?

Coefficients

- Intercept: 50.09 Euros

```
#Back transforming the coefficients and intercept
print("Intercept: ${}".format(round(np.exp(intercept),2)))

print("\nCoefficients:")
for i, col in enumerate(x_train.columns):
    print("{}: {}".format(col,round(np.exp(coef[i]),2)))
```

Person Capacity

X 1.05

Bedrooms

X 1.21

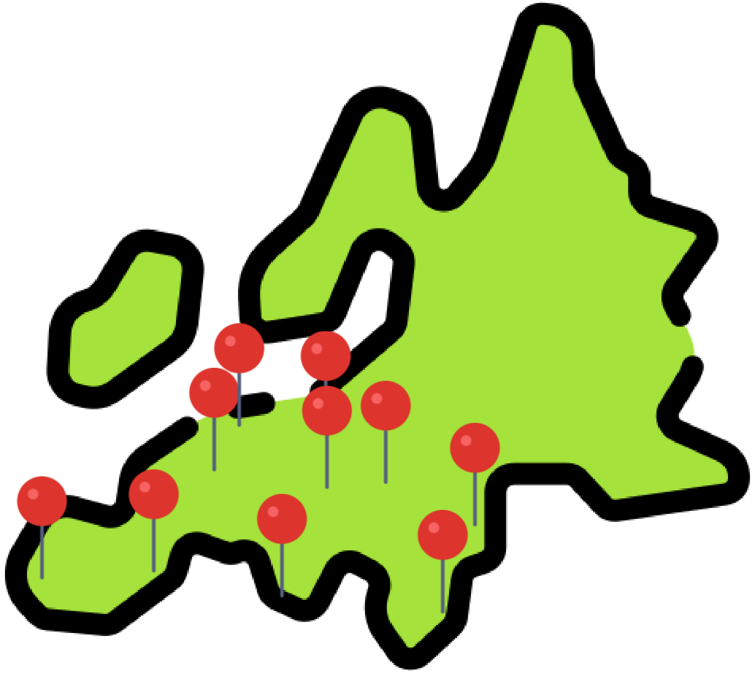
Norm. Attraction Index

X 1.51

Room Type (Home/Apart)

X 1.29

Why is our Model Poor?



Dataset contained features from multiple cities:

- Amsterdam, Athens, Barcelona, Berlin, Budapest, Lisbon, Paris, Rome & Vienna
- Variability between each city, difficult to create a general model across all cities.
- **"This dataset encapsulates the diverse tapestry..."**

Business Implications

What does this mean for Airbnb?

Ensures Host Loyalty

- Educate hosts on the importance of location and amenities
- Appealing listings for higher satisfaction and reviews
- Satisfied hosts = long-term customers

Transparency builds Trust

- Ensures pricing meets customer expectations for a satisfying experience
- Hosts can maximise their revenue margins

Top of the competition

- Ensures Airbnb remains competitive against other home rental marketplaces
- Allows Airbnb to effectively resolve customer complaints

Future opportunities for Airbnb

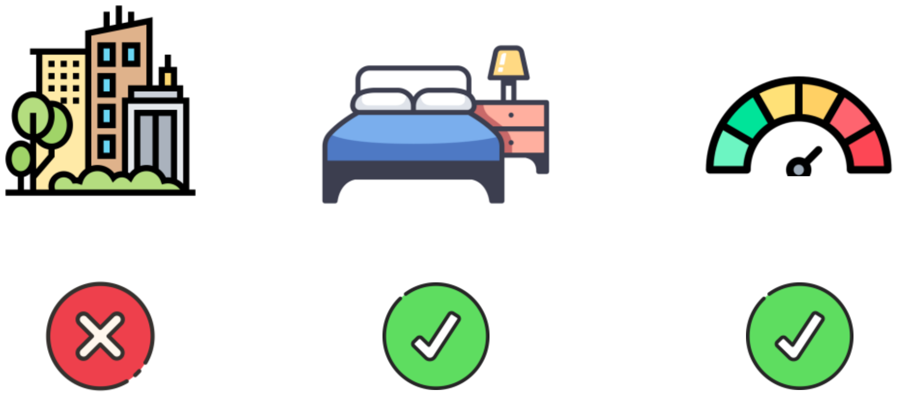
Machine learning integration into the platform

- Allow users to better estimate the costs of their rental stay
- Allow hosts to get an approximation of how much they should charge

Conclusion

Going back to our hypothesis…

We hypothesised that City Center, Bedrooms, and Attraction Index are significant determinants to European Airbnb pricing.



The factors with most influence are:

- Person Capacity: 1.05
- Bedrooms: 1.21
- Normalised Attraction Index: 1.51
- Room_Type_Entire home/apt: 1.29

airbnb

Thank you + Q&A

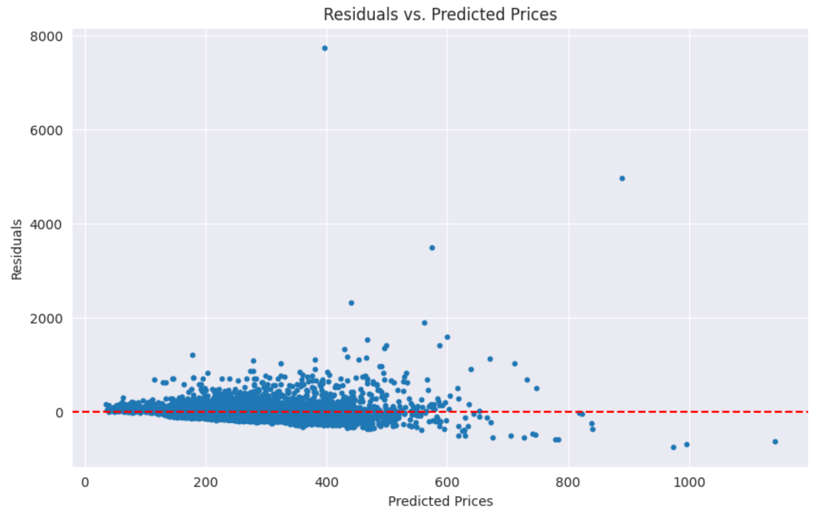
Appendices

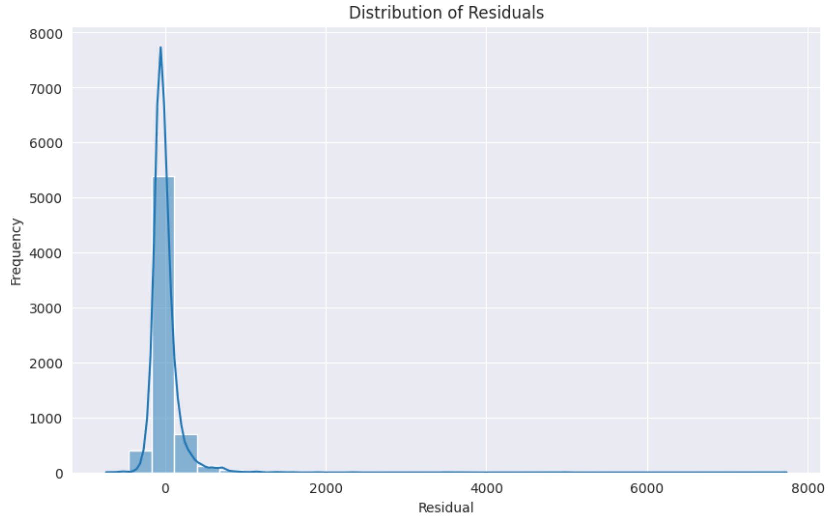
Appendix – AirBnB dataset extract

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Price	Person Capacity	Multiple Rooms	Business	Cleanliness Rating	Bedrooms	City Center (km)	Metro Distance (km)	Attraction Index	Normalised Attraction Index	Restaurant Index	Normalised Restaurant Index	Room_Type_Entire home/apt	Room_Type_Private room	Room_Type_Shared room	Guest Satisfaction
2	194.0336981	2	1	0	10	1	5.022963798	2.539380003	78.69037927	4.166707868	98.25389587	6.846472824	0	1	0	Excellent
3		4	0	0	8	1	0.488389289	0.239403923	631.1763783	33.42120862	837.2807567	58.34292774	0	1	0	Good
4		2	0	1	9	1	5.748311915	3.651621289	75.27587691	3.9859077	95.38695493	6.646700255	0	1	0	Good
5	433.529398	4	0	1	9	2	0.384862013	0.439876076	493.2725344	26.11910845	875.0330976	60.97356517	0	1	0	Excellent
6	485.5529257	2	0	0	10	1	0.544738183	0.318692647	552.8303244	29.272733	815.30574	56.81167696	0	1	0	Excellent
7	552.8085675	3	0	0	8	2	2.131420081	1.904668241	174.7889568	9.255191399	225.2016624	15.69237584	0	1	0	Excellent
8	215.1243175	2	0	0	10	1	1.881091564	0.729746739	200.1676516	10.59901016	242.7655237	16.91625096	0	1	0	Excellent
9	2771.307384	4	0	0	10	3	1.686806965	1.458403566	208.8081086	11.05652809	272.3138229	18.97521897	1	0	0	Excellent
10	1001.80442	4	0	0	9	2	3.719141399	1.196112353	106.2264562	5.624761439	133.8762019	9.328686362	1	0	0	Excellent
11	276.5214538	2	1	0	10	1	3.142361426		206.2528615	10.92122606	238.2912578	16.60447768	0	1	0	Good
12	909.4743749	2	0	0	10	1	1.009922025		409.8581245	21.70226002	555.1142756	38.68116138	1	0	0	Excellent
13	319.6400534	2	1	0	10	1	2.182707104	1.590381363	191.5013392	10.14012315	229.2974006	15.97777277	0	1	0	Excellent
14	675.6028402	4	0	0	8	1	2.933045843	0.628073047	214.9233419	11.38033376	269.624904	18.78785123	1	0	0	Good
15	552.8085675	2	0	0	10	1	1.305493932	1.342162408	325.2559516	17.22251876	390.912052	27.2393142	1	0	0	Excellent
16	209.0314719	2	1	0	8	1	7.304535267	3.720813886	59.77618069	3.165188486	75.70105653	5.274958532	0	1	0	Excellent
17	368.8514986	2	0	0	10	1	1.031100609	0.557884535	359.9219322	19.05810546	439.950562	30.65638813	0	1	0	Excellent
18	368.8514986	2	0	0	10	1	1.327797157	0.119528107	539.0128842	28.54109037	573.896572	39.98993881	0	1	0	Excellent
19	337.9185902	2	1	0	10	1	1.366334238	0.53493346	576.0828256	30.5039684	845.957616	58.94754378	0	1	0	Excellent
20	313.5472078	2	1	0	10	1	1.28975923	0.552116488	528.9405557	28.00775388	1023.904785	71.34715853	0	1	0	Excellent
21	447.5898109	2	1	0	9	1	1.057619331	1.065339126	422.8529062	22.39034235	476.6969918	33.21693223	1	0	0	Excellent
22		2	1	0	10	1	2.870632844		169.6957545	8.985502956	210.2676921	14.6517553	0	1	0	Excellent
23	933.8457573	4	0	0	10	2	1.014066427	0.377103687	477.7940702	25.29951349	664.0532506	46.27218589	1	0	0	Excellent
24		2	1	0	10	1	1.247083667	1.098774338	267.3259896	14.15508878	366.4646498	25.53578404	0	1	0	Excellent
25	377.2877464	2	0	0	10	1	1.167492301	0.983743003	278.8005954	14.76267678	383.5778926	26.72825942	0	1	0	Excellent
26	245.5885455	2	1	0	10	1	4.230634297	2.68714404	92.2136022	4.882771508	116.2171437	8.098177779	0	1	0	Excellent
27		2	1	0	8	1	4.180814737	2.808437972	93.27076367	4.93874891	117.759656	8.205662253	0	1	0	Excellent
28	295.0343308	2	1	0	10	1	3.366018582	0.225882299	170.0286974	9.003132506	217.1552314	15.13168895	0	1	0	Excellent
29	295.0343308	2	0	0	10	1	4.127280019	0.839325385	129.2296995	6.842798454	164.405009	11.45597756	0	1	0	Excellent
30	1032.971668	4	1	0	9	2	2.161607614	1.424169427	183.3652586	9.709312281	222.3896893	15.49643351	1	0	0	Excellent
31	270.4286083	4	0	0	10	1	4.885477641		91.666212	4.853786833	114.6743433	7.990673226	0	1	0	Excellent
32	524.6877417	2	0	1	9	1	1.126326972	1.121070546	386.3649665	20.4582817	462.8059339	32.24898332	1	0	0	Good
33	599.6766105	4	0	0	9	3	3.363590725	0.697422494	156.2371148	8.272859043	198.8741211	13.85783489	1	0	0	Excellent
34	516.0171537	4	0	0	9	2	1.071168346	0.382535403	487.8730216	25.83320066	702.9288761	48.98109541	1	0	0	Excellent
35	602.2543529	3	0	0	10	2	3.856290113	0.727962239	148.724103	7.875040072	184.2598483	12.83949133	1	0	0	Excellent
36	504.0658027	4	0	0	9	2	4.075247886	1.873410991	98.63663409	5.222875315	122.1255419	8.509883465	1	0	0	Excellent
37	1609.917278	6	1	0	10	3	2.988588786	1.750002981	207.9524737	11.01122166	241.7781267	16.84744772	1	0	0	Excellent
38		2	1	0	9	0	4.147086686	0.038354694	118.7788847	6.289420865	151.3763929	10.54812485	1	0	0	Excellent
39	202.7042861	2	0	0	10	1	5.611750156	3.911722316	71.34500904	3.777765634	89.08448606	6.207535156	0	1	0	Excellent
40	796.288051	4	0	0	10	2	2.854991669	1.702583373	202.8431045	10.74067716	271.3813254	18.9102412	1	0	0	Excellent
41		2	0	0	9	1	6.411657372	4.143226904	67.21539749	3.559099959	83.99842553	5.853131141	0	1	0	Excellent
42	319.6400534	2	0	1	10	1	0.540353868	0.338657742	558.6099219	29.57876654	816.9103767	56.92349036	0	1	0	Excellent
43		4	0	1	9	1	0.495828246	0.332742213	717.4464438	37.98926591	848.4048419	59.11807	0	1	0	Excellent

Appendix – Residuals Distribution for Base Model

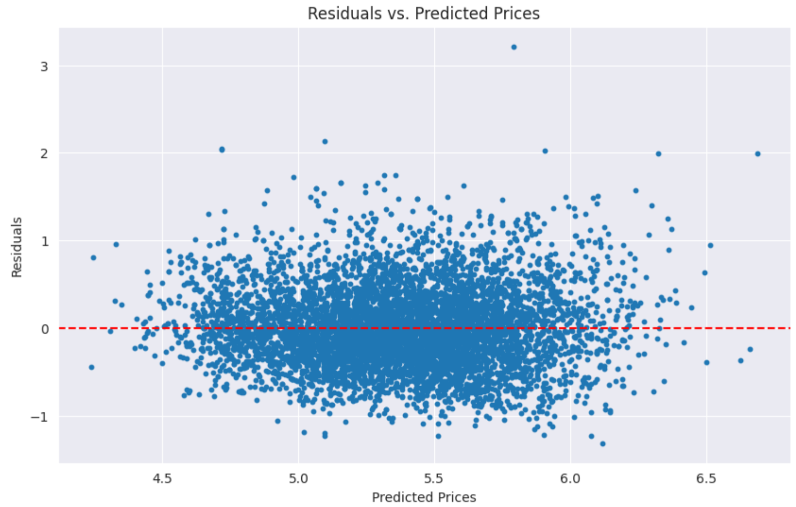
```
residuals = y_test - preds

#Plotting Residuals vs. Predicted values
plt.figure(figsize=(10, 6))
plt.scatter(preds, residuals, s=10)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Predicted Prices')
plt.ylabel('Residuals')
plt.title('Residuals vs. Predicted Prices')
plt.show()

#Histogram of Residuals
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True, bins=30)
plt.title('Distribution of Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```

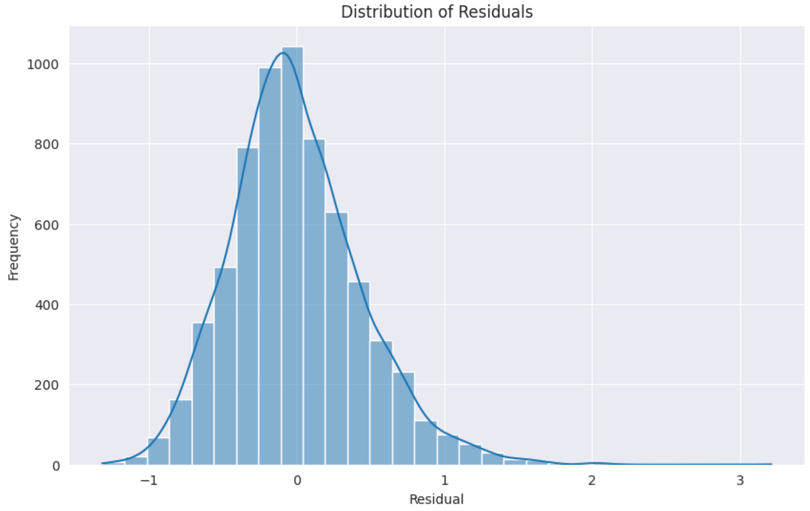




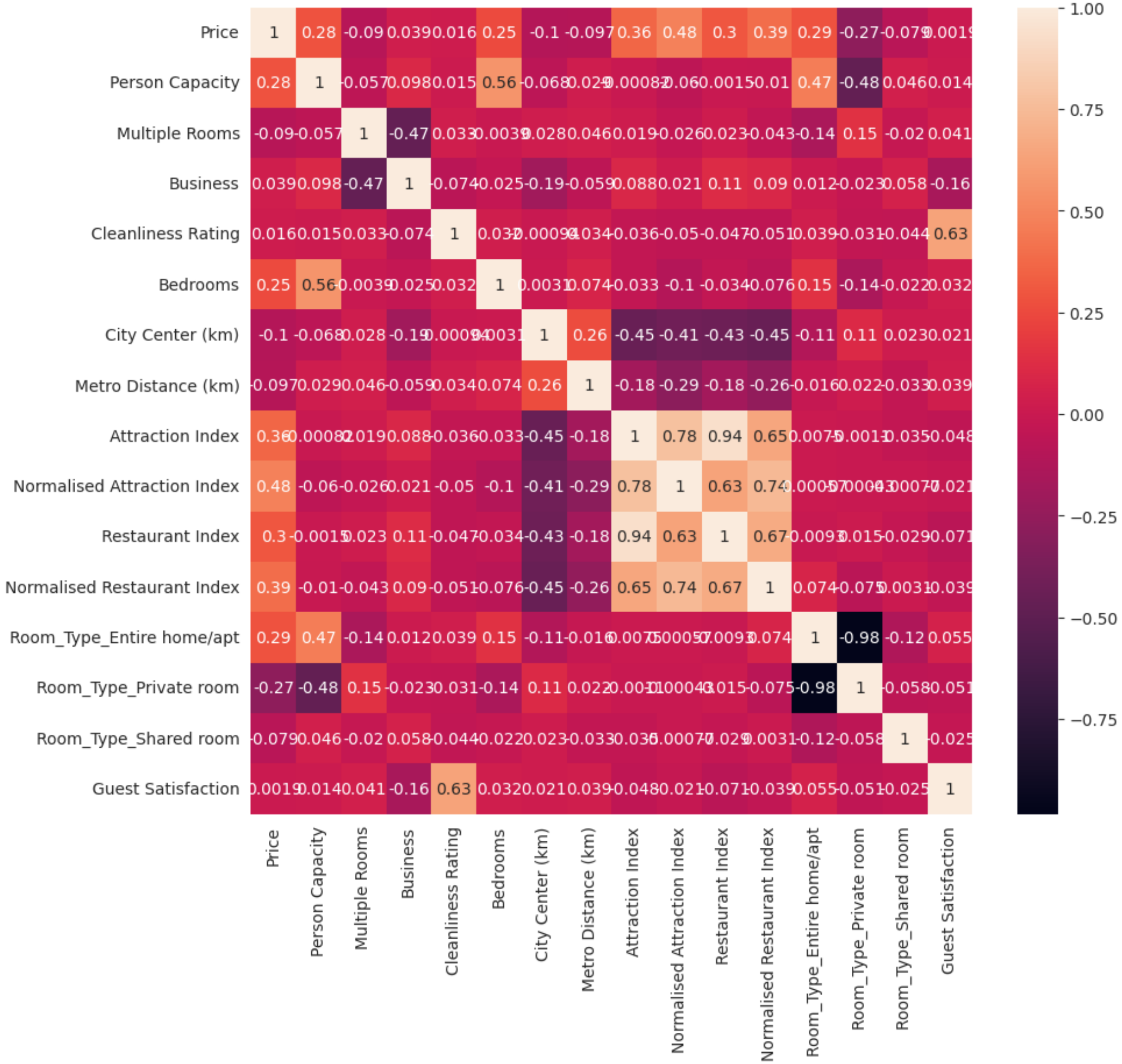
Appendix – Residuals Distribution for Logged Model

```
residuals = y_test - preds

#Plotting Residuals vs. Predicted values
plt.figure(figsize=(10, 6))
plt.scatter(preds, residuals, s=10)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Predicted Prices')
plt.ylabel('Residuals')
plt.title('Residuals vs. Predicted Prices')
plt.show()

#Histogram of Residuals
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True, bins=30)
plt.title('Distribution of Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```





Appendix – Correlation Analysis for Logged Model



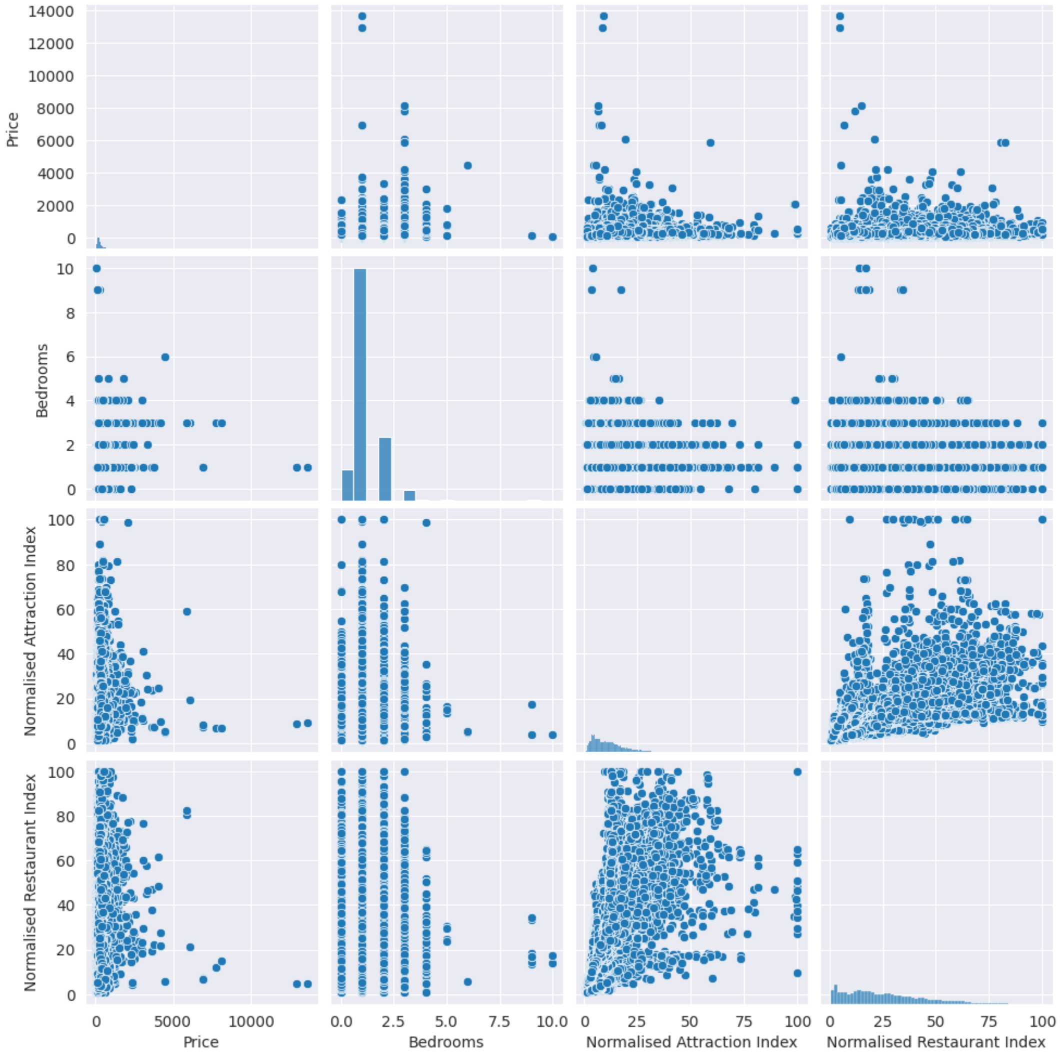
City Center (km): -0.1

Metro Distance (km): -0.097

Normalised Attraction Index: 0.48

Normalised Restaurant Index: 0.39

Appendix – Original pair plot



Appendix – Logged pair plot