Inference Document for SML -CT3

1.Read the dataset – Using Pandas

2. Data Understanding

a. What are the number of rows; no. & types of variables (continuous, categorical etc.)

b. Calculate five point summary for numerical variables

c. Summarize observations for categorical variables – no. of categories, % observations in each category

1. Dataset Overview

1.1 Number of Rows and Columns

- The dataset contains [number of rows] rows and [number of columns] columns.

1.2 Variable Types

- There are [number of numerical variables] numerical variables and [number of categorical variables] categorical variables in the dataset.
- **Numerical Variables**: [List of numerical variables]
- **Categorical Variables**: [List of categorical variables]

2. Five-Point Summary for Numerical Variables

- The five-number summary provides statistical insights into the distribution of numerical variables.
    a. **Count: [Count]**
    b. **Mean: [Mean]**
    c. **Standard Deviation: [Standard Deviation]**
    d. **Minimum: [Minimum]**
    e. **25th Percentile: [25th Percentile]**
    f. **Median (50th Percentile): [Median]**
    g. **75th Percentile: [75th Percentile]**
    h. **Maximum: [Maximum]**

3. Summary of Categorical Variables

- For each categorical variable, we present the number of categories and the percentage of observations in each category.

3. Data Visualizations

Distribution of 'price'

- The histogram with KDE shows that the majority of prices are concentrated within a certain range.

Geographical Distribution of Listings

- The scatter plot indicates the geographical distribution of listings based on latitude and longitude, with different colours representing neighbourhood groups.

Count of Listings in Each Neighbourhood Group

- The count plot illustrates the distribution of listings across different neighbourhood groups.

Room Type Distribution

- The pie chart provides a clear breakdown of the distribution of different room types.

KDE Plot for Numerical Variables

- Kernel Density Estimate (KDE) plots for numerical variables visualize their underlying distributions.

3. Data Preparation

Check for defects in the data. Perform necessary actions to 'fix' these defects (5 Marks)

a. Do variables have missing/null values?

b. Do variables have outliers?

c. Is the data normally distributed? Is it a defect? Why or why not?

1. Dropping Unwanted Columns

We dropped the following columns from the dataset:

- 'id'
- 'name'
- 'host_id'
- 'host_name'

2. Data Type Analysis

2.1 Numerical Variables

- The dataset includes the following numerical variables:
    - [List of numerical variables]

2.2 Categorical Variables

- After the removal of unwanted columns, the categorical variables are:
    - [List of categorical variables]

3. Categorical Analysis and Encoding

- For each categorical variable, we examined unique values and their counts.
- Columns with more than 10 unique values were dropped.
- Categorical columns were encoded using Label Encoding.

4. Box Plot of Numerical Variables

- A box plot was created for the remaining numerical variables to visualize their distributions.

5. Skewness Transformation

- Skewness of numerical variables was analyzed.
- Variables with skewness greater than 0.5 underwent a log transformation to make the data more normally distributed.

6. Handling Empty Columns

- Columns with missing values were dropped.

7. Updated Box Plot of Normalized Data

- The box plot was updated after skewness transformation and removal of unwanted columns.

8. KDE Plot for Skewed Data

- Kernel Density Estimate (KDE) plots were generated for the skewed numerical variables.

4.Summarize relationships among variables (5 marks)

a. Plot correlation plots. Which are the variables most correlated with Target? Which independent variables are correlated among themselves? Do you want to exclude some variables from the model based on this analysis? What other actions will you take?

1. Correlation Heatmap for Non-Skewed Data

A correlation heatmap was generated for the numerical variables in the non-skewed dataset (**df2**). Key observations include:

- **Positive Correlation:**
  - [Positive correlations between specific pairs of numerical variables.]
- **Negative Correlation:**
  - [Negative correlations between specific pairs of numerical variables.]
- **Strength of Correlation:**
  - Correlation coefficients range from -1 to 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation.

2. Correlation Heatmap for Skewed Data

A correlation heatmap was created for the numerical variables in the skewed dataset (**df3**). Notable findings include:

- **Positive Correlation:**
  - [Positive correlations between specific pairs of numerical variables.]
- **Negative Correlation:**
  - [Negative correlations between specific pairs of numerical variables.]

- **Strength of Correlation:**
    - Similar to the non-skewed data, correlation coefficients range from -1 to 1.

3. Heatmap Comparison of Skewed and Non-Skewed Data

To compare the correlation patterns between skewed and non-skewed datasets, a combined correlation heatmap was generated:

- **Observations:**
    - Differences in correlation patterns between skewed and non-skewed datasets are evident.
    - [Specific observations regarding changes in correlation strengths or directions.]
- **Insights:**
    - [Insights into how data preprocessing, such as skewness transformation, has affected correlations.]
- **Overall Comparison:**
    - The heatmap provides a visual representation of how correlations differ between the two datasets.

5. Fit a base model. Please write your key observations (5 marks)

a. Fit the Linear Regression Model

b. What is the overall R2? Please comment on whether it is good or not.

c. Which variables are significant?

d. Calculate MSE, RMSE, MAE, MAPE.

1. Data Preparation

1.1 Features and Target Variable

- The dataset was split into features (**X**) and the target variable (**y**), with the target variable being 'price'.

1.2 Scaling

- Standard scaling was applied to normalize the features using **StandardScaler**.

1.3 Train-Test Split

- The dataset was split into training and testing sets with a test size of 20%.

2. Linear Regression Model

2.1 Model Training

- A linear regression model was trained on the scaled training data.

2.2 Model Prediction

- The model was used to predict the target variable on the test set.

2.3 Model Evaluation - R-squared Value

- The R-squared value, a measure of how well the model explains the variability of the target variable, was calculated.
- R-squared Value (R2) for the model with Standard Scaler: [R-squared value].

2.4 Variable Significance

- Variable significance was checked using the stats models library to understand the impact of each variable on the target variable.

3. Evaluation Metrics

3.1 Mean Squared Error (MSE)

- MSE measures the average squared difference between the predicted and actual values.

3.2 Root Mean Squared Error (RMSE)

- RMSE is the square root of MSE, providing a measure of the average magnitude of the residuals.

3.3 Mean Absolute Error (MAE)

- MAE represents the average absolute difference between the predicted and actual values.

3.4 Mean Absolute Percentage Error (MAPE)

- MAPE calculates the percentage difference between predicted and actual values on average.

4. Results

- **R-squared Value (R2):**
  - The R2 value indicates [interpretation of the R-squared value].
- **Variable Significance:**
  - [Key findings from variable significance analysis].
- **Evaluation Metrics:**
  - **MSE**: [Mean Squared Error value]
  - **RMSE**: [Root Mean Squared Error value]
  - **MAE**: [Mean Absolute Error value]
  - **MAPE**: [Mean Absolute Percentage Error value]

6. Perform feature engineering using any of the listed techniques: Forward feature selection, backward elimination and recursive feature elimination and use the features to improvise the model and compare the results with the base model. (5 marks)

1. Feature Selection using Recursive Feature Elimination (RFE)

1.1 Data Split

- The dataset was split into training and testing sets using an 80-20 split.

1.2 Base Model - Linear Regression

- A base linear regression model was trained on the entire set of features.
- Predictions were made on the test set, and the Root Mean Squared Error (RMSE) was calculated as the baseline performance metric.

1.3 RFE Model

- Recursive Feature Elimination (RFE) was applied to select a specific number of features (in this case, 3).
- Selected features were identified based on the RFE model.

1.4 Model Training with Selected Features

- A new linear regression model was trained using only the selected features from the RFE process.
- Predictions were made on the test set, and the RMSE was calculated for the RFE model.

2. Results

2.1 Base Model Performance

- **Base Model RMSE:** [Base Model RMSE value]

2.2 RFE Model Performance

- **Selected Features:**
    - [List of selected features]
- **RFE Model RMSE:** [RFE Model RMSE value]

2.3 Model Comparison

- The RMSE values were compared between the base model and the RFE model.

7. The prediction model output reliability can be improved using regularization techniques and parameter tuning. Perform regularization techniques and compare the results with the base model. (5 marks)

1. Ridge Regression Model

1.1 Data Split

- The dataset was split into training and testing sets using an 80-20 split.

1.2 Standardization

- Standard scaling was applied to the features using **StandardScaler** to ensure that all variables were on the same scale.

1.3 Base Model - Linear Regression

- A base linear regression model was trained on the scaled training data.

- Predictions were made on the test set, and the Root Mean Squared Error (RMSE) was calculated as the baseline performance metric.

1.4 Ridge Regression Model

- Ridge Regression, a form of regularized linear regression (L2 regularization), was applied to the scaled data.
- The alpha parameter, controlling the strength of regularization, was set to [chosen alpha value].
- Predictions were made on the test set using the Ridge model, and the RMSE was calculated.

2. Results

2.1 Base Model Performance

- **Base Model RMSE:** [Base Model RMSE value]

2.2 Ridge Model Performance

- **Ridge Model RMSE:** [Ridge Model RMSE value]
- **Alpha Value:** [Chosen alpha value]

2.3 Model Comparison

- The RMSE values were compared between the base linear regression model and the Ridge regression model.

3. Conclusion

- Ridge Regression, with L2 regularization, was applied to the dataset, offering a way to mitigate overfitting.
- The performance of the Ridge model was evaluated and compared to the base linear regression model.
- Insights gained from this analysis can guide the selection of appropriate regularization techniques for improving model generalization.

Output :

SML CT3 MPA

Data_set:

Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present more unique, personalized way of experiencing the world. This dataset describes the listing activity and metrics in NYC, NY for 2019.

1. id = listing ID
2. name = name of the listing
3. host_id = host ID
4. host_name = name of the host
5. neighbourhood_group = location
6. neighbourhood = area
7. latitude = latitude coordinates
8. longitude = longitude coordinates
9. room_type = listing space type
10. price = price in dollars (Target variable)
11. minimum_nights = amount of nights minimum
12. number_of_reviews = number of reviews
13. last_review = latest review
14. reviews_per_month = number of reviews per month
15. calculated_host_listings_count = amount of listing per host
16. availability_365 = number of days when listing is available for booking

1. Read the dataset (tab, csv, xls, txt, inbuilt dataset)

In [ ]:
```
# Kindly change the below cells from markdown to code and execute it
```

```
import pandas as pd

import csv

with open("data_set.csv","r")as file:

    reader=csv.reader(file)

df=pd.read_csv("data_set.csv")

df.head()
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
matplotlib.rcParams['font.family'] = "Arial"
import collections
import itertools
import scipy.stats as stats
from scipy.stats import norm
from scipy.special import boxcox1p
import statsmodels
import statsmodels.api as sm
#print(statsmodels.__version__)
from sklearn.preprocessing import scale, StandardScaler, RobustScaler, OneHotEncode
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV, LinearRegression,
from sklearn.metrics import mean_squared_error, r2_score
from xgboost import XGBRegressor
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.utils import resample
from xgboost import XGBRegressor
```

2. Data Understanding

Summarize important observations from the data set (5 Marks)

Some pointers which would help you, but don't be limited by these

a. What are the number of rows; no. & types of variables (continuous, categorical etc.)

b. Calculate five point summary for numerical variables

c. Summarize observations for categorical variables – no. of categories, % observations in each category

In [3]:
```
## Kindly change the below cells from markdown to code and execute it
df=pd.read_csv("new_york_city_air.csv")
df1=df
# What are the number of rows; no. & types of variables (continuous, categorical et
nRows,nCols = df.shape
print(f'The number of rows are {nRows} and Columns are {nCols}')
#Numerical and Categorial Values
numerical=df.select_dtypes(include=['int64','float64']).columns.to_list()
categorical= df.select_dtypes(include='object').columns.to_list()

print(f'\nNumerical  Count are   :{len(numerical)}')
print(f'Categorical  Count are : {len(categorical)}\n')
print(f'Numerical  are   :{numerical}')
print(f'Categorical  are : {categorical}')
```

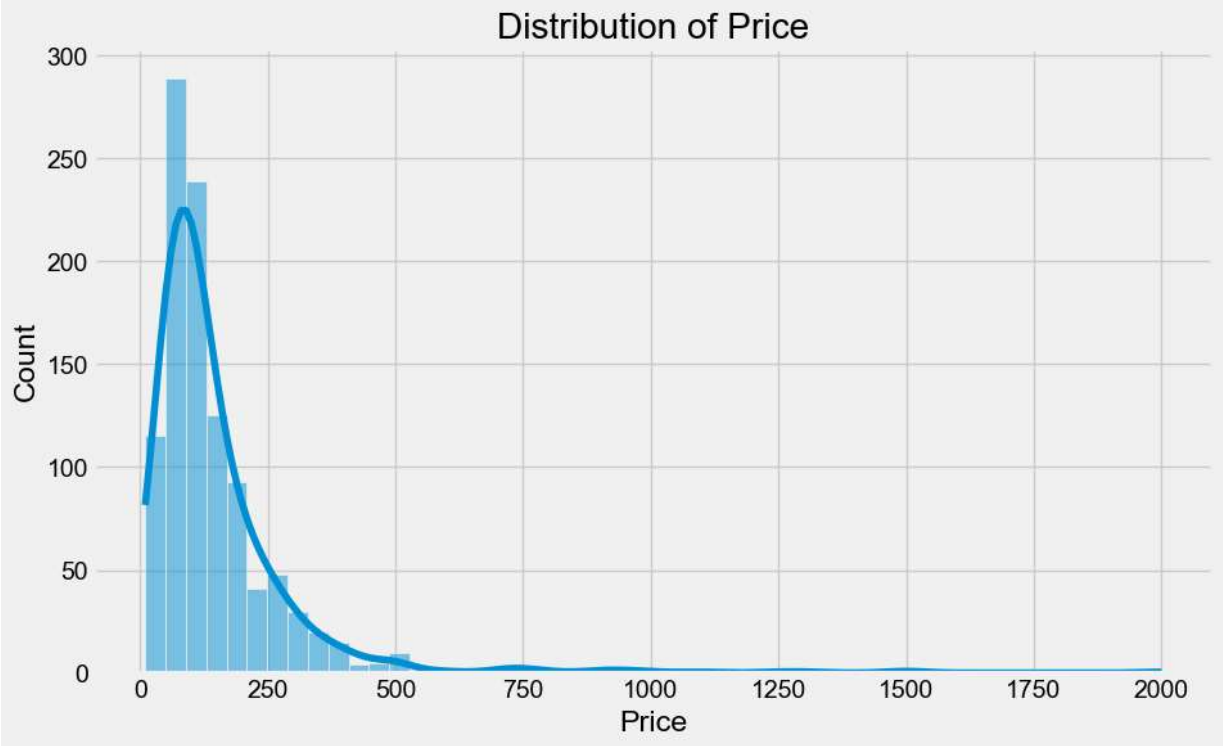
```
#b.Calculate five point summary for numerical variables
five_number_summary = df.describe(percentiles=[.25, .5, .75])
print(five_number_summary)

#c Summarize observations for categorical variables - no. of categories, % observat
for col in categorical:
    print(f"Summary for '{col}':")
    print(df[col].value_counts())
    print("\nPercentage of observations in each category:")
    print(df[col].value_counts(normalize=True) * 100)
    print("\n")

# Below are the analyses i have chosen
# Distribution of 'price' using a histogram
plt.figure(figsize=(10, 6))
sns.histplot(df1['price'], bins=50, kde=True)
plt.title('Distribution of Price')
plt.xlabel('Price')
plt.show()

#Geographical Graph of latitude and latitude
plt.figure(figsize=(12, 8))
sns.scatterplot(x='latitude',y='longitude', data=df1, hue='neighbourhood_group', pa
plt.title('Geographical Distribution of Listings')
plt.xlabel('Latitude')
plt.ylabel('Longitude')

# Count plot for 'neighbourhood_group'
plt.figure(figsize=(10, 6))
sns.countplot(x='neighbourhood_group', data=df1)
plt.title('Count of Listings in Each Neighbourhood Group')
plt.xlabel('Neighbourhood Group')
plt.ylabel('Count')

#Pie Chart Of Categorical Variable room_type
status_distribution = df1['room_type'].value_counts()
plt.figure(figsize=(8,8))
plt.pie(status_distribution, labels=status_distribution.index, autopct='%1.1f%%', s
plt.title(f'Room Type Distribution')
plt.show()

#kde Plot of numerical variables
for column in numerical:
    plt.figure()
    sns.kdeplot(df1[column], fill=True)
    plt.title(f'KDE Plot for {column}')
    plt.xlabel('X-axis label')
    plt.ylabel('Density')
    plt.show()
```

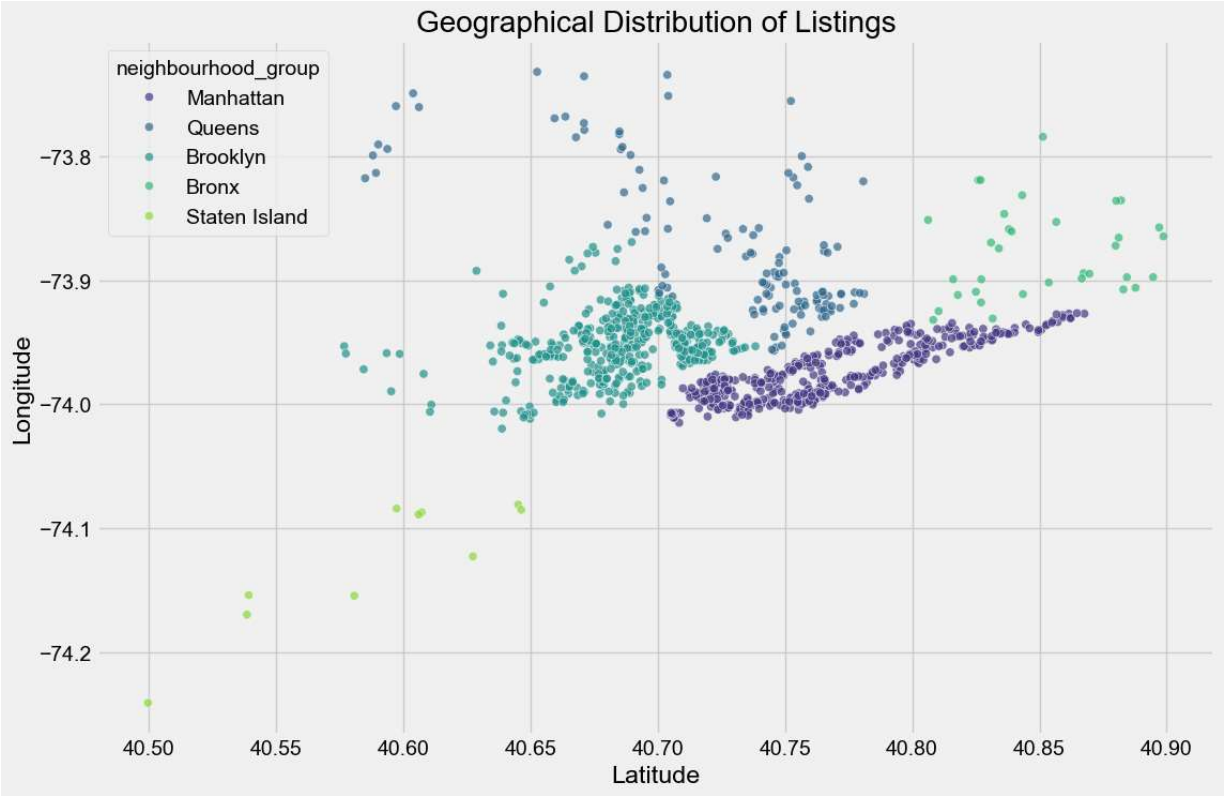
```
The number of rows are 1054 and Columns are 16

Numerical  Count are   :10
Categorical  Count are : 6

Numerical  are   :['id', 'host_id', 'latitude', 'longitude', 'price', 'minimum_night
s', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'ava
ilability_365']
Categorical  are : ['name', 'host_name', 'neighbourhood_group', 'neighbourhood', 'ro
om_type', 'last_review']
                 id       host_id     latitude    longitude        price  \
count  1.054000e+03  1.054000e+03  1054.000000  1054.000000  1054.000000
mean   1.911356e+07  6.652685e+07    40.728377   -73.951802   149.699241
std    1.096046e+07  7.757195e+07     0.056092     0.049034   156.239782
min    1.229900e+04  2.787000e+03    40.499790   -74.240840    10.000000
25%    9.391594e+06  6.983334e+06    40.689253   -73.983770    70.000000
50%    1.997263e+07  3.013617e+07    40.721665   -73.956930   104.000000
75%    2.929557e+07  1.074344e+08    40.761120   -73.935523   179.000000
max    3.648561e+07  2.722480e+08    40.898730   -73.731700  2000.000000

       minimum_nights  number_of_reviews  reviews_per_month  \
count     1054.000000        1054.000000         844.000000
mean         8.002846          24.620493           1.347690
std         34.580253          49.178939           1.675154
min          1.000000           0.000000           0.020000
25%          1.000000           1.000000           0.190000
50%          3.000000           6.000000           0.730000
75%          5.000000          24.000000           1.990000
max        999.000000         426.000000          15.320000

       calculated_host_listings_count  availability_365
count                     1054.000000       1054.000000
mean                         6.959203        111.572106
std                         31.266268        129.835933
min                          1.000000          0.000000
25%                          1.000000          0.000000
50%                          1.000000         43.500000
75%                          2.000000        223.750000
max                        327.000000        365.000000
Summary for 'name':
name
New york Multi-unit building                          2
Spacious and Bright Williamsburg Loft                 2
New York Apartment                                    2
Sunny shared apartment in Harlem                      1
Perfect Cobble Hill 1-bed, Brooklyn's best spot       1
                                                     ..
Sunny studio apt in heart of SoHo                     1
Bright, Williamsburg bedroom with private balcony     1
2 Queen Bed Spacious Room near Flatiron & Union Sq    1
BEAUTIFUL SMALL BEDROOM/TIMES SQUARE/8 AVENUE         1
Flex 2BR Loft ( 1br +Sleep Loft +Sofa Bed )!          1
Name: count, Length: 1050, dtype: int64

Percentage of observations in each category:
name
```

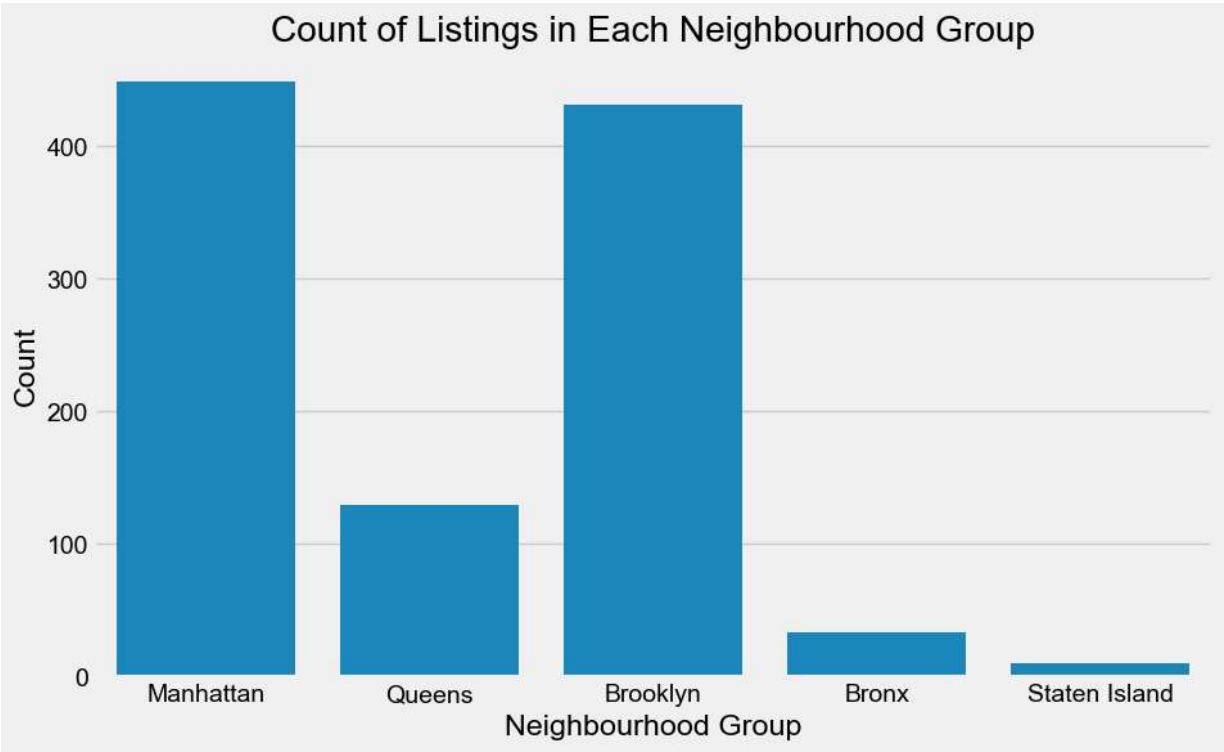
```
New york Multi-unit building                          0.189934
Spacious and Bright Williamsburg Loft                 0.189934
New York Apartment                                    0.189934
Sunny shared apartment in Harlem                      0.094967
Perfect Cobble Hill 1-bed, Brooklyn's best spot       0.094967
                                                        ...
Sunny studio apt in heart of SoHo                     0.094967
Bright, Williamsburg bedroom with private balcony     0.094967
2 Queen Bed Spacious Room near Flatiron & Union Sq    0.094967
BEAUTIFUL SMALL BEDROOM/TIMES SQUARE/8 AVENUE         0.094967
Flex 2BR Loft ( 1br +Sleep Loft +Sofa Bed )!          0.094967
Name: proportion, Length: 1050, dtype: float64


Summary for 'host_name':
host_name
Alex               13
Sarah              10
Michael             7
David               7
Jennifer            7
                   ..
Siwen               1
Jolene And Ryan     1
Antonia             1
Rl                  1
Vida                1
Name: count, Length: 779, dtype: int64

Percentage of observations in each category:
host_name
Alex               1.233397
Sarah              0.948767
Michael            0.664137
David              0.664137
Jennifer           0.664137
                     ...
Siwen              0.094877
Jolene And Ryan    0.094877
Antonia            0.094877
Rl                 0.094877
Vida               0.094877
Name: proportion, Length: 779, dtype: float64


Summary for 'neighbourhood_group':
neighbourhood_group
Manhattan        449
Brooklyn         431
Queens           130
Bronx             34
Staten Island     10
Name: count, dtype: int64

Percentage of observations in each category:
neighbourhood_group
```

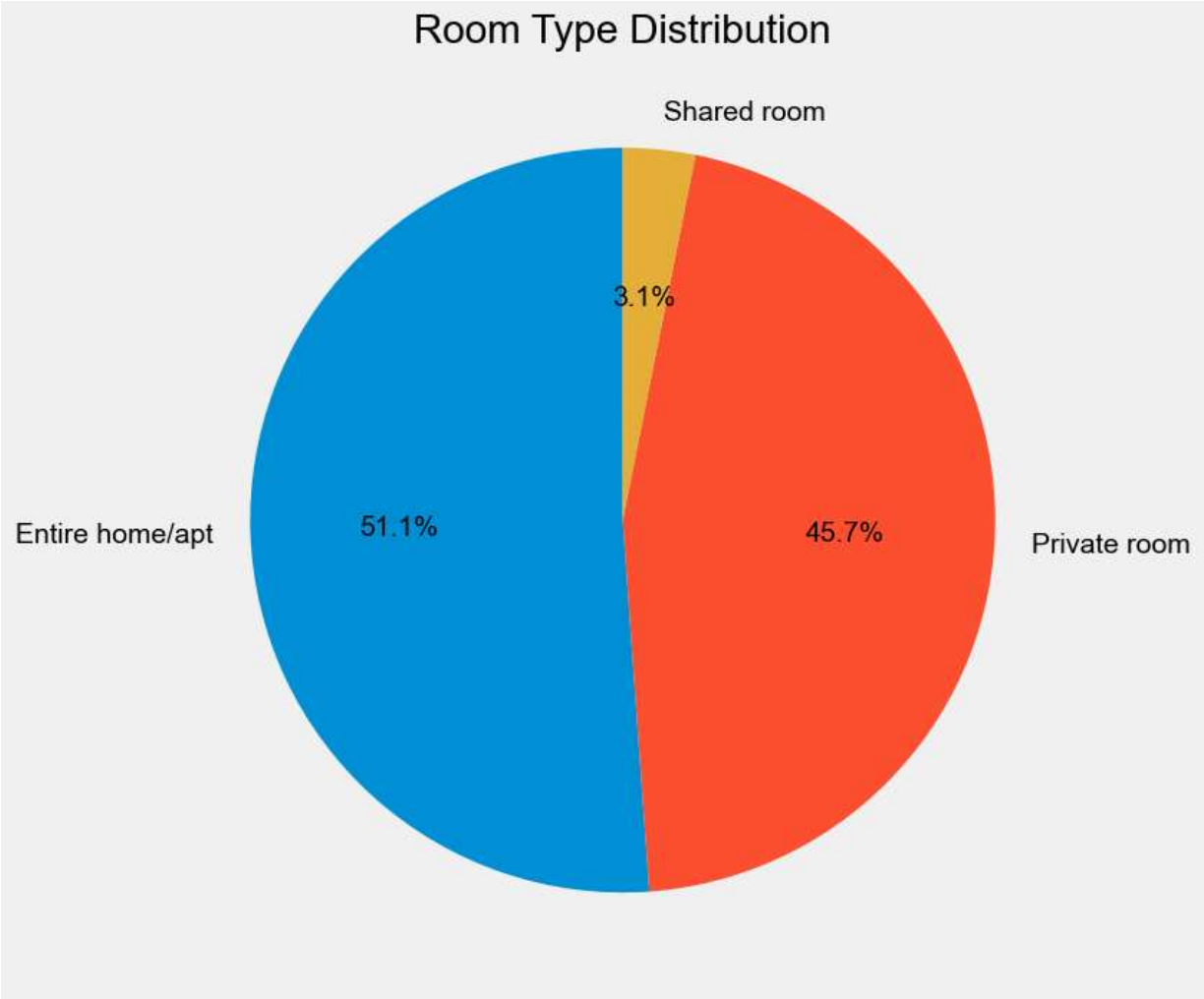
```
Manhattan        42.599620
Brooklyn         40.891841
Queens           12.333966
Bronx             3.225806
Staten Island     0.948767
Name: proportion, dtype: float64


Summary for 'neighbourhood':
neighbourhood
Williamsburg          92
Bedford-Stuyvesant    76
Bushwick              53
Upper West Side       49
Hell's Kitchen        46
                      ..
Tribeca                1
Kew Gardens            1
Morris Park            1
Randall Manor          1
Huguenot               1
Name: count, Length: 123, dtype: int64

Percentage of observations in each category:
neighbourhood
Williamsburg          8.728653
Bedford-Stuyvesant    7.210626
Bushwick              5.028463
Upper West Side       4.648956
Hell's Kitchen        4.364326
                          ...
Tribeca               0.094877
Kew Gardens           0.094877
Morris Park           0.094877
Randall Manor         0.094877
Huguenot              0.094877
Name: proportion, Length: 123, dtype: float64


Summary for 'room_type':
room_type
Entire home/apt    539
Private room       482
Shared room         33
Name: count, dtype: int64

Percentage of observations in each category:
room_type
Entire home/apt    51.13852
Private room       45.73055
Shared room         3.13093
Name: proportion, dtype: float64


Summary for 'last_review':
last_review
```

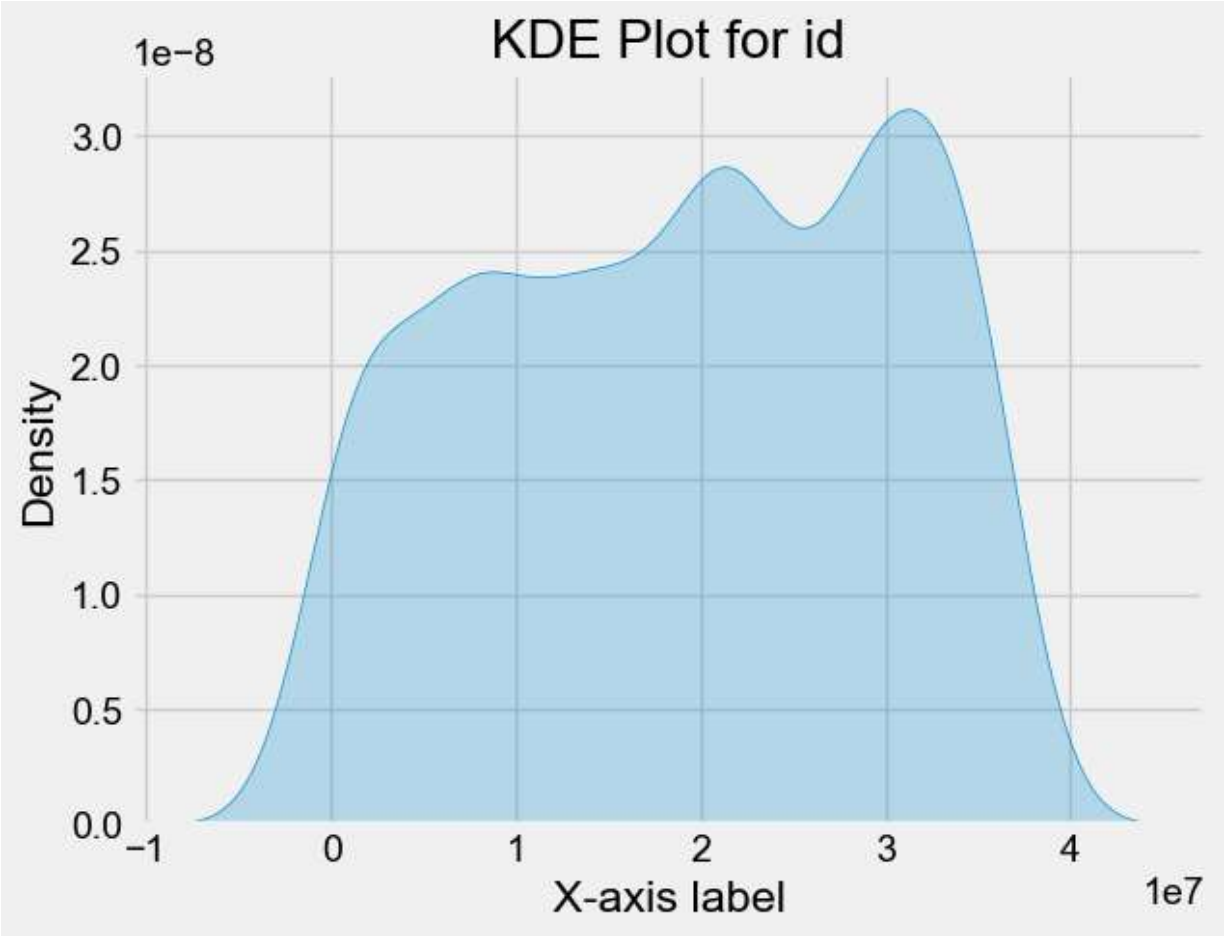
```
01-07-2019    32
24-06-2019    29
30-06-2019    27
23-06-2019    26
07-07-2019    25
              ..
23-02-2019     1
14-09-2016     1
20-05-2019     1
04-07-2016     1
07-10-2018     1
Name: count, Length: 360, dtype: int64

Percentage of observations in each category:
last_review
01-07-2019    3.791469
24-06-2019    3.436019
30-06-2019    3.199052
23-06-2019    3.080569
07-07-2019    2.962085
                ...
23-02-2019    0.118483
14-09-2016    0.118483
20-05-2019    0.118483
04-07-2016    0.118483
07-10-2018    0.118483
Name: proportion, Length: 360, dtype: float64
```

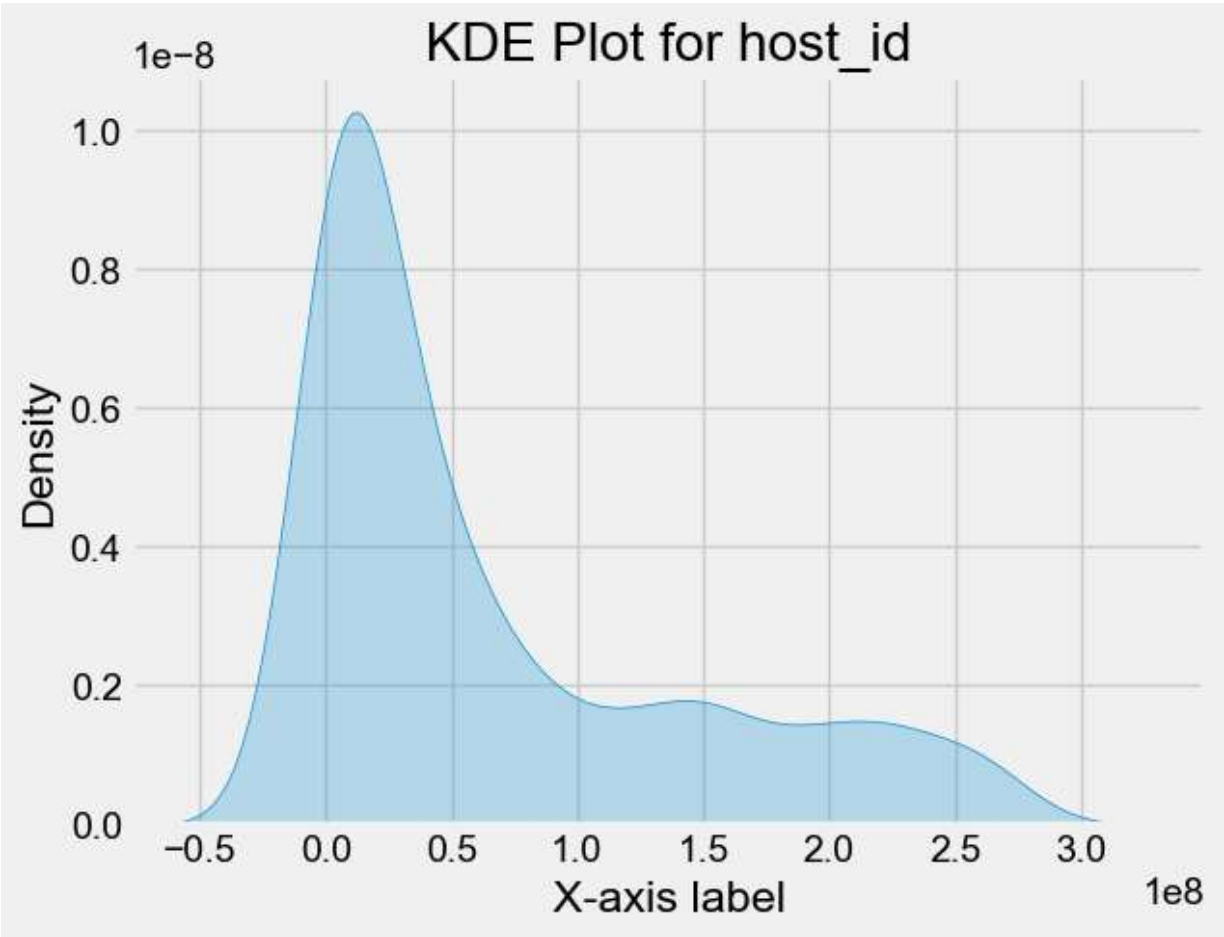


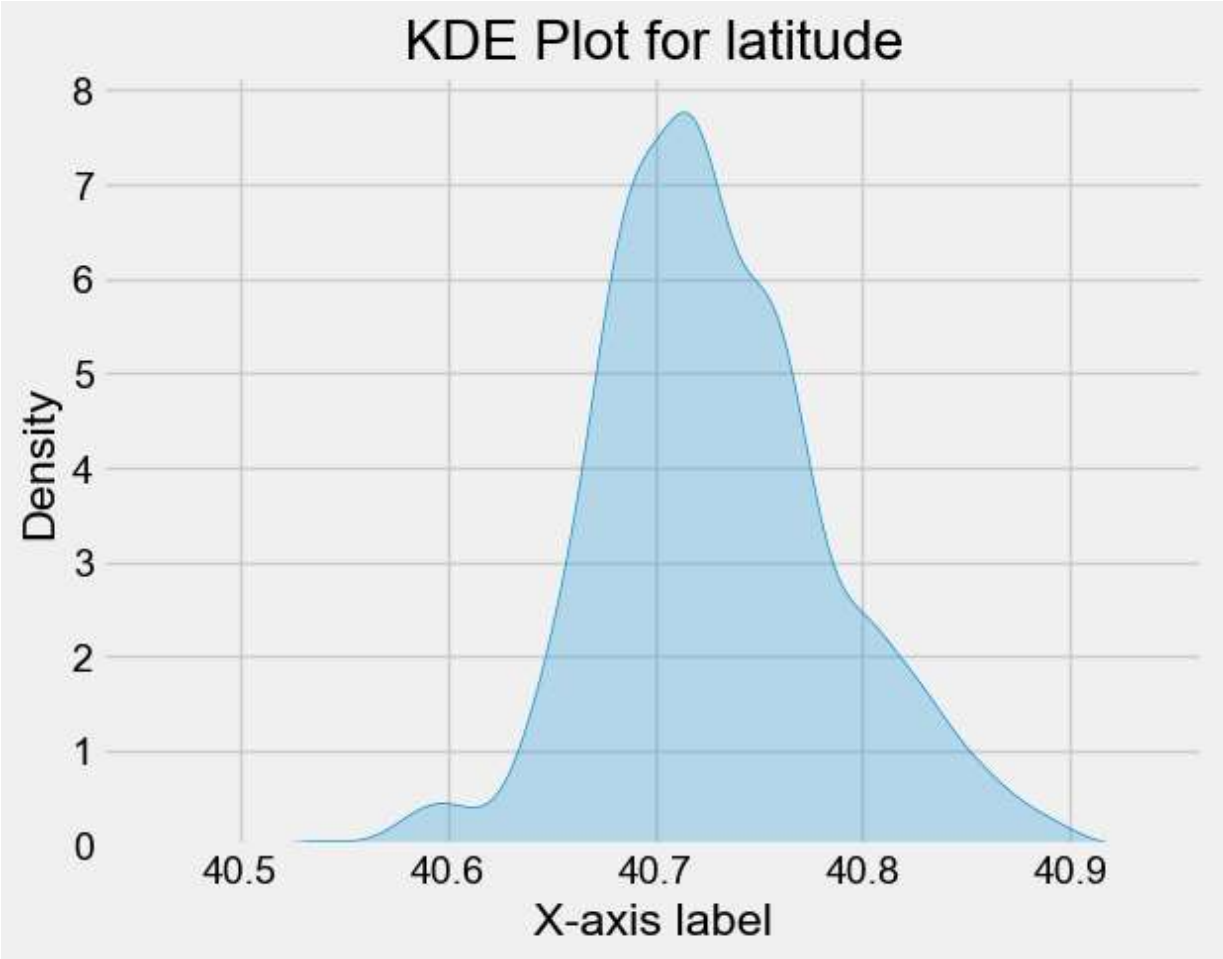

Geographical Distribution of Listings
neighbourhood_group
Manhattan
Queens
Brooklyn
Bronx
Staten Island
Longitude
−73.8
−73.9
−74.0
−74.1
−74.2
40.50
40.55
40.60
40.65
40.70
40.75
40.80
40.85
40.90
Latitude

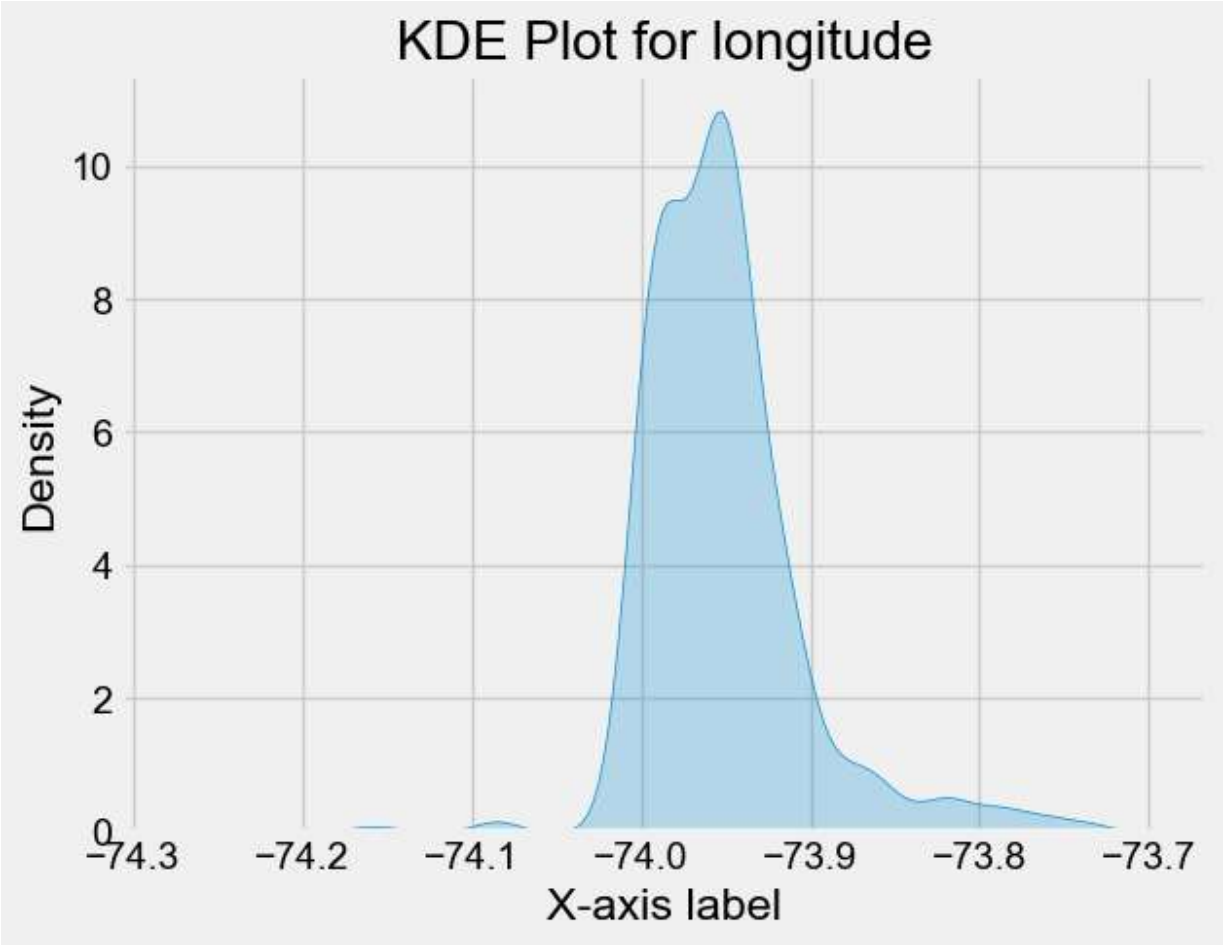

Count of Listings in Each Neighbourhood Group
Count
400
300
200
100
0
Manhattan
Queens
Brooklyn
Bronx
Staten Island
Neighbourhood Group

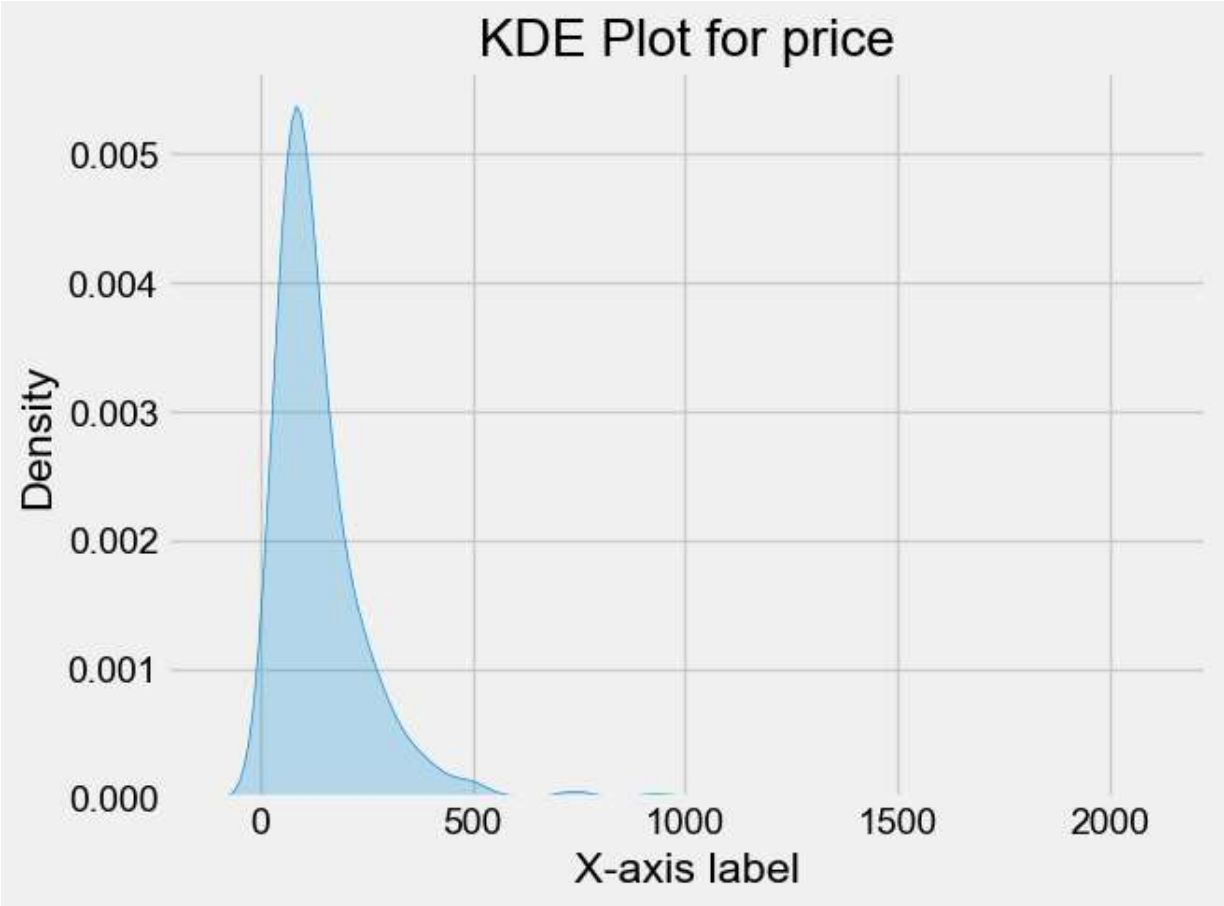

Room Type Distribution
Shared room
3.1%
Entire home/apt
51.1%
45.7%
Private room

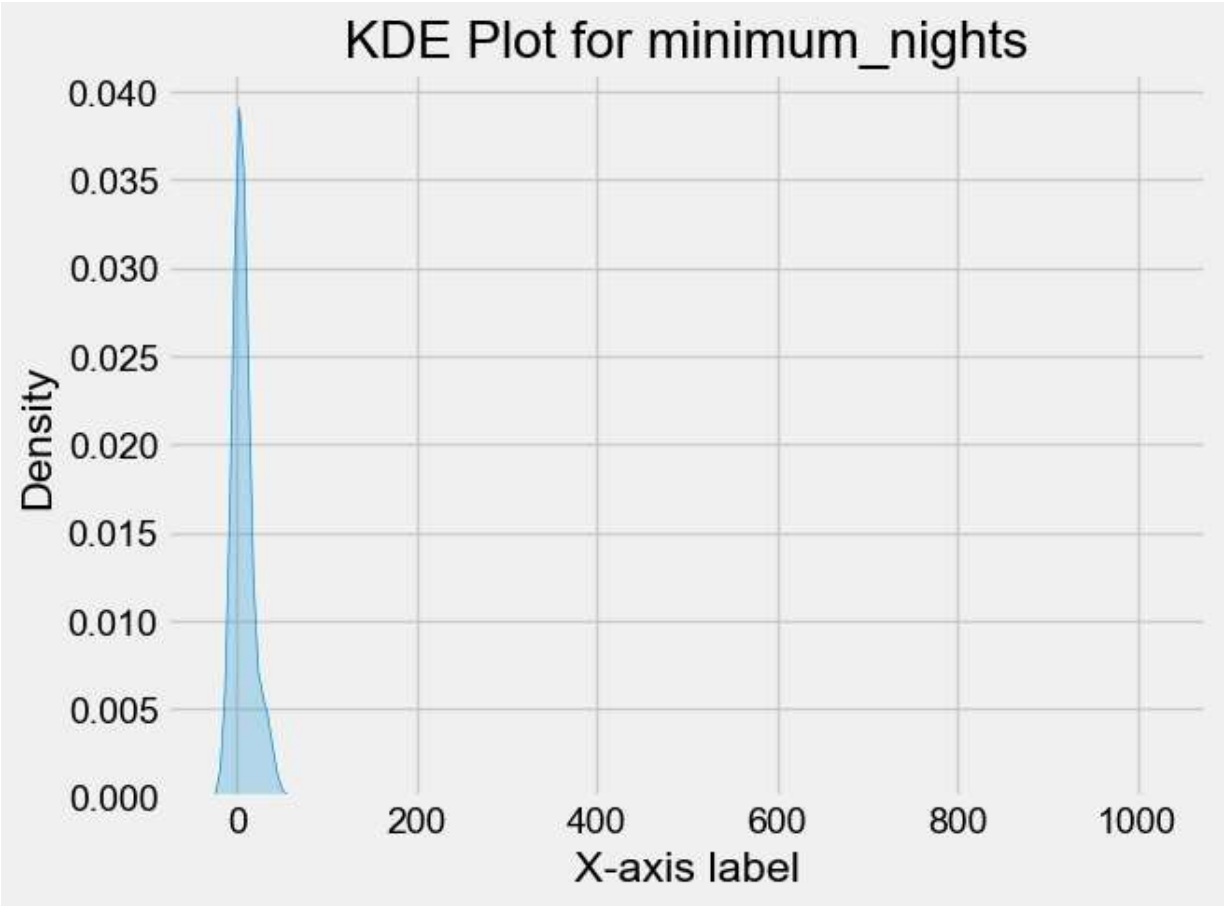

KDE Plot for id
1e−8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
−1
0
1
2
3
4
X-axis label
1e7

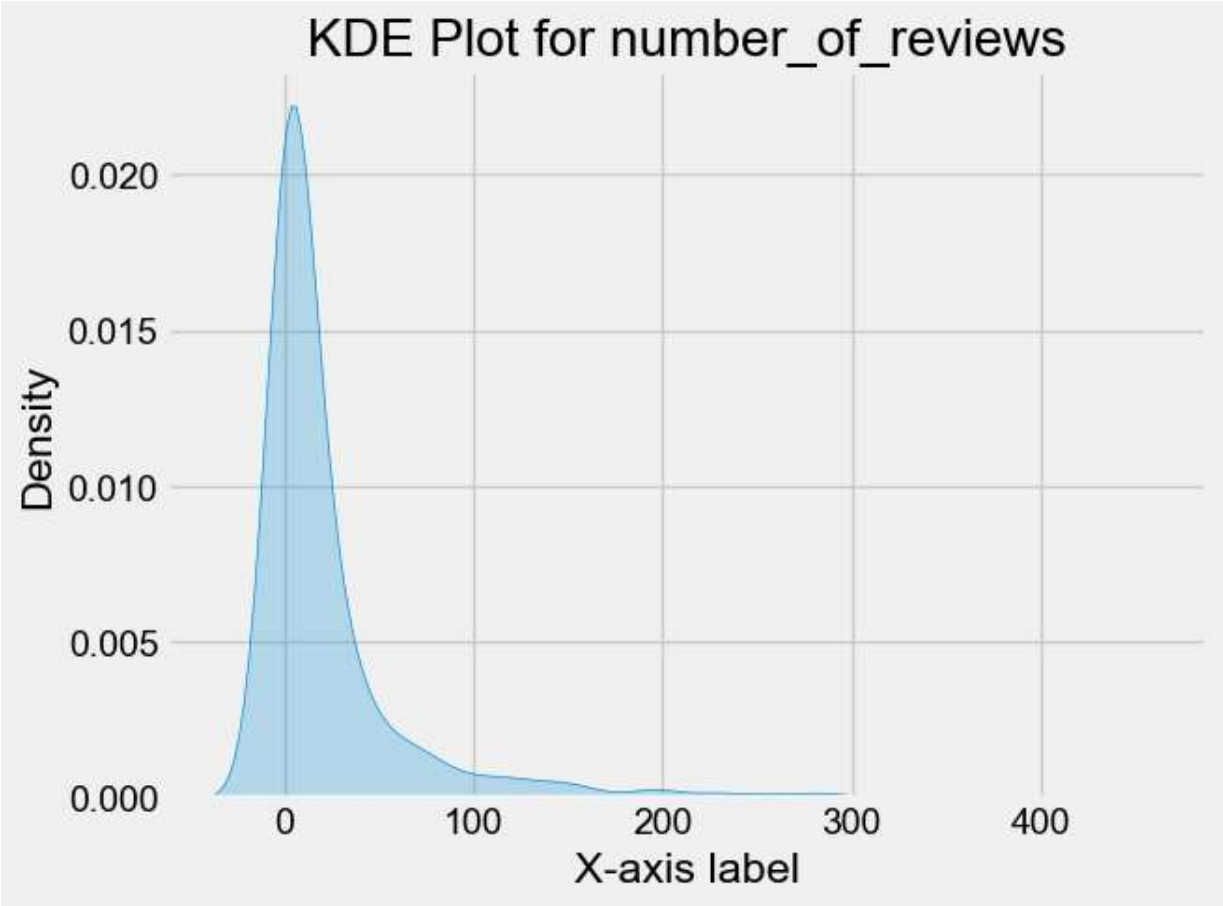

KDE Plot for host_id
1e−8
1.0
0.8
0.6
0.4
0.2
0.0
Density
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X-axis label
1e8

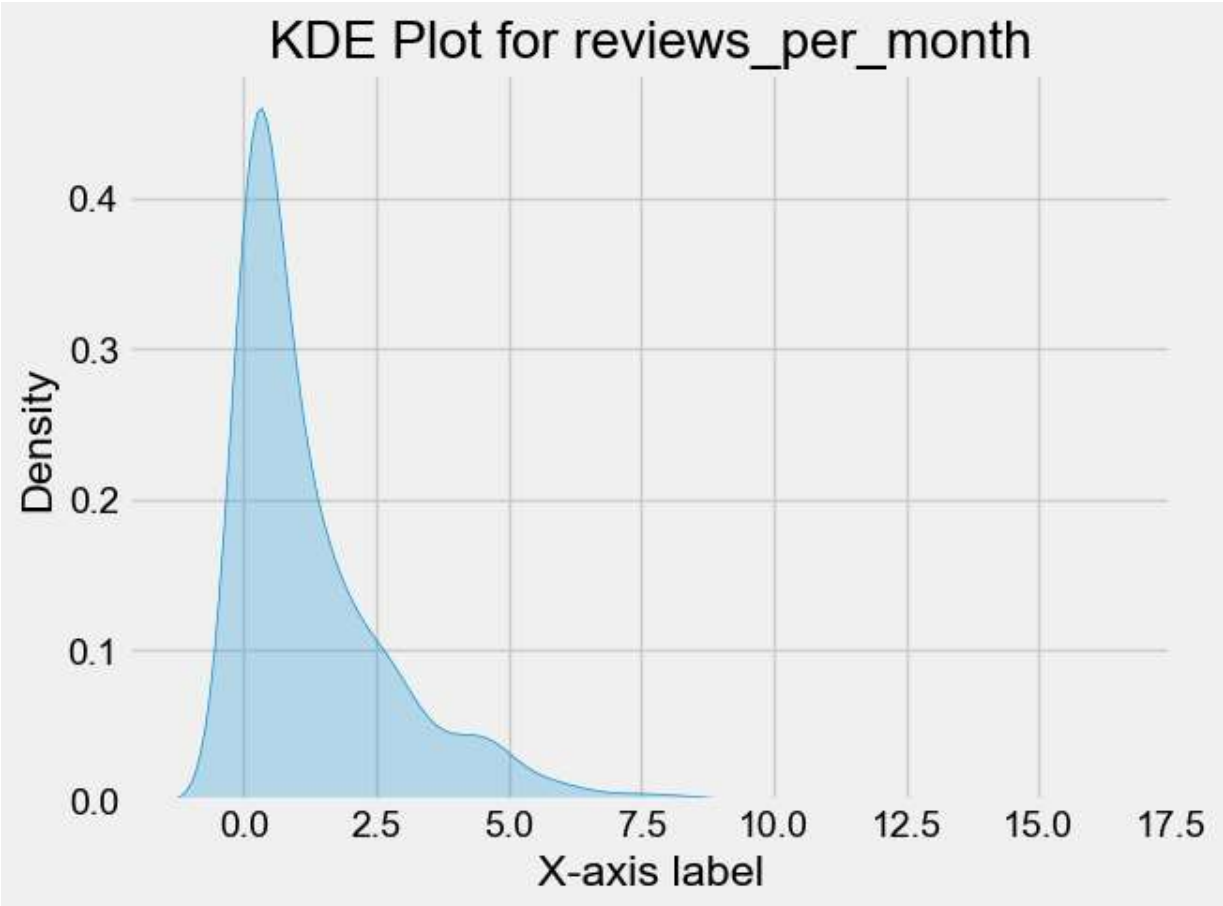

KDE Plot for latitude
Density
X-axis label
0
1
2
3
4
5
6
7
8
40.5
40.6
40.7
40.8
40.9

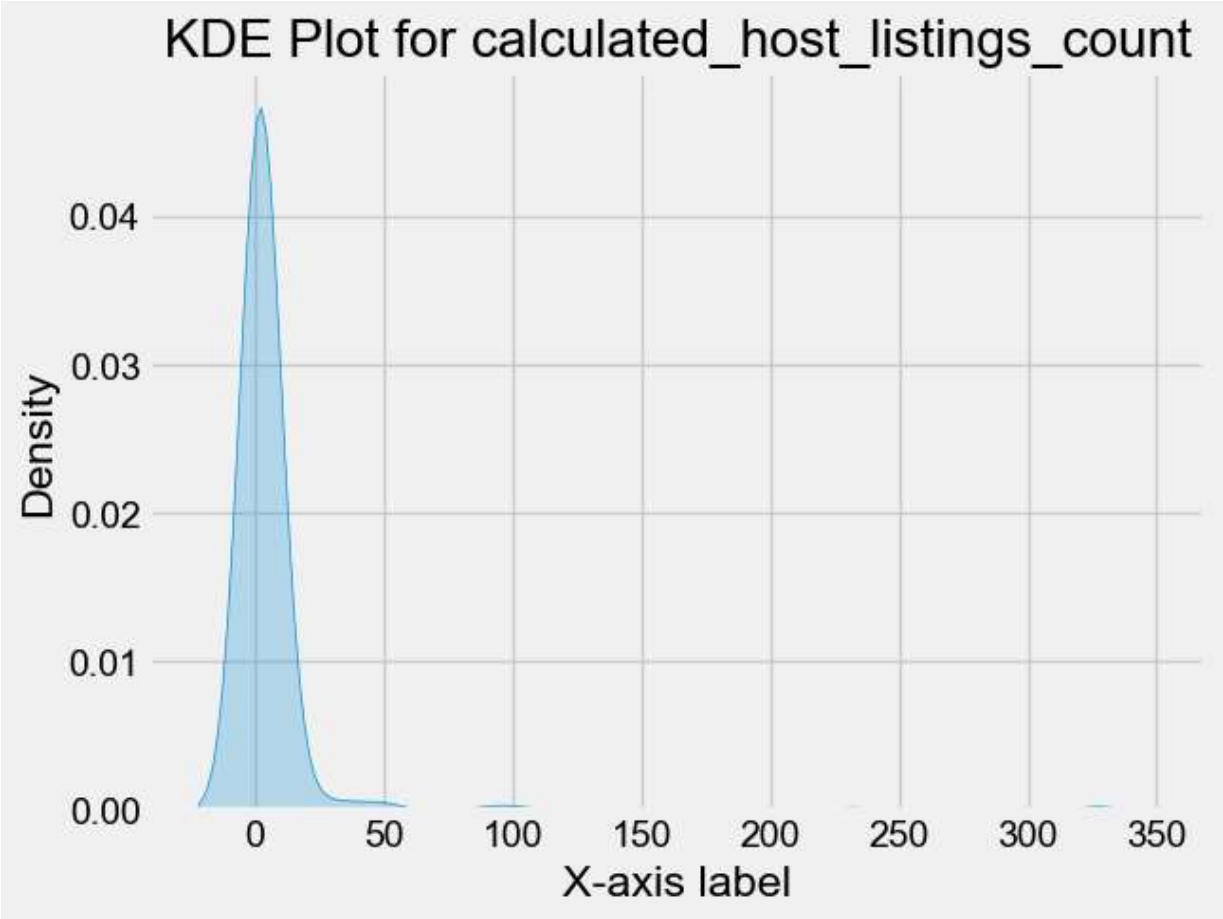

KDE Plot for longitude
Density
X-axis label
0
2
4
6
8
10
−74.3
−74.2
−74.1
−74.0
−73.9
−73.8
−73.7

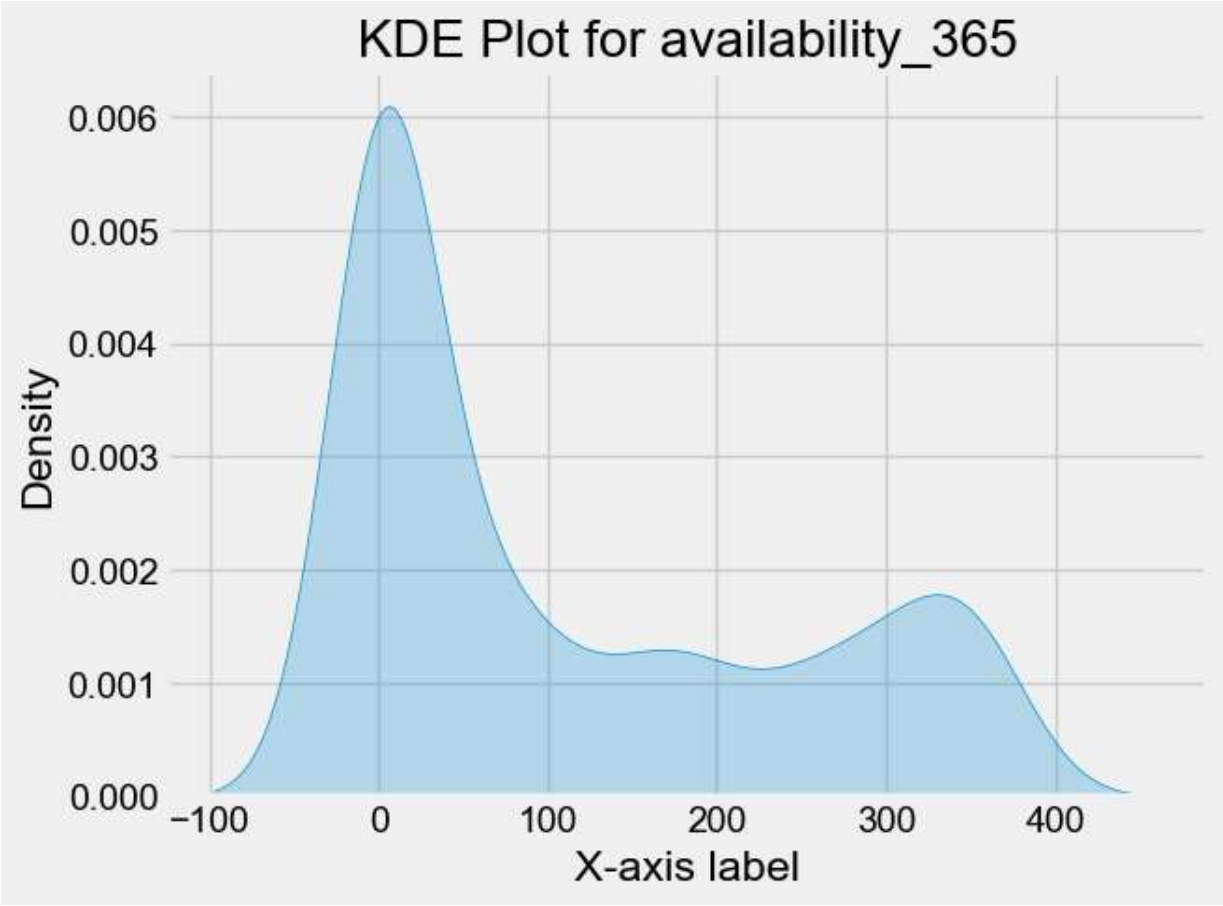

KDE Plot for price
Density
0.005
0.004
0.003
0.002
0.001
0.000
0
500
1000
1500
2000
X-axis label


KDE Plot for minimum_nights
Density
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
X-axis label


KDE Plot for number_of_reviews
Density
0.020
0.015
0.010
0.005
0.000
0
100
200
300
400
X-axis label


KDE Plot for reviews_per_month
Density
0.4
0.3
0.2
0.1
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
X-axis label


KDE Plot for calculated_host_listings_count
Density
0.04
0.03
0.02
0.01
0.00
0
50
100
150
200
250
300
350
X-axis label


KDE Plot for availability_365
Density
0.006
0.005
0.004
0.003
0.002
0.001
0.000
−100
0
100
200
300
400
X-axis label

3. Data Preparation

Check for defects in the data. Perform necessary actions to 'fix' these defects (5 Marks)

Some pointers which would help you, but don't be limited by these

a. Do variables have missing/null values?

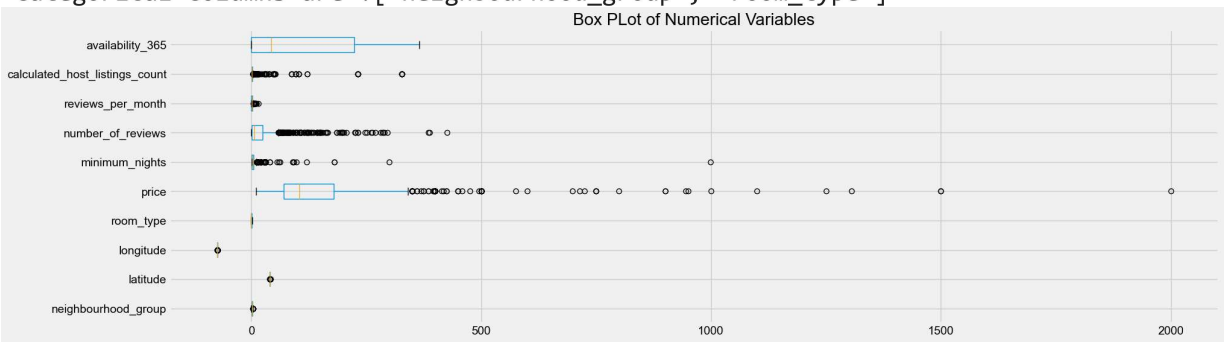
b. Do variables have outliers?

c. Is the data normally distributed? Is it a defect? Why or why not?

In [4]:
```
#Drop the unwanted columns
columns_to_drop = ['id', 'name','host_id','host_name']
df1 = df1.drop(columns=columns_to_drop)
print(f'Dropped Columns :{columns_to_drop}')

#Numerical and Categrical Data types
numerical_variables=df1.select_dtypes(include=['int64','float64']).columns.to_list(
categorical_variables =df1.select_dtypes(include='object').columns.to_list()
df2=df1

#Perform Categorical Analysis to remove and Replace the values

for column in categorical_variables:
    unique_values =df2[column].unique()
    unique_value_counts = df2[column].value_counts()

    #print(f'Unique values for {column}: {unique_values}')
    print(f'Unique value counts for {column}:\n{unique_value_counts}\n')

    if(len(unique_values)>=10):
        df2.drop(column, axis=1, inplace=True)
        categorical_variables.remove(column)
        print(f'Dropped column {column}')

print(f'Categorical Columns are :{categorical_variables}')

#Enoding values
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
for column in categorical_variables:
    df2[column] = encoder.fit_transform(df2[column])
#Box PLot of Numerical Variables
plt.figure(figsize=(20, 6))
plt.title("Box PLot of Numerical Variables")
df2.boxplot(vert=False)
plt.show()

# Copying df2 to Skewed Data Frame
skewed_df3=pd.DataFrame(df2)

#Make data Normally distributed based on skewnesis
```

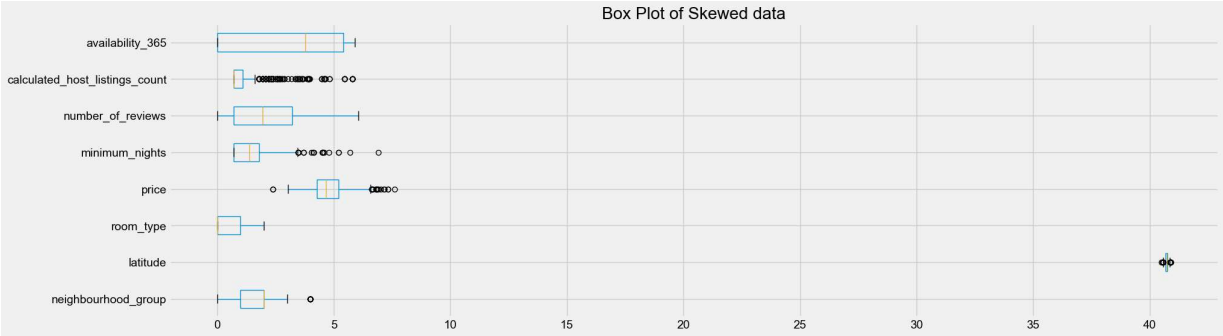
```
for column in skewed_df3.columns:
    skewness = skewed_df3[column].skew()
    print(f'Skewness {column}: {skewness}')

    if abs(skewness) > 0.5:
        # Apply log transformation if the skewness is greater than 0.5 (you can adj
        skewed_df3[column] = np.log1p(skewed_df3[column])
    else:
        # If skewness is not significant, leave the column unchanged in df2
        skewed_df3[column] = skewed_df3[column]

#Drop Empty column
skewed_df3.dropna(axis=1, inplace=True)

#Box Plot of Skewed Data
plt.figure(figsize=(20, 6))
skewed_df3.boxplot(vert=False)
plt.title("Box Plot of Skewed data")
plt.show()

#Droppiing unwanted data based on box plot
skewed_df3.drop('latitude',axis=1, inplace=True)

#Updated Box PLot of Skewed data
plt.figure(figsize=(20, 6))
skewed_df3.boxplot(vert=False)
plt.title("Updated Box PLot of Normalized  data ")
plt.show()

#Assigning skewed_df3 to df3
df3=skewed_df3

#Kde for skewed data
skewed_columns=df3.columns.to_list()
common_values = list(set(skewed_columns) & set(numerical_variables))
print(common_values)
for column in common_values:
    plt.figure(figsize=(10, 6))
    sns.kdeplot(skewed_df3[column] ,label='Skewed Data ', color='brown')
    plt.title(f'KDE Plot for {column}')
    plt.xlabel('X-axis label')
    plt.ylabel('Density')
    plt.legend()
    plt.show()
```

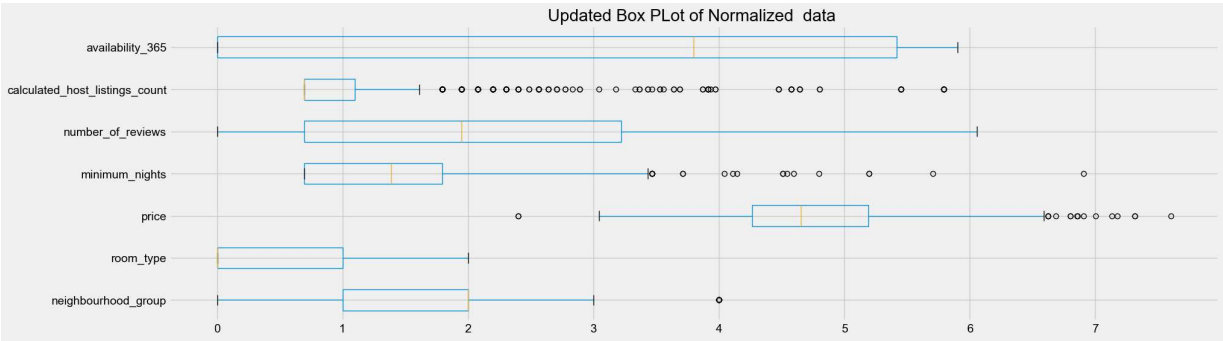
```
Dropped Columns :['id', 'name', 'host_id', 'host_name']
Unique value counts for neighbourhood_group:
neighbourhood_group
Manhattan        449
Brooklyn         431
Queens           130
Bronx             34
Staten Island     10
Name: count, dtype: int64

Unique value counts for neighbourhood:
neighbourhood
Williamsburg          92
Bedford-Stuyvesant    76
Bushwick              53
Upper West Side       49
Hell's Kitchen        46
                      ..
Tribeca                1
Kew Gardens            1
Morris Park            1
Randall Manor          1
Huguenot               1
Name: count, Length: 123, dtype: int64

Dropped column neighbourhood
Unique value counts for last_review:
last_review
01-07-2019    32
24-06-2019    29
30-06-2019    27
23-06-2019    26
07-07-2019    25
              ..
23-02-2019     1
14-09-2016     1
20-05-2019     1
04-07-2016     1
07-10-2018     1
Name: count, Length: 360, dtype: int64

Dropped column last_review
Categorical Columns are :['neighbourhood_group', 'room_type']
```



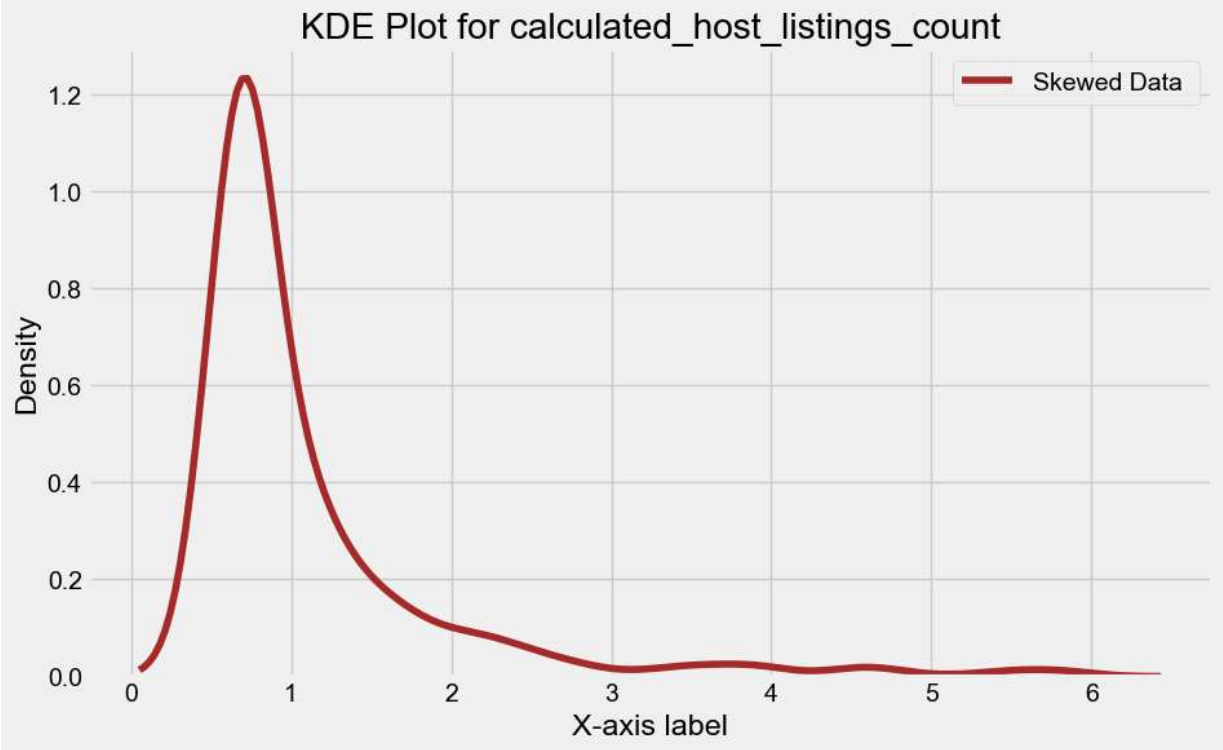

```
Skewness neighbourhood_group: 0.3411919218332452
Skewness latitude: 0.25703417390533895
Skewness longitude: 0.9878346284260408
```

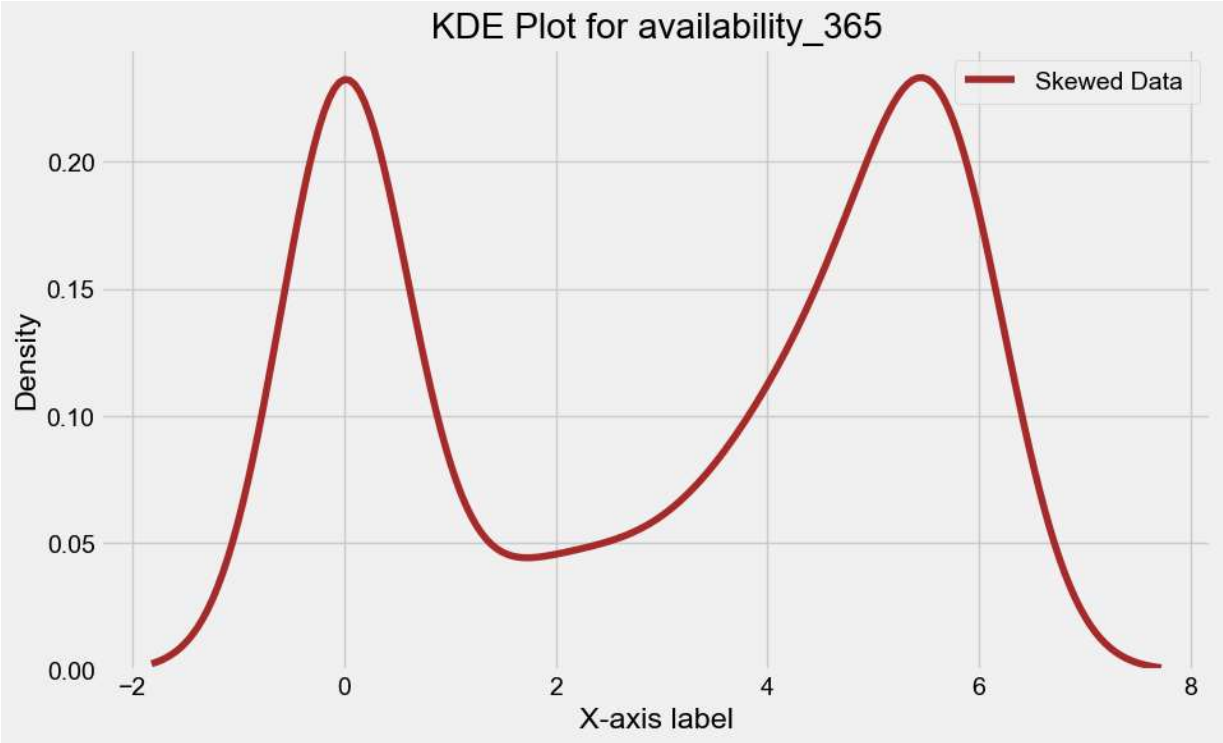
```
C:\Users\vigne\AppData\Roaming\Python\Python312\site-packages\pandas\core\arraylike.
py:396: RuntimeWarning: invalid value encountered in log1p
  result = getattr(ufunc, method)(*inputs, **kwargs)
Skewness room_type: 0.4605838811306319
Skewness price: 4.981155542768506
Skewness minimum_nights: 23.322089752532857
Skewness number_of_reviews: 3.7131116161914086
Skewness reviews_per_month: 2.4308933682122706
Skewness calculated_host_listings_count: 8.233808107546356
Skewness availability_365: 0.7598883201948172
```

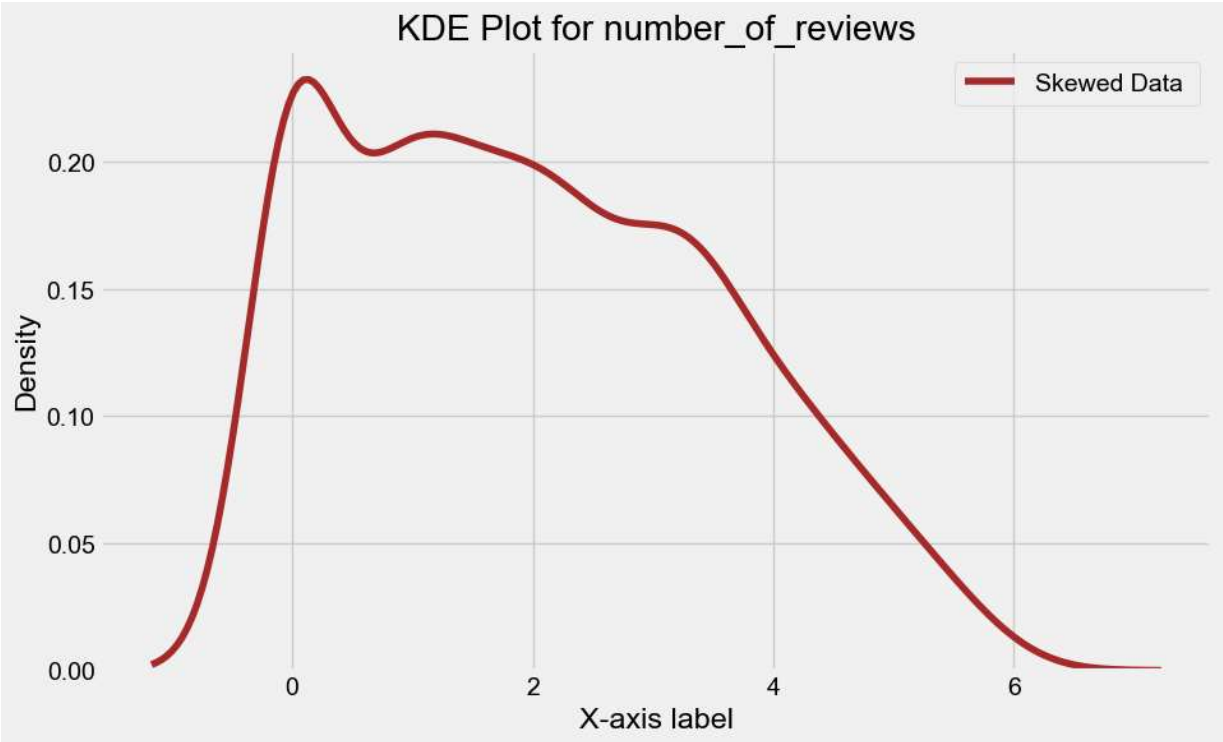


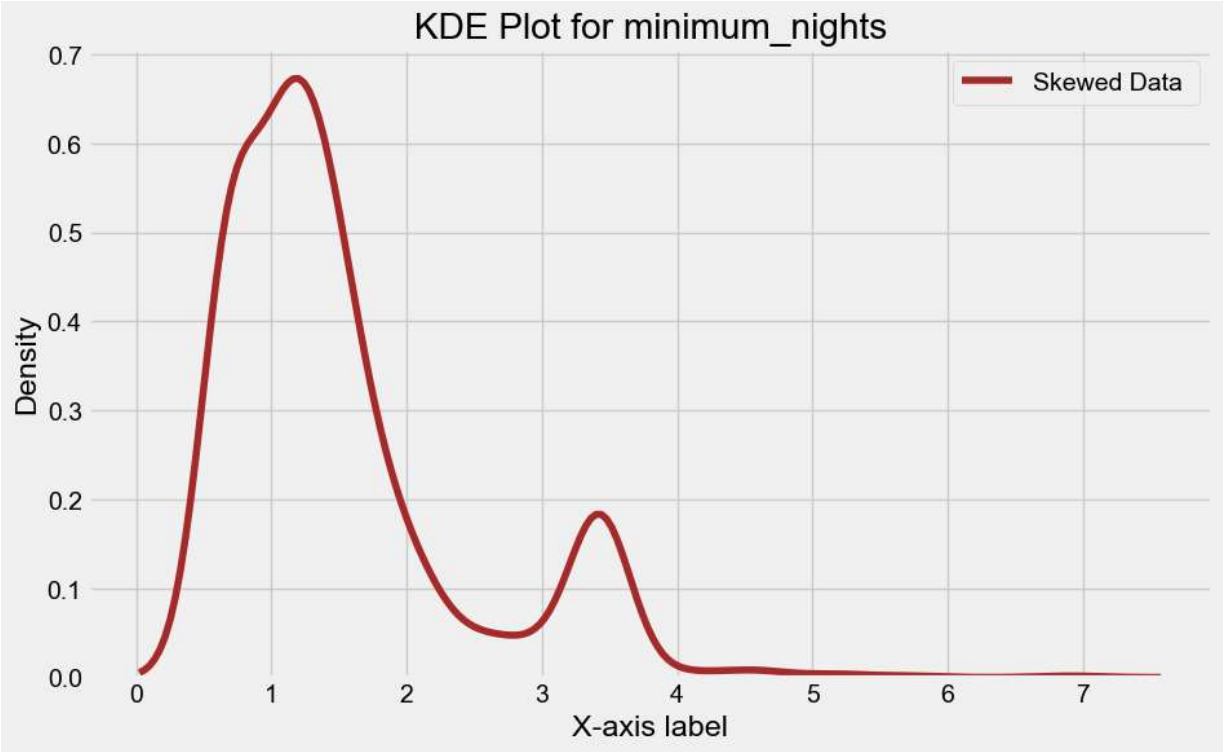


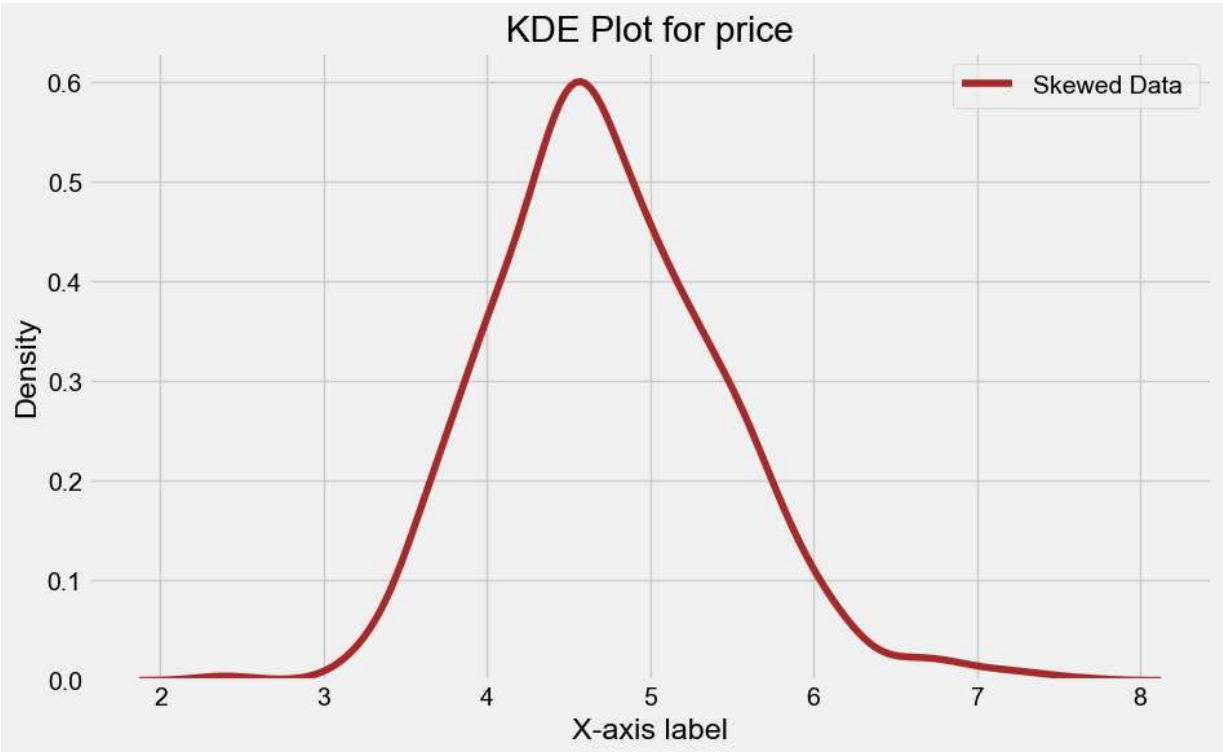

```
['calculated_host_listings_count', 'availability_365', 'number_of_reviews', 'minimum
_nights', 'price']
```




KDE Plot for availability_365
Skewed Data
Density
0.00
0.05
0.10
0.15
0.20
−2
0
2
4
6
8
X-axis label


KDE Plot for number_of_reviews
Skewed Data
Density
0.00
0.05
0.10
0.15
0.20
0
2
4
6
X-axis label



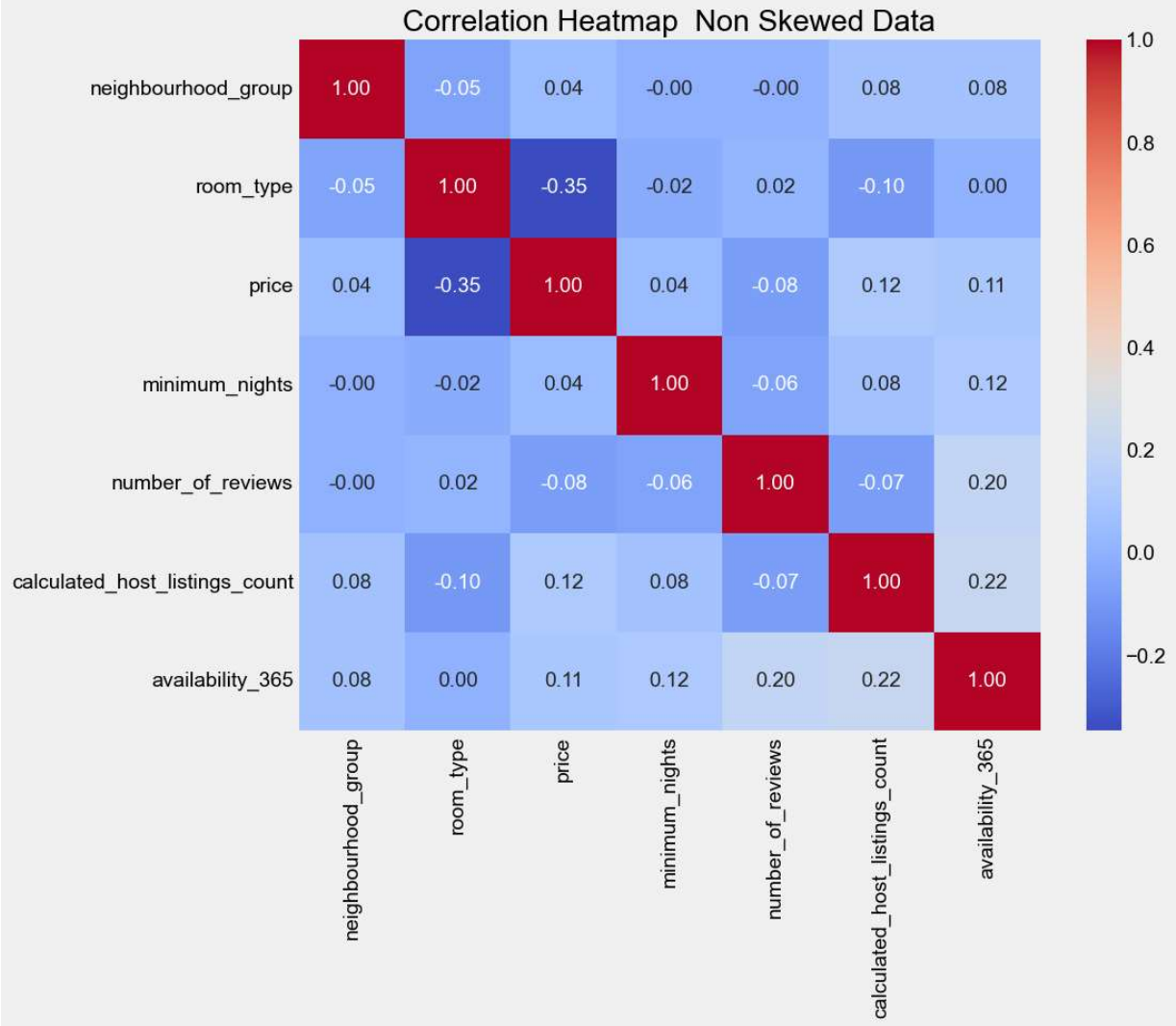


4. Summarize relationships among variables (5 marks)

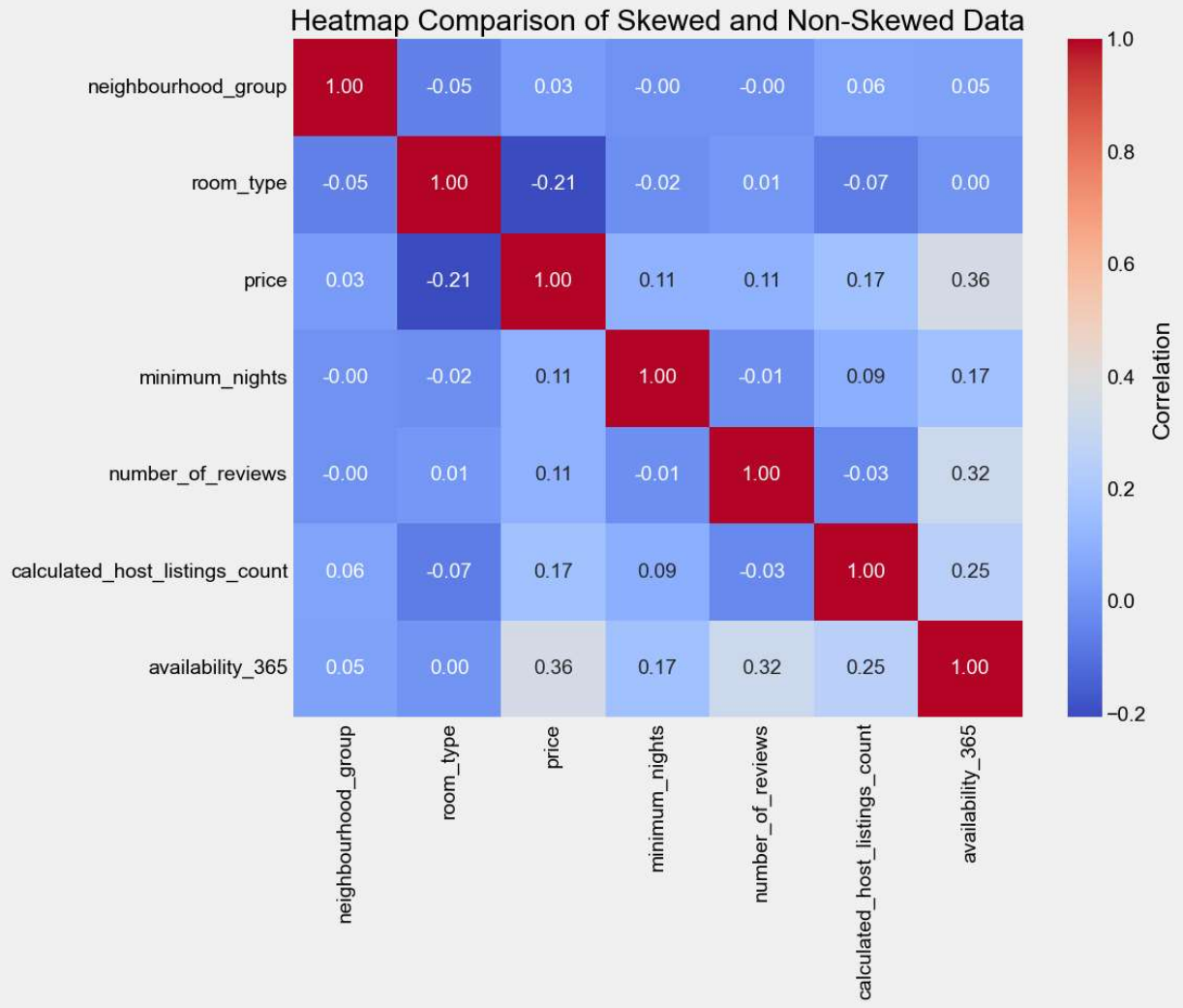
a. Plot correlation plots. Which are the variables most correlated with Target? Which independent variables are correlated among themselves? Do you want to exclude some variables from the model based on this analysis? What other actions will you take?

In [5]:
```
#Correlation heatmap for numerical variables  Non Skewed Data
plt.figure(figsize=(10, 8))
sns.heatmap(df2[df3.columns].corr(), annot=True, cmap='coolwarm', fmt='.2f')
```

```
plt.title('Correlation Heatmap  Non Skewed Data')
plt.show()

# Correlation heatmap for numerical variables   Skewed Data
plt.figure(figsize=(10, 8))
sns.heatmap(df3.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap  Skewed Data')
plt.show()

#Correlation of combined data df3 and df2
df_combined = pd.concat([df3,df2[df3.columns]], keys=['Skewed', 'Non-Skewed'])
correlation_matrix = df_combined.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", cbar_kws={'
plt.title("Heatmap Comparison of Skewed and Non-Skewed Data")
plt.show()
```




Correlation Heatmap Skewed Data
neighbourhood_group
room_type
price
minimum_nights
number_of_reviews
calculated_host_listings_count
availability_365
1.00 -0.05 0.06 0.03 -0.04 0.11 0.06
-0.05 1.00 -0.61 -0.15 -0.00 0.03 -0.00
0.06 -0.61 1.00 0.02 -0.07 0.07 0.10
0.03 -0.15 0.02 1.00 -0.26 0.33 0.14
-0.04 -0.00 -0.07 -0.26 1.00 -0.12 0.31
0.11 0.03 0.07 0.33 -0.12 1.00 0.33
0.06 -0.00 0.10 0.14 0.31 0.33 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6



5. Fit a base model. Please write your key observations (5 marks)

a. Fit the Linear Regression Model

b. What is the overall R2? Please comment on whether it is good or not.

c. Which variables are significant?

d. Calculate MSE, RMSE, MAE, MAPE.

In [26]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

#X and Y variables
X=df3.drop('price',axis=1)
y = df3[['price']]

print("\nX Variables :",X.columns.to_list())
print("y variable  :",y.columns.to_list())

#Standard Scaler
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

#test and train on
print(f'\nscaler type : {scaler}')
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2, random_sta

#Shape of train and test data
print(f'\nX_train :{X_train.shape}')
print(f'y_train :{y_train.shape}')
print(f'\nX_test :{X_test.shape}')
print(f'y_test :{y_test.shape}')

##Model fit Linear regression
model=LinearRegression()
model.fit(X_train,y_train)

#Predict model
print(f'\nModel  :{model}')
y_pred=model.predict(X_test)

#R2 square value
r2 = r2_score(y_test, y_pred)
print(f'\nR-Squared Value on (R2) {scaler} : {r2}')
#'R2 value is Good'

# Check variable significance using statsmodels
X_train_with_intercept = sm.add_constant(X_train)
model_stats = sm.OLS(y_train, X_train_with_intercept).fit()
print('')
print(model_stats.summary())

# Calculate MSE, RMSE, MAE, MAPE.
# Calculate MSE
mse = mean_squared_error(y_test, y_pred)

# Calculate RMSE
rmse = np.sqrt(mse)

# Calculate MAE
mae = mean_absolute_error(y_test, y_pred)

# Calculate MAPE
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

#Print the result
print(f'\nMSE  - Mean Square Error {scaler}:{mse}')
print(f'RMSE - Root Mean Squared Error   {scaler} : {rmse}')
print(f'MAE  - Mean Absolute Error  {scaler}:{mae}')
print(f'MAPE - Mean Absolute Percentage Error  {scaler} : {mape}')
```

```
X Variables : ['neighbourhood_group', 'room_type', 'minimum_nights', 'number_of_revi
ews', 'calculated_host_listings_count', 'availability_365']
y variable  : ['price']

scaler type : StandardScaler()

X_train :(843, 6)
y_train :(843, 1)

X_test :(211, 6)
y_test :(211, 1)

Model  :LinearRegression()

R-Squared Value on (R2) StandardScaler() : 0.46137438854757484

                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.402
Model:                            OLS   Adj. R-squared:                  0.398
Method:                 Least Squares   F-statistic:                     93.80
Date:                Tue, 19 Dec 2023   Prob (F-statistic):           5.20e-90
Time:                        11:05:43   Log-Likelihood:                -693.87
No. Observations:                 843   AIC:                             1402.
Df Residuals:                     836   BIC:                             1435.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.7278      0.019    248.018      0.000       4.690       4.765
x1             0.0083      0.019      0.434      0.664      -0.029       0.046
x2            -0.4454      0.019    -23.023      0.000      -0.483      -0.407
x3            -0.1224      0.021     -5.889      0.000      -0.163      -0.082
x4            -0.1180      0.022     -5.440      0.000      -0.161      -0.075
x5             0.0445      0.022      2.022      0.044       0.001       0.088
x6             0.1150      0.022      5.193      0.000       0.072       0.158
==============================================================================
Omnibus:                      256.379   Durbin-Watson:                   2.103
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1675.148
Skew:                           1.214   Prob(JB):                         0.00
Kurtosis:                       9.465   Cond. No.                         1.84
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.

MSE  - Mean Square Error StandardScaler():0.24433723753597197
RMSE - Root Mean Squared Error   StandardScaler() : 0.49430480225865897
MAE  - Mean Absolute Error  StandardScaler():0.369852673342861
MAPE - Mean Absolute Percentage Error  StandardScaler() : 7.714385043959887
```

6. Perform feature engineering using any of the listed techniques: Forward feature selection, backward elimination

and recursive feature elimination and use the features to improvise the model and compare the results with the base model. (5 marks)

In [27]:
```
#Load Libraries RFE Recursive Feature Elimination
from sklearn.feature_selection import RFE

#train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

#Base Model -Linear Regression
base_model = LinearRegression()
base_model.fit(X_train, y_train)
base_predictions = base_model.predict(X_test)
base_rmse = mean_squared_error(y_test, base_predictions, squared=False)
print(f'\nBase Model :{base_model}')
print(f"Base Model RMSE: {base_rmse}")

# Choose the number of features to select
num_features_to_select = 3

# Create RFE model and select features
rfe_model = RFE(estimator=LinearRegression(), n_features_to_select=num_features_to_
rfe_model.fit(X_train, y_train)

# Selected features
selected_features = X.columns[rfe_model.support_]
print("\nSelected Features:", selected_features)

# Fit a model using the selected features
rfe_X_train = rfe_model.transform(X_train)
rfe_X_test = rfe_model.transform(X_test)

rfe_model = LinearRegression()
rfe_model.fit(rfe_X_train, y_train)
rfe_predictions = rfe_model.predict(rfe_X_test)
rfe_rmse = mean_squared_error(y_test, rfe_predictions, squared=False)
print(f'\nREF Model :{rfe_model}')
print(f"RFE Model RMSE: {rfe_rmse}")

print(f"\nBase Model RMSE: {base_rmse}")
print(f"RFE Model RMSE: {rfe_rmse}")
```

```
Base Model :LinearRegression()
Base Model RMSE: 0.4943048022586591

Selected Features: Index(['room_type', 'minimum_nights', 'calculated_host_listings_c
ount'], dtype='object')

REF Model :LinearRegression()
RFE Model RMSE: 0.49830086385617256

Base Model RMSE: 0.4943048022586591
RFE Model RMSE: 0.49830086385617256
```

7. The prediction model output reliability can be improved using regularization techniques and parameter tuning. Perform regularization techniques and compare the results with the base model. (5 marks)

In [28]:
```
#load library of RIDGE
#Ridge Regression, also known as Tikhonov regularization or L2 regularization
from sklearn.linear_model import  Ridge

#Step 2: Fit Train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Standardize Scaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Base Model
base_model = LinearRegression()
base_model.fit(X_train_scaled, y_train)
base_predictions = base_model.predict(X_test_scaled)
base_rmse = mean_squared_error(y_test, base_predictions, squared=False)
print(f'\nBase Model :{base_model}')
print(f"Base Model RMSE: {base_rmse}")

#Regularized Model (Ridge Regression)
# You can tune the alpha parameter for Ridge regularization

alpha = 0.1
ridge_model = Ridge(alpha=alpha)
ridge_model.fit(X_train_scaled, y_train)
ridge_predictions = ridge_model.predict(X_test_scaled)
ridge_rmse = mean_squared_error(y_test, ridge_predictions, squared=False)
print(f'\nRidge Model :{ridge_model}')
print(f"Ridge Model RMSE: {ridge_rmse}")

#print the result
print(f"\nBase Model RMSE: {base_rmse}")
print(f"Ridge Model RMSE: {ridge_rmse}")
```

```
Base Model :LinearRegression()
Base Model RMSE: 0.4943048022586589

Ridge Model :Ridge(alpha=0.1)
Ridge Model RMSE: 0.4943012036116637

Base Model RMSE: 0.4943048022586589
Ridge Model RMSE: 0.4943012036116637
```

In [ ]: