# Spatial Prediction on Airbnb in New York City

Haesong Choi

December 13, 2019

## 1 Introduction

Since it launched in 2009, Airbnb company has been surprisingly increasing and it covers more than 81,000 cities and 191 countries worldwide. Airbnb is a home-sharing platform that connects people who want to rent out their homes with guests who are looking for affordable accomodations. Airbnb allows home-owners and renters to put their properties online, so that guests can pay to stay in them. Hosts are generally required to set their own expected average night price for their listings. As users have been continuously growing on Airbnb, hosts may find it hard to properly price their property. Trying to monitor and understand the underlying pricing dynamics of the Airbnb is very important, particularly in big cities like New York where there is lots of competition on both the suppy and demand side.

This study aims to analyze over 48,000 listings in the New York City area in order to accurately predict the optimal listing price by using the spatial basis function expansion. To verify how accurate this model is, we compare MSE of simple regression model, ridge regression model and lasso regression model with that of the spatial basis function expansion. For regression models, we first apply the forward stepwise selection to the model since there are ten features of Airbnb for predictors. Then, we use cross validation to prevent underfitting or overfitting, and run the regression models.

## 2 Statistical Model

### 2.1 Regression analysis

Regression analysis is a powerful analysis, widely used for predicting the unknown variable by fitting a linear equation. The basic multiple regression model for our analysis is as follows:

$$\begin{aligned}\log(\text{price}+1) &= \beta_0 + \beta_1 x_1 + \cdots + \beta_{10} x_{10} \qquad (2.1)\\ x_1 &= \text{neighborhood group}\\ x_2 &= \text{neighborhood}\\ x_3 &= \text{latitude}\\ x_4 &= \text{longitude}\\ x_5 &= \text{room type}\\ x_6 &= \text{minimum nights}\\ x_7 &= \text{number of reviews}\\ x_8 &= \text{reviews per month}\\ x_9 &= \text{host listings count}\\ x_{10} &= \text{availability among 365 days}\end{aligned}$$

Since all variables are not normally distributed, we replaced the dependent variable with the logarithm of (price + 1) and made the other variables normalized. In particular, this paper fits the model through forward stepwise selection to remove redundant predictors and improve our results for the model.

## 2.2 Forward stepwise selection

For computational reasons, the best subset cannot be applied for any $n$ large predictors due to the $2^n$. Also, irrelevant predictors decrease the precision of the estimated coefficients and predicted values. Therefore, the actual set of predictor variables used in the final regression model is often determined by variable selection analysis. This paper applied forward stepwise selection to our regression model. The algorithm for forward stepwise selection is as follows:

- Let $\mathcal{M}_0$ denote the null model which contains no predictors.
- For $k = 1, 2, \cdots, n-1$, consider all $n-k$ models in $\mathcal{M}_k$ and choose the best model that minimizes the criterion of interest.
- For all remaining steps, add one predictor to the previous model.
- Stop when addition of another predictor increases information criteria value.

For the information criteria, AIC, BIC, and Mallow's $C_p$ are used. $n$ denotes the number of size for dataset and $p_c$ denotes the number of (predictors + 1).

### 2.2.1 AIC

The AIC (Akaike Information Criterion) is defined for a large class of models fit by maximum likelihood. Small values of AIC are preferred, so better candidate sets will have smaller RSS and a smaller number of terms $p_c$. The AIC is given by

$$AIC = n\log(RSS_c/n) + 2p_c \tag{2.2}$$

### 2.2.2 BIC

The BIC (Bayes Information Criterion) is derived from a Bayesian point of view. It looks like AIC except the second term. Small values of BIC are preferred, so better candidate sets will have smaller RSS and a smaller number of terms $p_c$. The BIC is given by

$$AIC = n\log(RSS_c/n) + p_c \log n \tag{2.3}$$

### 2.2.3 Mallow's $C_p$

Mallow's $C_p$ is named after Colin Lingwood Mallows and is defined as:

$$C_p = RSS_c/\hat{\sigma}^2 + 2p_c - n \tag{2.4}$$

where $\hat{\sigma}^2$ is an estimate of the variance of the error $\epsilon$. It balances between lack of fit and complexity. If we fix the complexity, we consider only the choices $x_c$ with a fixed number of terms and the choice with the smallest RSS is the preferred choice.

## 2.3 Cross Validation

When they make use of a statistical model, people usually fit the model on a training set and make predictions on a data that was not trained. When the model is fitted too much to the training data, the model can be said to be not generalized and overfitted. On the other hand, underfitting means that model does not fit the training data and misses the trends in the data. To prevent overfitting or underfitting, this study applied cross validation to our model. A dataset is usually split into training data and test data. Cross validation is very similar to train/test split, but it is applied to more subsets. The dataset is split into $k$ subsets. We use $k-1$ subsets to train a dataset and leave the last subset for test data.

## 2.4 Spatial Basis Function

To present the spatial relationship between variables, this study adjusts spatial basis function. For nonstationary modeling, we usually use basis functions to estimate the spatial covariance structure. Basis function procedures represent a function $x$ which depends on $s$ by a linear expansion,

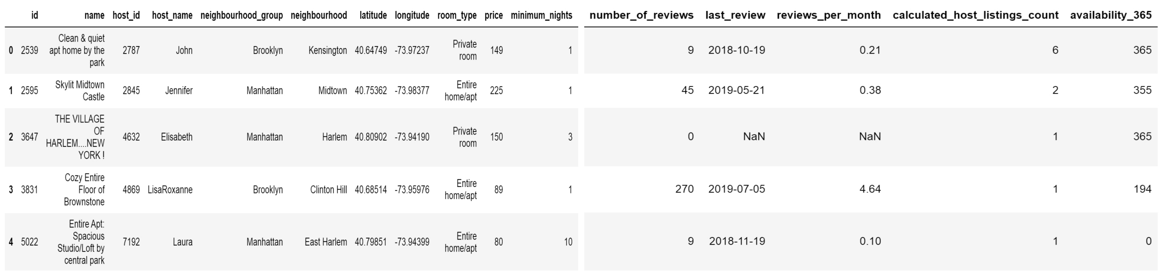
$$y(\boldsymbol{s}) = \sum_{i=1}^{r} \lambda_i \pi_i(\boldsymbol{s}) \tag{2.5}$$

where $r$ basis functions are known. In general, the spatial basis function expansion can be developed as follows. Let $g(s)$ is a known r-dimensional real value function. Then define $y(\boldsymbol{s}) = g(\boldsymbol{s})'\boldsymbol{\eta}$, where $\boldsymbol{\eta}$ has mean zero, cov $= \boldsymbol{\Sigma}$. Then $E(y(\boldsymbol{s}, \boldsymbol{u})) = g(\boldsymbol{s})'\boldsymbol{\Sigma}g(\boldsymbol{u})$, which is not a function of $\boldsymbol{s} - \boldsymbol{u}$. If $g(\boldsymbol{s})'$ is orthonormal over $D$, $y(\boldsymbol{s}) = \sum_{i=1}^{r} g_i(\boldsymbol{s})'\boldsymbol{\eta}_i$ is a truncated KL expansion. Applying the truncated KL expansion to $\omega_i(\cdot)$,$\omega_i(\boldsymbol{s}) = \sum_{j=1}^{r} \pi_{ij}(\boldsymbol{s})\alpha_i$. Then, we obtain

$$y(\boldsymbol{s}) = \sum_{i=1}^{r}\sum_{j=1}^{r} \alpha_i \eta_i \pi_{ij}(\boldsymbol{s}) \tag{2.6}$$

## 3 The Dataset

The dataset used for this project comes from New York City Airbnb Open Data in Kaggle.com. The dataset was scraped on 20 July 2019 and contains information on all New York Airbnb listings that were live on five boroughs that New York City encompasses: The Bronx, Brooklyn, Manhattan, Queens, and Staten Island. As seen in Figure 1, there are 488,895 observations with 16 explanatory variables. This paper removed insignificant variables such as ID, Name, Host ID, Host name, and Last review and made use of 10 variables as predictors. (Neighbourhood group, Neighbourhood, Latitude, Longitude, Room type, Minimum nights, Number of reviews, Reviews per month, Calculated host listing count, and Availability among 365 days)



| | id | name | host_id | host_name | neighbourhood_group | neighbourhood | latitude | longitude | room_type | price | minimum_nights | number_of_reviews | last_review | reviews_per_month | calculated_host_listings_count | availability_365 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2539 | Clean & quiet apt home by the park | 2787 | John | Brooklyn | Kensington | 40.64749 | -73.97237 | Private room | 149 | 1 | 9 | 2018-10-19 | 0.21 | 6 | 365 |
| **1** | 2595 | Skylit Midtown Castle | 2845 | Jennifer | Manhattan | Midtown | 40.75362 | -73.98377 | Entire home/apt | 225 | 1 | 45 | 2019-05-21 | 0.38 | 2 | 355 |
| **2** | 3647 | THE VILLAGE OF HARLEM....NEW YORK ! | 4632 | Elisabeth | Manhattan | Harlem | 40.80902 | -73.94190 | Private room | 150 | 3 | 0 | NaN | NaN | 1 | 365 |
| **3** | 3831 | Cozy Entire Floor of Brownstone | 4869 | LisaRoxanne | Brooklyn | Clinton Hill | 40.68514 | -73.95976 | Entire home/apt | 89 | 1 | 270 | 2019-07-05 | 4.64 | 1 | 194 |
| **4** | 5022 | Entire Apt: Spacious Studio/Loft by central park | 7192 | Laura | Manhattan | East Harlem | 40.79851 | -73.94399 | Entire home/apt | 80 | 10 | 9 | 2018-11-19 | 0.10 | 1 | 0 |

Figure 1: The dataset

For a dependent variable, price was used in this paper. Since it is positively skewed, we took log-transformation to make the price variable follow normal distribution, so that it met the assumption of regression analysis. Figure 2 presents the distribution of price variable and that of logrithm of (price+1).

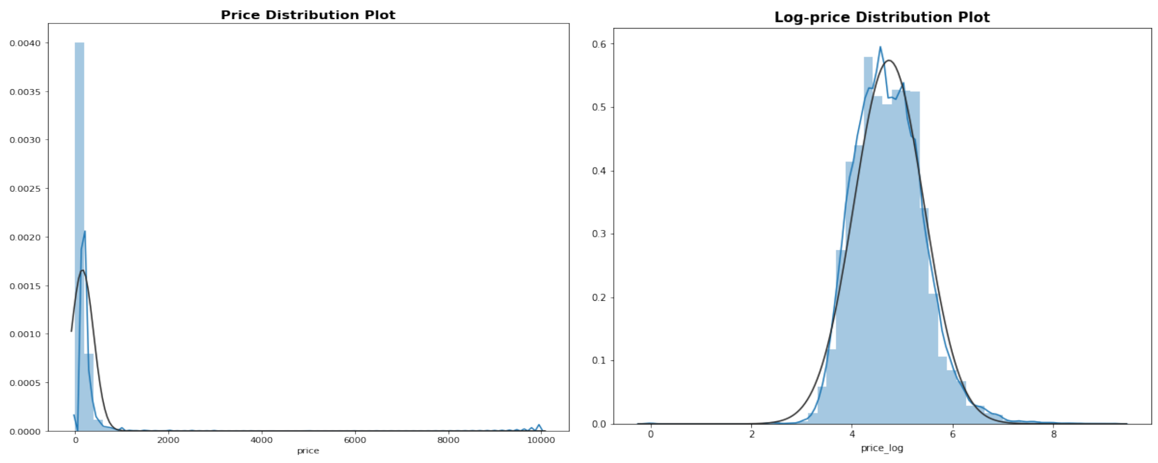
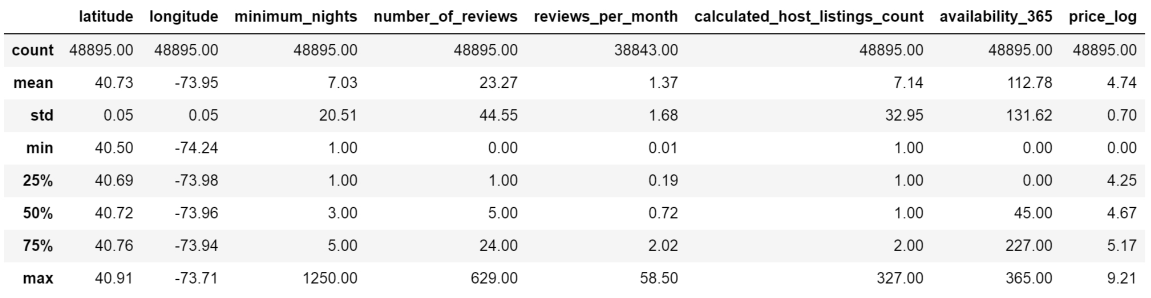


Figure 2: The Price variable

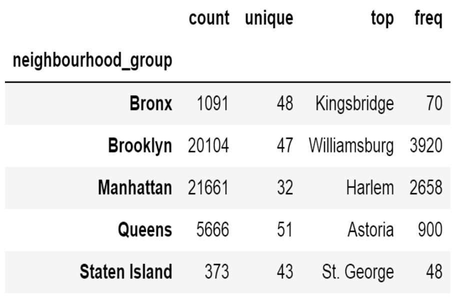
Figure 3 exhibits the descriptive statistics for all variables. Reviews per month variable has 10,052 missing data which means 0 review. If there were no reviews for the listing, data simply will not exist. Therefore, we appended it with 0 for missing values. For the categorical data, Room type consists of entire home/apt (25,409 data), private room (22,326 data), and shared room (1,160 data). It shows that entire home and apartment are rented the most.



| | latitude | longitude | minimum_nights | number_of_reviews | reviews_per_month | calculated_host_listings_count | availability_365 | price_log |
|---|---|---|---|---|---|---|---|---|
| **count** | 48895.00 | 48895.00 | 48895.00 | 48895.00 | 38843.00 | 48895.00 | 48895.00 | 48895.00 |
| **mean** | 40.73 | -73.95 | 7.03 | 23.27 | 1.37 | 7.14 | 112.78 | 4.74 |
| **std** | 0.05 | 0.05 | 20.51 | 44.55 | 1.68 | 32.95 | 131.62 | 0.70 |
| **min** | 40.50 | -74.24 | 1.00 | 0.00 | 0.01 | 1.00 | 0.00 | 0.00 |
| **25%** | 40.69 | -73.98 | 1.00 | 1.00 | 0.19 | 1.00 | 0.00 | 4.25 |
| **50%** | 40.72 | -73.96 | 3.00 | 5.00 | 0.72 | 1.00 | 45.00 | 4.67 |
| **75%** | 40.76 | -73.94 | 5.00 | 24.00 | 2.02 | 2.00 | 227.00 | 5.17 |
| **max** | 40.91 | -73.71 | 1250.00 | 629.00 | 58.50 | 327.00 | 365.00 | 9.21 |

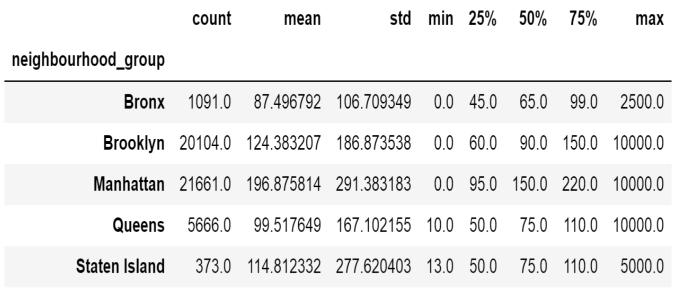
Figure 3: The Descriptive Statistics

Figure 4 represents the number of dataset upon each neighborhood group and the descriptive statistics for price upon neighborhood group. In Table 4a, Brooklyn has the most number of neighbors (3,920 data), followed by Manhattan (2,658 data). Averagely, Manhattan has the higher price among the other neighborhood groups, which is completely expected. Moreover, we found that the top 10 neighbors which averagely have expensive airbnbs are located in Manhattan (11,783 data) and Brooklyn (7,743 data).



| neighbourhood_group | count | unique | top | freq |
|---|---|---|---|---|
| Bronx | 1091 | 48 | Kingsbridge | 70 |
| Brooklyn | 20104 | 47 | Williamsburg | 3920 |
| Manhattan | 21661 | 32 | Harlem | 2658 |
| Queens | 5666 | 51 | Astoria | 900 |
| Staten Island | 373 | 43 | St. George | 48 |

(a) The number of dataset in each Neighbourhood group



| neighbourhood_group | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| Bronx | 1091.0 | 87.496792 | 106.709349 | 0.0 | 45.0 | 65.0 | 99.0 | 2500.0 |
| Brooklyn | 20104.0 | 124.383207 | 186.873538 | 0.0 | 60.0 | 90.0 | 150.0 | 10000.0 |
| Manhattan | 21661.0 | 196.875814 | 291.383183 | 0.0 | 95.0 | 150.0 | 220.0 | 10000.0 |
| Queens | 5666.0 | 99.517649 | 167.102155 | 10.0 | 50.0 | 75.0 | 110.0 | 10000.0 |
| Staten Island | 373.0 | 114.812332 | 277.620403 | 13.0 | 50.0 | 75.0 | 110.0 | 5000.0 |

(b) The Descriptive Statistics for Price divided by Neighbour group

Figure 4: The Neighbourhood group

## 4 Analysis

### 4.1 Regression Analysis with Forward step-wise selection

#### 4.1.1 Correlation between variables

Before applying the forward step-wise selection, we had Pearson correlation coefficient with the correlation matrix to check how the predictors are correlated each other. In Figure 6, the values of correlations between predictors are very small except that between number of reviews and reviews per month ($r = 0.59$). We also calculated eigenvalues to check the existence of multicollinearity. We found that none of variables was close to zero, such that there does not exist the problem of multicollinearity.

#### 4.1.2 Residual Plots

Secondly, we checked the residual plot to validate our model. If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data. Figure 9 represents the residual plots for ten predictors with the dependent variable, price in our multiple regression. In case of ideal residual plot, the red line would be horizontal. As seen in 9, there are not many outliers in each feature but, the points in residual plots on neighbor group with price and longitude with price, minimum nights with price and reviews per months with price are not randomly dispersed. This result may lead the model to be underfitted. To avoid underfitting, we normalized the data set and used polynomial transformation. (9 is shown in the last page.)

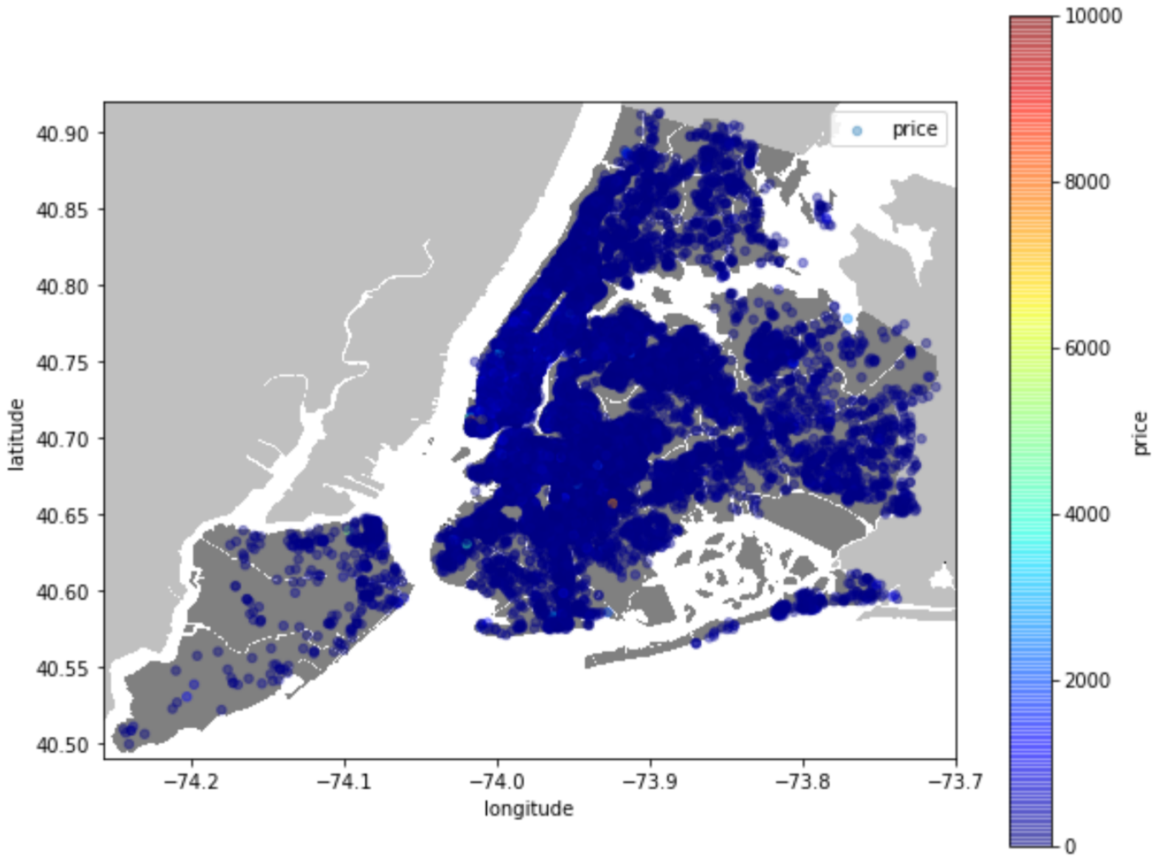


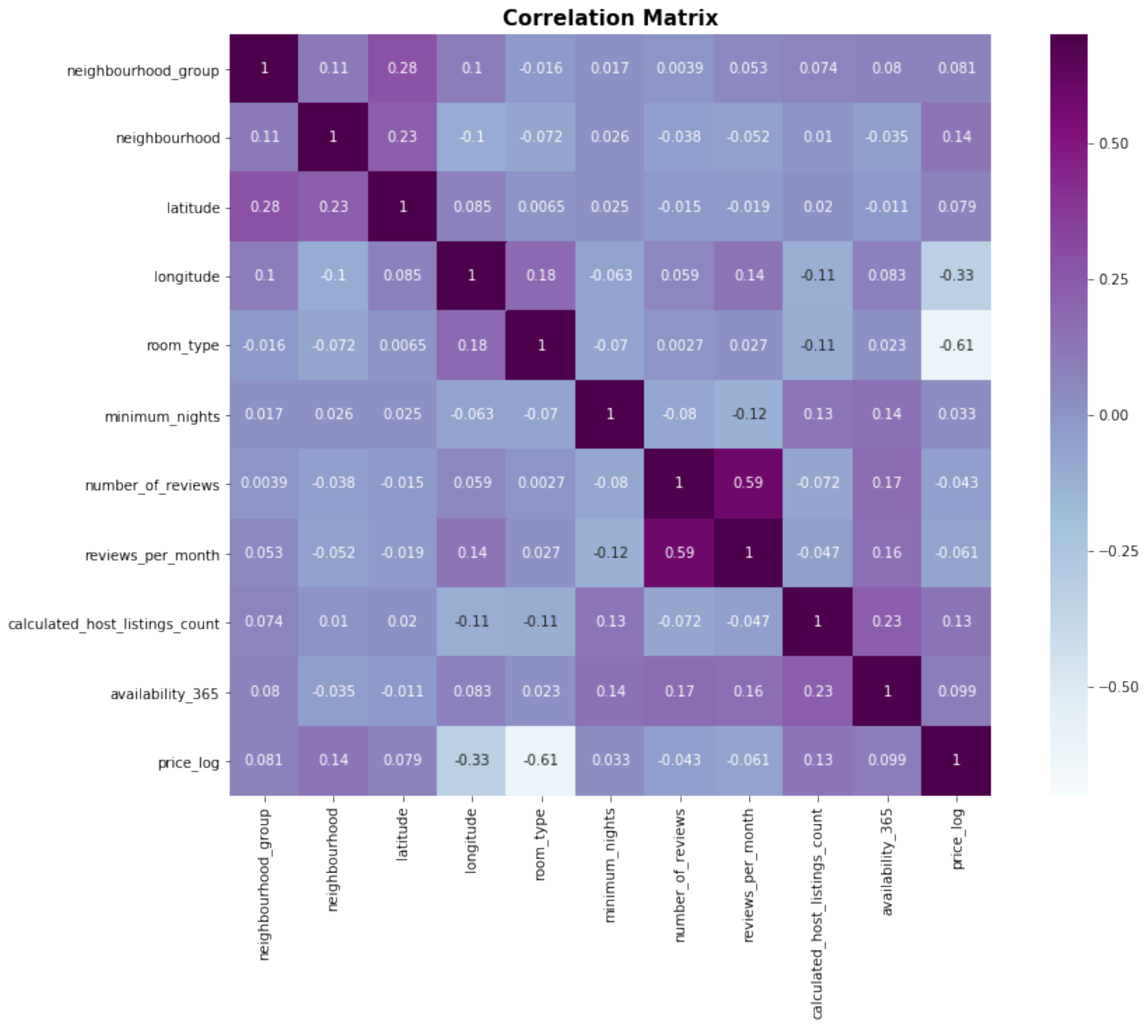
Figure 5: The correlation Matrix for Variables in Regress Model

### 4.1.3 Forward step-wise selection

Since this model has ten variables for predictors this study applied forward step-wise selection to our multiple regression model. We compared RSS, $C_p$, AIC, BIC, and adjusted $R^2$ for each group of predictors and found that there is no group that increases information criteria value. Only in BIC analysis, information criteria value was increased but, overally, the information criteria values for all predictors were the smallest values. Therefore, we conclude that all predictors are significantly useful for our model.

## 4.2 Comparing multiple regression with spatial basis function expansion

In this section, we compare MSE (Mean Squared Residual) of multiple regression with that of spatial basis function expansion. The less MSE is, The more accurately the model predicts the response variable, price of Airbnb. To examine them precisely, we also obtained the value of Ridge regression, Lasso regression as a substitute of simple regression. The result is



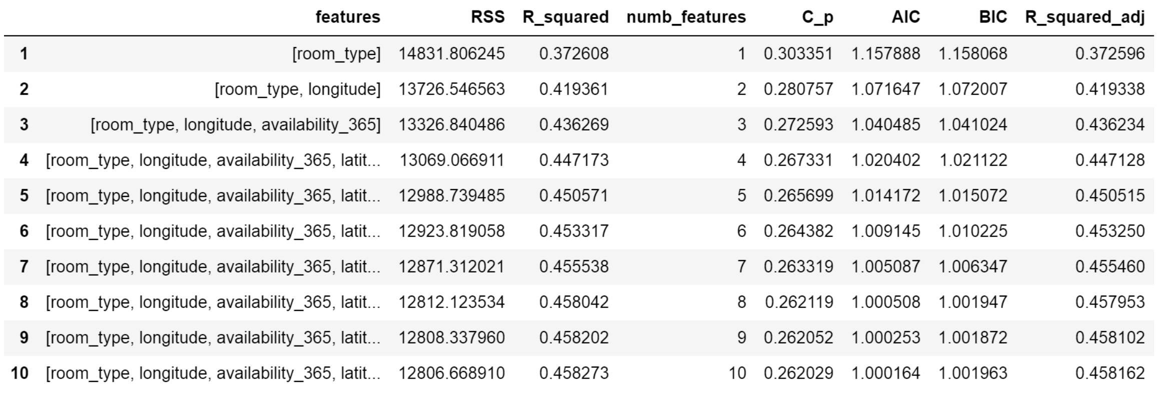
| | neighbourhood_group | neighbourhood | latitude | longitude | room_type | minimum_nights | number_of_reviews | reviews_per_month | calculated_host_listings_count | availability_365 | price_log |
|---|---|---|---|---|---|---|---|---|---|---|---|
| neighbourhood_group | 1 | 0.11 | 0.28 | 0.1 | -0.016 | 0.017 | 0.0039 | 0.053 | 0.074 | 0.08 | 0.081 |
| neighbourhood | 0.11 | 1 | 0.23 | -0.1 | -0.072 | 0.026 | -0.038 | -0.052 | 0.01 | -0.035 | 0.14 |
| latitude | 0.28 | 0.23 | 1 | 0.085 | 0.0065 | 0.025 | -0.015 | -0.019 | 0.02 | -0.011 | 0.079 |
| longitude | 0.1 | -0.1 | 0.085 | 1 | 0.18 | -0.063 | 0.059 | 0.14 | -0.11 | 0.083 | -0.33 |
| room_type | -0.016 | -0.072 | 0.0065 | 0.18 | 1 | -0.07 | 0.0027 | 0.027 | -0.11 | 0.023 | -0.61 |
| minimum_nights | 0.017 | 0.026 | 0.025 | -0.063 | -0.07 | 1 | -0.08 | -0.12 | 0.13 | 0.14 | 0.033 |
| number_of_reviews | 0.0039 | -0.038 | -0.015 | 0.059 | 0.0027 | -0.08 | 1 | 0.59 | -0.072 | 0.17 | -0.043 |
| reviews_per_month | 0.053 | -0.052 | -0.019 | 0.14 | 0.027 | -0.12 | 0.59 | 1 | -0.047 | 0.16 | -0.061 |
| calculated_host_listings_count | 0.074 | 0.01 | 0.02 | -0.11 | -0.11 | 0.13 | -0.072 | -0.047 | 1 | 0.23 | 0.13 |
| availability_365 | 0.08 | -0.035 | -0.011 | 0.083 | 0.023 | 0.14 | 0.17 | 0.16 | 0.23 | 1 | 0.099 |
| price_log | 0.081 | 0.14 | 0.079 | -0.33 | -0.61 | 0.033 | -0.043 | -0.061 | 0.13 | 0.099 | 1 |

Figure 6: The correlation Matrix for Variables in Regress Model

shown in 1. For the result of spatial basis function expansion, we conclude that the price is spatially and significantly related with other features. For example, we want to predict the price of Airbnb in (40.90,-74.22), we obtained 554.34. Also, MSE of this model is smaller than those of other models such as simple, Ridge,and Lasso regression, such that spatial basis function expansion is the most accurate model for the prediction of price.

| Spatial Basis | Simple | Ridge | Lasso |
|---|---|---|---|
| 0.4268 | 0.5308 | 0.5226 | 0.5204 |

Table 1: The MSE of Four Models



| | features | RSS | R_squared | numb_features | C_p | AIC | BIC | R_squared_adj |
|---|---|---|---|---|---|---|---|---|
| **1** | [room_type] | 14831.806245 | 0.372608 | 1 | 0.303351 | 1.157888 | 1.158068 | 0.372596 |
| **2** | [room_type, longitude] | 13726.546563 | 0.419361 | 2 | 0.280757 | 1.071647 | 1.072007 | 0.419338 |
| **3** | [room_type, longitude, availability_365] | 13326.840486 | 0.436269 | 3 | 0.272593 | 1.040485 | 1.041024 | 0.436234 |
| **4** | [room_type, longitude, availability_365, latit... | 13069.066911 | 0.447173 | 4 | 0.267331 | 1.020402 | 1.021122 | 0.447128 |
| **5** | [room_type, longitude, availability_365, latit... | 12988.739485 | 0.450571 | 5 | 0.265699 | 1.014172 | 1.015072 | 0.450515 |
| **6** | [room_type, longitude, availability_365, latit... | 12923.819058 | 0.453317 | 6 | 0.264382 | 1.009145 | 1.010225 | 0.453250 |
| **7** | [room_type, longitude, availability_365, latit... | 12871.312021 | 0.455538 | 7 | 0.263319 | 1.005087 | 1.006347 | 0.455460 |
| **8** | [room_type, longitude, availability_365, latit... | 12812.123534 | 0.458042 | 8 | 0.262119 | 1.000508 | 1.001947 | 0.457953 |
| **9** | [room_type, longitude, availability_365, latit... | 12808.337960 | 0.458202 | 9 | 0.262052 | 1.000253 | 1.001872 | 0.458102 |
| **10** | [room_type, longitude, availability_365, latit... | 12806.668910 | 0.458273 | 10 | 0.262029 | 1.000164 | 1.001963 | 0.458162 |

Figure 7: The correlation Matrix for Variables in Regress Model

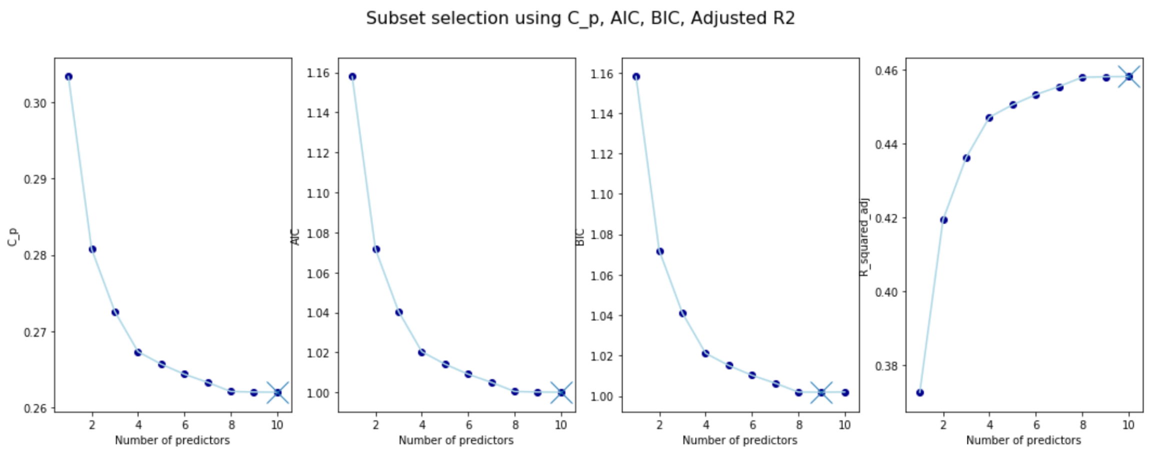


Figure 8: The correlation Matrix for Variables in Regress Model

## 5 Discussion

This study verified how the spatial basis function expansion is accurate for the prediction of the price of Airbnb. This result would be useful for both renters and guests as well as Airbnb company for the analysis of Airbnb features and price. In addition, we could predict the number of reviews or reviews per month as a dependent variable instead of price. We can check if the location and these variables are spatially related or which model is the most adaptive for the prediction of them. Thirdly, we use another model for large spatial dataset instead of the spatial basis function expansion because we managed only 150 knots of the Gaussian kernel in the spatial basis function expansion, such that the results would be better.

## 6 Bibiliography

Banerjee, S.,Carlin, B.P., Gelfand, A.E., Hierarchical Modeling and Analysis for Spatial Data, Second Edition
Jay M. V., Modern Spatial Statistics: Basis Functions, Convolutions, and Big Data
Jia Y., Risser M, Saha A., Basis Function Models for Nonstationary Spatial Modeling
Bradely Note in class

## 7 Python and R Code

```
# coding: utf-8

# In[20]:



import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
import itertools
import time

from scipy import stats
from scipy.stats import norm
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier

from sklearn.linear_model import LinearRegression
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
from sklearn import metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from math import sqrt
from sklearn.metrics import r2_score


# In[2]:


data = pd.read_csv('input/AB_NYC_2019.csv',delimiter = ',')
data.head(5)
data.describe()


# In[5]:


data.columns
len(data)
len(data.columns)
data.shape #the number of rows and columns


# In[6]:


data.dtypes


# In[7]:


#looking to find out how many nulls are found in each column in dataset
data.isnull().sum()
```

```
# In[3]:


#The above distribution graph shows that there is a right-skewed distribution on
#log transformation will be used to make this feature less skewed.
#Since division by zero is a problem, log+1 transformation
plt.figure(figsize=(10,10))
sns.distplot(data['price'],fit=norm)
plt.title("Price Distribution Plot",size=15, weight='bold')


# In[4]:


data['price_log'] = np.log(data.price+1)


# In[10]:


plt.figure(figsize=(10,8))
sns.distplot(data['price_log'],fit=norm)
plt.title("Log-price Distribution Plot",size=15, weight='bold')

plt.figure(figsize=(6,6))
stats.probplot(data['price_log'],plot=plt)
plt.show()


# In[5]:


#dropping columns that are not significant
data1 = data.drop(['id','name','host_id','host_name', 'last_review','price'], ax
#examing the changesa
data1.head(5)
```

```
df = pd.DataFrame(data1)
df.head(5)
data2 = df.round(2)
data2.describe().round(2)


# In[6]:


df.room_type.value_counts()
df.neighbourhood_group.value_counts()
#df.neighbourhood.value_counts()
#pd.DataFrame(df.neighbourhood_group.unique())
df['neighbourhood'].groupby(df['neighbourhood_group']).describe()


# In[ ]:


df.neighbourhood_group.unique()
len(set(data.neighbourhood_group)) #we use "set" to remove duplicates
df.room_type.unique()
len(set(data.room_type))
len(set(df.neighbourhood))


# In[7]:


pd.DataFrame(df.neighbourhood_group.unique())
df.room_type.value_counts()


# In[8]:


df.isnull().sum()
```

```
#Number of reviews features has some missing data.
#It will be replaced with zero.


# In[41]:


#replacing all NaN values in 'reviews_per_month' with 0
df.fillna({'reviews_per_month':0}, inplace=True)
df.reviews_per_month.isnull().sum()


# In[42]:


df['neighbourhood_group'] = df['neighbourhood_group'].astype("category").cat.cod
df['neighbourhood'] = df['neighbourhood'].astype("category").cat.codes
df['room_type'] = df['room_type'].astype("category").cat.codes
df.info()


# In[10]:


plt.figure(figsize=(15,10))
corr=df.corr(method='pearson')
sns.heatmap(corr, annot=True, square=True,cmap="BuPu",vmin=-0.7, vmax=0.7)
plt.title("Correlation Matrix", size =15, weight='bold')


# In[11]:


#multicollinearity will help to measure the relationship between explanatory var
multicollinearity, V=np.linalg.eig(corr)
multicollinearity
#None of the eigenvalues is close to zero. no multicollinearity exists in the da
```

```
# In[45]:


#Residual Plots
#detect outliers, non-linear data for regression models.
#the residual plots for each explanatory variables with the price.
df_x, df_y = df.iloc[:,:-1], df.iloc[:,-1]


# In[46]:


def fit_linear_reg(X,Y):
#Fit linear regression model and return RSS and R squared values
model_k = linear_model.LinearRegression(fit_intercept = True)
model_k.fit(X,Y)
RSS = mean_squared_error(Y,model_k.predict(X)) * len(Y)
R_squared = model_k.score(X,Y)
return RSS, R_squared


# In[44]:


export_csv = df.to_csv('input/df.csv', index = None, header=True)


# In[47]:


#Initialization variables
Y = df_y
X = df_x
k = 10
```

```
remaining_features = list(X.columns.values)
features = []
RSS_list, R_squared_list = [np.inf], [np.inf] #Due to 1 indexing of the loop...
features_list = dict()

for i in range(1,k+1):
best_RSS = np.inf

for combo in itertools.combinations(remaining_features,1):

RSS = fit_linear_reg(X[list(combo) + features],Y)    #Store temp result

if RSS[0] < best_RSS:
best_RSS = RSS[0]
best_R_squared = RSS[1]
best_feature = combo[0]

#Updating variables for next loop
features.append(best_feature)
remaining_features.remove(best_feature)

#Saving values for plotting
RSS_list.append(best_RSS)
R_squared_list.append(best_R_squared)
features_list[i] = features.copy()


# In[48]:


print('Forward␣stepwise␣subset␣selection')
print('Number␣of␣features␣|', 'Features␣|', 'RSS')
display([(i,features_list[i], round(RSS_list[i])) for i in range(1,5)])


# In[49]:
```

```
df1 = pd.concat([pd.DataFrame({'features':features_list}),pd.DataFrame({'RSS':R
df1['numb_features'] = df1.index


# In[50]:


#Initializing useful variables
m = len(Y)
p = 11
hat_sigma_squared = (1/(m - p -1)) * min(df1['RSS'])

#Computing
df1['C_p'] = (1/m) * (df1['RSS'] + 2 * df1['numb_features'] * hat_sigma_squared
df1['AIC'] = (1/(m*hat_sigma_squared)) * (df1['RSS'] + 2 * df1['numb_features']
df1['BIC'] = (1/(m*hat_sigma_squared)) * (df1['RSS'] + np.log(m) * df1['numb_f
df1['R_squared_adj'] = 1 - ( (1 - df1['R_squared'])*(m-1)/(m-df1['numb_features'
df1


# In[51]:


df1['R_squared_adj'].idxmax()
df1['R_squared_adj'].max()


# In[52]:


variables = ['C_p', 'AIC','BIC','R_squared_adj']
fig = plt.figure(figsize = (18,6))

for i,v in enumerate(variables):
ax = fig.add_subplot(1, 4, i+1)
ax.plot(df1['numb_features'],df1[v], color = 'lightblue')
ax.scatter(df1['numb_features'],df1[v], color = 'darkblue')
if v == 'R_squared_adj':
ax.plot(df1[v].idxmax(),df1[v].max(), marker = 'x', markersize = 20)
else:
ax.plot(df1[v].idxmin(),df1[v].min(), marker = 'x', markersize = 20)
ax.set_xlabel('Number of predictors')
ax.set_ylabel(v)

fig.suptitle('Subset selection using C_p, AIC, BIC, Adjusted R2', fontsize = 16)
plt.show()


# In[31]:


f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,0],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,1],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})


# In[13]:


f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,2],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})


# In[33]:


f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,3],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
```

*# In[34]:*

```
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,4],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
```

*# In[35]:*

```
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,5],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
```

*# In[36]:*

```
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,6],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
```

*# In[37]:*

```
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,7],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
```

*# In[38]:*

```
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,8],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})


# In[39]:


f, axes = plt.subplots(figsize=(10,7))
sns.residplot(df_x.iloc[:,9],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})


# In[45]:


df_x_transformed = StandardScaler().fit_transform(df_x)
x_train, x_test, y_train, y_test = train_test_split(df_x_transformed, df_y, test


# In[60]:


a = pd.DataFrame(df_x_transformed)
f, axes = plt.subplots(figsize=(10,7))
sns.residplot(a.iloc[:,3],df_y,lowess=True, scatter_kws={'alpha': 0.5},
line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})


# In[46]:


### Linear Regression ###

def linear_reg(input_x, input_y, cv=5):
## Defining parameters
model_LR= LinearRegression()
```

```
parameters = {'fit_intercept':[True,False], 'normalize':[True,False], 'copy_X':[

## Building Grid Search algorithm with cross-validation and Mean Squared Error s

grid_search_LR = GridSearchCV(estimator=model_LR,
param_grid=parameters,
scoring='neg_mean_squared_error',
cv=cv,
n_jobs=-1)

## finding the best parameters.

grid_search_LR.fit(input_x, input_y)
best_parameters_LR = grid_search_LR.best_params_
best_score_LR = grid_search_LR.best_score_
print(best_parameters_LR)
print(best_score_LR)


# linear_reg(nyc_model_x, nyc_model_y)


# In[48]:


kfold_cv=KFold(n_splits=5, random_state=42, shuffle=False)
for train_index, test_index in kfold_cv.split(df_x_transformed,df_y):
X_train, X_test = df_x_transformed[train_index], df_x_transformed[test_index]
y_train, y_test = df_y[train_index],df_y[test_index]


# In[52]:


Poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
X_train = Poly.fit_transform(X_train)
```

```
X_test = Poly.fit_transform(X_test)


# In[53]:


##Linear Regression
lr = LinearRegression(copy_X= True, fit_intercept = True, normalize = True)
lr.fit(X_train, y_train)
lr_pred= lr.predict(X_test)


# In[55]:


print('MAE: %f'% mean_absolute_error(y_test, lr_pred)) #the difference between p
print('RMSE: %f'% np.sqrt(mean_squared_error(y_test, lr_pred)))
print('R2 %f' % r2_score(y_test, lr_pred)) #the goodness of fit measure


# In[13]:


#what hosts have the most listings on Airbnb platform and taking advantage of th
#Interestingly, the first host has more than 300+ listings.
top_host = df.host_id.value_counts().head(10)
top_host


# In[14]:


df.calculated_host_listings_count.max()


# In[15]:
```

```
f, ax = plt.subplots(figsize=(8,6))
viz1 = top_host.plot(kind = 'bar')
viz1.set_title('Hosts with the most listings in NYC')
viz1.set_ylabel('Count of listings')
viz1.set_xlabel('Host IDs')


# In[28]:


df['price'].groupby(df['neighbourhood_group']).describe()


# In[17]:


#we make subgroup for each neighbourhood_group
sub_1 = df.loc[df['neighbourhood_group'] == 'Bronx']
sub_2 = df.loc[df['neighbourhood_group'] == 'Brooklyn']
sub_3 = df.loc[df['neighbourhood_group'] == 'Manhattan']
sub_4 = df.loc[df['neighbourhood_group'] == 'Queens']
sub_5 = df.loc[df['neighbourhood_group'] == 'Staten Island']


# In[18]:


plt.figure()
#plt.xlim(0,500)
fig = sns.distplot(sub_2['price'],fit=norm);


# In[19]:


res = stats.probplot(sub_1['price'], plot=plt)
```

# In[29]:

```
#neighbourhood where airbnb concentrates on (top 10)
#pd.DataFrame
a = pd.DataFrame(df['neighbourhood'].value_counts().head(10).index.tolist())
#a = df.sort_values(by=['neighbourhood'], ascending=False).head(10)
neigh = []
neighh = []
#for i in range (0,10):
aa0 = a.iloc[0]
aa1 = a.iloc[1]
aa2 = a.iloc[2]
aa3 = a.iloc[3]
neigh1 = df.loc[df['neighbourhood']== a.iloc[0][0]]
neigh2 = df.loc[df['neighbourhood']== aa1[0]]
```

# In[30]:

```
#Finally I made it.
j = 0
neigh10 = []
for i in range (0,10):
globals()["aa" + str(j)] = a.iloc[j]
globals()["neigh" + str(j)] = df.loc[df['neighbourhood']== globals()["aa" + str(
j += 1
```

# In[31]:

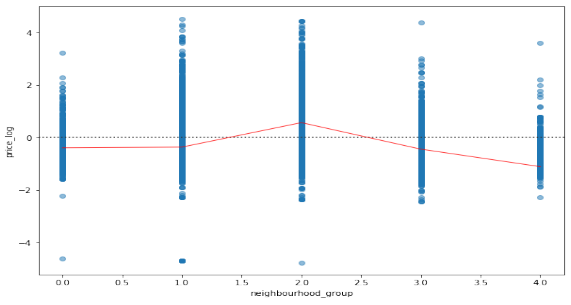
```
neigh10 = pd.concat([neigh1, neigh2, neigh3, neigh4, neigh5, neigh6, neigh7, nei
neigh10.neighbourhood_group.value_counts()
```

```
# In[34]:


import urllib
#initializing the figure size
plt.figure(figsize=(10,8))
#loading the png NYC image found on Google and saving to my local folder along w
i=urllib.request.urlopen('https://upload.wikimedia.org/wikipedia/commons/e/ec/Ne
nyc_img=plt.imread(i)
#scaling the image based on the latitude and longitude max and mins for proper c
plt.imshow(nyc_img,zorder=0,extent=[-74.258, -73.7, 40.49,40.92])
ax=plt.gca()
#using scatterplot again
df.plot(kind='scatter', x='longitude', y='latitude', label='price', c='price', a
plt.legend()
plt.show()


# In[ ]:



jupyter nbconvert-to script notebook_name.ipynb
```

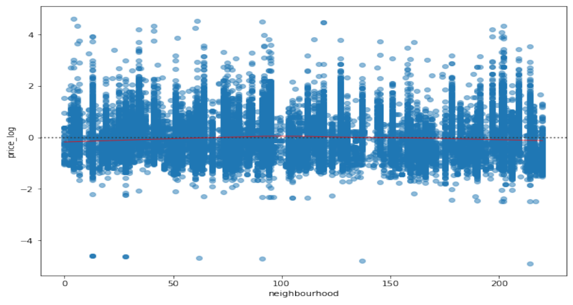
```
rm(list=ls())



library(maps)
library(mapproj)
library(geoR)
library(ggplot2)
library(fields)
library(rgdal)
library(maptools)
library(spdep)
library(RColorBrewer)
library(classInt)
library(MBA)
library(fields)
library(dplyr)
library(matlib)

#load data
mydata=read.csv("C://Users/user/Documents/AB_NYC_2019.csv",header = TRUE, sep=","
#data

#load n*p design matrix
z = mydata[!duplicated(mydata[7:8]), ]
#z7 = (z[,7] - min(z[7]))/(max(z[7])-min(z[7]))
#z8 = (z[,8] - min(z[8]))/(max(z[8])-min(z[8]))
#X=cbind(1,z7,z8) ##load the coordinates of the spatial locations
X=cbind(1,z[7:8])
#estimate the beta without taking into account correlations
X=as.matrix(X)
XpX=t(X)%*%X
invXpX=solve(XpX) #nonsingular
price = as.matrix(log(z[,10]+1))
hist(price)
betahat=(invXpX)%*%t(X)%*%price #Price
betahat
#detrend
ztilde=price-X%*%betahat


#load the coordinates of the spatial locations
coord=z[7:8]
coord1 = as.matrix(coord)
#load the knots of the Gaussian kernel
knots=cover.design(coord,150)   #level3.csv
#measurement error square (defined in the Question)
#epssquare=5.6062

#Compute the n*r spatial basis function matrix
```

```
r=dim(knots[1:100,])[1]
source("C://Users/user/Documents/Create_GK.R")
source("C://Users/user/Documents/FRKmisslocs2.R")
source("C://Users/user/Documents/EM3.R")
S=Create_GK(coord,cbind(knots[,2],knots[,1]),15,1)
x.res=100
y.res=100
#surf=mba.surf(cbind(coord,S[1]),no.X = x.res,no.Y = y.res,h=5,m=2,extend = FALS
#image.plot(surf,xaxs="r",yaxs="r",xlab="lat",ylab="long")
out.EM=EM3(S,ztilde,0,4)
K=matrix(sapply(out.EM[1],as.numeric),r,r)
sigtemp=as.vector(out.EM[2])
sigtemp=sigtemp[[1]]
sigxi=as.numeric(sigtemp[length(sigtemp)])

######Kriging Predictor######
Ksym=(K+t(K))/2 #truncated KL expansion, positive definite
Sp=Create_GK(t(as.matrix(c(40.90,-74.22),1,2)),cbind(knots[,2],knots[,1]),30,1)

out.FRK=FRKmisslocs2(ztilde,S,Sp,Ksym,sigxi,0) #S=G(basis function) in our note,
Yhattilde=matrix(sapply(out.FRK[1][1],as.numeric),1,1)
MSPE=matrix(sapply(out.FRK[2][1],as.numeric),1,1)

Yhat=t(as.matrix(c(1,40.90,-74.22)))%*%betahat+Yhattilde #our rogers lat, lon

MSPE #mean squared prediction errors
Yhat #predicted value
#' ###(b) 2. The predicted value and mean squared prediction errors when incorpo
#compute the EM estimates
out.EM=EM3(S,ztilde,epssquare,4) #changed
K=matrix(sapply(out.EM[1],as.numeric),r,r) #sapply: apply a function over a vect
sigtemp=as.vector(out.EM[2])
sigtemp=sigtemp[[1]]
sigxi=as.numeric(sigtemp[length(sigtemp)])

######Kriging Predictor######
Ksym=(K+t(K))/2
```

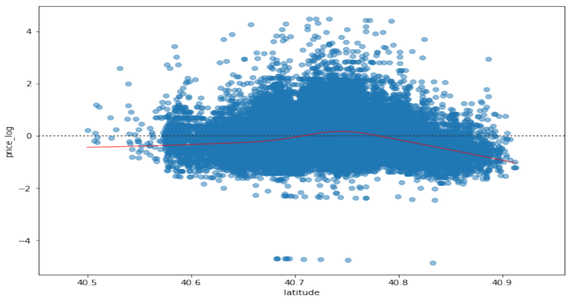
```
Sp=Create_GK(t(as.matrix(c(30.444,-84.299),1,2)),cbind(knots[,2],knots[,1]),30,1

out.FRK=FRKmisslocs2(ztilde,S,Sp,Ksym,sigxi,epssquare) #changed
Yhattilde=matrix(sapply(out.FRK[1][1],as.numeric),1,1)
MSPE=matrix(sapply(out.FRK[2][1],as.numeric),1,1)

Yhat=t(as.matrix(c(1,30.444,-84.299)))%*%betahat+Yhattilde

MSPE #mean squared prediction errors
Yhat #predicted value
```



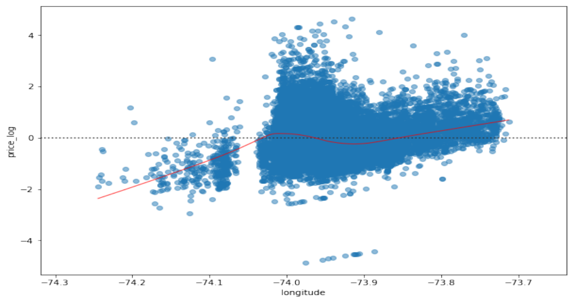
(a) Neighbor group with Price



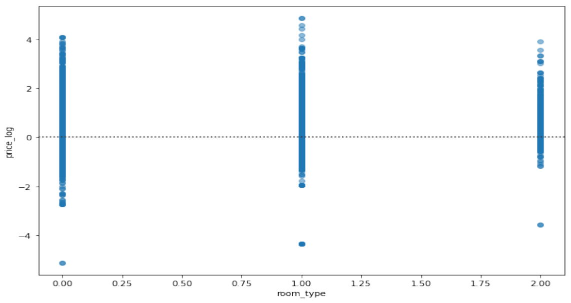
(b) Neighbor with Price



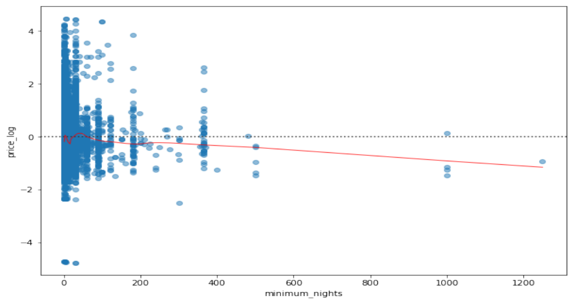
(c) latitude with Price



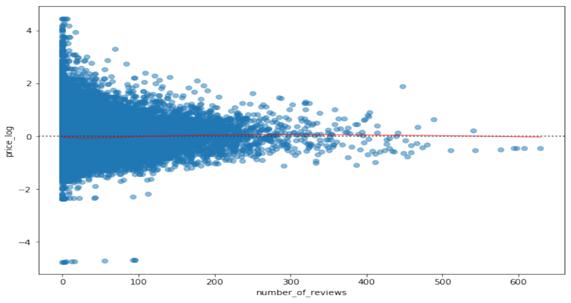
(d) longitude with Price



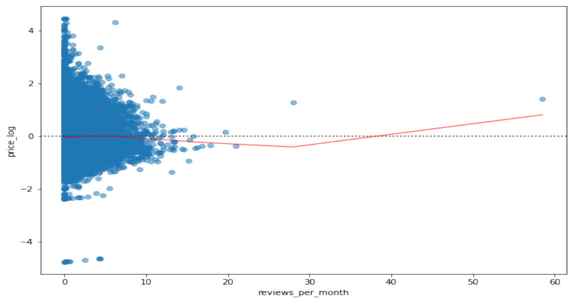
(e) Room type with Price



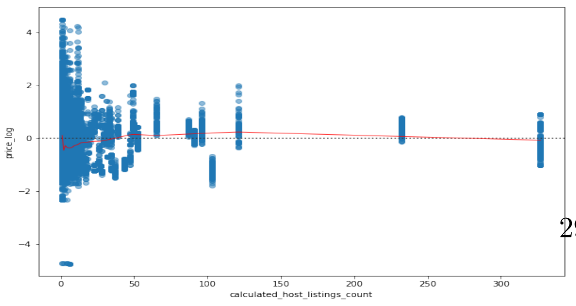
(f) Minumum Nights with Price

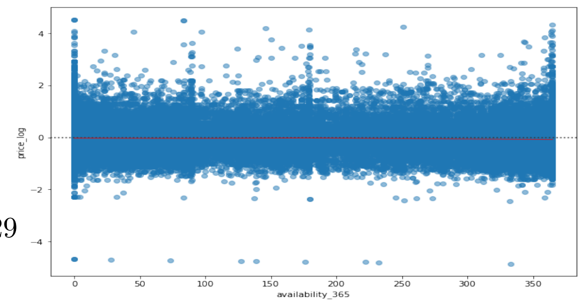


(g) Number of Reviews with Price



(h) Reviews per Months with Price





(i) Host listings count with Price

(j) Availability among 365 with Price

Figure 9: The Residual Plots for Each Explanatory variables with the Dependent variable,