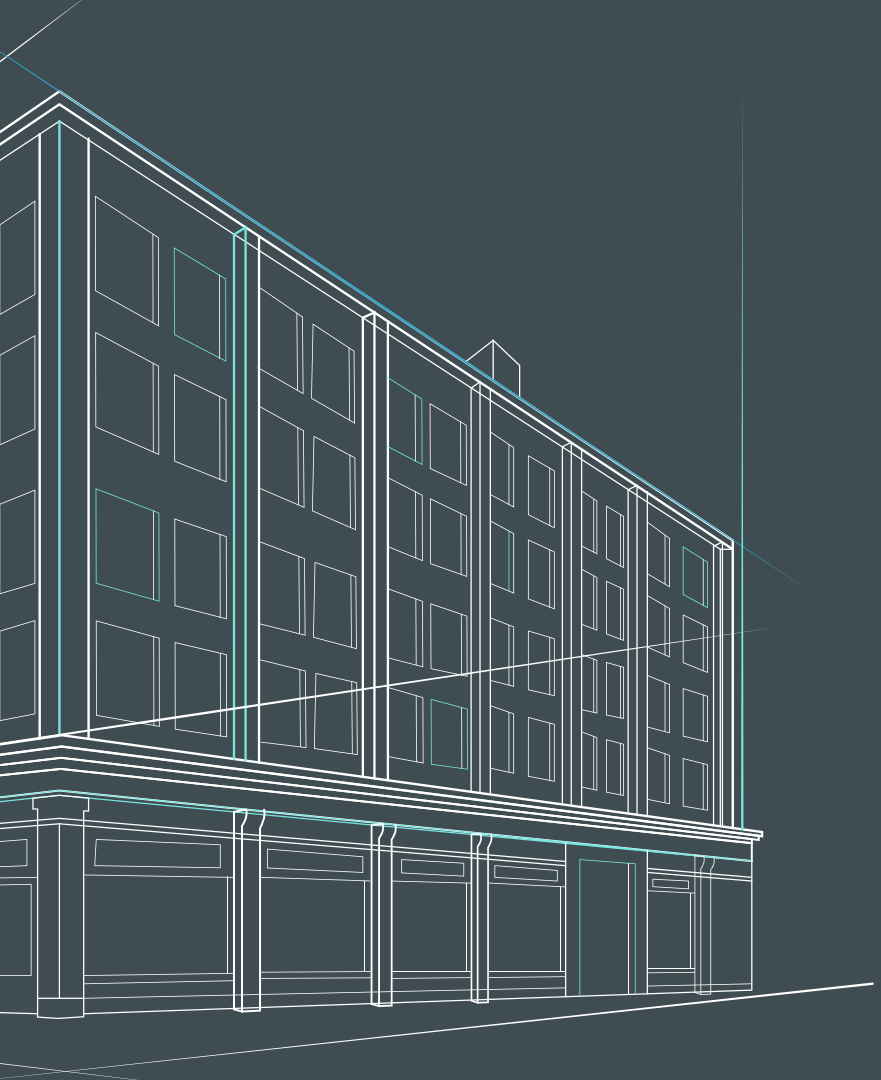


SC1015 MINI-PROJECT

A140_Team 7

Tan Jing Jie
Toh Xin Ying
Liu Yu-Hsing

CONTENTS

1. Motivation

01

SG's Rental Price ↑ 33.2 % from Jan 2022 to Jan 2023

02

AirBnb does not have predict price features

airbnb

03

Rentals do not know the average price to set to earn optimal profit

04

Rentals needs to set good price to attract customers

2. Problem Statement

Can we predict a good rental price for a house for AirBNB given its location, room type and property type etc?

2. Data Preparation

```
df = pd.read_csv('data/listings.csv')
df.head()
```

id	listing_url	scrape_id	last_scraped	source	name	description	neighborhood_overview	picture_url	host_id
71609	https://www.airbnb.com/roo...	20221229070856	2022-12-29	city scrape	Ensuite Room (Room 1 & 2) ...	For 3 rooms.Book room 1&2 ...	NaN	https://a0.muscache.com/pi...	367042
71896	https://www.airbnb.com/roo...	20221229070856	2022-12-29	city scrape	B&B Room 1 near Airport &...	The space Voca...	NaN	https://a0.muscache.com/pi...	367042
71903	https://www.airbnb.com/roo...	20221229070856	2022-12-29	city scrape	Room 2-near Airport & EXPO	Like your own home, 24hrs ...	Quiet and view of the play...	https://a0.muscache.com/pi...	367042
275343	https://www.airbnb.com/roo...	20221229070856	2022-12-29	city scrape	Amazing Room with window 1...	Awesome location and host ...	NaN	https://a0.muscache.com/pi...	1439258
275344	https://www.airbnb.com/roo...	20221229070856	2022-12-29	city scrape	15 mins to Outram MRT Sing...	Lovely home for the specia...	Bus stop Food center...	https://a0.muscache.com/pi...	1439258

airbnb

Size of Dataset

3037 Rows
75 Columns

2. Data Preparation

OpenStreetMap spatial data

```
city = gpd.read_file('data/singapore.geojson')
city.head()
```

✓ 6.6s

	osm_id	name	ref	amenity	highway	shop	leisure	railway	other_tags	geometry
0	25451918	NaN	2A	NaN	motorway_junction	NaN	NaN	NaN	NaN	POINT (103.97394 1.32757)
1	25455287	NaN	14A	NaN	motorway_junction	NaN	NaN	NaN	NaN	POINT (103.87400 1.29544)
2	25455292	NaN	10B	NaN	motorway_junction	NaN	NaN	NaN	NaN	POINT (103.90850 1.30101)
3	25455304	NaN	1	NaN	motorway_junction	NaN	NaN	NaN	NaN	POINT (103.97848 1.33468)
4	26778790	NaN	NaN	NaN	traffic_signals	NaN	NaN	NaN	NaN	POINT (103.85002 1.30152)

airbnb

Airbnb Reviews data

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	71609	793880	2011-12-19	1456140	Max	The rooms were clean and tidy. Beds very comfo...
1	71609	1731810	2012-07-17	1804182	Zac	Good space and quite an interesting home in a ...
2	71609	2162194	2012-09-01	3113461	Zahra	It was a comfortable place. Belinda was a kind...
3	71609	2190615	2012-09-04	1432123	Helmut	We are four mature age travellers and stayed f...
4	71609	3221837	2013-01-02	2759938	Jack	Belinda is a great host! She helps you wheneve...

2. Data Preparation

01
Useful Data

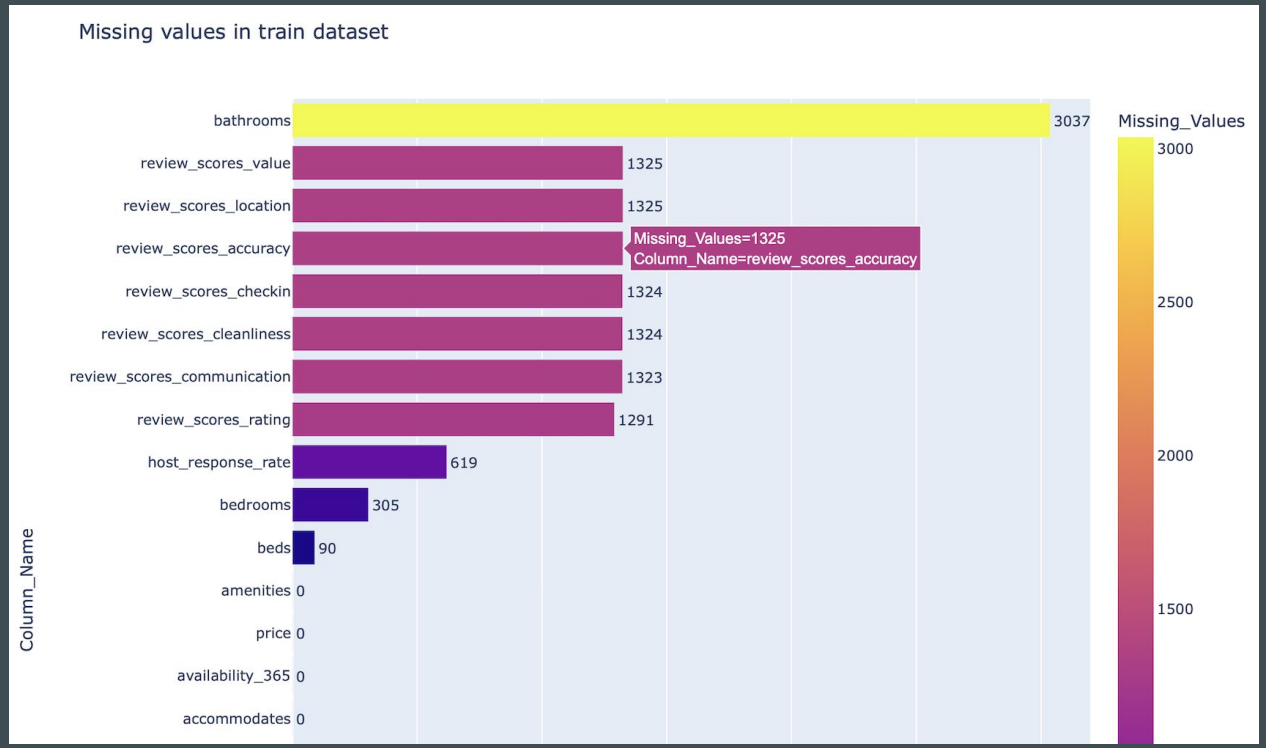
- Dealing with Missing Values

02
Not Useful Data

- Drop Unnecessary Columns
- Feature Engineering

2. Data Preparation

Missing Values



Bathrooms

Drop this empty column

The Rest

Replaced with median

2. Data Preparation

Remove outliers of price

Exclude outliers to prevent confusing the models

Trim outliers and $0 rental prices

```
# Remove $0 rental
df = df.query('price != 0')
# Remove outliers
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df['price'] < (Q1 - 1.5 * IQR)) | (df['price'] > (Q3 + 1.5 * IQR)))]
df[['price']].head(n=2772)
```

	price
1	81.0
2	81.0
3	52.0
4	49.0
5	175.0
...	...
3029	45.0
3030	45.0
3031	186.0
3034	83.0
3035	434.0

2772 rows × 1 columns

2. Data Preparation

Change date format

Extract the number of years

	host_since	host_since_years
0	2011-01-29	12
1	2011-01-29	12
2	2011-01-29	12
3	2011-11-24	12
4	2011-11-24	12
...	...	...
3032	2022-10-22	1
3033	2018-05-06	5
3034	2012-04-25	11
3035	2020-04-10	3
3036	2022-12-23	1

3037 rows × 2 columns

2. Data Preparation

Categorical Data

One-Hot Encoding

Separate categorical data to more columns.

⇒ Replaced with 0 & 1

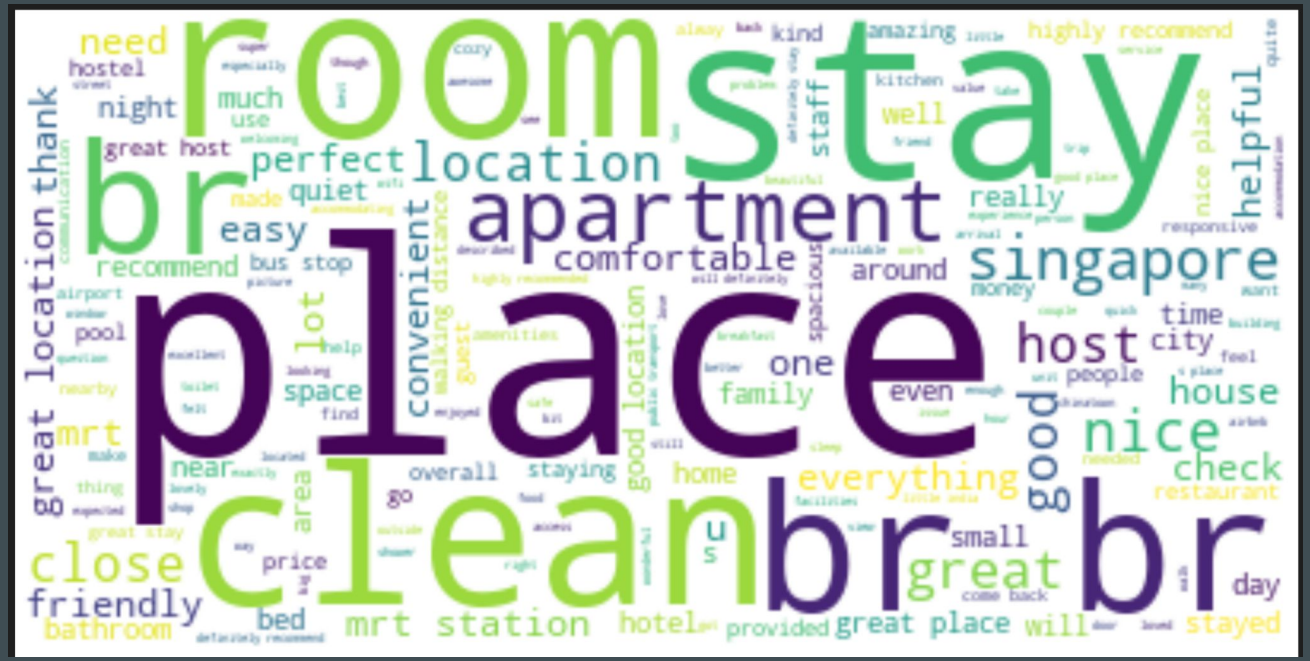
```
col = ['neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'property_type', 'room_type']
df[col] = df[col].astype('category')

transformer = make_column_transformer(
    (
        OneHotEncoder(),
        make_column_selector(dtype_include="category")
    ), remainder='passthrough', verbose_feature_names_out=False)
transformed = transformer.fit_transform(df)
df = pd.DataFrame(transformed, columns=transformer.get_feature_names_out())
df[['property_type_Apartment','property_type_Hotel','property_type_House','property_type_Other']].head(n=len(df))
```

	property_type_Apartment	property_type_Hotel	property_type_House	property_type_Other
0	0.0	0.0	1.0	0.0
1	0.0	0.0	1.0	0.0
2	1.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0
4	0.0	0.0	1.0	0.0
...	...	...	...	...
2767	0.0	1.0	0.0	0.0
2768	0.0	0.0	0.0	1.0
2769	0.0	1.0	0.0	0.0
2770	1.0	0.0	0.0	0.0
2771	0.0	1.0	0.0	0.0

2772 rows × 4 columns

3. Exploratory Analysis

Insights: Words that frequently appear in reviews



Features that appear in columns

- MRT
- Bedrooms

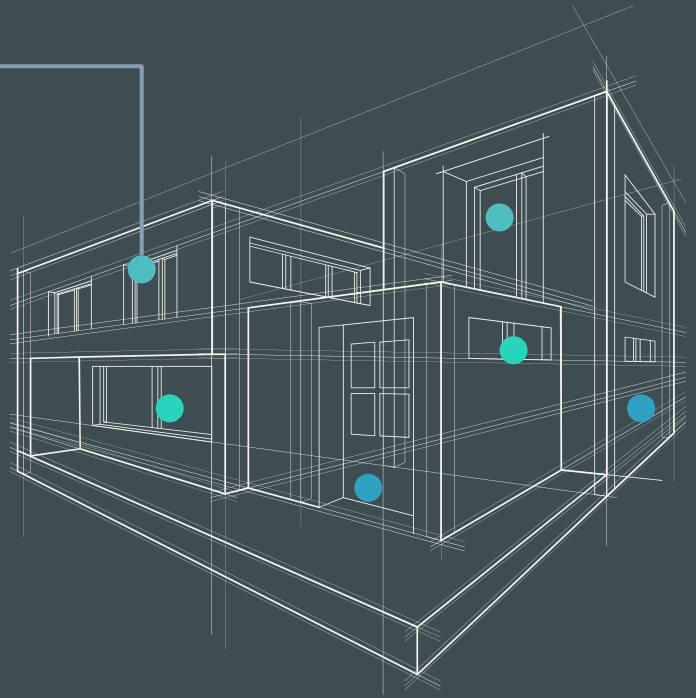
Features that appear in Amenities column

- Pool
- Kitchen
- Safe
- Wifi
- Breakfast

3. Exploratory Analysis

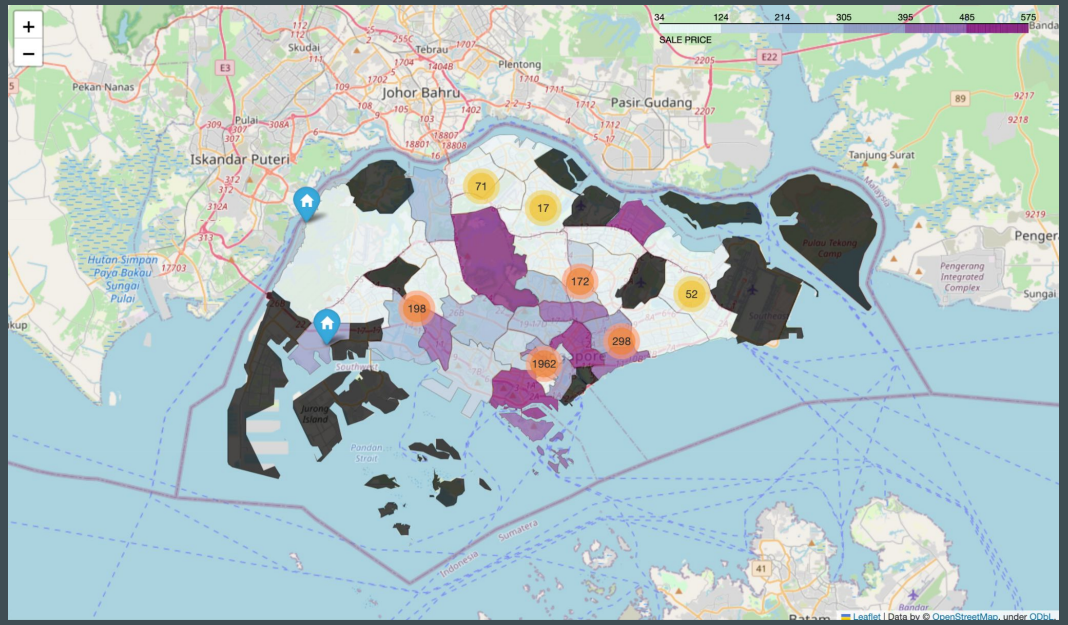
1. Locations

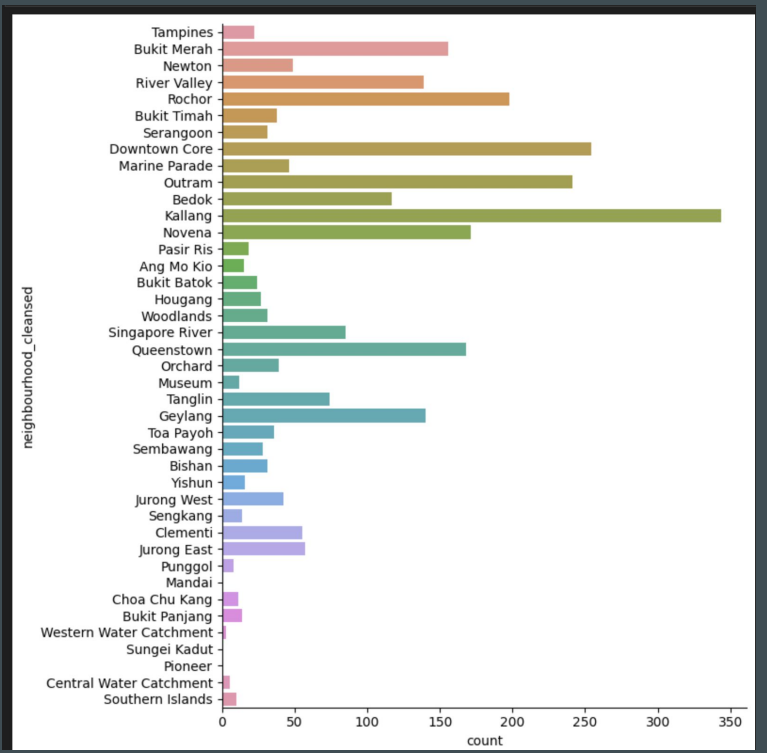
K Mean Clustering



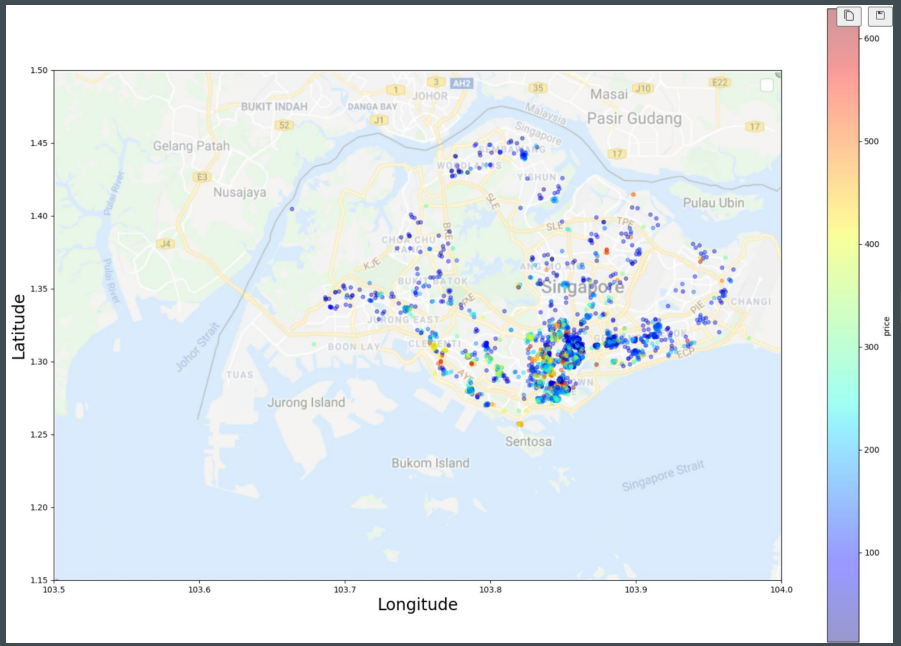
3. Exploratory Analysis

The data for each location is imbalance

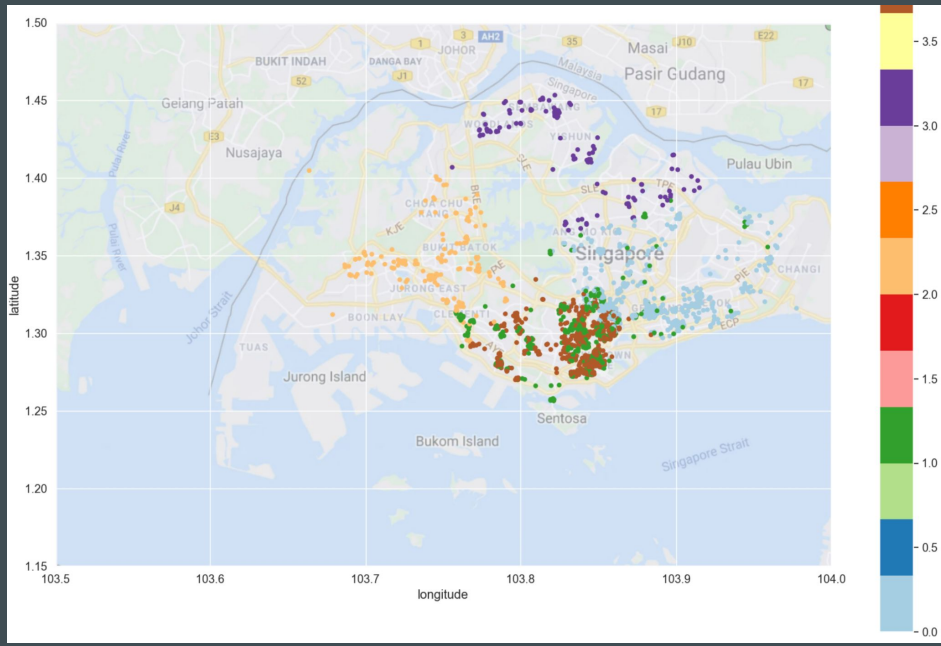




3. Exploratory Analysis

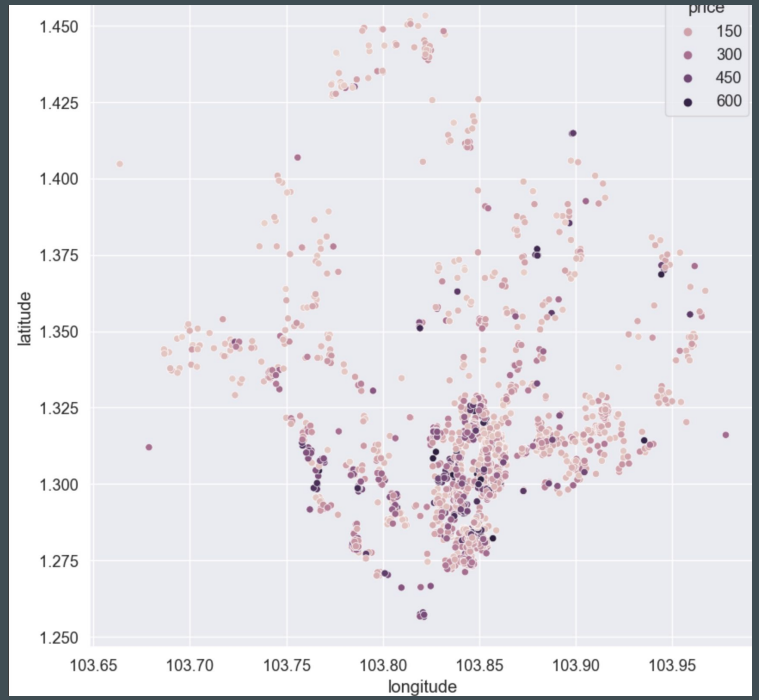


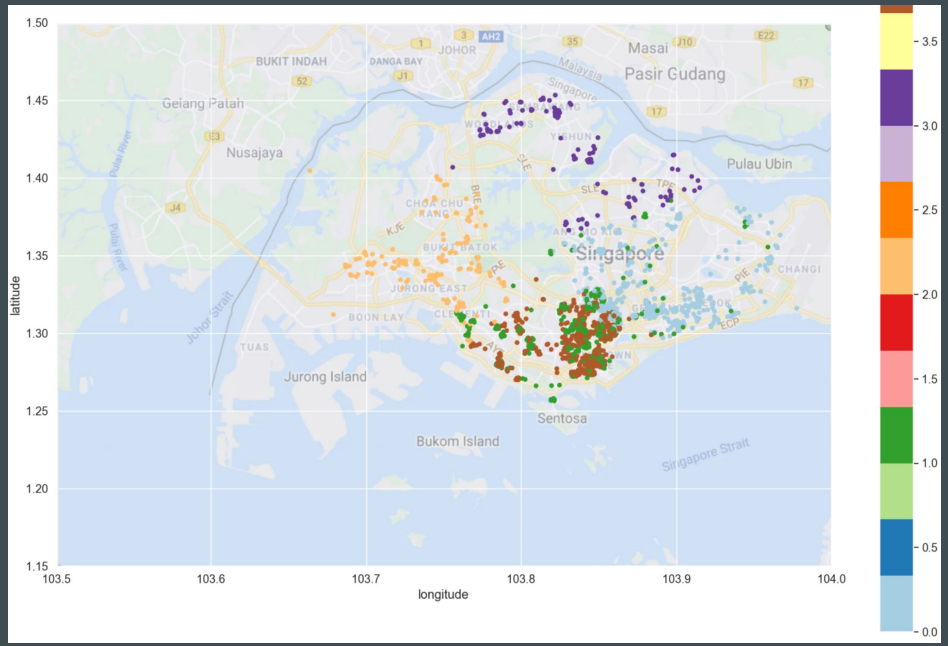
K mean Clustering: Create new regions based on Longitude, Latitude and Price



3. Exploratory Analysis

K mean clustering is accurate

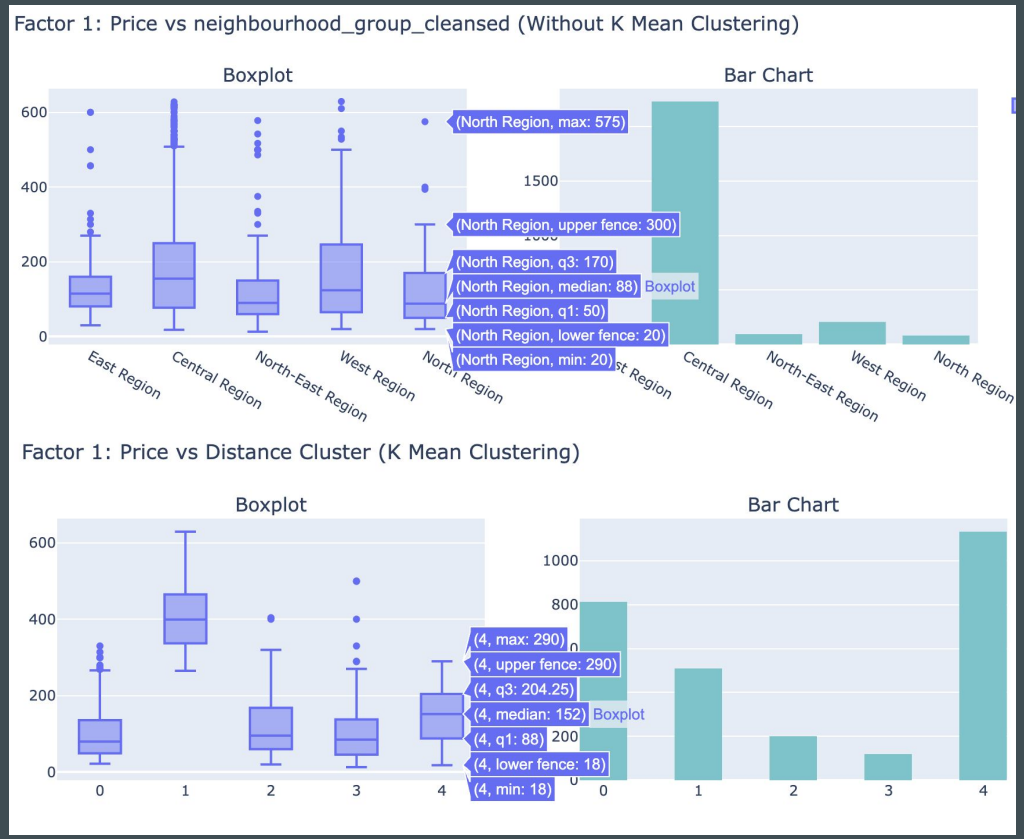




3. Exploratory Analysis

neighbourhood_group_cleansed	neighbourhood_group_cleansed	price
Central Region	2231	155.000000
West Region	207	124.000000
East Region	157	115.000000
North-East Region	95	90.000000
North Region	82	88.000000

distanceCluster	distanceCluster	price
1	510	399.000000
4	1133	152.000000
2	198	95.500000
3	119	85.000000
0	812	80.000000



3. Exploratory Analysis

1. Locations

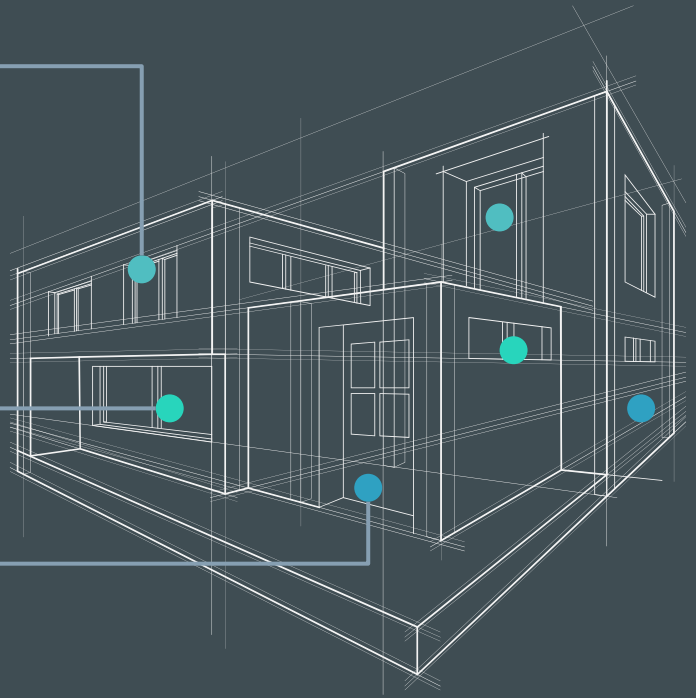
K Mean Clustering

2. Property Type

Reduce features for column

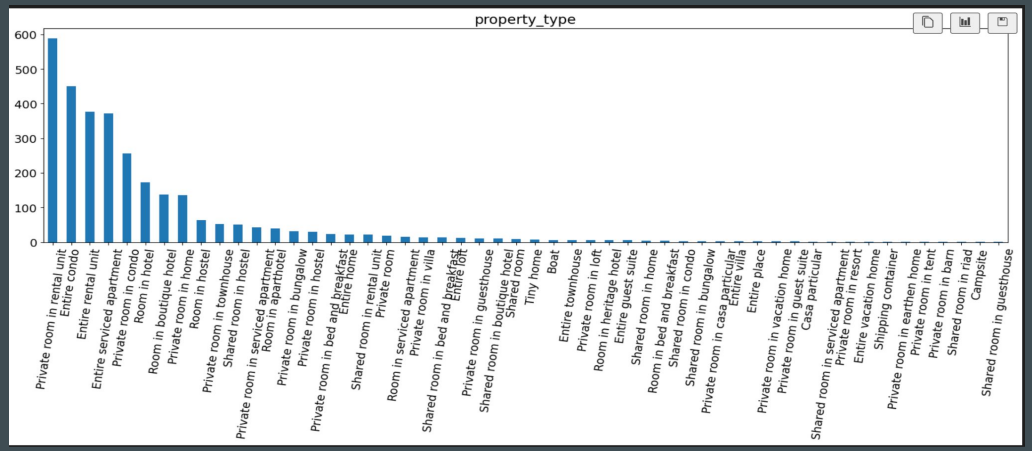
3. Room Type

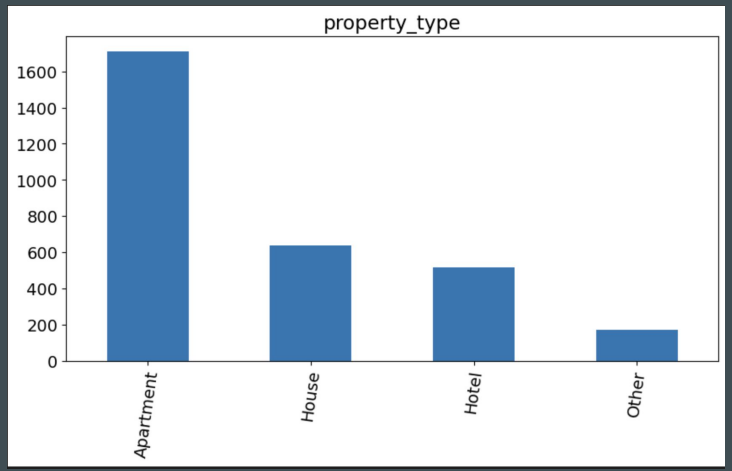
Correlation Matrix



3. Exploratory Analysis

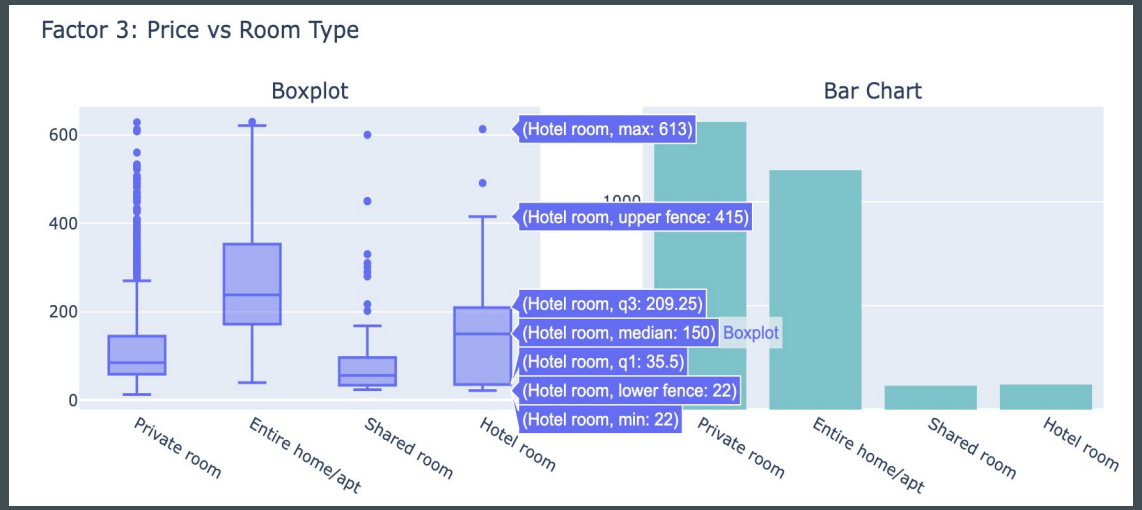
Property Type: Imbalance Data



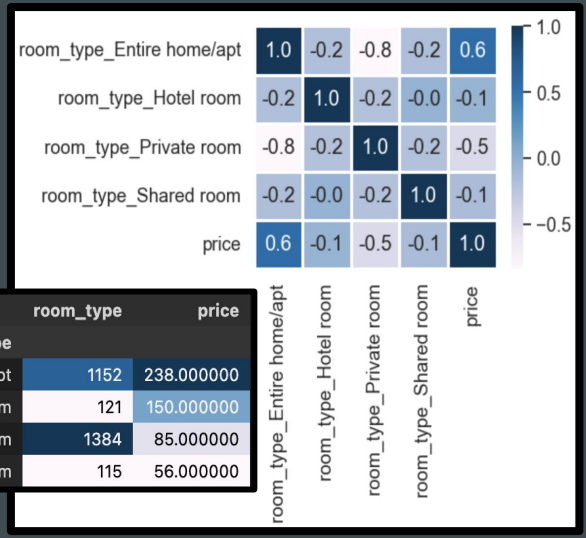


3. Exploratory Analysis

Room Type correlation



room_type	room_type	price
Entire home/apt	1152	238.000000
Hotel room	121	150.000000
Private room	1384	85.000000
Shared room	115	56.000000



3. Exploratory Analysis

1. Locations

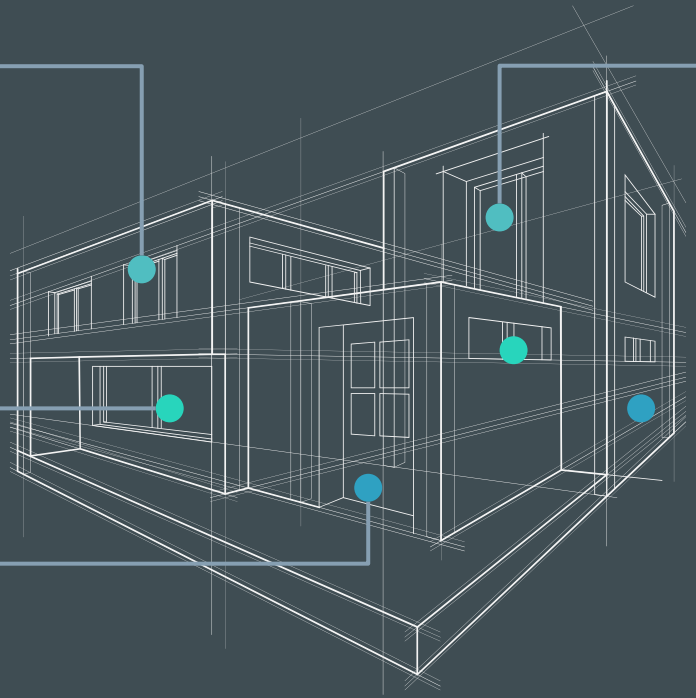
K Mean Clustering

2. Property Type

Reduce features for columns

3. Room Type

Correlation Matrix



4. Amenities

Reduce features by using Correlation heatmap

3. Exploratory Analysis

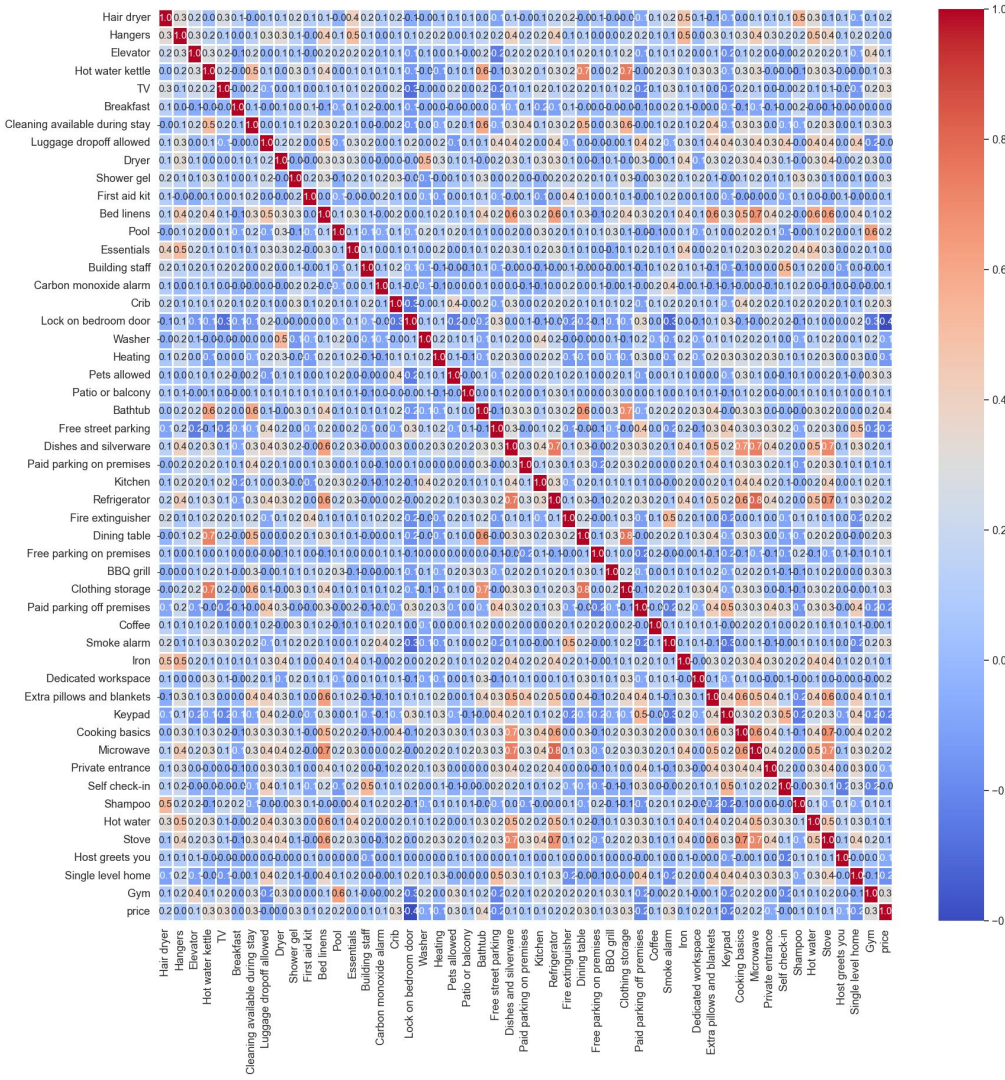
Amenities correlation

Drop the columns with corr < 0.15 & corr>-0.15.

+ Code

```
# calculate correlation matrix with price column
corr = df_showAmmenities.corr()['price']

low_corr_cols = corr[(corr < 0.15)&(corr > -0.15)].index.tolist()
print("Low correlation columns: ", low_corr_cols)
print(len(low_corr_cols))
```

```
Low correlation columns:  ['Hangers', 'Elevator', 'Breakfast', 'Luggage dro
25
```



3. Exploratory Analysis

1. Locations

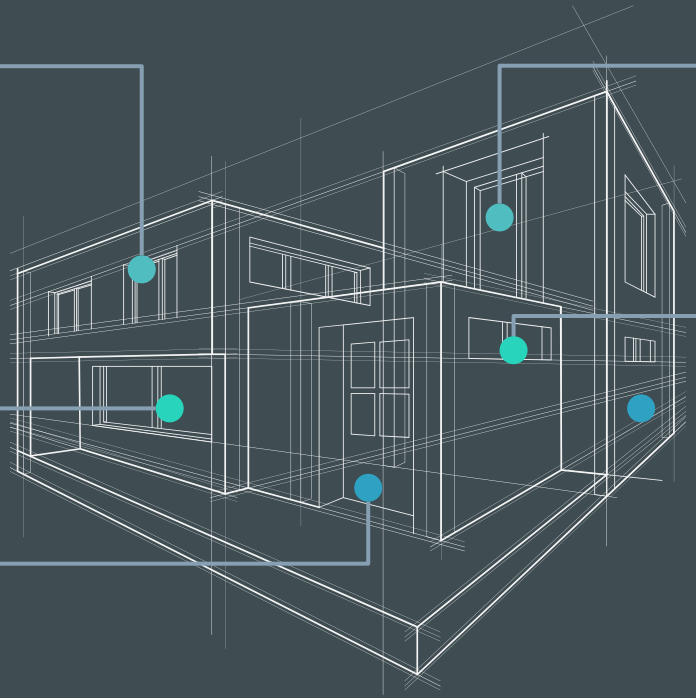
Latitude & Longitude
One-Hot Encoding
K Mean Clustering

2. Property Type

Reduce features for column

3. Room Type

Correlation Matrix



4. Amenities

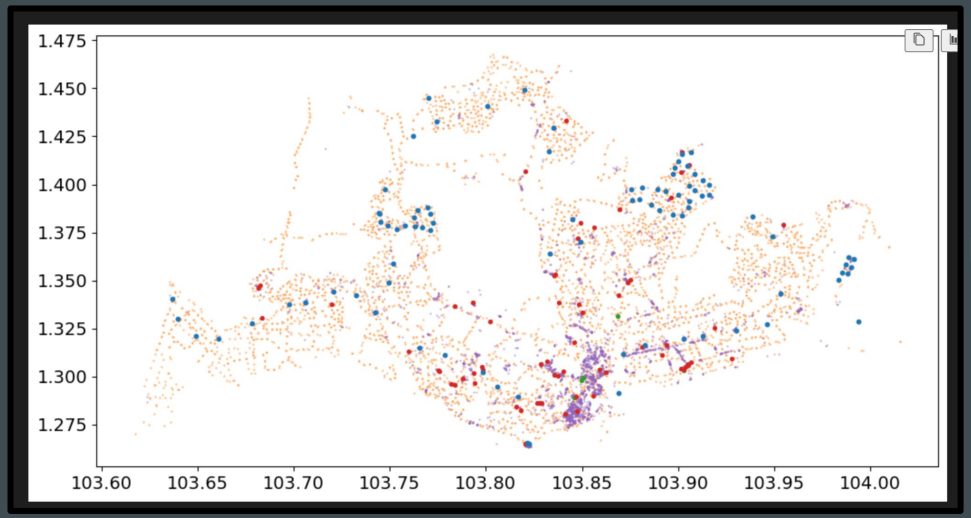
Reduce features by using Correlation heatmap

5. Facility

Distance from Key Locations

3. Exploratory Analysis

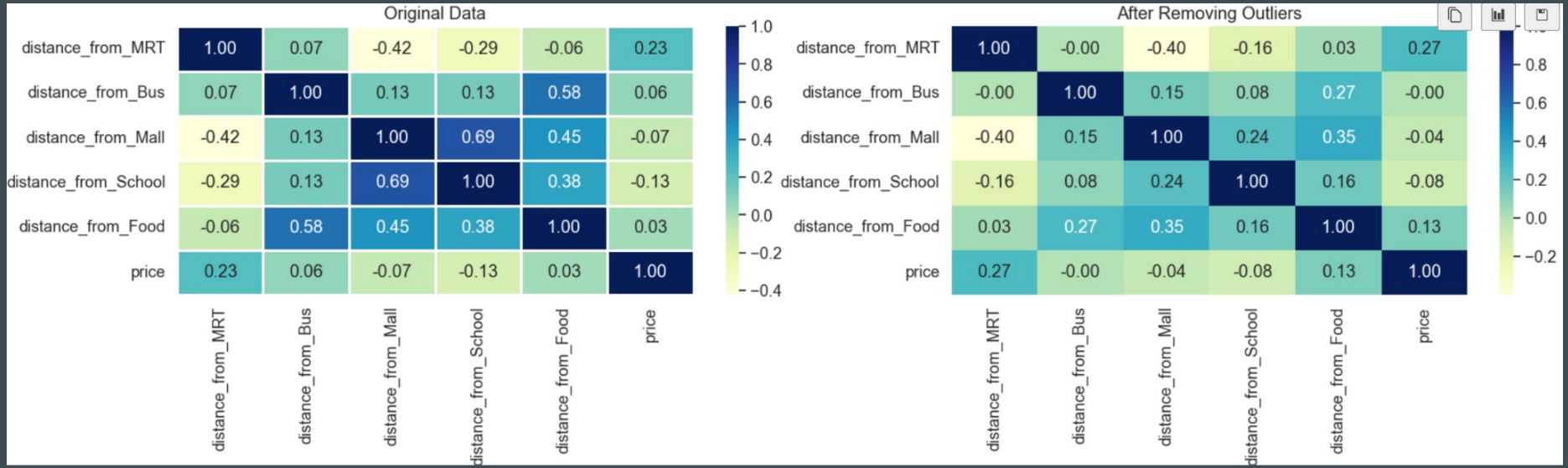
- **Collected Longitude and Latitude amenities data from OpenStreetMap**
- **Make use of AirBnB locations to calculate the distance away from the amenities**



	distance_from_MRT	distance_from_Bus	distance_from_Mall	distance_from_School
0	0.842953	0.180102	7.976199	3.515767
1	0.880282	0.205346	7.952045	3.786036
2	0.549557	0.117975	4.230675	1.095807
3	0.609339	0.040906	3.862544	0.685090
4	0.738518	0.210803	7.816646	3.809634
...	...	...	...	...
2767	2.442173	0.119477	0.213772	0.206431
2768	2.442173	0.119477	0.213772	0.206431
2769	2.442173	0.119477	0.213772	0.206431
2770	0.895359	0.130348	3.099577	0.759436
2771	0.880842	0.919777	4.480271	0.863616

2772 rows × 4 columns

3. Exploratory Analysis



3. Exploratory Analysis

1. Locations

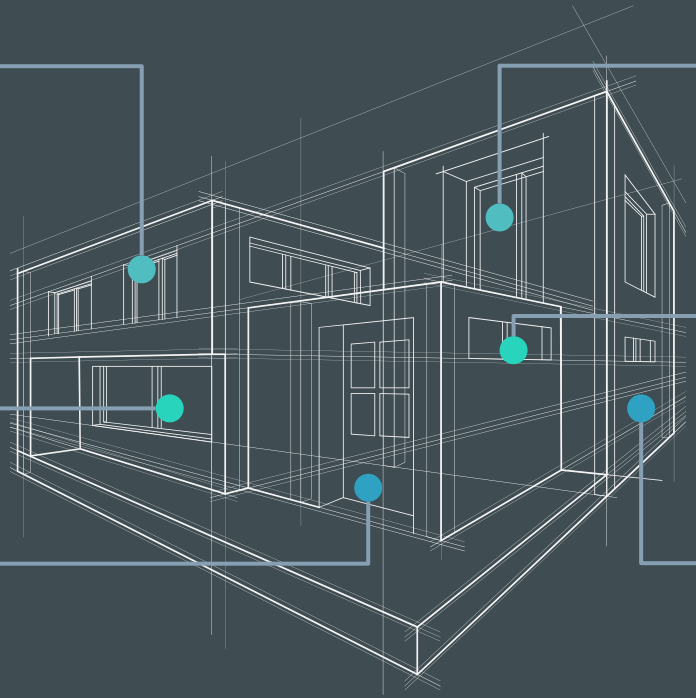
K Mean Clustering

2. Property Type

Reduce features for columns

3. Room Type

Correlation Matrix



4. Amenities

Reduce features by using Correlation heatmap

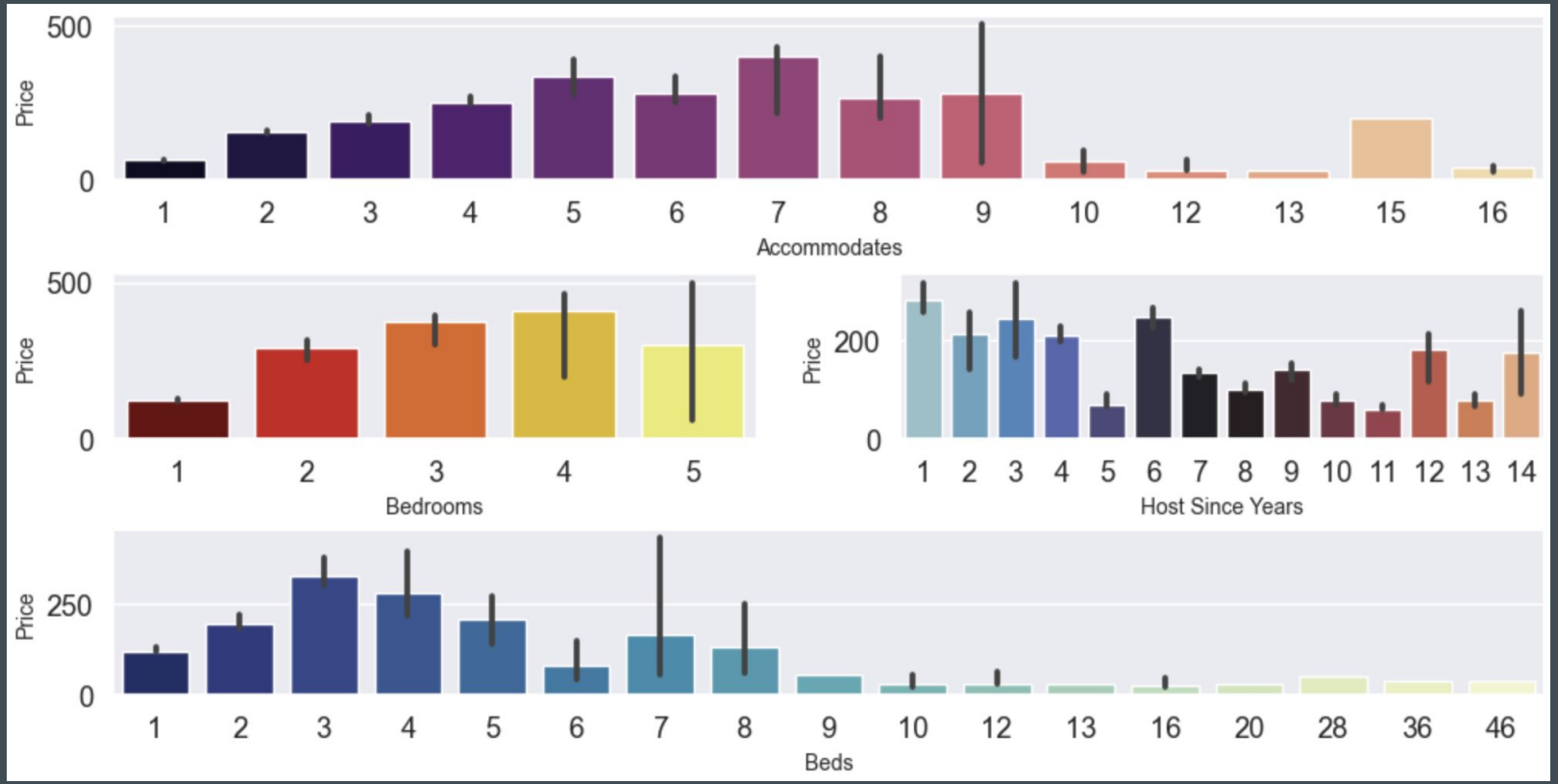
5. Facility

Distance from Key Locations

6. Other Factors

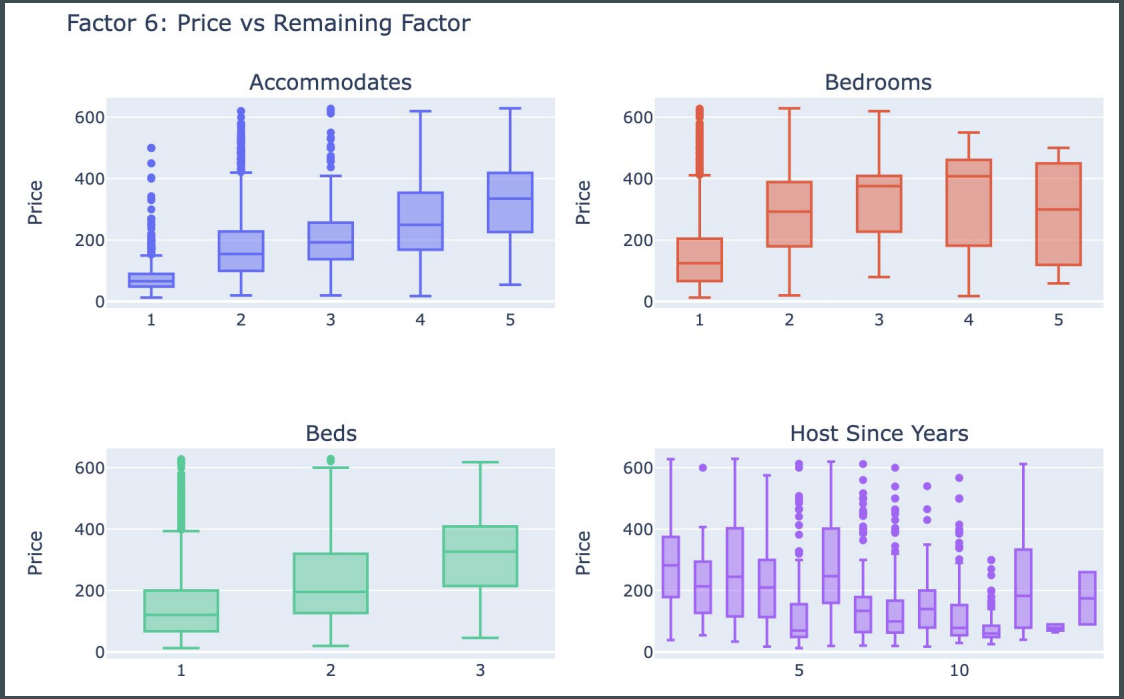
Numeric Data:
Discrete

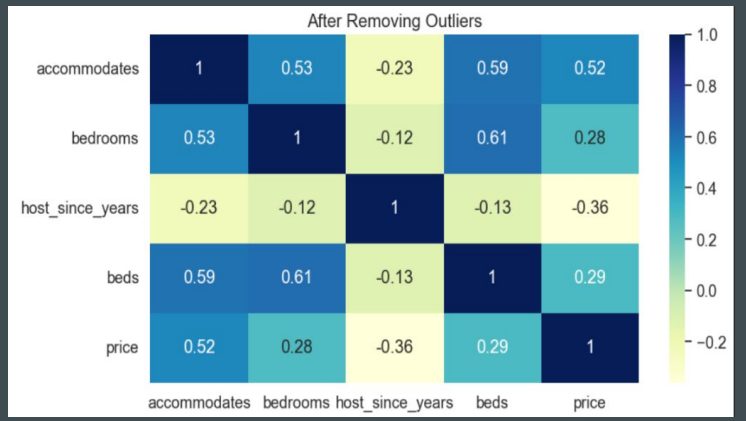
3. Exploratory Analysis



3. Exploratory Analysis

- **Remove Outliers**





3. Exploratory Analysis

1. Locations

K Mean Clustering

2. Property Type

Reduce features from columns

3. Room Type

Correlation Matrix

4. Amenities

Reduce features by using correlation map

5. Facility

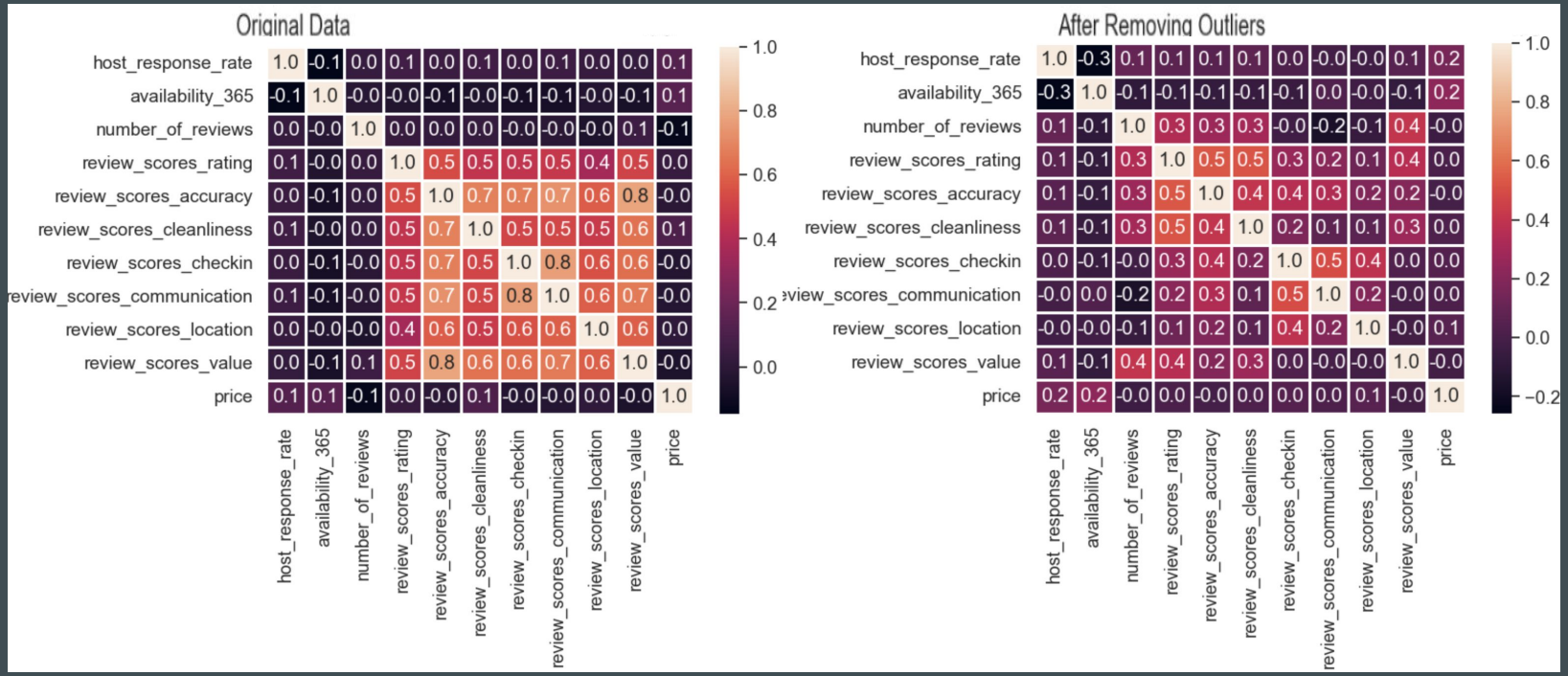
Distance from Key Locations

6. Other Factors

Numeric Data:
Discrete
Continuous

3. Exploratory Analysis

3. Exploratory Analysis



Drop columns that have no correlation with price

Reduced 75 to 16 columns (Exclude one hot encoding)	
Property Type	number_of_reviews
Room Type	distance_from_mrt
Host_response time	distance_from_bus
accommodates	distance_from_mall
amenities	distance_from_school
bedrooms	distance_from_food
beds	Distance Cluster
price	
availability_365	

2. Data Preparation

Connectivity

Saturn is the ringed one. It’s composed of hydrogen and helium

Comfort

Earth is the third planet from the Sun in the Solar System

Privacy

Neptune is the farthest planet from the Sun

Services

Uranus is the seventh one from the Sun. It’s the coldest

3. Exploratory Analysis

Unpack amenities (One-Hot Encoding)

```
amenities_set = set()
# # Creating a set of all possible amenities
amenities_json = df.amenities.apply(json.loads)
amenities_list = ["|".join(x) for x in list(amenities_json)]

for amenities in amenities_list:
    for amenity in amenities.split(sep="|"):
        amenities_set.add(amenity)

pd.Series(list(amenities_set))
```

```
0      42" HDTV with Netflix, sta...
1                           Hot tub
2              Sound system with aux
3       Bose bluetooth speaker  B...
4        Singapore Brand's body soap
                   ...
444    Shared hot tub - available...
445                             Gym
446       50" TV with standard cable
447    Coffee maker: espresso mac...
448             Children's dinnerware
Length: 449, dtype: object
```

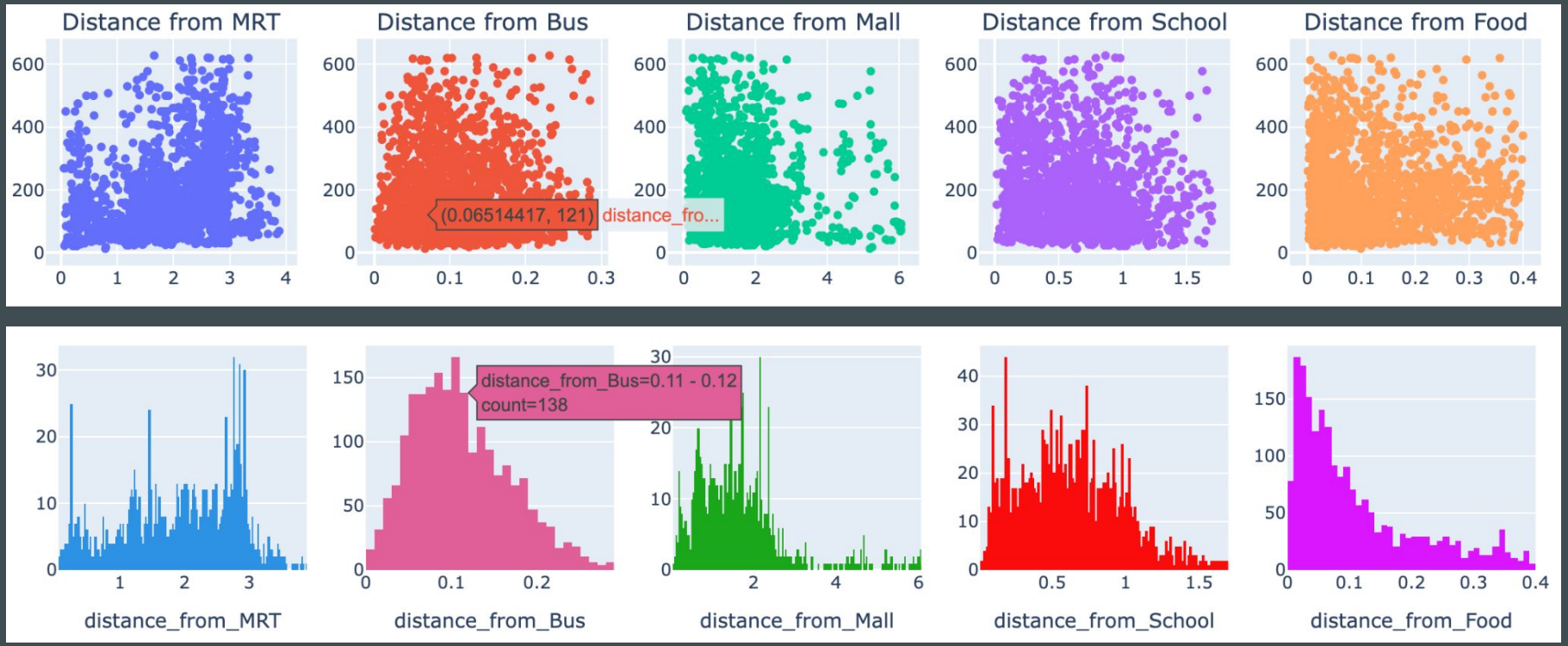
Remove mostly empty columns (from amenities encoding)

Any column that has mostly true or mostly false values can be treated as a constant, and hence dropped.

```
# A filter to remove all columns with > 90% false values or > 90% true values (constant).
to_be_removed = []

for col in df.iloc[:, remember_row:].columns:
    if (df[col].mean() > 0.9) or (df[col].mean() < 0.1):
        to_be_removed.append(col)
display(pd.Series(to_be_removed))
print(f"Number of cols removed {len(to_be_removed)}")

df.drop(columns=to_be_removed, axis=1, inplace=True)
```

```
0      42" HDTV with Netflix, sta...
1                           Hot tub
2              Sound system with aux
3       Bose bluetooth speaker  B...
4        Singapore Brand's body soap
                   ...
391                 Dryer – In unit
392    Shared hot tub - available...
393       50" TV with standard cable
394    Coffee maker: espresso mac...
395             Children's dinnerware
Length: 396, dtype: object

Number of cols removed 396
```

3. Exploratory Analysis



3. Exploratory Analysis

```
df = pd.read_csv('data/clean_listingfinal.csv', index_col=0)
df.head(n=len(df))
```

✓ 0.0s Python

	property_type_Apartment	property_type_Hotel	property_type_House	property_type_Other	room_type_Entire home/apt	room_type_Hotel room	room_type_Private room	room_type_Shared room	host_response_rate	accommodates	...	C
0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.90	1	...	
1	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.90	2	...	
2	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.00	1	...	
3	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.00	1	...	
4	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.90	4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
2767	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.97	1	...	
2768	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.97	1	...	
2769	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.97	4	...	
2770	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.98	2	...	
2771	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.00	4	...	

2772 rows × 48 columns

Complete Data Cleaning and Exploration

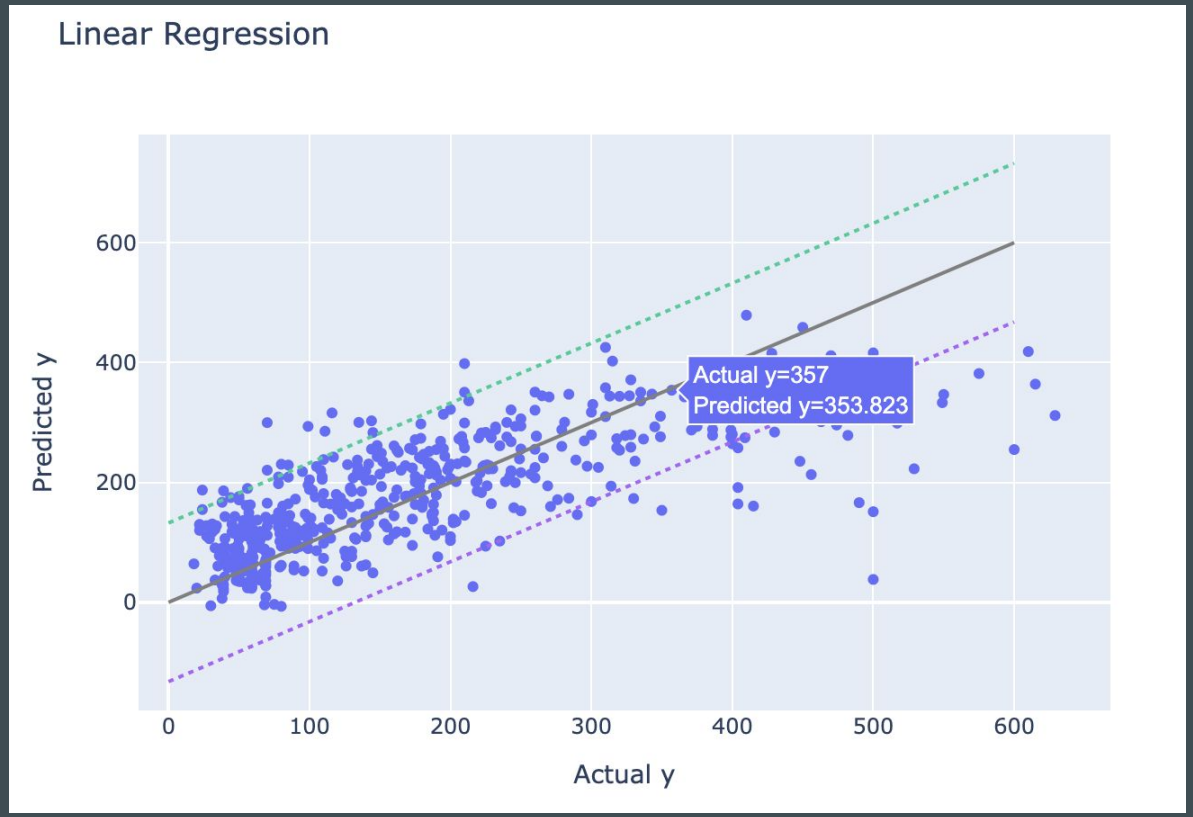
4. Machine Learning

- Linear Regression
- Gradient Boosting Regressor
- XGBoost Regression
- Neural Networks

4. Machine Learning

	Implementation	Goal
Linear Regression	Ordinary least square	Minimise residual sums of squares
Gradient Boosting Regression / XGBoost Regression	Boosting: base estimators built sequentially Gradient: differential loss function	Minimise loss function (square error)
Multi-layer perceptron Regression	Hidden layers with weights and biases, trained using backpropagation.	Minimise loss function (square error)

4. Machine Learning



Train Score: 0.562
Train RMSE: 88.195

Test Score: 0.579
Test RMSE: 83.287

4. Machine Learning

Train Score: 1.000
Train RMSE: 1.450

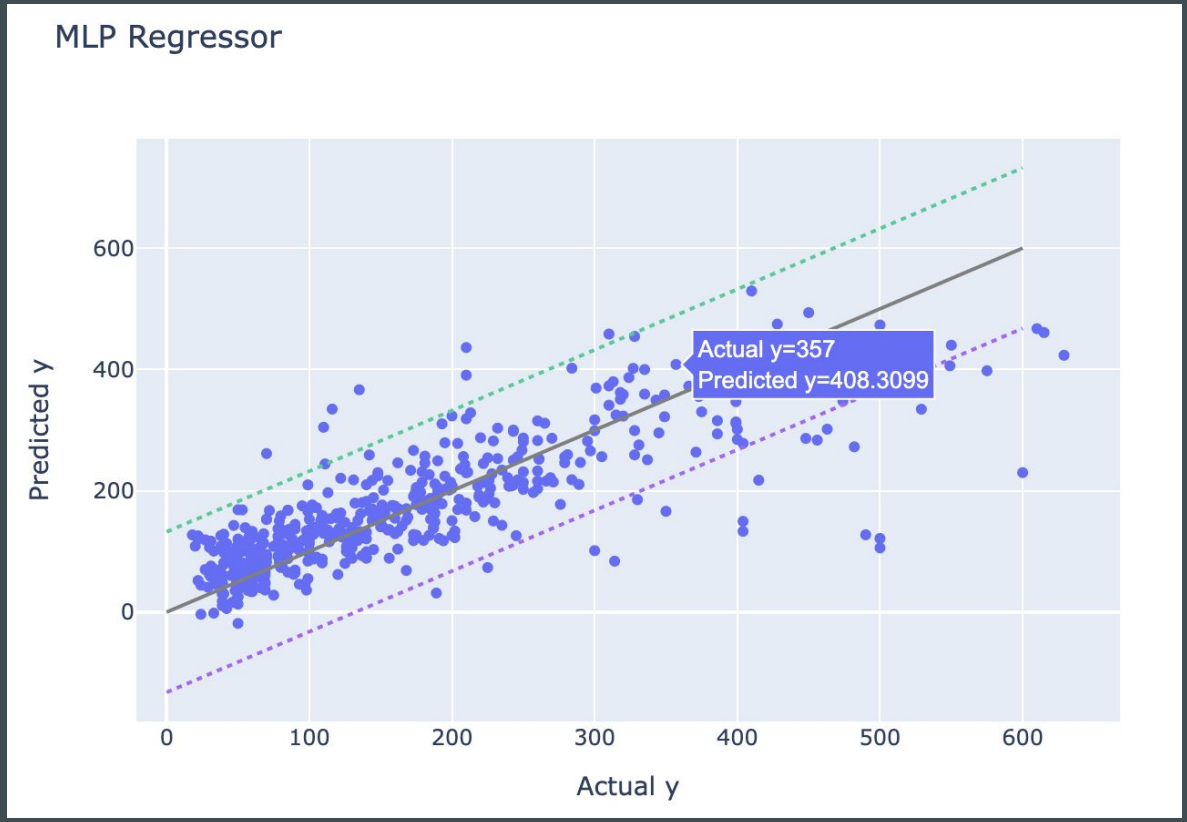
Test Score: 0.850
Test RMSE: 49.688

4. Machine Learning

Train Score: 0.994
Train RMSE: 10.340

Test Score: 0.846
Test RMSE: 50.425

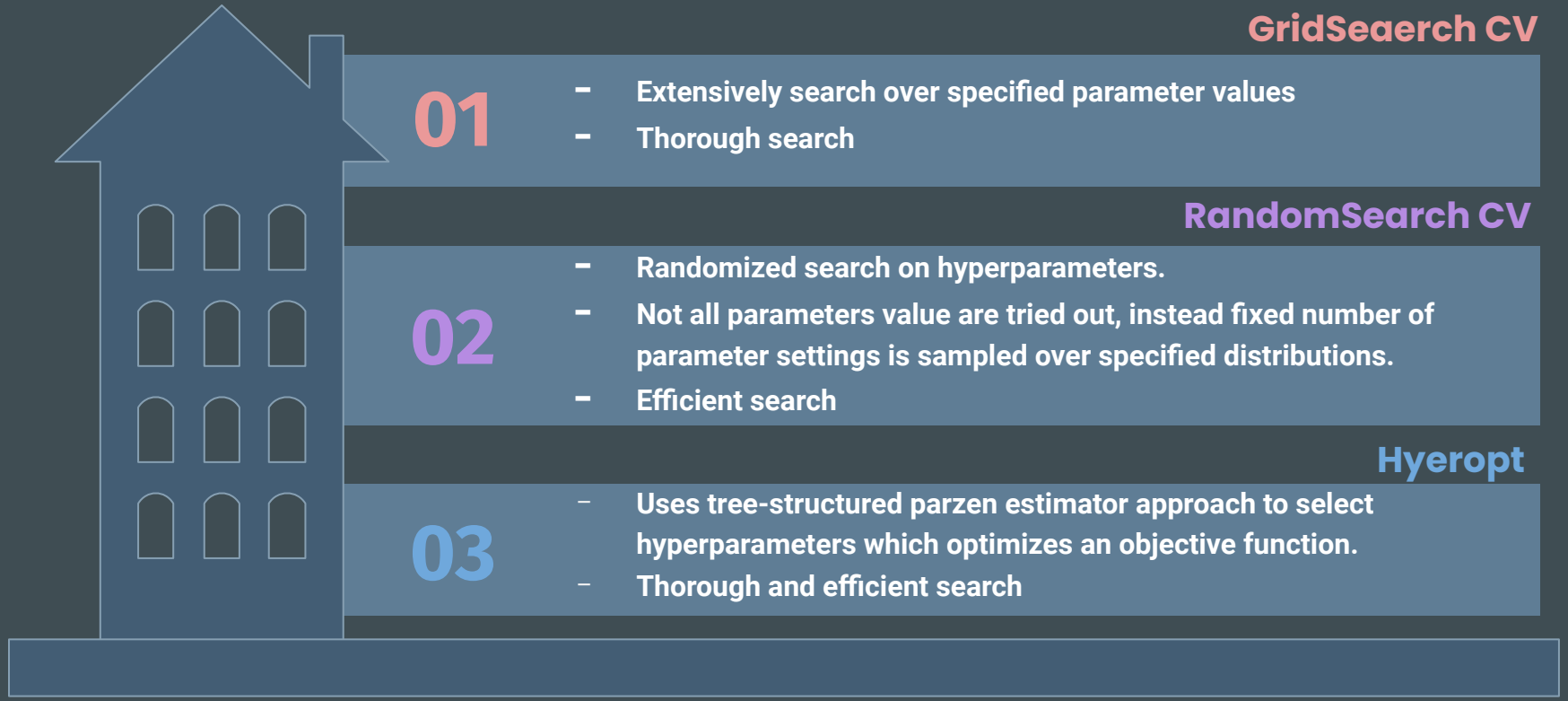
4. Machine Learning



Train Score: 0.755
Train RMSE: 65.978

Test Score: 0.703
Test RMSE: 69.977

5. Optimization



GridSeaerch CV

01

- Extensively search over specified parameter values
- Thorough search

RandomSearch CV

02

- Randomized search on hyperparameters.
- Not all parameters value are tried out, instead fixed number of parameter settings is sampled over specified distributions.
- Efficient search

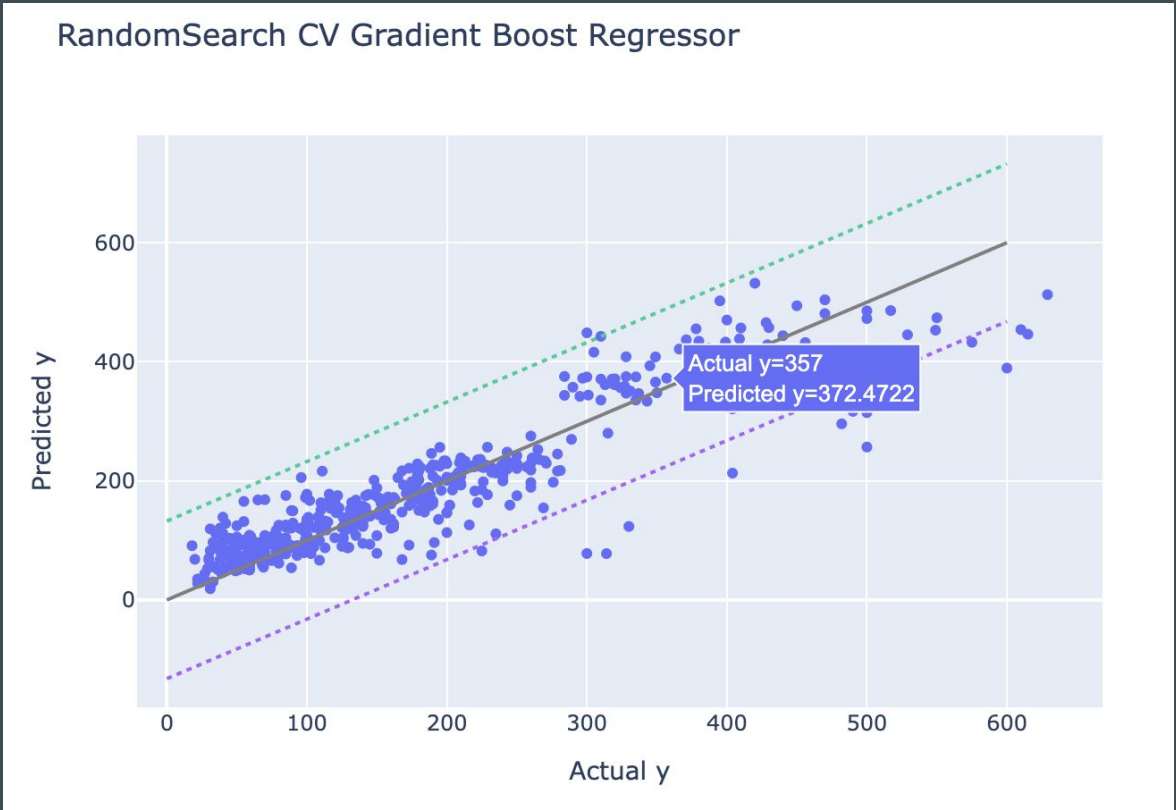
Hyeropt

03

- Uses tree-structured parzen estimator approach to select hyperparameters which optimizes an objective function.
- Thorough and efficient search

5. Optimization

Best parameters: {'learning_rate': 0.04917213242114807, 'max_bins': 14, 'max_leaf_nodes': 83, 'min_samples_leaf': 8}



Without CV:

Train Score: 1.000
Train RMSE: 1.387

Test Score: 0.860
Test RMSE: 48.055

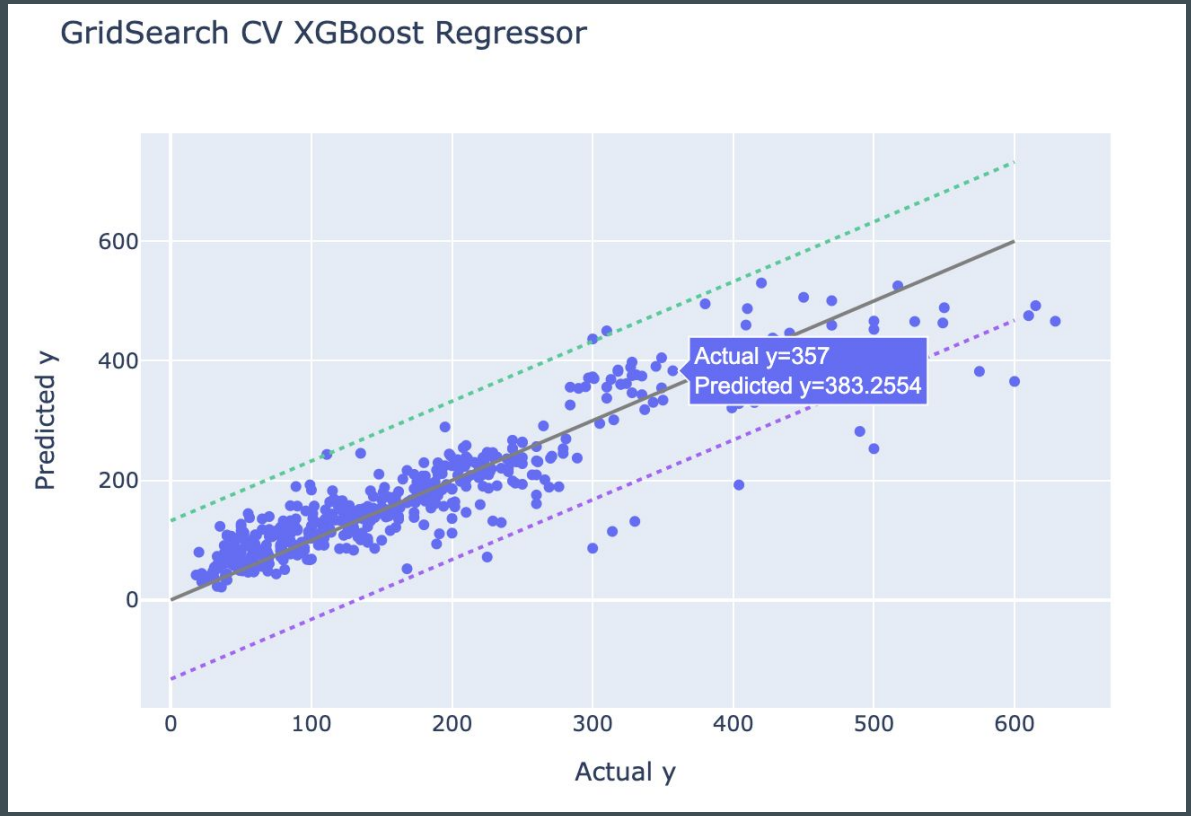
With CV:

Train Score: 0.978
Train RMSE: 19.716

Test Score: 0.873
Test RMSE: 45.802

5. Optimization

Best parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.03, 'max_leaves': 40, 'n_estimators': 1000, 'subsample': 1}



Without CV:

Train Score: 0.995
Train RMSE: 9.809

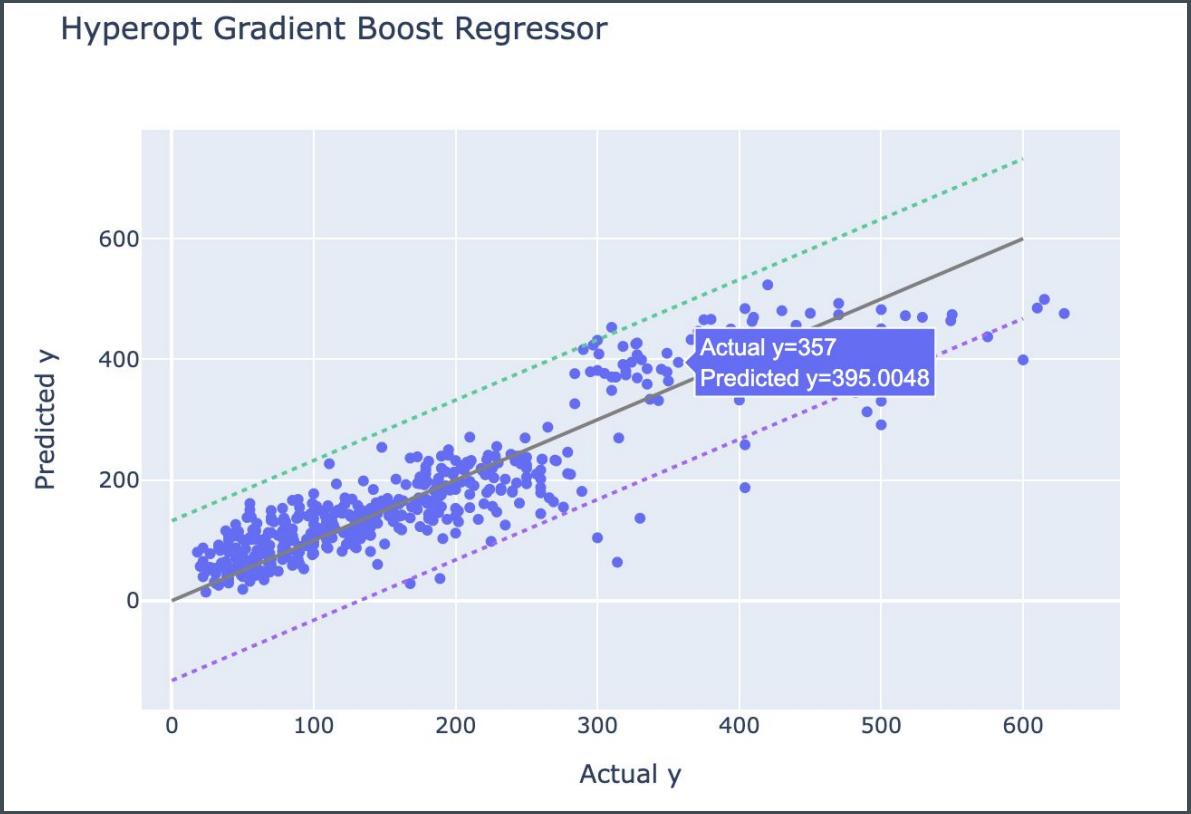
Test Score: 0.862
Test RMSE: 47.632

With CV:

Train Score: 0.995
Train RMSE: 9.827

Test Score: 0.867
Test RMSE: 46.738

5. Optimization



Without CV:

Train Score: 1.000
Train RMSE: 1.387

Test Score: 0.860
Test RMSE: 48.055

With CV:

Train Score: 0.869
Train RMSE: 48.209

Test Score: 0.848
Test RMSE: 50.002

5. Optimization

Without CV:

Train Score: 0.995
Train RMSE: 9.809

Test Score: 0.862
Test RMSE: 47.632

With CV:

Train Score: 0.998
Train RMSE: 5.990

Test Score: 0.875
Test RMSE: 45.358

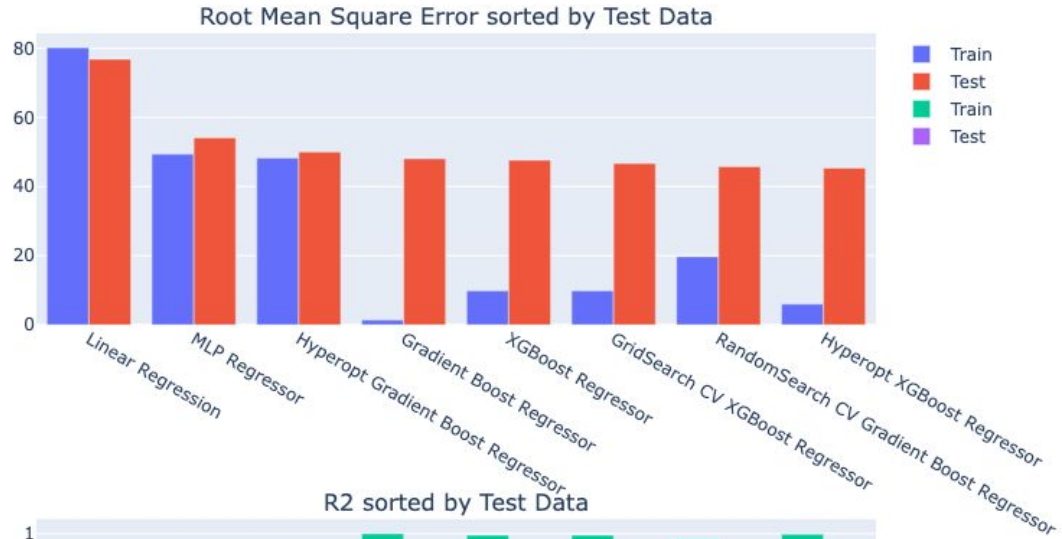
6. Ranking

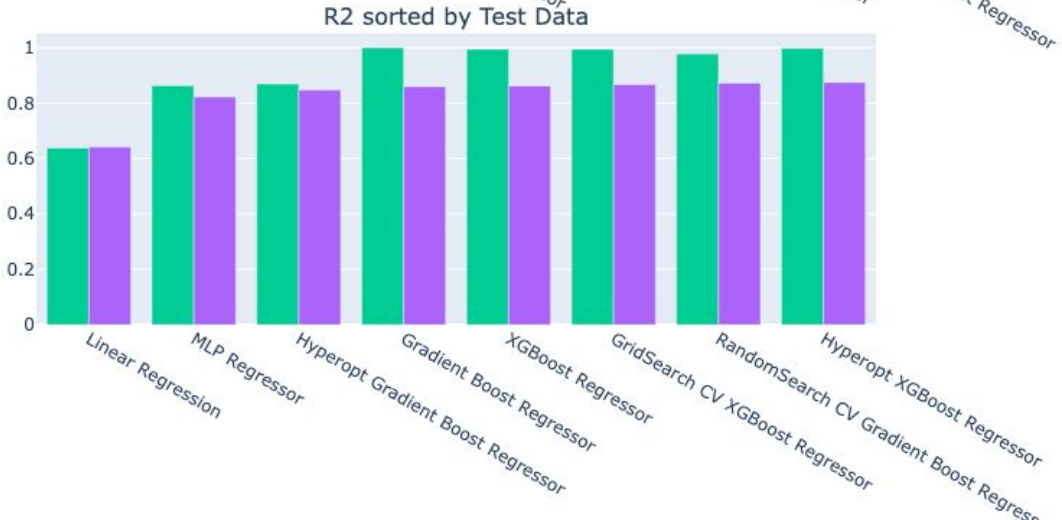
Least RMSE

Hyperopt XGBoost Regressor

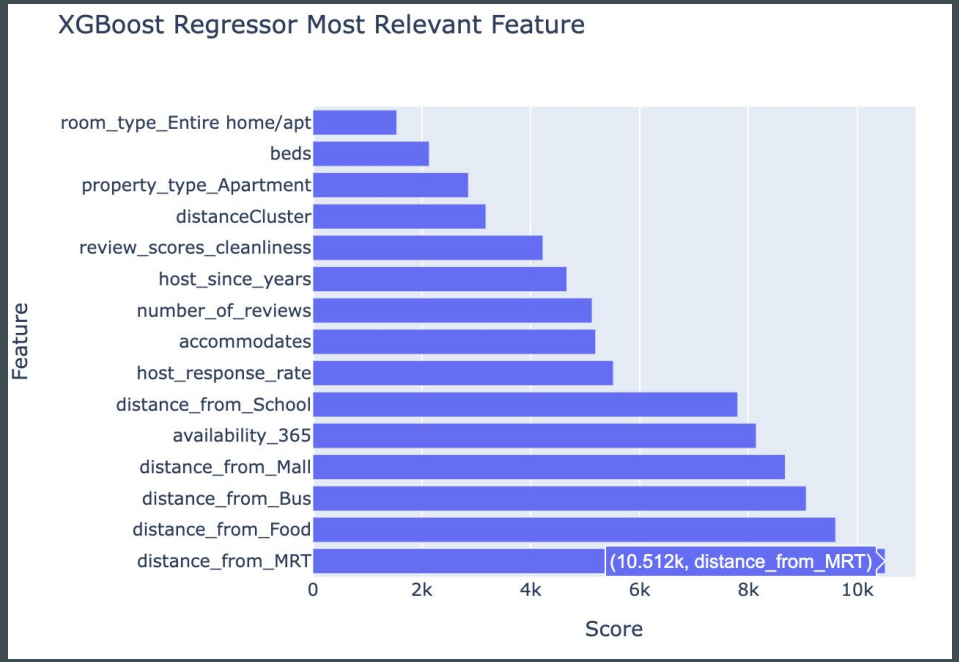
Best R^2

Hyperopt XGBoost Regressor





7. Insight



Most important feature

Location:
- Distance from MRT
- Distance from Bus
- Distance from Mall

Host trustworthiness:
- Availability
- Response rate
- Host since years

Listing features:
- Accommodates
- Property type
- Beds
- Room type

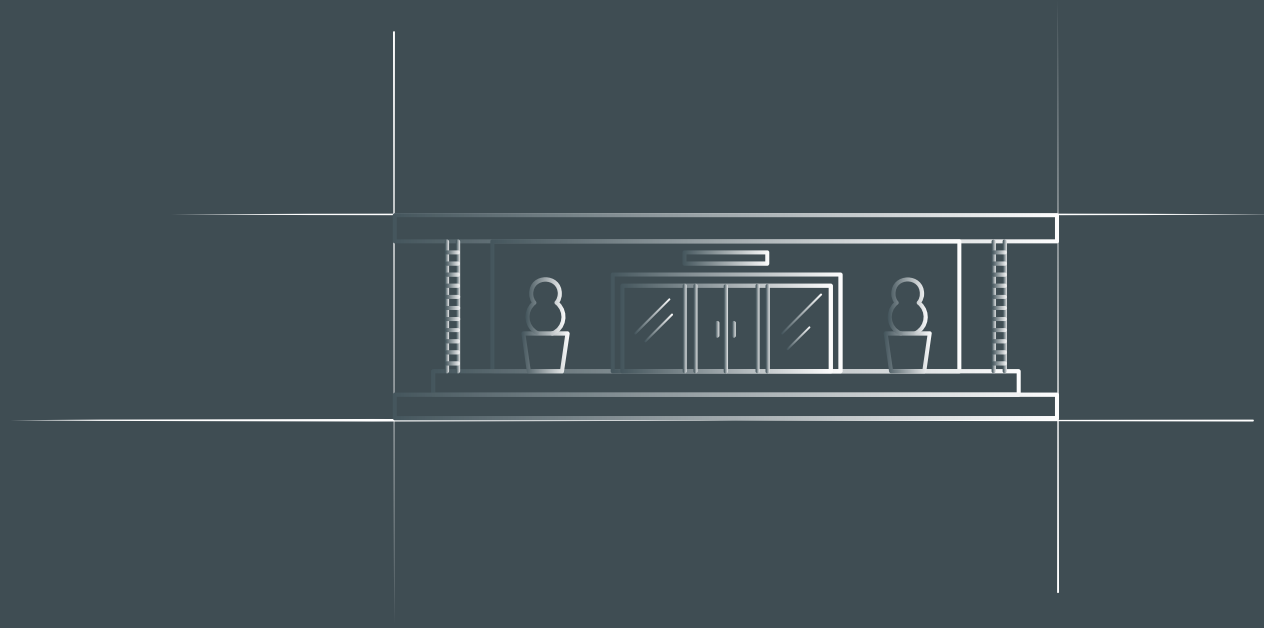
8. Conclusion

Did we manage to predict prices of listing?

Yes!

8. Conclusion

Did we manage to predict prices of listing?



THANK YOU!