

AIRBNB PRICE PREDICTION MODEL



MSIS 5223 PROGRAMMING FOR DATA SCIENCE

TEAM MEMBERS

Anudeep Subraveti20168818

Krithika Ganesh Kumar....20162108

Roshith Elangovan.....20161234

Under the guidance of

Dr. Bryan Hammer

CONTENTS:

1. Executive Summary.....	3
2. Statement of Scope.....	3
3. Project Schedule.....	4
4. Data Preparation.....	6
• Data Access.....	6
• Data Consolidation.....	6
• Data Cleaning.....	6
• Data Transformation.....	9
• Data Reduction.....	10
5. Descriptive Statistics.....	12
6. Data Dictionary.....	37
7. Modelling Techniques	38
8. Data splitting and Sub sampling.....	39
9. Model Building	39
• Linear Regression	41
• Clustering	42
• Association Rules	45
10. Evaluation	46
11. Problems and Constraint.....	47
12. Conclusion.....	48

EXECUTIVE SUMMARY:

Airbnb is an online rental market, where the owners can list their property for rent. Often, it is difficult for the owners to determine the listing price as it should be competitive and beneficial. This study will help the owners to value their property and set a price for the listing that is to be posted on Airbnb website. To address the problem, we propose to build a predictive model that helps the new host to value their property. The optimum price will be predicted by considering factors such as amenities (Wi-Fi, Gym, Kitchen, Tv), convenience to the nearest neighborhood facilities, property type and listing attributes. The end goal of this project is to let the Airbnb home providers set an optimum price for their asserts.

STATEMENT OF SCOPE:

Delen is a property owner in New York area who wants to rent his apartment on Airbnb website and is unsure of the price that he can rent his apartment for. As he checked the website for other listings, he found that the price varied, and was baffled.

1. PROJECT OBJECTIVES:

- This project aims to assist owners like Delen in determining the optimal price for the rental property.
- The input information for prediction should be facile and well understood by the owners.
- The price will be the target variable on predictor variables such as apartment specific amenities.
- Sample considered for this project is the listings and data available on Airbnb specific to New York City and the model will generalized for the use of any owner wanting to list his/her property on the website.

2. THE TARGET AND PREDICTOR VARIABLES:

The target variable for the model is Price coded as a continuous variable.

According to our hypothesis, two factors that mainly affect the price are:

- The amenities that are offered to the guest by the host. This can be considered as variety of amenities ranging from Internet to Hot tub. The rent of the house is likely to be higher with more amenities.
- The facilities located in the neighborhood of the listing. This can range from atm, restaurants to cafes. The more walkable a facility from the house, the more priced the house could be.

These two variables will serve as predictor variables in determining the price (target) variable.

The hosts are aware of all the amenities available at his/her house. The facilities located in the neighborhood will be located, calculated, consolidated using OpenStreetMaps API.

PROJECT SCHEDULE:

We anticipate the project to be completed in approximately 100 days. We are a group of 3 and we decided to work together on every task rather than splitting the task and working on it individually.

We met everyday for an hour to understand, clean and work on the data. Furthermore, we spent 2 to 3 hours every week to discuss on the progress and optimizations required to build an effective model.

During Spring break, we are planning to work individually on analyzing techniques involved in building the model. The analysis will include the following,

- Deeper understanding of the modelling techniques
- Come up with reasons for choosing modelling technique
- Based on the modelling technique understand if any specific data preprocessing / transformation required

After the break, we are planning to meet to further understand and analyze everyone's perspective and select predictive modelling technique that provides better solution for the business problem.

GANTT CHART:

Project Work	January				February				March				April				May
	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
1. Project Proposal – Executive Summary																	
2. Statement of Scope																	
3. Data Access																	
4. Data Consolidation																	
5. Data Cleaning																	
6. Data Transformation																	
7. Data Reduction																	
8. Data Dictionary																	
Finalization and Submission of Deliverable I																	
9. Adjust Deliverable I Requirements																	
10. Select Modelling Technique																	
11. Data Splitting & Sub-Samples																	
12. Build Model																	
13. Assess Model																	
14. Formatting, Style, Grammar, Spelling																	
Compile Deliverable I and II and Submit																	
15. Compile Final Report																	

- - Yet to be completed
- - Completed as a part of deliverable 2

DATA PREPARATION:

1. DATA ACCESS

Airbnb is a massive repository of data about prices and listings. New York region is chosen because it is heavily populated and is enriched with tourist attractions where we can find interesting patterns followed by hosts in pricing their listings. The data has been obtained from [Insideairbnb.com](https://www.insideairbnb.com) in the csv format for the year of 2018. This website is official and trusted source from Airbnb. It consisted of 49057 records with 96 variables.

2. DATA CONSOLIDATION

We worked on

- Understanding the variables and related it to the business problem
- Check whether the current data is enough for the analysis.

Since all the data that is needed is present in a single file. We considered only one csv file.

3. DATA CLEANING

From the data set we are considering only 23 variables and the others are removed. Total number of missing values in the 23 variables is 235. There is 0.5% of missing values so the rows with missing values can be removed. Box whisker plot has been used to remove the outliers in the dataset.

- The format of actual values in price variables needs to be worked on for the analysis, so we converted the format of price values (\$56.00 to 56)
- We used the box plot to find out and remove the outliers using inter quartile range.

```
In [8]: l.isnull().sum()
```

```
Out[8]: id                                0
        neighbourhood_group_cleansed      0
        city                             53
        state                             13
        zipcode                           684
        latitude                          0
        longitude                          0
        property_type                     0
        room_type                         0
        accommodates                       0
        bathrooms                          76
        bedrooms                           49
        beds                              44
        bed_type                           0
        amenities                          0
        price                             0
        guests_included                   0
        extra_people                       0
        minimum_nights                     0
        maximum_nights                     0
        number_of_reviews                  0
        instant_bookable                   0
        cancellation_policy                 0
        calculated_host_listings_count     0
        dtype: int64
```

```
# Dealing with erroneous data
main_list.columns
```

```
Index(['index', 'id', 'neighbourhood_group_cleansed', 'city', 'state',
       'zipcode', 'latitude', 'longitude', 'property_type', 'room_type',
       'accommodates', 'bathrooms', 'bedrooms', 'beds', 'bed_type',
       'amenities', 'price', 'guests_included', 'extra_people',
       'minimum_nights', 'maximum_nights', 'number_of_reviews',
       'instant_bookable', 'cancellation_policy',
       'calculated_host_listings_count'],
      dtype='object')
```

```
#Removing value occuring once
main_list.cancellation_policy[3672]
```

```
'long_term'
```

```
pd.unique(main_list.cancellation_policy)
```

```
array(['strict_14_with_grace_period', 'moderate', 'flexible',  
      'super_strict_30', 'strict', 'super_strict_60'], dtype=object)
```

```
main_list.drop('index', inplace=True, axis =1)
```

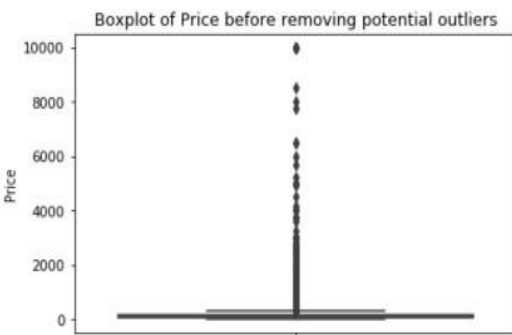
```
# replace all the values related to strict to a single variable "strict"  
main_list.replace('strict_14_with_grace_period','strict', inplace = True)  
main_list.replace('super_strict_30','strict', inplace = True)  
main_list.replace('super_strict_60','strict', inplace = True)
```

```
# Function to convert price string to numeric: $56.00 -> 56  
main_list.price  
convert_numeric_price = lambda x:int(re.sub(',', '', x.split('.')[0][1:]))  
main_list['price'] = main_list['price'].apply(convert_numeric_price)
```

```
# Function to convert price string to numeric: $56.00 -> 56  
main_list.extra_people  
main_list['extra_people'] = main_list['extra_people'].apply(convert_numeric_price)
```

```
# boxplot the price variable with outliers  
plot_before = sns.boxplot(y = main_list['price'])  
plot_before.set(ylabel='Price', title = 'Boxplot of Price before removing potential outliers')
```

```
[Text(0, 0.5, 'Price'),  
Text(0.5, 1.0, 'Boxplot of Price before removing potential outliers')]
```



```
# we see that there are outliers in the price variable and this needs be removed
```

```
#Data Reduction - Removing Outliers  
inter_quartile_range = main_list['price'].quantile(q=0.75)-main_list['price'].quantile(0.25)  
# Since the price can't be negative, we inserted a constarints  
low_lim = max(0, main_list['price'].quantile(0.25)-1.5*inter_quartile_range)  
up_lim = main_list['price'].quantile(0.75)+1.5*inter_quartile_range  
print ("Only prices between", low_lim, "and", up_lim,"are retained in the dataset.")
```

```
Only prices between 0 and 342.5 are retained in the dataset.
```



```
# We see that price between 0 and 342.5 in the IQR are retained

remove_indicies = main_list[main_list['price'] > up_lim].index
main_list.drop(remove_indicies, inplace=True)
main_list.reset_index(inplace=True, drop=True)

#The number of rows that are removed through inter Quartile range
len(remove_indicies)

2911

#2911 rows are removed through IQR

# boxplot the price variable without outliers
plot_after = sns.boxplot(y = main_list['price'])
plot_after.set(ylabel='Price', title = 'Boxplot of Price after removing potential outliers')

[Text(0, 0.5, 'Price'),
Text(0.5, 1.0, 'Boxplot of Price after removing potential outliers')]

Boxplot of Price after removing potential outliers
350
300
250
200
150
100
50
0
Price
```

#We see that the the values are all within the range and outliers are removed

4. DATA TRANSFORMATION

As a part of data transformation, we implemented the following steps:

- Converted the data types of categorical variables like (Property_type, room_type etc.) to category data type.
- Created dummy variables for all the categorial variables and generated new attributes that represents each level of every categorical variable.
- For example, a new attribute is constructed for Apartment level of Property_type variable.
- Converted the list of amenities in to different attributes, each representing single amenity available with respect each listing.

```
#Converting the data types
main_list['neighbourhood_group_cleansed'] = main_list['neighbourhood_group_cleansed'].astype('category')
main_list['property_type'] = main_list['property_type'].astype('category')
main_list['room_type'] = main_list['room_type'].astype('category')
main_list['bed_type'] = main_list['bed_type'].astype('category')
main_list['instant_bookable'] = main_list['instant_bookable'].astype('category')
main_list['cancellation_policy'] = main_list['cancellation_policy'].astype('category')
```

Dummy Variables:

```
# Creating dummy variables for neighbourhood_group_cleansed
dummy1= pd.get_dummies(main_list['neighbourhood_group_cleansed'], prefix='neighbourhood')
# Creating dummy variables for property_type
dummy2= pd.get_dummies(main_list['property_type'], prefix='property')
# Creating dummy variables for room_type
dummy3= pd.get_dummies(main_list['room_type'], prefix='roomtype')
# Creating dummy variables for instant_bookable
dummy4= pd.get_dummies(main_list['instant_bookable'], prefix='instant_book')
# Creating dummy variables for bed_type
dummy5= pd.get_dummies(main_list['bed_type'], prefix='bedtype')
```

```
#merging the dummy variables to main_list
dummy1.columns = ['neighbourhood_Bronx', 'neighbourhood_Brooklyn',
                  'neighbourhood_Manhattan', 'neighbourhood_Queens',
                  'neighbourhood_Staten_Island']
main_list = main_list.join(dummy1)
```

```
dummy2.columns = ['property_Aparthotel', 'property_Apartment',
                  'property_Bed_breakfast', 'property_Boat',
                  'property_Boutique_hotel', 'property_Bungalow', 'property_Bus',
                  'property_Cabin', 'property_Camper_RV',
                  'property_Casa_particular_Cuba', 'property_Castle', 'property_Cave',
                  'property_Condominium', 'property_Cottage', 'property_Earth_house',
                  'property_Guest_suite', 'property_Guesthouse', 'property_Hotel',
                  'property_Hotel', 'property_House', 'property_Island', 'property_Loft',
                  'property_Nature_lodge', 'property_Other', 'property_Resort',
                  'property_Serviced_apartment', 'property_Tent', 'property_Tiny house',
                  'property_Townhouse', 'property_Villa']
main_list = main_list.join(dummy2)
```

```
dummy3.columns = ['roomtype_Entire_home_apt', 'roomtype_Private_room',
                  'roomtype_Shared_room']
main_list = main_list.join(dummy3)
```

```
dummy4.columns = ['instant_book_False', 'instant_book_True']
main_list = main_list.join(dummy4)
```

5. DATA REDUCTION:

Data reduction involves dealing with the columns in the data set. We removed columns such as 'listing_url', 'thumbnail_url', 'host_since', 'host_location' etc., as these variables were not involved in predicting the price of the listing and they had missing values upto 80%. Country and city specific information were also removed as we are concerned in developing a generalized price prediction model.

```

#Removing the unnecessary columns for analysis
to_drop = ['scrape_id','listing_url','last_scraped','name', 'summary',
'space', 'description', 'experiences_offered', 'neighborhood_overview',
'notes', 'transit','interaction','thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url',
'host_id', 'host_url', 'host_name', 'host_since', 'host_location',
'host_about', 'host_response_time','host_acceptance_rate', 'host_is_superhost', 'host_thumbnail_url',
'host_picture_url', 'host_neighbourhood', 'host_listings_count',
'host_total_listings_count', 'host_verifications',
'host_has_profile_pic', 'host_identity_verified', 'street',
'neighbourhood','market',
'smart_location', 'country_code', 'country','is_location_exact','calendar_updated', 'has_availability',
'availability_30', 'availability_60', 'availability_90',
'availability_365', 'calendar_last_scraped','first_review', 'last_review',
'requires_license',
'license', 'jurisdiction_names','require_guest_profile_picture',
'require_guest_phone_verification','reviews_per_month','access', 'review_scores_rating', 'review_scores_accuracy',
'review_scores_cleanliness', 'review_scores_checkin',
'review_scores_communication', 'review_scores_location',
'review_scores_value','weekly_price', 'monthly_price','square_feet',
'neighbourhood_cleansed','host_response_rate','house_rules','security_deposit',
'cleaning_fee','is_business_travel_ready']
main_list.drop(to_drop, inplace = True, axis = 1)

```

Principal Component Analysis:

Principal Component analysis is a technique that creates linear combination of columns. PCA uses only numerical values, we collected numerical values from the data frame and ran the PCA model.

Eigen values were assessed after PCA.

```

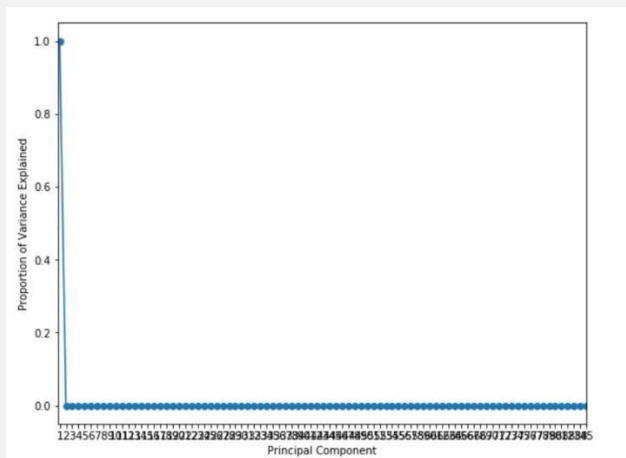
#PCA Analysis for Data Reduction.
main_list.columns
main_list.shape
data_pca = main_list[['latitude', 'longitude', 'accommodates',
'bathrooms', 'bedrooms', 'beds',
'guests_included', 'extra_people', 'minimum_nights', 'maximum_nights',
'number_of_reviews',
'calculated_host_listings_count', 'TV', 'ELEVATOR', 'INTERNET',
'CABLE TV', 'Wifi', 'Air conditioning', 'Kitchen', 'Smoke detector',
'Hot water', 'Microwave', 'Coffee maker', 'Family/kid friendly',
'Refrigerator', 'Fire extinguisher', 'Gym', 'HOT TUB', 'SHAMPOO',
'PARKING', 'SMOKING ALLOWED', 'HAIR DRYER', 'SUITABLE FOR EVENTS',
'HANGERS', 'IRON', 'DOORMAN', 'BREAKFAST', 'WHEEL',
'Buzzer/wireless intercom', 'Laptop friendly workspace',
'neighbourhood_Bronx', 'neighbourhood_Brooklyn',
'neighbourhood_Manhattan', 'neighbourhood_Queens',
'neighbourhood_Staten_Island', 'property_Aparthotel',
'property_Apartment', 'property_Bed_breakfast', 'property_Boat',
'property_Boutique_hotel', 'property_Bungalow', 'property_Bus',
'property_Cabin', 'property_Camper_RV', 'property_Casa_particular_Cuba',
'property_Castle', 'property_Cave', 'property_Condominium',
'property_Cottage', 'property_Earth_house', 'property_Guest_suite',
'property_Guesthouse', 'property_Hotel', 'property_Hotel',
'property_House', 'property_Island', 'property_Loft',
'property_Nature_lodge', 'property_Other', 'property_Resort',
'property_Serviced_apartment', 'property_Tent', 'property_Tiny house',
'property_Townhouse', 'property_Villa', 'roomtype_Entire_home_apartment',
'roomtype_Private_room', 'roomtype_Shared_room', 'instant_book_False',
'instant_book_True', 'bedtype_Airbed', 'bedtype_Couch', 'bedtype_Futon',
'bedtype_Pull_out_Sofa', 'bedtype_Real_Bed']]
data_pca.shape
main_list_PCA = pca(n_components=85).fit(data_pca)

```

```
#Obtain eigenvalues
main_list_PCA.explained_variance_

array([1.01913000e+14, 1.76528852e+03, 5.08006695e+02, 3.54350439e+02,
       2.23233669e+02, 3.66176636e+00, 7.61353779e-01, 6.14269044e-01,
       5.24223815e-01, 4.56368885e-01, 3.74022230e-01, 3.47932900e-01,
       2.97059306e-01, 2.73102739e-01, 2.34429088e-01, 2.12523193e-01,
       1.99415532e-01, 1.89076760e-01, 1.82961256e-01, 1.73938318e-01,
       1.67453211e-01, 1.58166503e-01, 1.57899832e-01, 1.30663412e-01,
       1.23531329e-01, 1.21480636e-01, 1.15773504e-01, 1.09218008e-01,
       1.08095717e-01, 1.06516923e-01, 1.01529282e-01, 1.01083444e-01,
       7.38239584e-02, 6.34585976e-02, 5.70406525e-02, 4.87306015e-02,
       4.74184701e-02, 4.36301158e-02, 3.43854757e-02, 3.33591858e-02,
       3.30138280e-02, 2.83128112e-02, 2.59684441e-02, 2.56206223e-02,
       2.22592163e-02, 2.16010630e-02, 2.11492606e-02, 1.26076353e-02,
       8.46369563e-03, 6.55735054e-03, 6.41873212e-03, 4.60206091e-03,
       3.18481158e-03, 2.31122941e-03, 2.08794874e-03, 1.92809299e-03,
       1.46724756e-03, 1.19450133e-03, 1.16726613e-03, 1.05722497e-03,
       7.47639952e-04, 6.80324229e-04, 5.74090138e-04, 5.63369815e-04,
       3.03298269e-04, 1.68348087e-04, 1.37979223e-04, 1.15308237e-04,
       9.21808137e-05, 7.14885340e-05, 6.61305547e-05, 4.73645997e-05,
       4.40906486e-05, 2.56842593e-05, 2.20821588e-05, 2.20768112e-05,
       2.20642076e-05, 2.20529043e-05, 1.01747301e-18, 1.01747301e-18,
       1.01747301e-18, 1.01747301e-18, 1.01747301e-18, 1.01747301e-18,
       8.21007931e-21])
```

Scree Plot:



From the Eigen values it is understood that only 6 components are retained. That is, in reality, we are dealing only with 6 variables and rest of the values can be thrown out.

DESCRIPTIVE STATISTICS:

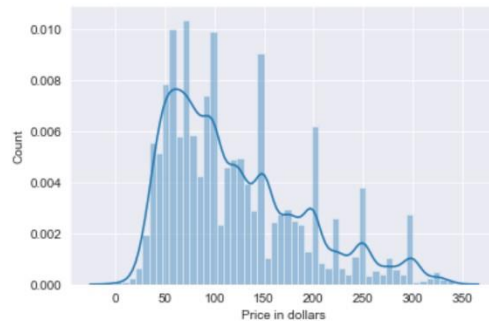
Before we start building the prediction model, it is important to understand the variables involved in the study and if there's any interesting patterns.

1. Price – Target variable:

```
sns.distplot(main_list['price'], kde=True, axlabel='Price in dollars')
plt.ylabel('Count')
main_list['price'].skew()
```

C:\Users\krith\AppData\Local\Continuum\anaconda3\lib\site-packages\scipy\stats\st
le sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` in
l be interpreted as an array index, `arr[np.array(seq)]`, which will result eithe
return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

77]: 0.9116284409035669



```
main_list.price.describe()
```

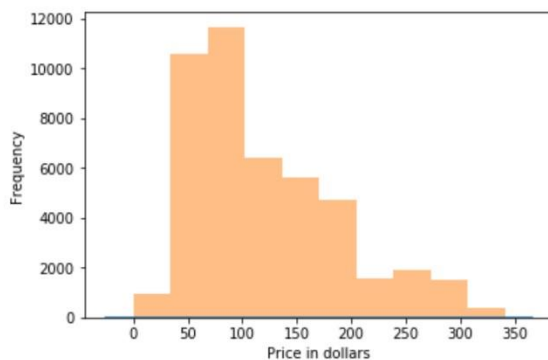
```
5]: count    45259.000000
    mean      121.063457
    std       68.344553
    min       0.000000
    25%       67.000000
    50%      100.000000
    75%      160.000000
    max      341.000000
    Name: price, dtype: float64
```

We found that the mean price value is \$121 per night and maximum is \$341 per night.

```
# Price
sns.distplot(main_list['price'], axlabel='Price in dollars')
plt.ylabel('Frequency Count')
main_list['price'].plot.hist(alpha=0.5)
main_list['price'].skew()
```

C:\Users\krith\AppData\Local\Continuum\anaconda3\lib\site-packa
le sequence for multidimensional indexing is deprecated; use `a
l be interpreted as an array index, `arr[np.array(seq)]`, which
return np.add.reduce(sorted[indexer] * weights, axis=axis) /

4]: 0.9116284409035669

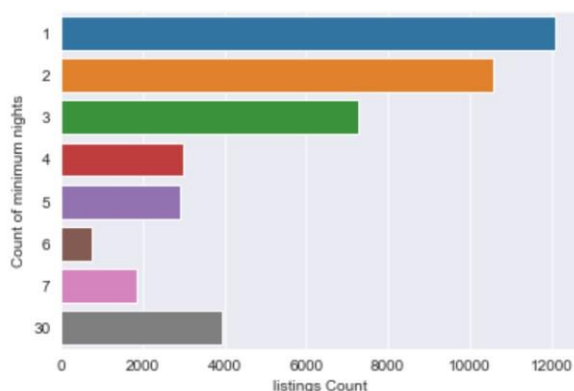


From the plot it is understood that most of the listings have price ranging from \$50-100 per night.

2. Minimum nights and Customer count:

```
# minimum nights
mini_nights = main_list['minimum_nights'].value_counts(dropna=False)[:8]
sns.barplot(x=mini_nights.values, y=mini_nights.index, orient = 'h')
plt.xlabel('listings Count')
plt.ylabel('Count of minimum nights')
```

6]: Text(0, 0.5, 'Count of minimum nights')

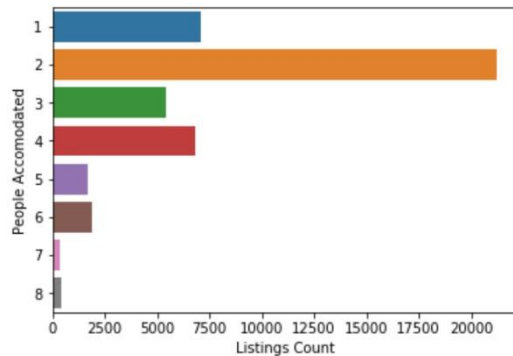


We can understand from the plot that most of the listings had the criteria that the minimum nights to be spent by the customer as 1 and there were few listings that required customers to spend at least 30 nights which makes no sense.

3. Number of people accommodated by the listings:

```
: #accommodates
accommodating = main_list['accommodates'].value_counts(dropna=False)[:8]
sns.barplot(x=accommodating.values, y=accommodating.index,orient = 'h')
plt.xlabel('Listings Count')
plt.ylabel('People Accommodated')
```

```
[56]: Text(0, 0.5, 'People Accommodated')
```



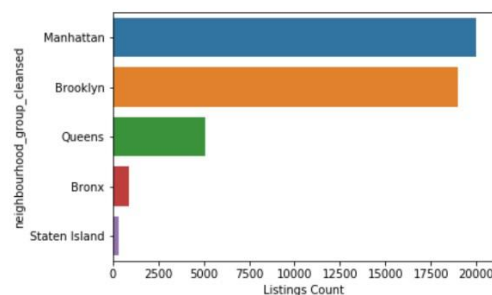
From the plot, we can see that most of the listings accommodate 2 people at a time per booking and many accommodate 1 and 4 people.

4. Based on Neighborhood groups:

As we know that New York has around 8 popular neighborhoods and the location of listings near the neighborhoods can inflate/deflate the prices of the listings.

```
#neighbourhood_group_cleansed
neighbourhood_group_cleansed = main_list['neighbourhood_group_cleansed'].value_counts(dropna=False)[:8]
sns.barplot(x=neighbourhood_group_cleansed.values, y=neighbourhood_group_cleansed.index,orient = 'h')
plt.xlabel('Listings Count')
plt.ylabel('neighbourhood_group_cleansed')
```

```
]: Text(0, 0.5, 'neighbourhood_group_cleansed')
```

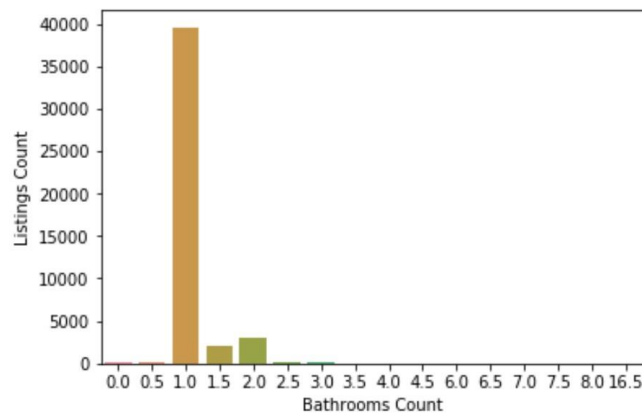


We see that Manhattan has most number of listings followed by Brooklyn and Queens.

5. Bathrooms in the apartment:

```
#Bathrooms
bathrooms_Count = main_list['bathrooms'].value_counts()
sns.barplot(x=bathrooms_Count.index, y=bathrooms_Count.values)
plt.ylabel('Listings Count')
plt.xlabel('Bathrooms Count')
```

```
Text(0.5, 0, 'Bathrooms Count')
```



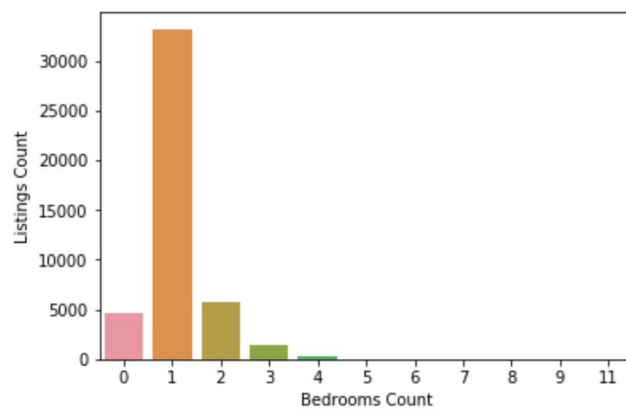
We see that most of the listings have one bath and few listings also have 1.5 and

2.0

6. Bedrooms in the apartment:

```
#Bedrooms
bedrooms = main_list['bedrooms'].value_counts()
sns.barplot(x=bedrooms.index, y=bedrooms.values)
plt.ylabel('Listings Count')
plt.xlabel('Bedrooms Count')
```

```
Text(0.5, 0, 'Bedrooms Count')
```



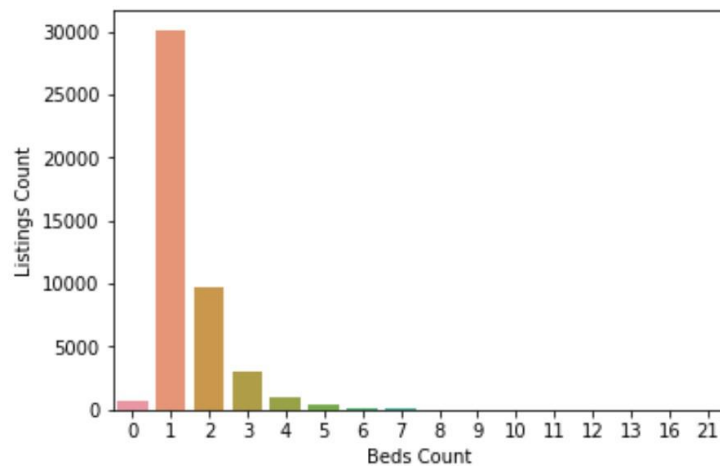
We see from the listings that most of the apartments had one bedroom and few had

0 which might probably be a hall in the house rented for cheaper price.

7. Beds in the apartments:

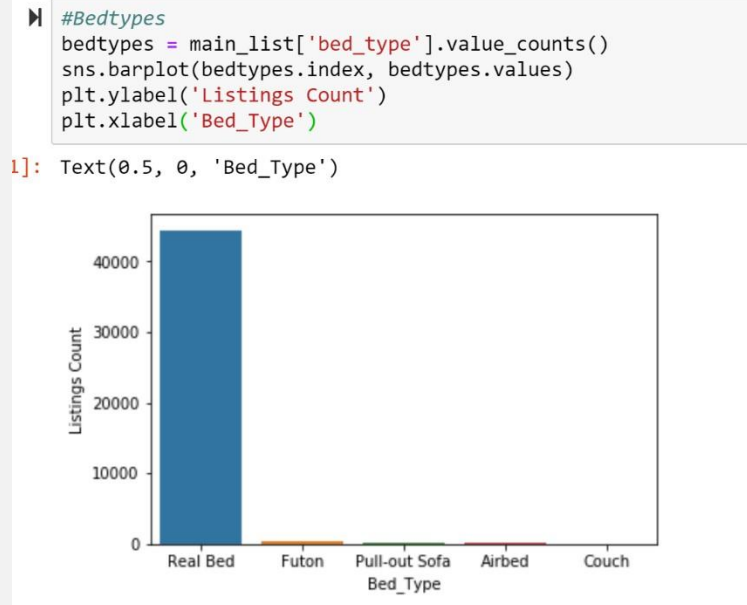
```
▶ #Beds
beds = main_list['beds'].value_counts()
sns.barplot(x=beds.index, y=beds.values)
plt.ylabel('Listings Count')
plt.xlabel('Beds Count')
```

```
]: Text(0.5, 0, 'Beds Count')
```



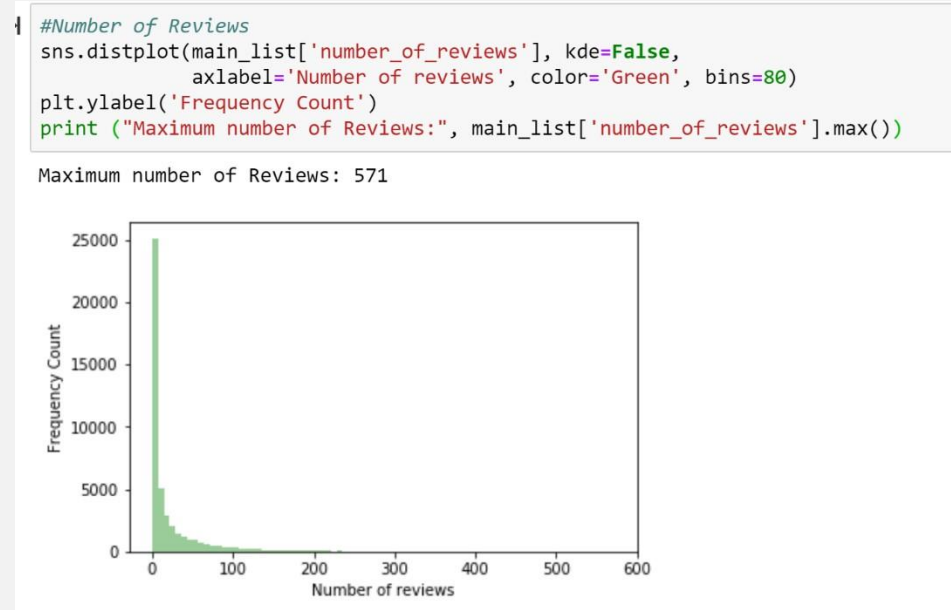
Most of the listings had one bed and few of them had two. There were a very few listings that had zero beds which makes no sense and can be interpreted for extra beds.

8. Bed Types:



From the plot, we can understand that almost 99% of the listings has Real bed.

9. Number of reviews:

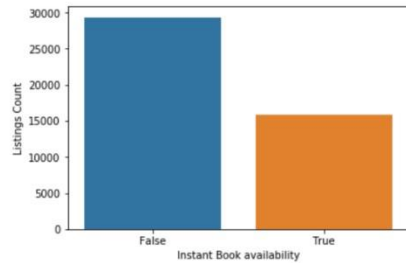


As we know that reviews are given by the guests who stayed in the apartment and it shows how popular the apartment is. Most of the listings had number of reviews ranging from 0 to 100.

10. Instant bookable:

```
#Instant Bookable
instant_bookable = main_list.instant_bookable.value_counts()
instant_book_dict = {'f':'False', 't':'True'}
sns.barplot([instant_book_dict[i] for i in instant_bookable.index], instant_bookable.values)
plt.xlabel('Instant Book availability')
plt.ylabel('Listings Count')
```

```
3]: Text(0, 0.5, 'Listings Count')
```



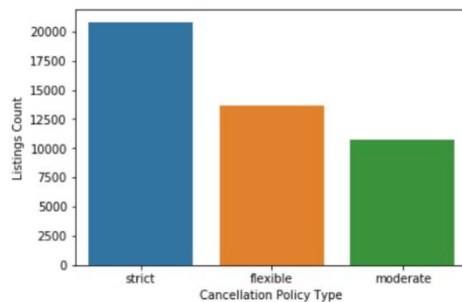
From the plot it can be perceived that listings with higher price might have instant booking feature and the listings with lower price might not, which will be clear in further analysis.

11. Cancellation policy:

The cancellation policy was categorized into flexible, moderate, strict.

```
#Cancellation Policy
sns.barplot(pd.Series(main_list['cancellation_policy'].value_counts()).index,
            pd.Series(main_list['cancellation_policy'].value_counts()).values,
            )
plt.xlabel('Cancellation Policy Type')
plt.ylabel('Listings Count')
```

```
3]: Text(0, 0.5, 'Listings Count')
```

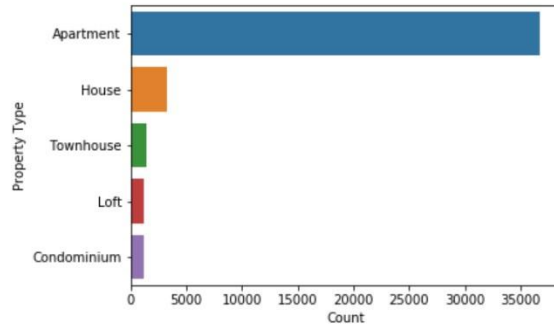


Around 50% of the listings had strict cancellation policy and others had flexible and moderate cancellation policies.

12. Property type:

```
# Property types
property_type = main_list['property_type'].value_counts()[1:5]
sns.barplot(y=property_type.index, x=property_type.values, orient='h')
plt.xlabel('Count')
plt.ylabel('Property Type')
```

```
]: Text(0, 0.5, 'Property Type')
```

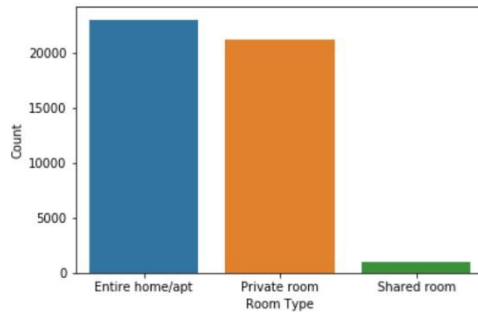


From the plot we see that most of the properties were apartments, few were houses, townhouse, loft and condominium.

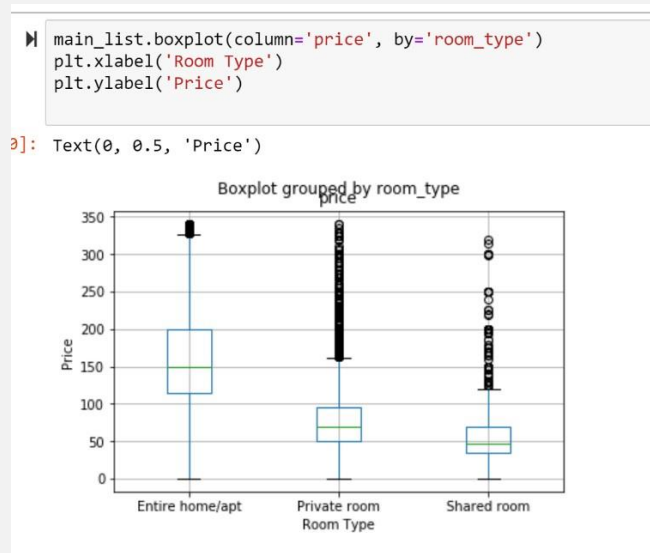
13. Room Types:

```
#Room types
type_of_room = main_list['room_type'].value_counts()
sns.barplot(type_of_room.index, type_of_room.values)
plt.xlabel('Room Type')
plt.ylabel('Count')
```

```
]: Text(0, 0.5, 'Count')
```



Most of the listings offered entire home and private room. Very few houses had shared room which might have lower price.



A box plot of room_type and price give us a view of the distribution of price for different room types.

- Entire home had average price of \$150 per night
- Private room had average price of around \$75 per night
- Shared room had average price of around \$50 per night

As amenities are important predictor variable for the analysis, we performed descriptive statistics of amenities variable and the results are as shown below,

```
#Descriptive statistics for amenities

amenities = [ 'TV', 'ELEVATOR', 'INTERNET',
              'CABLE TV', 'Wifi', 'Air conditioning', 'Kitchen', 'Smoke deTector',
              'HoT waTer', 'Microwave', 'Coffee maker', 'Family/kid friendly',
              'Refrigerator', 'Fire exTinguishe', 'GYM', 'HOT TUB', 'SHAMPOO',
              'PARKING ', 'SMOKING ALLOWED', 'HAIR DRYER', 'SUITABLE FOR EVENTS ',
              'HANGERS', 'IRON', 'DOORMAN', 'BREAKFAST',
              'Buzzer/wireless inTercom', 'LapTop friendly workspace']
```

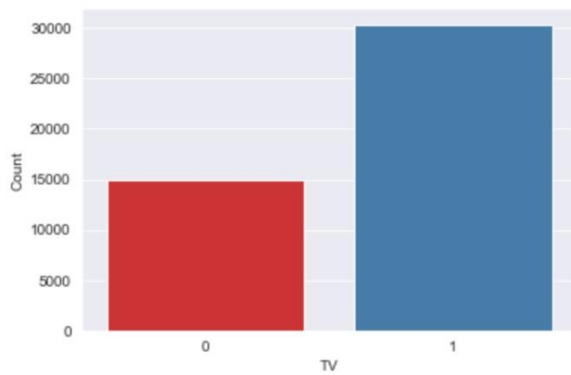
The above shown were the considered amenities. Amenities is categorical variable with the values 0 and 1.

- 0- Amenity present
- 1- Amenity absent

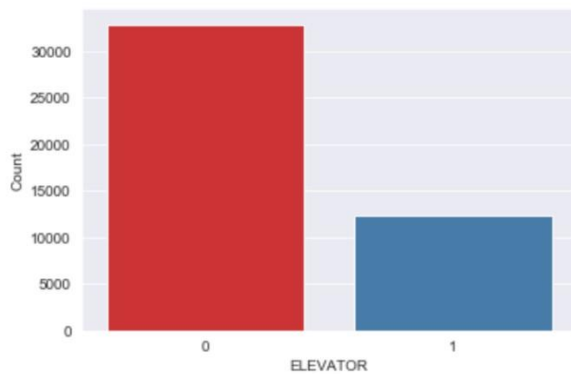
```
sns.set_style("darkgrid")
for amenity in amenities:
    print(main_list[amenity].value_counts())
    sns.barplot(pd.Series(main_list[amenity].value_counts()).index, pd.Series(main_list[amenity].value_counts()).values, palette=['red', 'blue'])
    plt.ylabel('Count')
    plt.xlabel(amenity)
    plt.show()
```

The amenities were looped through to generate visualizations

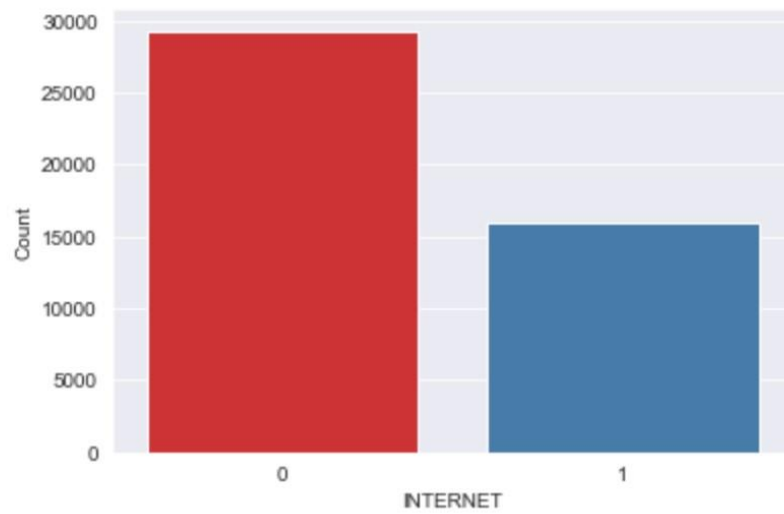
```
1    30376
0    14883
Name: TV, dtype: int64
```



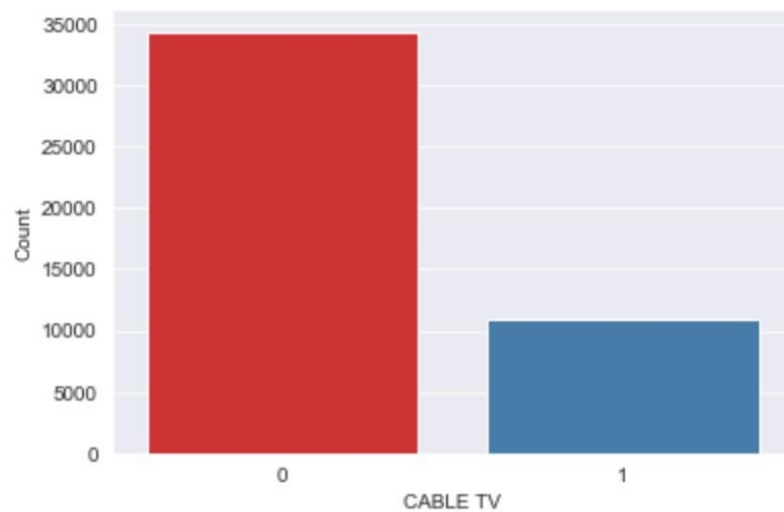
```
0    32897
1    12362
Name: ELEVATOR, dtype: int64
```



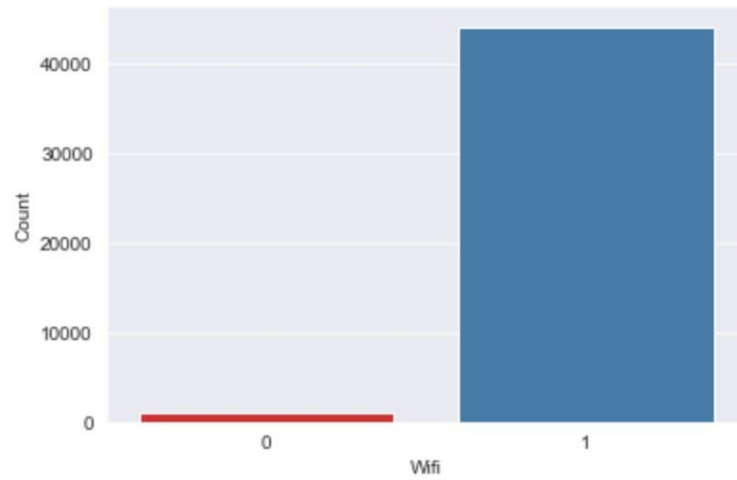
```
0    29305
1    15954
Name: INTERNET, dtype: int64
```



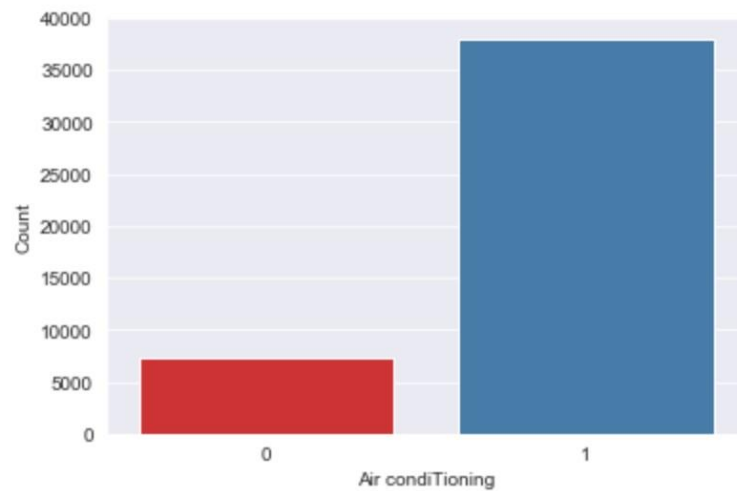
```
0    34360
1    10899
Name: CABLE TV, dtype: int64
```



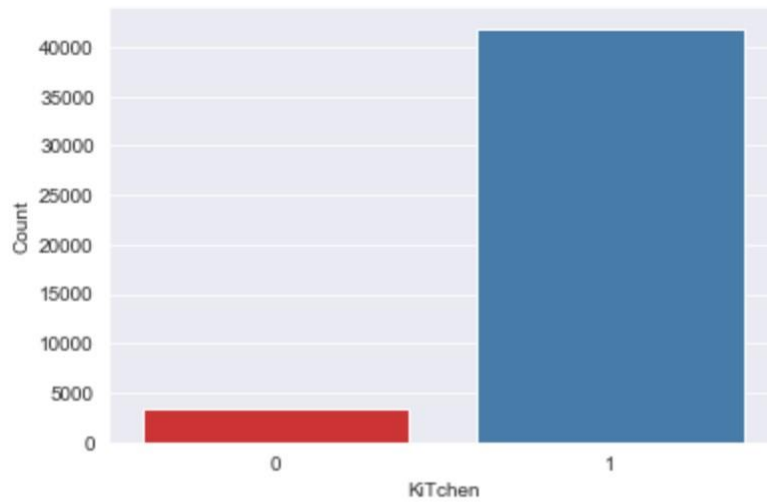
```
1    44184
0     1075
Name: Wifi, dtype: int64
```



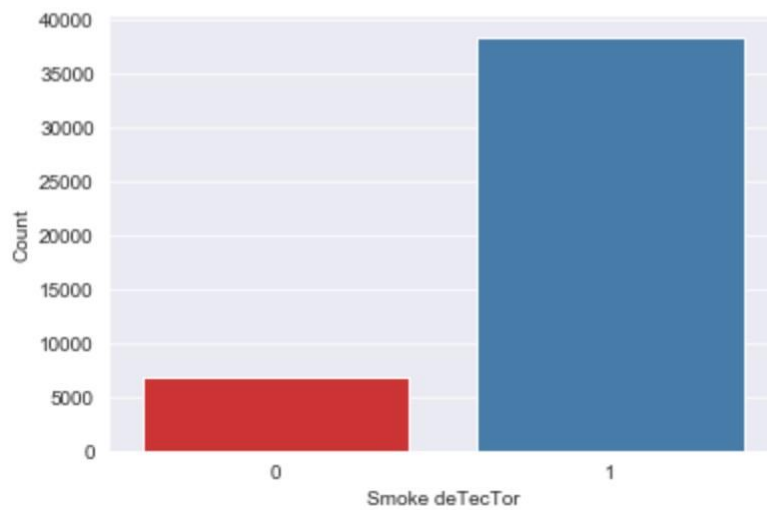
```
1    38038
0     7221
Name: Air condiTioning, dtype: int64
```



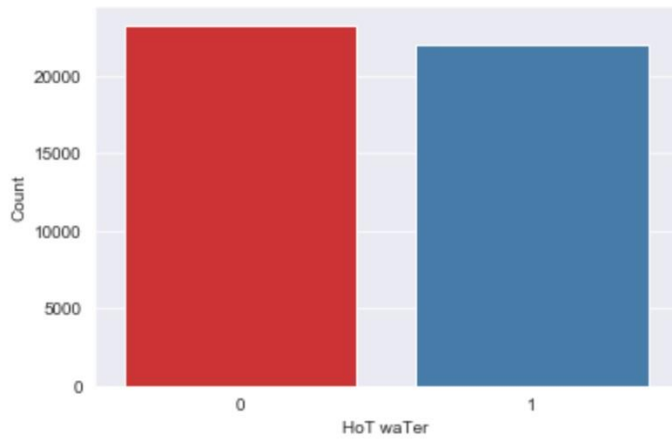

```
1    41873
0     3386
Name: KiTchen, dtype: int64
```



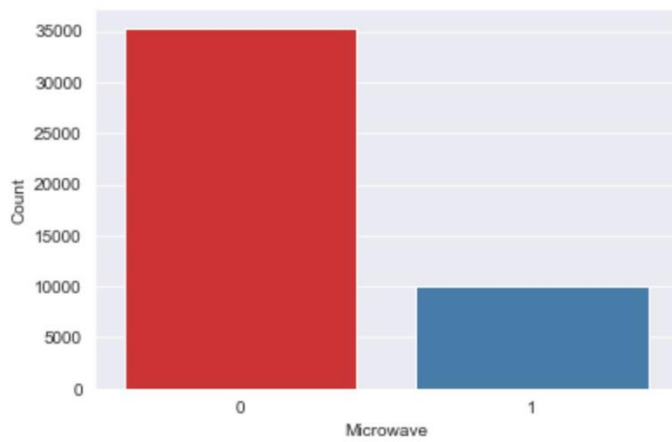
```
1    38468
0     6791
Name: Smoke deTecTor, dtype: int64
```



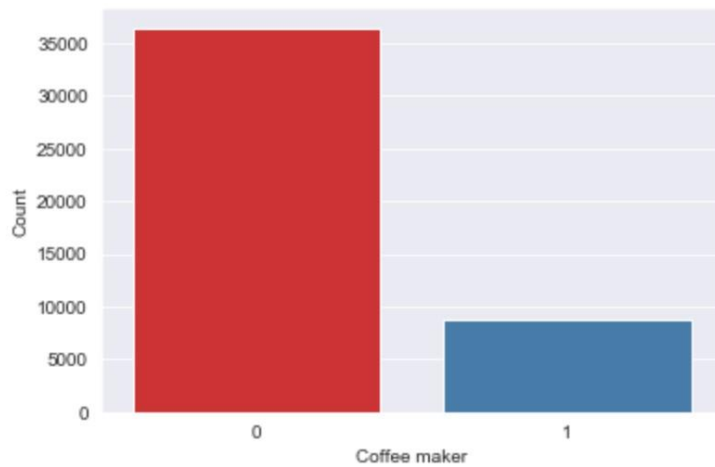
```
0    23261
1    21998
Name: HoT waTer, dtype: int64
```



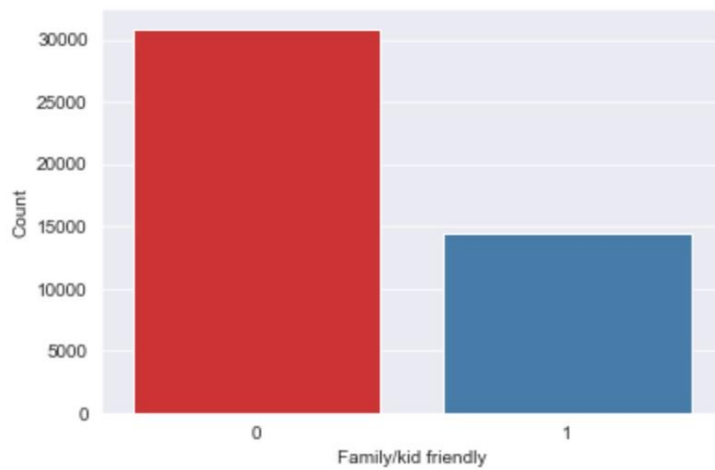
```
0    35310
1     9949
Name: Microwave, dtype: int64
```



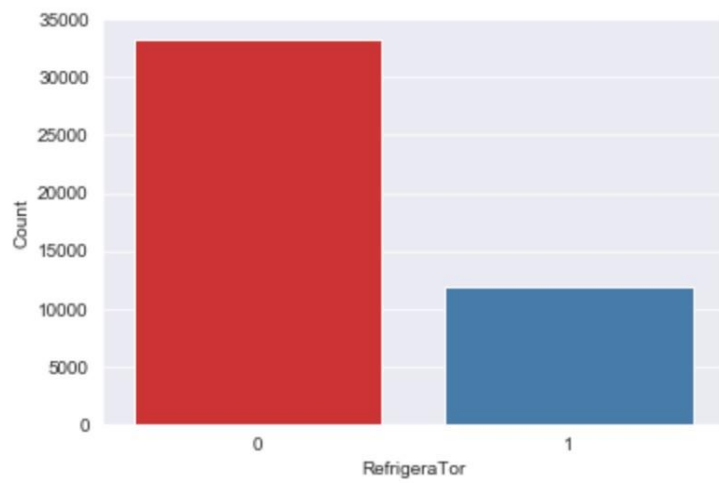
```
0    36441
1     8818
Name: Coffee maker, dtype: int64
```



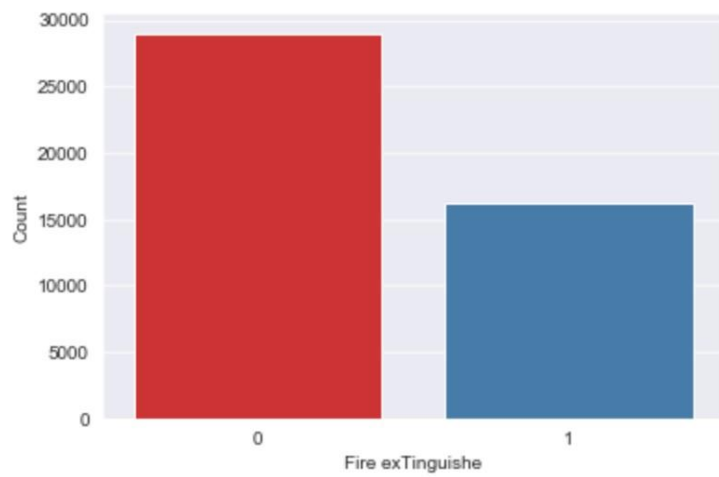
```
0    30863
1    14396
Name: Family/kid friendly, dtype: int64
```



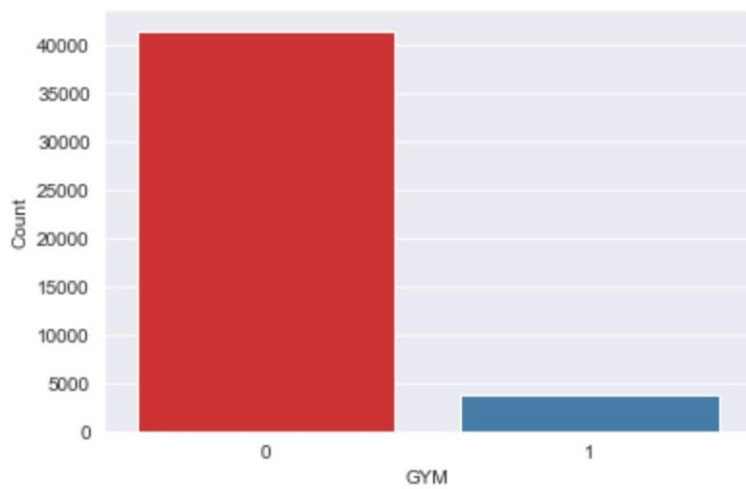
```
0    33326
1     11933
Name: Refrigerator, dtype: int64
```



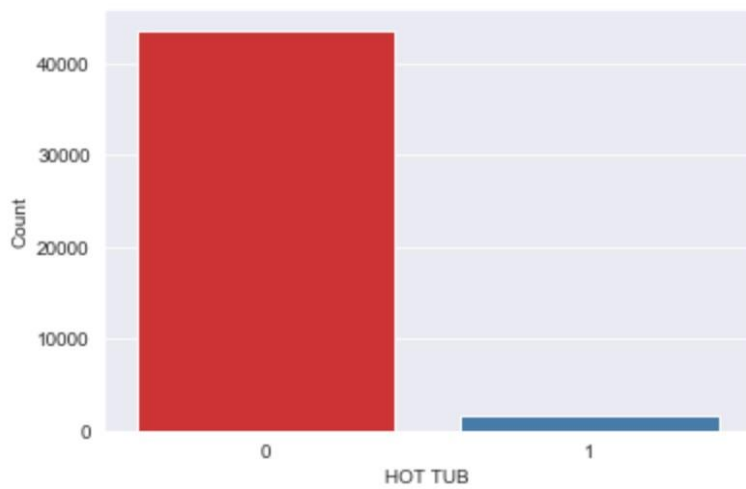
```
0    29004
1    16255
Name: Fire extinguisher, dtype: int64
```



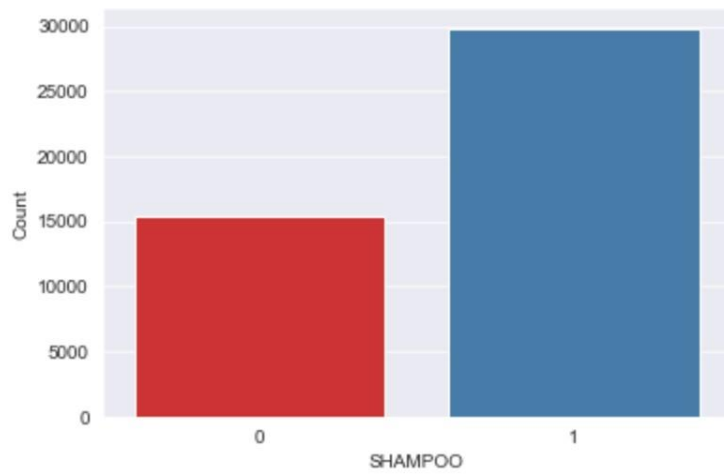
```
0    41443
1     3816
Name: GYM, dtype: int64
```



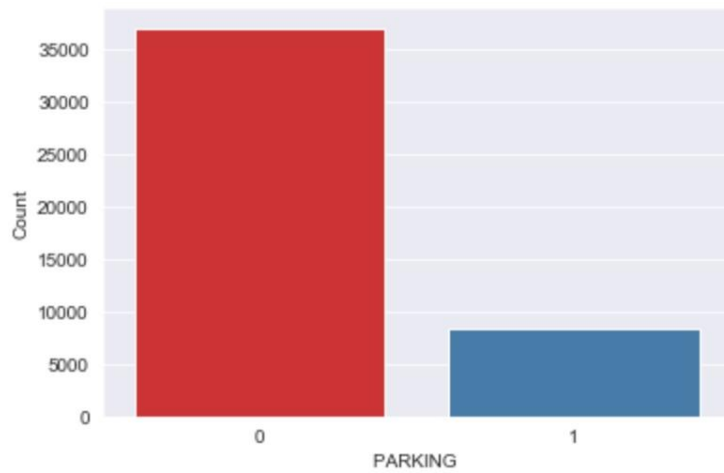
```
0    43653
1     1606
Name: HOT TUB, dtype: int64
```



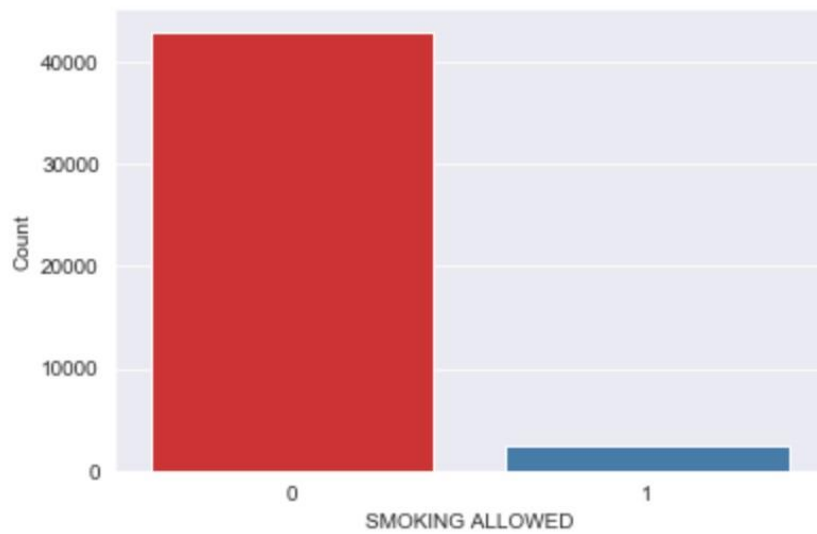
```
1    29827
0    15432
Name: SHAMPOO, dtype: int64
```



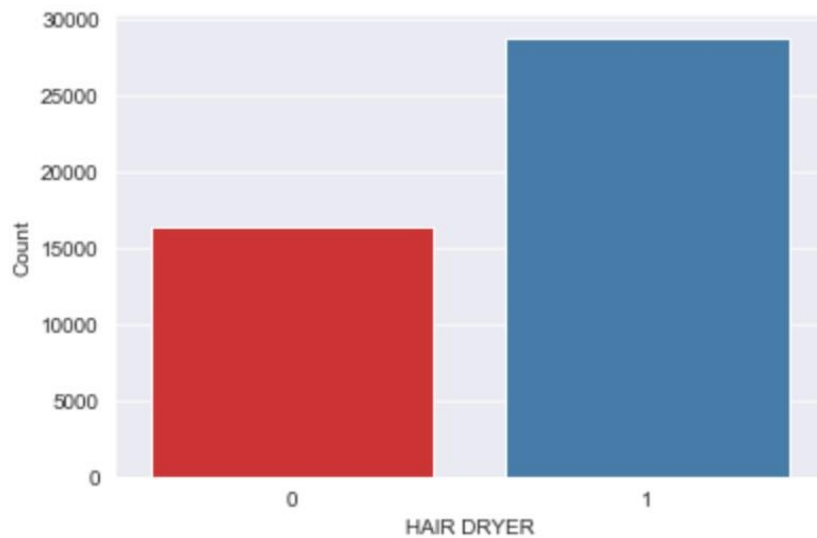
```
0    36987
1     8272
Name: PARKING, dtype: int64
```



```
0    42894
1     2365
Name: SMOKING ALLOWED, dtype: int64
```



```
1    28819
0    16440
Name: HAIR DRYER, dtype: int64
```



0 45259

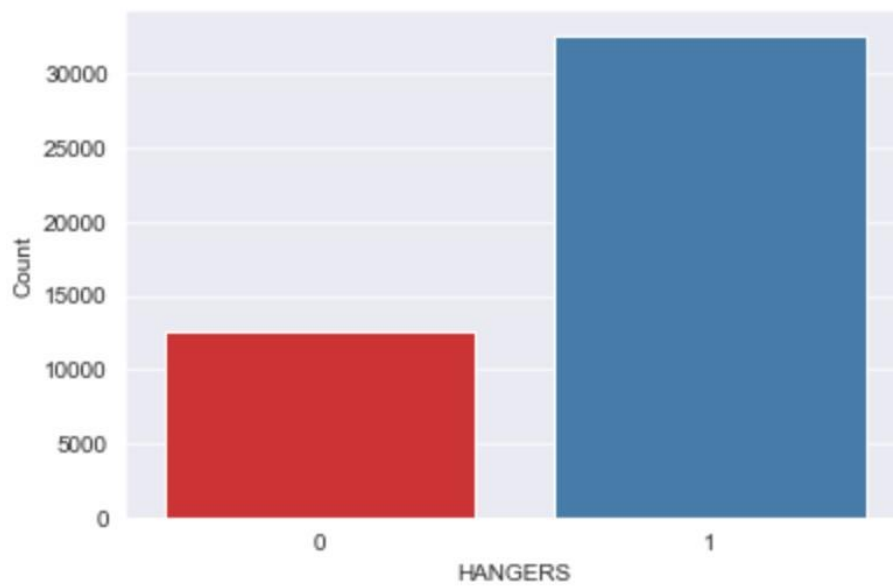
Name: SUITABLE FOR EVENTS , dtype: int64



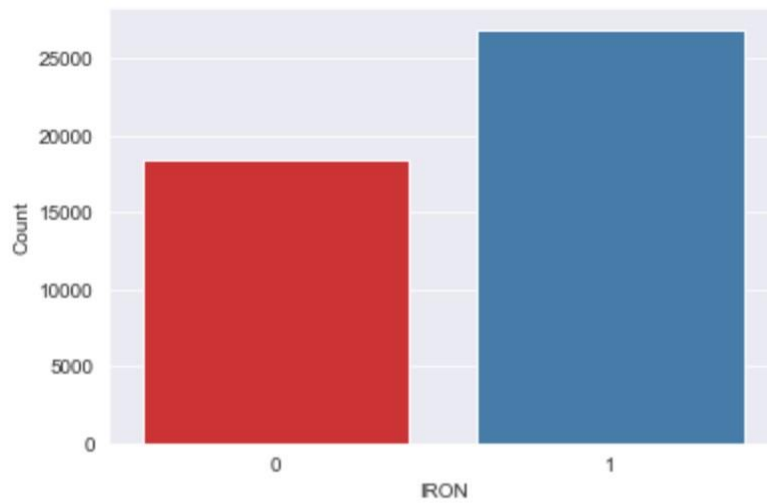
1 32642

0 12617

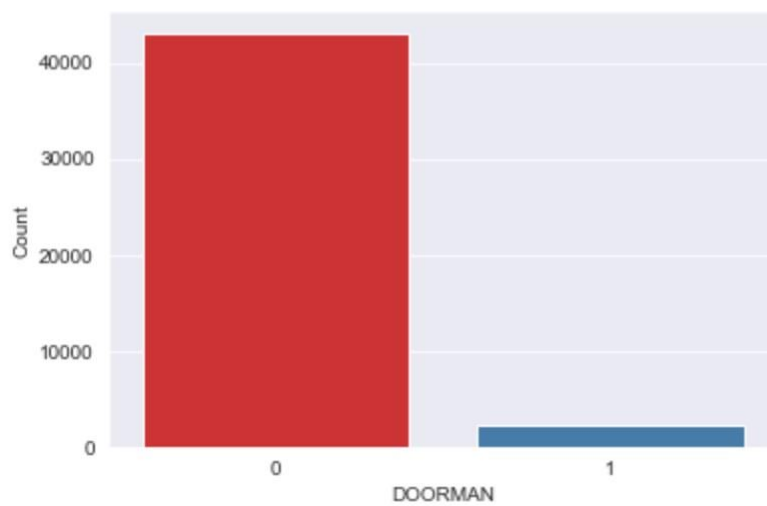
Name: HANGERS, dtype: int64



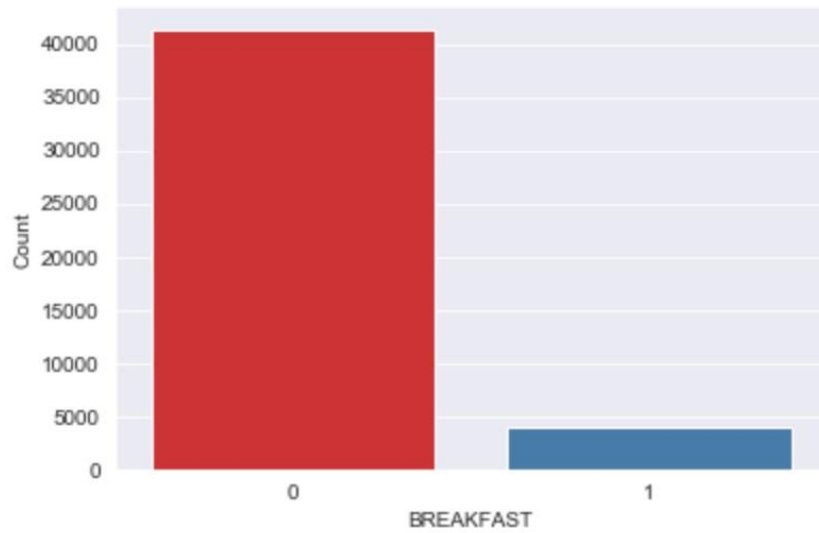

```
1    26880
0    18379
Name: IRON, dtype: int64
```



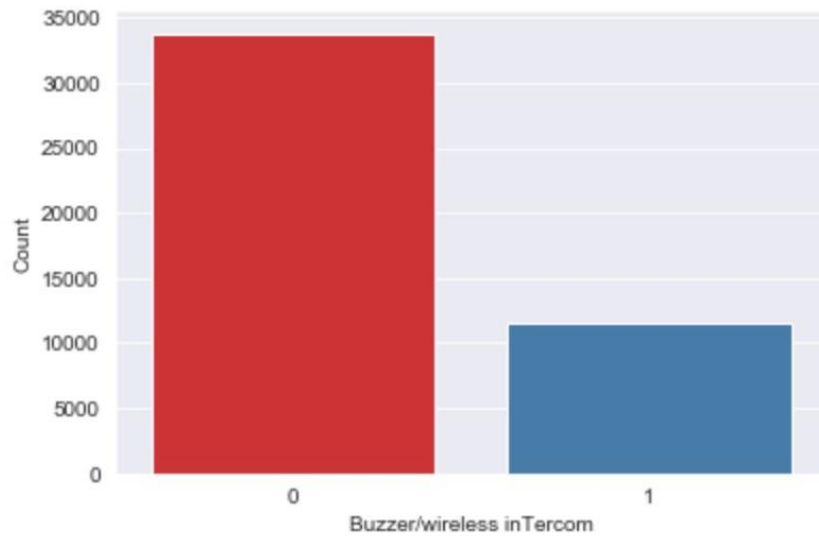
```
0    43094
1     2165
Name: DOORMAN, dtype: int64
```



```
0    41366
1     3893
Name: BREAKFAST, dtype: int64
```



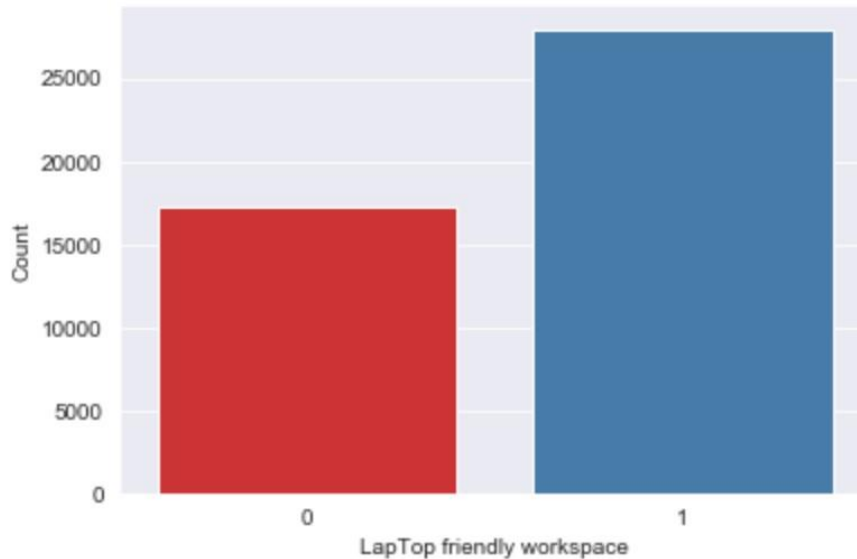
```
0    33775
1    11484
Name: Buzzer/wireless inTercom, dtype: int64
```



```

1    27960
0    17299
Name: LapTop friendly workspace, dtype: int64

```



We performed correlation analysis between the one of the predictor variables (amenities) and target variable (price)

```

# Correlation between target and independent variables
for amenity in amenities:
    # Pearson Correlation Coefficient
    print("Correlation between Price and ", amenity, " = ", np.corrcoef(main_list[amenity], main_list['price'])[0, 1])

Correlation between Price and TV = 0.2618849630322195
Correlation between Price and ELEVATOR = 0.19207576988369096
Correlation between Price and INTERNET = 0.07111127586618633
Correlation between Price and CABLE TV = 0.20333454369905982
Correlation between Price and Wifi = 0.024715107279662293
Correlation between Price and Air conditioning = 0.19755494791631423
Correlation between Price and Kitchen = 0.09509007308768323
Correlation between Price and Smoke detector = 0.05469440730138382
Correlation between Price and Hot water = 0.010048665763754148
Correlation between Price and Microwave = 0.04110074464106924
Correlation between Price and Coffee maker = 0.07934429776284783
Correlation between Price and Family/kid friendly = 0.2339711366597516
Correlation between Price and Refrigerator = 0.040302717503886316
Correlation between Price and Fire extinguisher = 0.020763061442707453
Correlation between Price and GYM = 0.19373515305776493
Correlation between Price and HOT TUB = 0.010731647455053651
Correlation between Price and SHAMPOO = 0.1009567872454867
Correlation between Price and PARKING = 0.06299595587625996
Correlation between Price and SMOKING ALLOWED = -0.06526142120454669
Correlation between Price and HAIR DRYER = 0.13478402060918165
Correlation between Price and SUITABLE FOR EVENTS = nan
Correlation between Price and HANGERS = 0.04263166798415906
Correlation between Price and IRON = 0.1236346047585998
Correlation between Price and DOORMAN = 0.14852383102451172
Correlation between Price and BREAKFAST = -0.007267530352270059
Correlation between Price and Buzzer/wireless intercom = 0.06694770300035383
Correlation between Price and LapTop friendly workspace = 0.09353195166997132

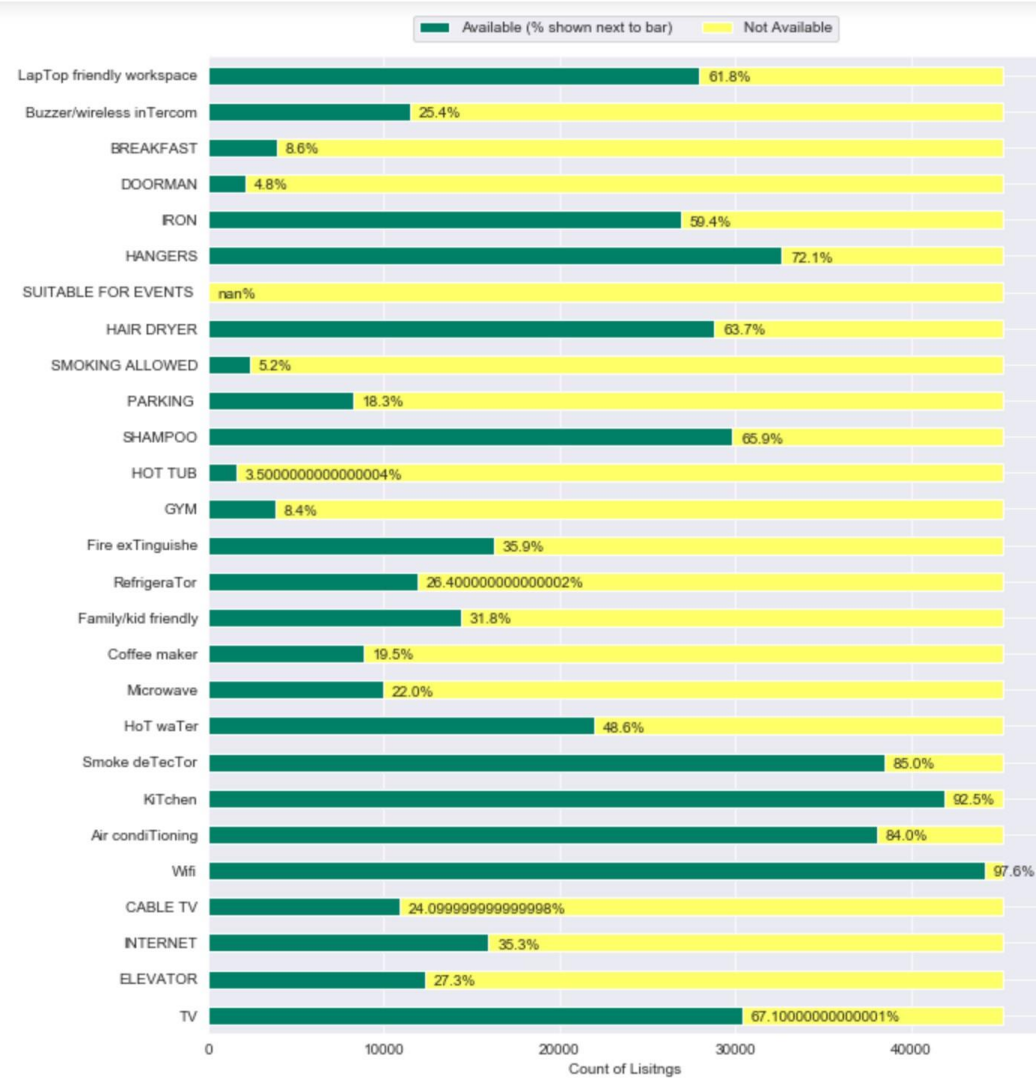
```

```

mainlist_amenities = pd.concat([main_list[amenity].value_counts() for amenity in amenities], axis=1)
ax = mainlist_amenities.sort_index(ascending=False).transpose().plot(kind='barh', stacked=True, figsize=(10,12), colormap='seismic')
ax.legend(['Available (% shown next to bar)', 'Not Available'], loc='upper center', ncol=2, bbox_to_anchor=(0.5, 1.05))
plt.xlabel('Count of Listings')
rects = ax.patches
# Adding Labels
labels = np.round(mainlist_amenities.transpose()[1]/mainlist_amenities.transpose().sum(axis=1), decimals=3)*100

for rect, label in zip(rects, labels):
    ax.text(x=rect.get_x()+rect.get_width()+500, y=rect.get_y(), s=str(label)+'%', ha='left', va='bottom')

```



The above plot shows the percentage of listings having each amenity.

DATA DICTIONARY:

The following table shows the data description, type and definition of the variables involved in the analysis

Variable Name	Type	Example	Description
ID	int	2515	This represents the unique ID for each listing in Airbnb.
NEIGHBOURHOOD_GROUP_CLEANS	Category	Manhattan	Indicates the neighbourhood groups in which the listing located
CITY	Factors	Brooklyn	City in which listing is located
STATE	Factors	NY	State in which listing is located
ZIPCODE	Factors	11218	Zipcode of the area, listing is located
LATITUDE	Float64	40.64748608	Latitude position of Listing, which will be used to find out nearby facilities to Listing
LONGITUDE	Float64	-73.97236954	Longitude position of Listing, which will be used to find out nearby facilities to Listing
PROPERTY_TYPE	Category	Apartment Guest suite	Defines the type of property of the listing.
ROOM_TYPE	Category	Private Room Entire Home	Defines the part of the listing which is available for rent like single room or entire home.
ACCOMMODATES	Int64	3	Indicates the number of people the listing can accommodate.
BATHROOMS	Float64	2	Number of bathrooms in each listing
BEDROOMS	Float64	1	Number of bedrooms available in each listing
BEDS	Float64	2	Number of beds in each listing
BED_TYPE	Category	Pull-out Sofa Real Bed	Type of bed available
AMENITIES	Object	TV,Wi-fi, Parking, Bathtub,Hairdryer	List of Amenities available in the listing
PRICE	Int64	59	Price of listing for each night
GUESTS_INCLUDED	Int64	2	Number of guests that are included for each booking
EXTRA_PEOPLE	Int64	3	Number of extra people that are allowed along with the guest for each booking
MINIMUM_NIGHTS	Int64	2	Minimum number of nights to book the listing
MAXIMUM_NIGHTS	Int64	10	Maximum number of nights a guest can book the listing
NUMBER_OF_REVIEWS	Int64	96	Number of reviews for each listing on the Airbnb
INSTANT_BOOKABLE	Category	t,f	Whether a customer can book the listing instantly, True or False
CANCELLATION_POLICY	Category	strict moderate flexible	defines the cancellation policy for the guests for the each booking
CALCULATED_HOST_LISTINGS_COUNT	Int64	4	How many listings does each host handles or own

MODELING TECHNIQUES:

Linear Regression:

The dependent variable chosen is price and the rest are independent variables. Since the range of values were huge we had to convert the numerical values to categorical values. For example, the variable `number_of_reviews` have lowest value of 0 and highest value of 401. This sort of trend was observed in `minimum_nights` too.

Clustering:

In this part we take our numeric variable and try to identify patterns in unsupervised manner. Here the Kmeans clustering algorithm was used. Also, we tune the value of k and find the optimum number of clusters. 11 variables that describes the distance from the 11 facilities are the input variables. The motive was to see how price varies as the distance varies. it is based on unsupervised learning.

Association Rules:

In this part we used Orange package in python by so the Apriori algorithm was implemented. To use this model, we need to have the variables to be discrete and since all the amenities features are either 0 or 1 association rule mining was used.

Assumptions

The key assumptions in Linear Regression are

- The relationship between the dependent variable and independent variables is considered linear
- All the variables should be multivariate normal
- There should be no or little multi- collinearity
- There should be little or no autocorrelation
- The data is always homoscedastic
- There are no outliers in the data (they have been removed in the preparation step)

The assumptions of clustering are

- The clusters are spherical
- Clusters are of same size

DATA SPLITTING AND SUB SAMPLING:

Data splitting is the process of partitioning the data into parts. This is done for us to split the data into training and testing. We have considered the split ratio to be 70-30 rather than 50-50 because we need to train the model with the majority of the data (70%) for it to have enough data to properly find out the key trends in the models. This is done for the parameters to be fine-tuned and to avoid over fitting. If the 50-50 partition is considered, then there won't be enough data for the model to be trained.

MODEL BUILDING:

Recursive Feature Elimination:

After getting rid of the non-significant variables we are left with 52 variables for modelling. We use the concept of Recursive feature elimination and eliminate variables one by one which also decreases mean square error.

A graph has been plotted for number of features vs Mean Square Error for better understanding

```

from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.metrics import mean_squared_error

num_features = df.shape[1]-1 #Start from num of features = 52
y_true = df['price']
X=df.ix[:, 1:]
mse_dict = {}
while num_features>0: # Iteratively Reduce number of features
    linear_reg_model = LinearRegression(fit_intercept = True)
    selector = RFE(linear_reg_model, n_features_to_select=num_features)
    selector = selector.fit(X, y_true) # Find the set of best n features

    # Reduce train_dataset X to the selected features only
    X=selector.transform(X)

    # Fit model and find MSE error
    linear_reg_model.fit(X, y=y_true)
    y_pred = linear_reg_model.predict(X)
    mse_dict[num_features] = mean_squared_error(y_true, y_pred)
    num_features-=1

from matplotlib import pyplot as plt
%matplotlib inline
mse_df = pd.DataFrame(mse_dict.items(), columns=['num_parameters','mse'], index=mse_dict.keys())
plt.plot(mse_df['mse'])
plt.grid()
plt.xlabel('Number of features')
plt.ylabel('Mean Squared Error')

```

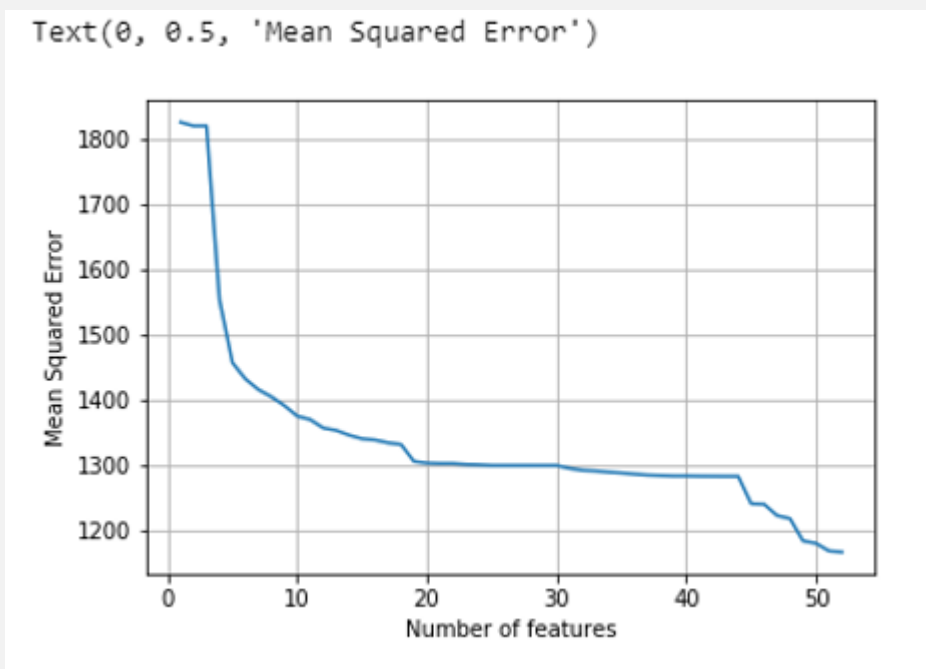


Fig.9.1 Graph of Number of features vs MSE

When only one feature is considered the MSE is 2754.

From the graph we can see that the MSE decreases as the number of features increases. This trend is not significant after a certain number of features. A stagnant line is observed from 14 to 20 number of features and beyond 21 the decrease in MSE is not very significant as it tends to become a straight line parallel to x-axis. The MSE from the model at number of features = 21 is 1424.

According to the graph it can be noticed that most of the neighborhood facility features are not significant as the features after 20th column were neighborhood facilities.

This is an interesting result because our initial hypothesis was that the Price is greatly affected by the number of facilities that is close to the place. And this idea was got from the fact that the real estate pricing of a place would increase if it has more amenities nearby, since it easier for people to access these amenities. However, it can be interpreted form the graph that none of the amenities really matter for the price.

Linear Regression:

- Some variables which were categorical were combined into groups. Like for property_type only the common categories (house, apartment, condominium) others were categorized as Others and bed_type was converted into common types (as Couch, Airbed, Pull-out sofa, etc. were grouped together and named 'Other').
- The concept of One-Hot encoding was for categorical variables with more than one category.
- Statsmodels Library was used instead of sklearn since statsmodels outputs from the model and co-efficient significance along with other evaluation matrices. Model fitting was done using ordinary least square.

Importance of the coefficients:

Variable	Coefficient	Interpretation
Bedrooms	19.1	Unit increase in bedroom results in additional charges of \$19.1 in rent
Elevatorinbuilding	8.2	Presence of elevator indicates that the price would increase by \$8.2
minimum_nights_MN1	2.9	If the minimum number of nights is greater than 3, for example 7, the host is likely to give a discounted price. This explains the negative coefficient. Apart from that, the price is increased more if the minimum number of nights required is 2 rather than 1
minimum_nights_MN2	4.8	
minimum_nights_MN3	4.1	
minimum_nights_MN_	-2.4	
instant_bookable_f	7.6	This price goes up for properties which does not need an instant bookable feature by \$7.6
instant_bookable_t	1.7	
cancellation_policy_flexible	2.8	The cancellation policy is stricter for most of the established and renowned poperties owned by real estate owners. These properties have higher prices. These properties have higher prices. The model explains this by having a greater coefficient of 4.5 for strict cancellation policy
cancellation_policy_moderate	2.1	
cancellation_policy_strict	4.5	
bed_type_Other	2.1	There are lot of bed types including couches, hence the price increases by \$7.2 for real beds and \$2.1 for other types
bed_type_Real_Bed	7.2	
room_type_Entire_home/apt	36.6	The price increases by \$36.6 if the property type is entire house or apartment. The prices are usually higher for entire apartment. Similarly, for shared room, the price goes down by \$22.2
room_type_Private_room	-5.1	
room_type_Shared_room	-22.2	
calculated_host_listings_count_M	3.3	Multiple listings are maintained by real estate owners and they have competitive price when compared to normal people. Hence price goes up by \$5.9 if the number of host listing is one
calculated_host_listings_count_S	5.9	

Fig.9.2 Important Variables

Important variables for the Price to vary is got from the table above.

The most important variable is Bedrooms, because for a unit change in bedroom there 19.1\$ increase in the price.

Regression Equation:

$$\begin{aligned}
 \text{Price} = & 9.3 + 19.1 * (\text{bedrooms}) + 8.2 * (\text{elevator}) + 2.9 * (\text{min_MN1}) \\
 & + 4.8 * (\text{min_nights_MN2}) + 4.1 * (\text{min_nights_MN3}) + 2.4 * (\text{min_nights_MN}) \\
 & + 7.6 * (\text{instant_bookable f}) + 1.7 * (\text{instant_bookable t}) + 6.6 * (\text{num_page_saved H}) \\
 & + 14.1 * (\text{num_saved L}) + 1.9 * (\text{num_page_saved M}) \\
 & + 2.8 * (\text{cancellation_policy_flexible}) + 2.1 * (\text{cancellation_policy_moderate}) \\
 & + 4.5 * (\text{cancellation_policy_strict}) + 2.1 * (\text{bed_type_other}) + 7.2 * (\text{bed_type_real}) \\
 & + 36.6 * (\text{room_type_entire_home}) + 5.1 * (\text{room_type_private}) \\
 & + 22.2 * (\text{room_type_shared}) + 3.3 * (\text{calculated_host_listings_count M}) \\
 & + 5.9 * (\text{calculated_host_listings_count S})
 \end{aligned}$$

Clustering:

The data must be normalized; this is done because different variables have different ranges. Mix and match normalization is used with the formula below.

$$X_{\text{normalized}} = (X - X_{\text{min}}) / (X_{\text{max}} - X_{\text{min}})$$

X_{max} . maximum value in column.

X_{min} . minimum value in column.

```
from sklearn.cluster import KMeans
inertia_dict = {}
for n in range(1,11):
    cluster_model = KMeans(n_clusters=n, init='k-means++', random_state=13)
    cluster_model.fit(df)
    inertia_dict[n] = cluster_model.inertia_
```

inertia_dict # Steepest slope is from 1 to 2. So 2 is the optimal number of clusters

```
{1: 3674.6250903914543,
 2: 2078.7167437297094,
 3: 1733.4498839613684,
 4: 1519.9381643339473,
 5: 1343.3754904568329,
 6: 1230.4384869370087,
 7: 1136.848413082595,
 8: 1060.136929887839,
 9: 1009.6786807705128,
10: 966.20818079913488}
```

Number of Clusters	1	2	3	4	5	6	7	8	9	10
Inertia	3674	2078	1733	1519	1343	1230	1136	1060	1009	966

Fig.9.3 Inertia for different k values

This section explains about how the numeric values are used in an unsupervised manner to identify patterns. There are 11 input variables based on the distance of neighborhood facilities. This was done on the idea that the listings could be grouped based on the distance. And since the neighborhood facilities have been eliminated in the regression because of the recursive feature elimination we thought

that they could be clustered together with price to get different clusters. The two distinct clusters, first one is the houses near the facilities and other one is houses away from the facilities.

K means Clustering

All the variables considered are of numeric values. We plot a graph for number of clusters vs inertia. As the number of clusters varied from 1 to 10 we can see that the inertia decreases. But this decrease is significant only till the number of clusters is 2. Beyond 2 the line tends to be parallel with the x axis and so they can be considered non-significant. Here there are two clusters one is house near the facilities (price is lower) and other is house where the facilities are far away (price is lower).

```
In [9]: ax = pd.Series(inertia_dict.values(), index=inertia_dict.keys()).plot(marker='o')
ax.set(xlabel='Number of Clusters (k)', ylabel='Inertia')
```

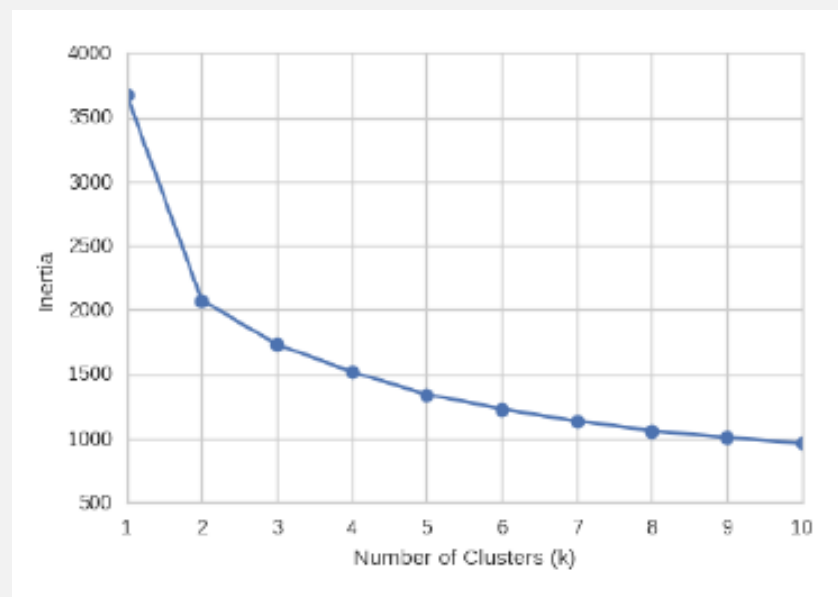


Fig.9.4 Graph Number of clusters vs Inertia

Variable	Cluster	
	1	2
price	80	110
restaurant	533	202
atm	1593	760
cinema	2630	1339
hospital	3882	1607
nightclub	8253	1544
park	1785	1747
mall	7024	2204
gallery	11459	2411
museum	6111	1277
supermarket	756	323
bus_stop	194	186

Fig.9.5 Cluster distance table

We consider cluster 1 is cheap and cluster 2 is expensive

From the table we can interpret that the cluster 2 has amenities are at a closer distance that the cluster 1.

For example, restaurant is 202 meters on avg in cluster 2(expensive) and is 533 meters on avg in cluster 1 (Cheaper).

```
In [12]: X = pd.DataFrame(X, columns=['dimension_1', 'dimension_2'])
X['label'] = cluster_model.labels_

In [13]: ax = sns.lmplot(x='dimension_1', y='dimension_2', data=X, fit_reg=False, hue='label', legend=False, palette='Set1')
plt.legend(['Cluster_1 (Listings with facilities nearby)', 'Cluster_2 (Listings with facilities far away)'])
```

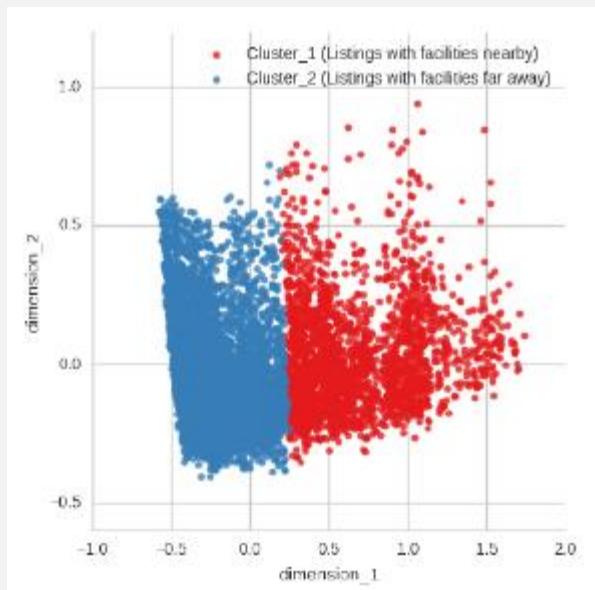


Fig.9.6 Scatter plot based on PCA

From the graph it can be interpreted that the two clusters are distinct.

Association Rule:

Since all the amenities is either 0 or 1 we can easily use Association Rule. This is to identify which set of amenities are concurrently provided by the hosts. For example, with hot water available (hot_water=1) then Hot Tub(hot_tub=1) is also available. And find out what other amenities occur together.

```
In [3]: # We only require the columns which are in-house amenities (which are binary) and
# the room type (Full House/Private Room/Shared Room)
# and the property type (Condominium/House/Apartment)
# Subset the data to include only these columns
cols = [u'Elevator in building', u'Internet', u'Family/kid friendly', u'Wireless Internet',
        u'Buzzer/wireless intercom', u'Kitchen', u'Doorman', u'Wheelchair accessible',
        u'Cable TV', u'Hot tub', u'Gym', u'Pool', u'TV', u'Dryer', u'Washer', u'Essentials', u'Shampoo',
        u'Heating', u'Air conditioning', u'Pets allowed', u'Suitable for events',
        u'Smoking allowed', u'Indoor fireplace', u'Breakfast', u'Laptop friendly workspace', u'Iron', u'Hangers',
        u'Hair dryer', u'Private living room', u'Private entrance']#, 'property_type', 'room_type']
df = df.ix[:,cols]
```

Listing ID	Elevator	Gym	Heating	Air-conditioning
1	1	1	1	0
2	0	1	1	1
3	0	0	1	1
4	1	0	0	1

Fig.9.7 Amenities in matrix format

Listing ID	Amenities
1	Elevator, Gym, Heating
2	Gym, Heating, Air-conditioning
3	Heating, Air-conditioning
4	Elevator, Air-conditioning

Fig9.8 Amenities in grouped format

```
In [11]: # Get frequent itemsets
data = Orange.data.Table("association-mining/airbnb-amenities.basket")

s_threshold = 0.5
ind = Orange.associate.AssociationRulesSparseInducer(support=s_threshold, storeExamples = True) # Specify
the support threshold
itemsets = ind.get_itemsets(data)

print "Number of Frequent Itemsets that satisfy the support threshold of (%4.2f) is %d" % (s_threshold, len(itemsets))

Number of Frequent Itemsets that satisfy the support threshold of (0.50) is 578

In [12]: for itemset, tids in itemsets[:15]: # First 15 itemsets
print "(%4.2f) %s" % (len(tids)/float(len(data)),
                    ", ".join(data.domain[item].name for item in itemset))
```

Support	Confidence	Rule
0.51	0.95	Iron, Kitchen, Wireless Internet → Essentials
0.54	0.91	Laptop friendly workspace → Hangers
0.76	0.98	Air conditioning, Kitchen → Heating
0.85	0.98	Essentials, Wireless Internet → Heating

Fig 9.9 Association Rules with Property/ Room Type

From the support and confidence values of the set of amenities, we can say that owners who want to be competitive in terms of rent and the amenities provided, if the host has Iron, kitchen and wireless internet at the house and has the budget to invest on one more amenity, he can choose essentials such as bedsheet and pillow covers to be more aligned with his competitors. This can be said with 95% confidence.

EVALUATION

Linear Regression

OLS Regression Results

Dep. Variable:	price	R-squared:	0.584
Model:	OLS	Adj. R-squared:	0.582
Method:	Least Squares	F-statistic:	200.8
Date:	Mon, 29 Apr 2019	Prob (F-statistic):	0.00
Time:	17:51:42	Log-Likelihood:	-45533.
No. Observations:	9202	AIC:	9.120e+04
Df Residuals:	9137	BIC:	9.166e+04
Df Model:	64		
Covariance Type:	nonrobust		

Fig 10.1 OLS Regression Result

From the summary R-squared is 0.584 and F-statistic is 200.8 that's larger than F-stat on 5% alpha. So it can be concluded that the model is significant.

Verifying the assumptions of Linear Regression:

Residuals plots:

```
In [36]: linear_reg_model.fit(X=df_new.ix[:,1:], y=y_true)
y_pred = linear_reg_model.predict(df_new.ix[:,1:])
residuals = y_true-y_pred
plt.hist(residuals, bins=50)
plt.grid()
plt.ylabel("Frequency")
plt.xlabel("Residuals")
print "The residuals are approximately normal distributed with mean = ", residuals.mean()
```

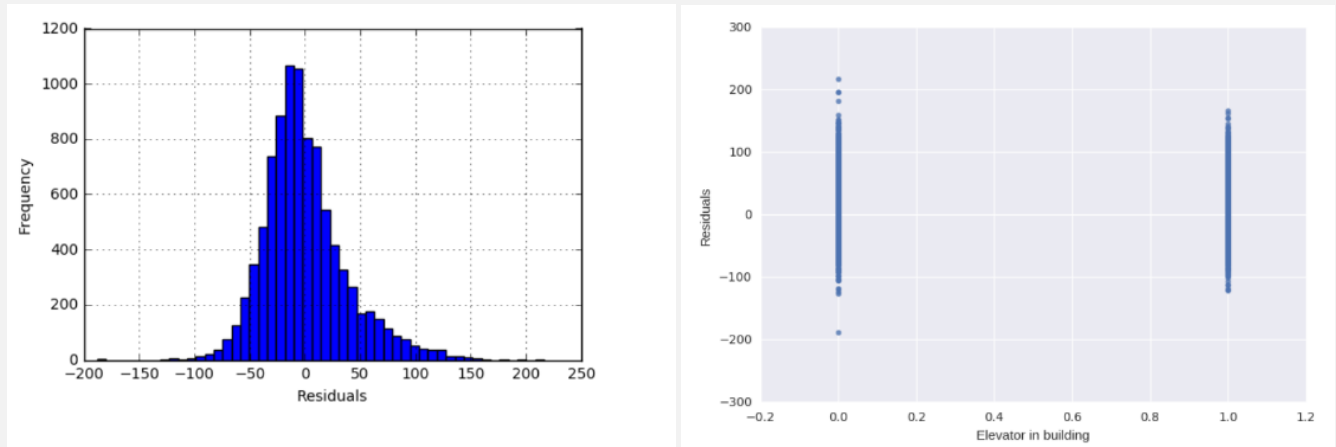



Fig10.2 Residual plots

Residual Histograms: From the residual histograms we can see that the mean is approximately normally distributed with a mean value of -0.0017. The graph and the mean value is approximately equal to 0, which can be used to evaluate and conclude that there is no violation of assumptions.

Residuals vs X: We plotted elevator in building against the residuals to check if the value is either 0 and 1. It does not show any pattern from which we can learn that there's no over/underestimation.

The model was also evaluated for Linear Regression using KFold Cross validation, with 10 folds. The results are promising since the average cross validation was 1429 is closer to the cross validation when entire data set is used for test and train which is 1424. It can be concluded that there is no overfitting

Fold	1	2	3	4	5	6	7	8	9	10	Avg
MSE	1472	1395	1392	1440	1536	1281	1499	1390	1407	1460	1429

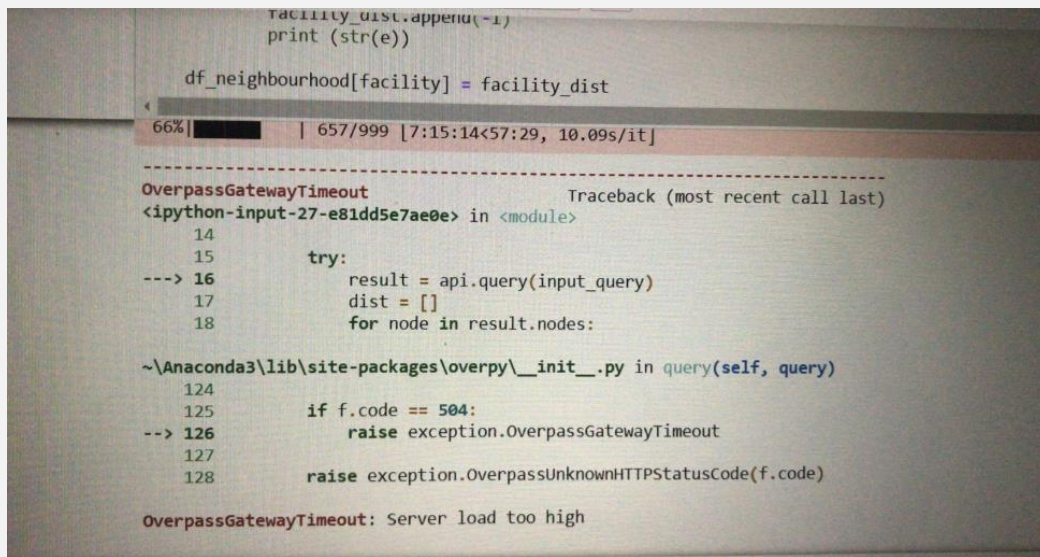
Fig.10.2 Cross Validation for each fold

Model Evaluation for Clustering:

From the model scatter plot it can be clearly seen that there are only 2 clusters, one can be labelled cheap and other one can be labelled expensive. This provides a good result in terms of clusters when treated as unsupervised machine learning problem

PROBLEMS AND CONSTRAINTS:

- We faced issues with large datasets while mapping distance to the neighborhood using OpenStreet API
- The team had to split the dataset into smaller chunks and run multiple iterations
- Though we ran smaller chunks each iteration consumed lot of time and failed multiple times



```
facility_dist.append(-1)
print (str(e))

df_neighbourhood[facility] = facility_dist

66%|██████████| 657/999 [7:15:14<57:29, 10.09s/it]

-----
OverpassGatewayTimeout                                Traceback (most recent call last)
<ipython-input-27-e81dd5e7ae0e> in <module>
    14
    15     try:
--> 16         result = api.query(input_query)
    17         dist = []
    18         for node in result.nodes:

~\Anaconda3\lib\site-packages\overpy\__init__.py in query(self, query)
    124
    125         if f.code == 504:
--> 126             raise exception.OverpassGatewayTimeout
    127
    128             raise exception.OverpassUnknownHTTPStatusCode(f.code)

OverpassGatewayTimeout: Server load too high
```

Fig 11.1 Output screen shot

- So we decided to limit the number of rows around 9700 and proceeded with the analysis.
- 9700 rows consumed 48 hours continuously to run and give the distance values.

CONCLUSION

The primary aim of the project was to build a predictive model for the owners to help them determine the rental price of the home they wish to rent. The model was based on the hypothesis that rental price increases with decrease in distance between the house and neighborhood facilities. This hypothesis was proven wrong by linear regression model as the neighborhood was not significant predictors of price. On the other hand, the hypothesis was proved right by clustering algorithm as the price was more if the neighborhood distance was less and the price was less when the neighborhood was far from the house. So it wholly depends on the technique used to study the problem