```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
# from google.colab import drive
# from google.colab import files
import statsmodels.api as sm
from matplotlib import pyplot as plt
import numpy as np
from datetime import datetime
import re
import sklearn
from sklearn.linear_model import (
    LinearRegression,
    TheilSenRegressor,
    RANSACRegressor,
    HuberRegressor,
)
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

import math
import random


#Test
#new


# from google.colab import drive
# drive.mount('/content/drive')


#reading data
dfMain=pd.read_csv('AB_NYC_2019.csv')
df = dfMain.copy()
df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latit
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68

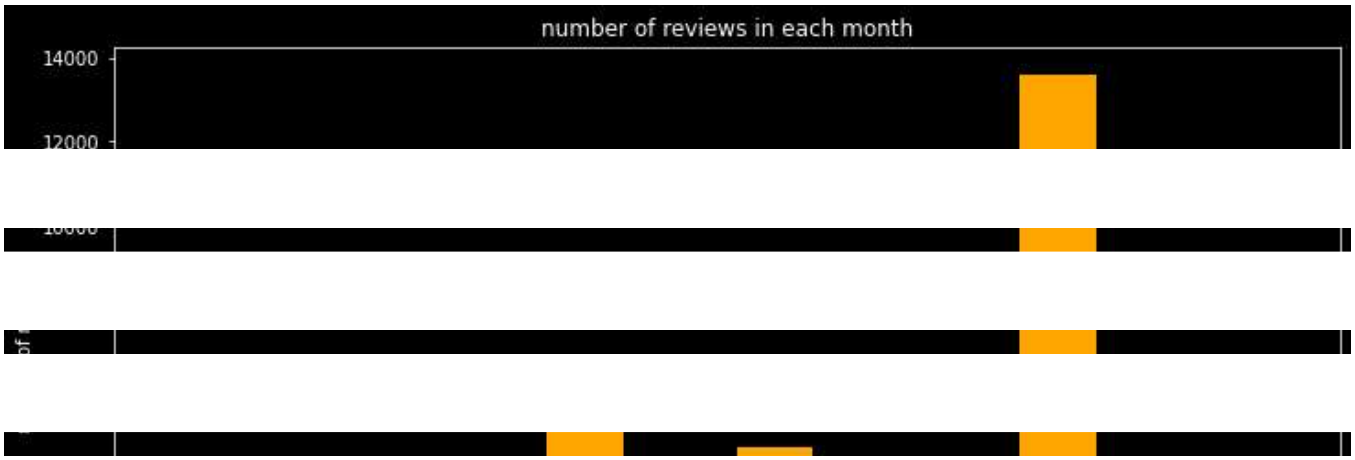
Data Preprocessing

central park

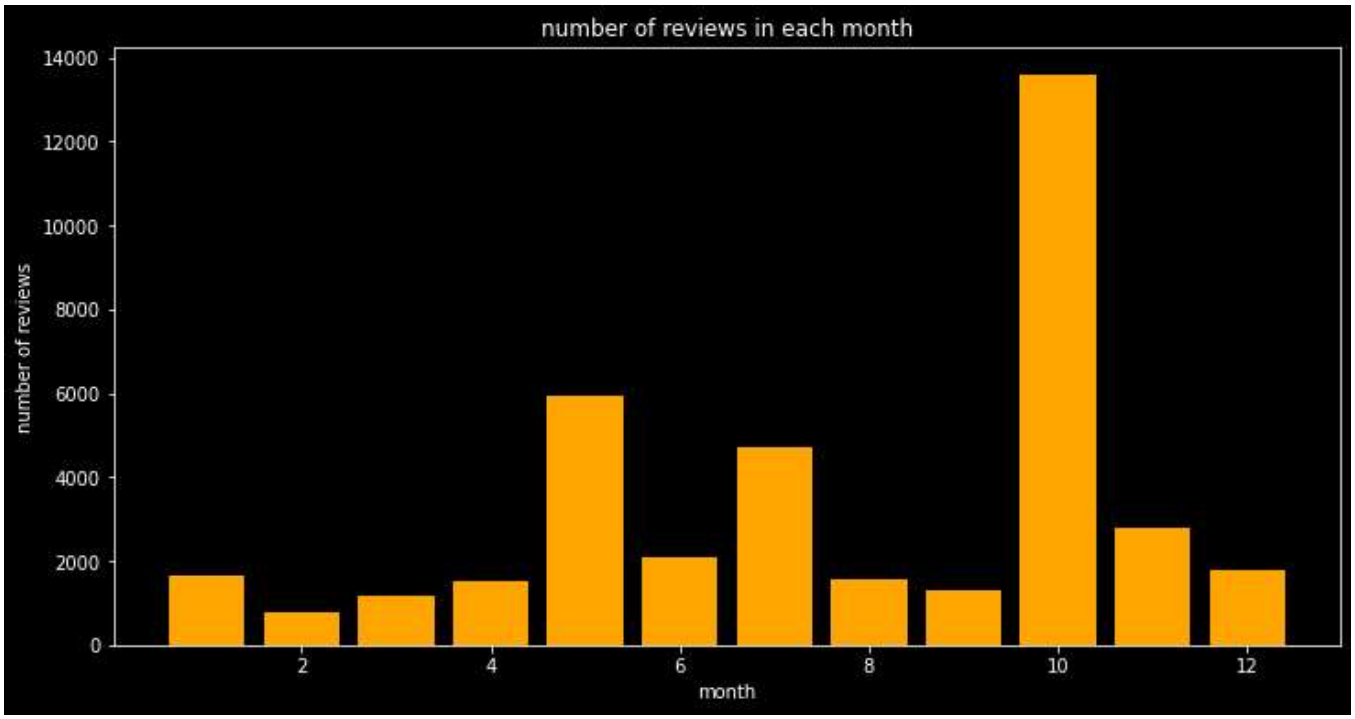
```
df.dropna()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem
		Large Cozy 1 BR				

```
dfBackUp = df.copy()
df['last_review'] = pd.to_datetime(df['last_review'])
df['last_review_month'] = pd.DatetimeIndex(df['last_review']).month
monthes = df['last_review_month'].dropna()
monthes.value_counts()
plt.figure(figsize=(12,6))
plt.bar(x=monthes.unique() ,height=monthes.value_counts()  , color= 'orange')
plt.title('number of reviews in each month')
plt.xlabel('month')
plt.ylabel('number of reviews')
plt.show()
df = dfBackUp
```




```
plt.figure(figsize=(12,6))
plt.bar(x=monthes.unique() ,height=monthes.value_counts()  , color= 'orange')
plt.title('number of reviews in each month')
plt.xlabel('month')
plt.ylabel('number of reviews')
plt.show()
```




```
dfMain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   name                            48879 non-null  object
 2   host_id                         48895 non-null  int64
```

```
 3   host_name                       48874 non-null  object
 4   neighbourhood_group             48895 non-null  object
 5   neighbourhood                   48895 non-null  object
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object
 9   price                           48895 non-null  int64
 10  minimum_nights                  48895 non-null  int64
 11  number_of_reviews               48895 non-null  int64
 12  last_review                     38843 non-null  object
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

```
df['last_review'].value_counts()
```

```
2019-06-23    1413
2019-07-01    1359
2019-06-30    1341
2019-06-24     875
2019-07-07     718
              ...
2015-01-09       1
2018-01-29       1
2013-03-31       1
2017-11-16       1
2014-09-07       1
Name: last_review, Length: 1764, dtype: int64
```

the columns are :

-id : listing ID

-name : name of the listing

-host ID

-host_name

-neighbourhood_group : location

-neighbourhood : area

-latitude : latitude coordinates

-longitude : longitude coordinates

-room_type

-price : price in dollars

-minimum_nights

-number_of_reviews

-last_review : the date of last history

-reviews_per_month

-calculated_host_listings_count : number of lisiting for the host

-availability_365 : number of available days in year

Handling nulls

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

```
sns.heatmap(df.isnull(), cbar = False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72012d9890>
```


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

```
id                                  0
name                               16
host_id                             0
host_name                          21
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10052
reviews_per_month               10052
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

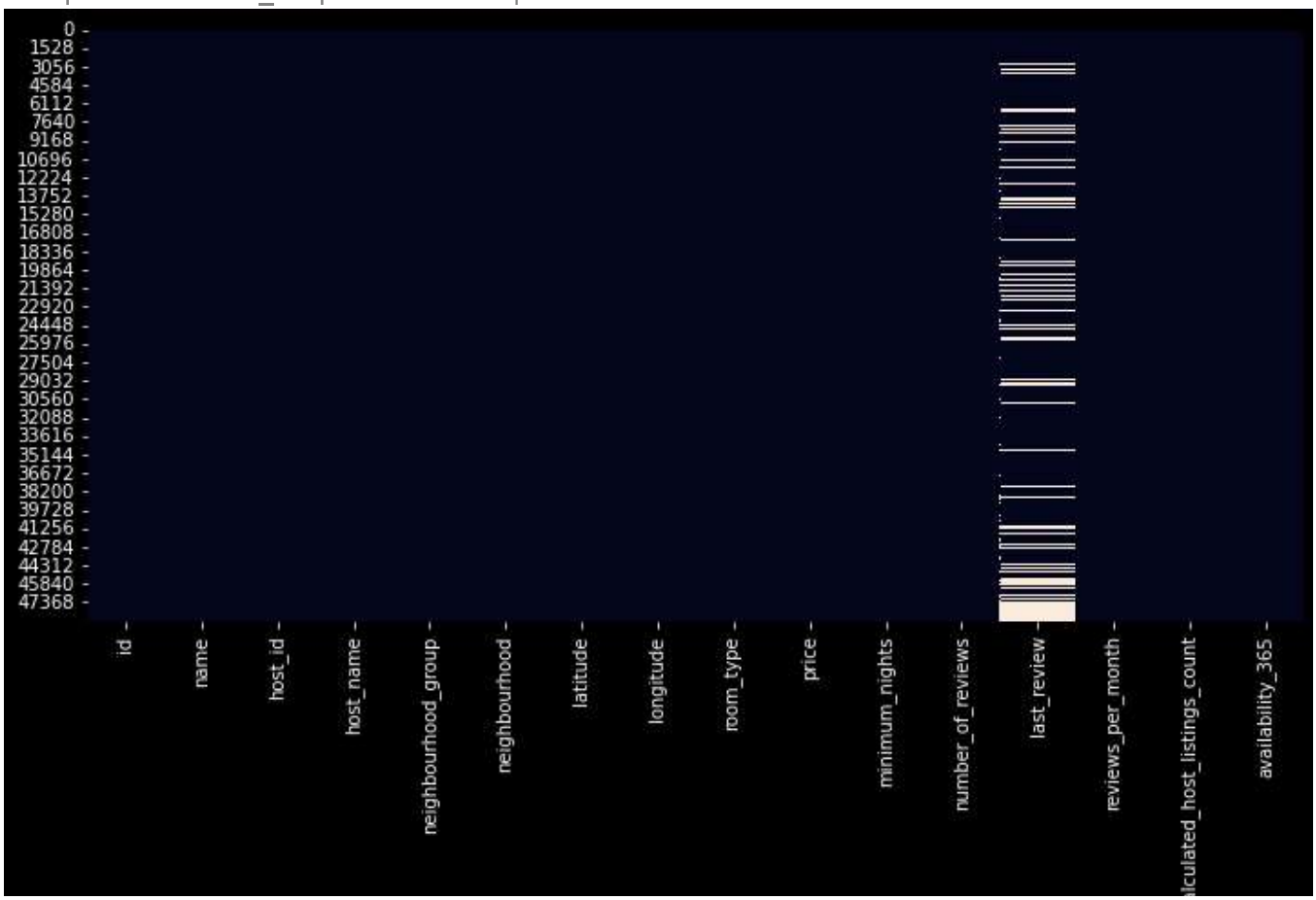
As shown before, the nulls are only in'last_review' and 'reviews_per_month' and these nulls are results of 0s in 'number_of_reviews'

so now 'number_of_reviews' will be filled by 0

```
df['reviews_per_month'].replace(np.nan , 0 , inplace=True)
```

```
sns.heatmap(df.isnull(), cbar = False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f71ff1cef90>
```



now the 'last_review' for the samples that have no reviews will be filled

but first it`s better to be categorized by year and season

then the nulls will be filled by 'no reviews'

```
# determining the rows season from its month
SP = df['last_review'].str.extract(r'(\d+-0[3-5]-\d+)').dropna()
SU = df['last_review'].str.extract(r'(\d+-0[6-8]-\d+)').dropna()
AU = df['last_review'].str.extract(r'(\d+-(?:09|10|11)-\d+)').dropna()
WI = df['last_review'].str.extract(r'(\d+-(?:12|01|02)-\d+)').dropna()
```

```
WI_values = [re.sub(r'\d+-\d+-\d+' , 'Winter-' + str(x)[0:4]  , str(x)) for x in dfMain['last
SP_values = [re.sub(r'\d+-\d+-\d+' , 'Spring-' + str(x)[0:4]  , str(x)) for x in dfMain['last
SU_values = [re.sub(r'\d+-\d+-\d+' , 'Summer-' + str(x)[0:4]  , str(x)) for x in dfMain['last
AU_values = [re.sub(r'\d+-\d+-\d+' , 'Autumn-' + str(x)[0:4]  , str(x)) for x in dfMain['last
```

```
# assigning the values
df['last_review'][WI.index] = WI_values
df['last_review'][SP.index] = SP_values
df['last_review'][SU.index] = SU_values
df['last_review'][AU.index] = AU_values
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  """
```

```
#filling nulls with 'no reviews'
df['last_review'].replace(np.nan , 'no reviews' , inplace = True)
```

```
df['last_review'].value_counts()
```

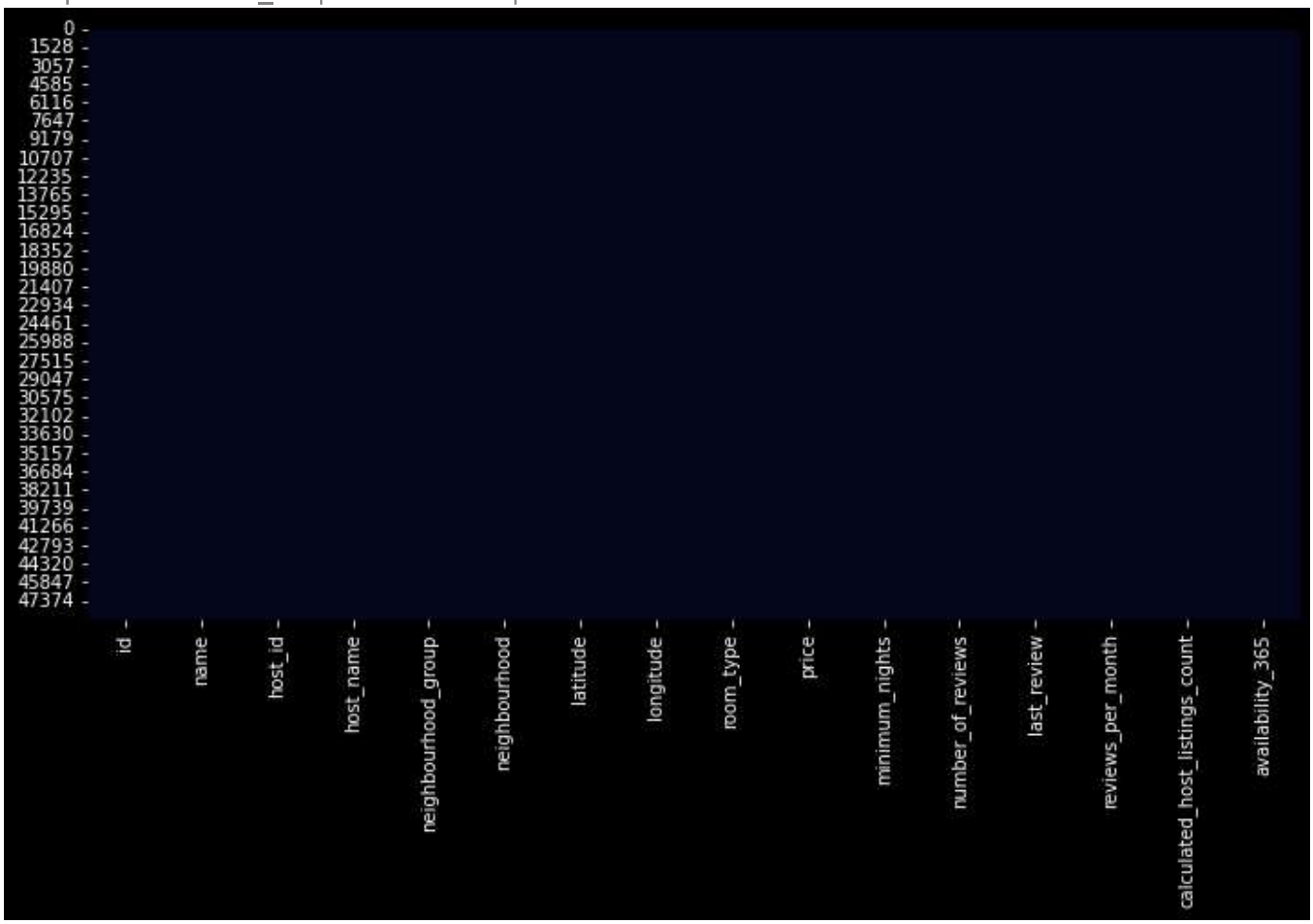
```
Summer-2019    17326
no reviews     10052
Spring-2019     5973
Winter-2019     1910
Autumn-2018     1909
Winter-2018     1642
Summer-2018     1576
Autumn-2017      934
Spring-2018      923
Summer-2017      919
Summer-2016      895
Winter-2017      815
Autumn-2016      703
Winter-2016      614
Autumn-2015      585
Spring-2017      537
Spring-2016      495
Summer-2015      398
Winter-2015      287
Spring-2015      123
Autumn-2014       77
Summer-2014       63
Spring-2014       31
Winter-2014       28
Autumn-2013       18
Spring-2013       12
Autumn-2012       11
```

```
Summer-2013        9
Winter-2013        9
Summer-2012        7
Winter-2012        6
Spring-2011        3
Autumn-2011        2
Winter-2011        2
Spring-2012        1
Name: last_review, dtype: int64
```

```
df.dropna(inplace= True)
```

```
sns.heatmap(df.isnull(), cbar = False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f71f420d210>
```



encoding and labling

```
df_encoded = df.copy()
```

encoding 'last_review'

this column will be encoded keeping its timing orders and 'no reviews' will be 0

```
seasons = ['Winter' , 'Spring' , 'Summer' , 'Autumn' ]
code = 1
for year in range(2011 ,2020):
  for season in seasons :
    df_encoded['last_review'].replace(str(season)+'-'+str(year) , code , inplace=True)
    code += 1

df_encoded['last_review'].replace('no reviews' , 0 , inplace = True)


df_encoded['last_review'].value_counts()
```

```
35    17321
0     10037
34     5972
32     1909
33     1908
29     1641
31     1574
28      934
30      923
27      919
23      894
25      813
24      702
21      612
20      584
26      537
22      495
19      395
17      286
18      123
16       77
15       63
14       31
13       28
12       18
10       12
8        11
9         9
11        9
7         7
5         6
2         3
4         2
1         2
6         1
Name: last_review, dtype: int64
```

```
df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48858 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48858 non-null  int64  
 1   name                            48858 non-null  object 
 2   host_id                         48858 non-null  int64  
 3   host_name                       48858 non-null  object 
 4   neighbourhood_group             48858 non-null  object 
 5   neighbourhood                   48858 non-null  object 
 6   latitude                        48858 non-null  float64
 7   longitude                       48858 non-null  float64
 8   room_type                       48858 non-null  object 
 9   price                           48858 non-null  int64  
 10  minimum_nights                  48858 non-null  int64  
 11  number_of_reviews               48858 non-null  int64  
 12  last_review                     48858 non-null  int64  
 13  reviews_per_month               48858 non-null  float64
 14  calculated_host_listings_count  48858 non-null  int64  
 15  availability_365                48858 non-null  int64  
dtypes: float64(3), int64(8), object(5)
memory usage: 6.3+ MB
```

encoding 'room_type'

```
df_encoded['room_type'] = df_encoded['room_type'].astype('category').cat.codes
print(f'\nbefore:\n')
print(df['room_type'].value_counts())
print(f'\nafter:\n')
print(df_encoded['room_type'].value_counts())
```

```
before:

Entire home/apt    25393
Private room       22306
Shared room         1159
Name: room_type, dtype: int64

after:

0    25393
1    22306
```

```
2     1159
Name: room_type, dtype: int64
```

```
df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48858 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48858 non-null  int64  
 1   name                            48858 non-null  object 
 2   host_id                         48858 non-null  int64  
 3   host_name                       48858 non-null  object 
 4   neighbourhood_group             48858 non-null  object 
 5   neighbourhood                   48858 non-null  object 
 6   latitude                        48858 non-null  float64
 7   longitude                       48858 non-null  float64
 8   room_type                       48858 non-null  int8   
 9   price                           48858 non-null  int64  
 10  minimum_nights                  48858 non-null  int64  
 11  number_of_reviews               48858 non-null  int64  
 12  last_review                     48858 non-null  int64  
 13  reviews_per_month               48858 non-null  float64
 14  calculated_host_listings_count  48858 non-null  int64  
 15  availability_365                48858 non-null  int64  
dtypes: float64(3), int64(8), int8(1), object(4)
memory usage: 6.0+ MB
```

one-hot encoding 'neighbourhood_group'

```
df['neighbourhood_group'].value_counts()
```

```
Manhattan        21643
Brooklyn         20089
Queens            5664
Bronx             1089
Staten Island      373
Name: neighbourhood_group, dtype: int64
```

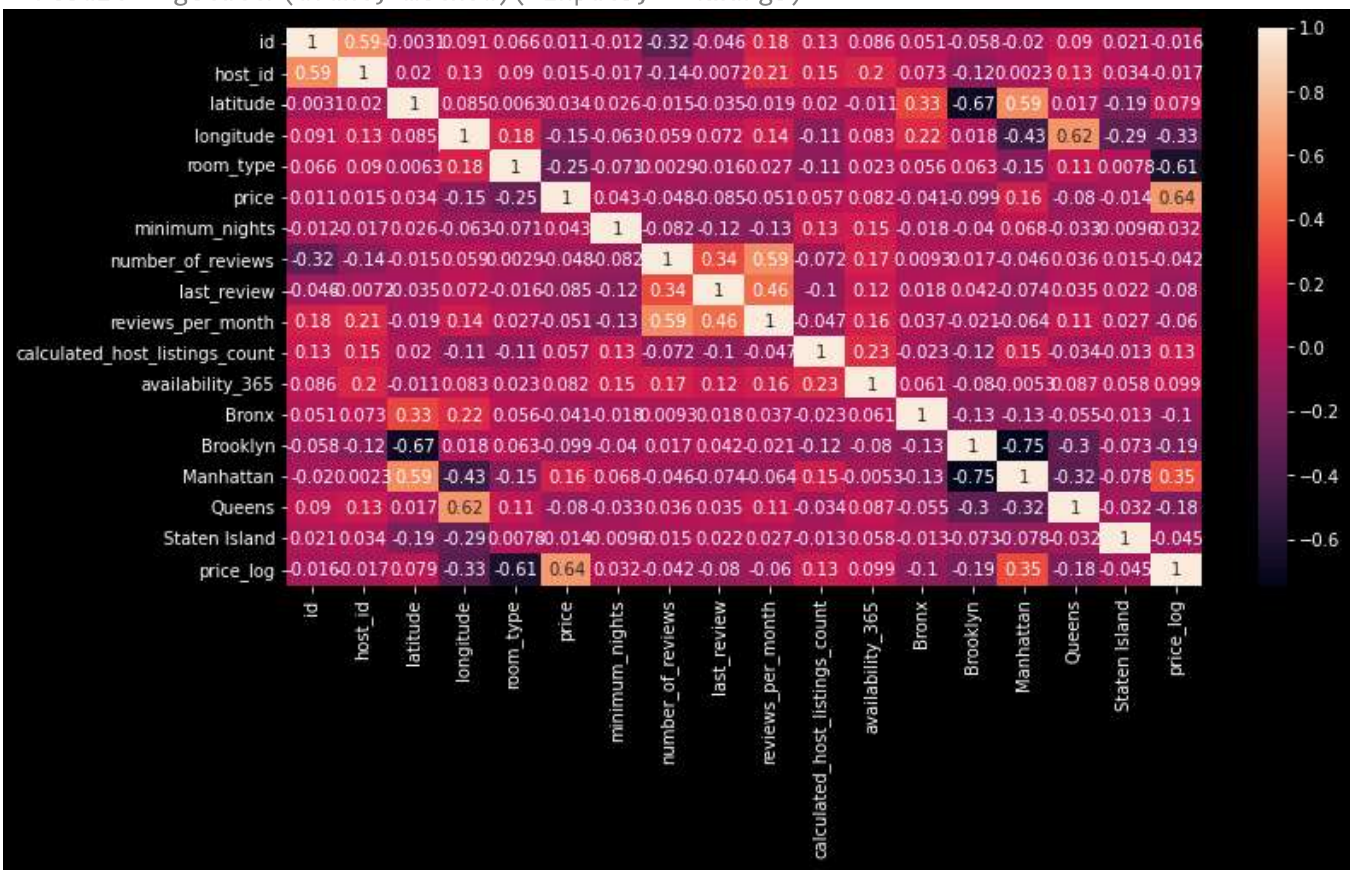
```
oneHotEncoder = pd.get_dummies(df_encoded['neighbourhood_group'])
```

```
df_encoded = df_encoded.join(oneHotEncoder)
```

EDA

```
# df_encoded.drop(['id' , 'host_id'] ,axis=1 , inplace=True)
df_encoded['price_log'] = np.log(df_encoded['price'])
plt.figure(figsize=(12, 6))
plt.style.use('dark_background')
sns.heatmap( df_encoded.corr() , annot = True )
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:726: RuntimeWarning: divide
  result = getattr(ufunc, method)(*inputs, **kwargs)
```




```
df_encoded.corr()
```

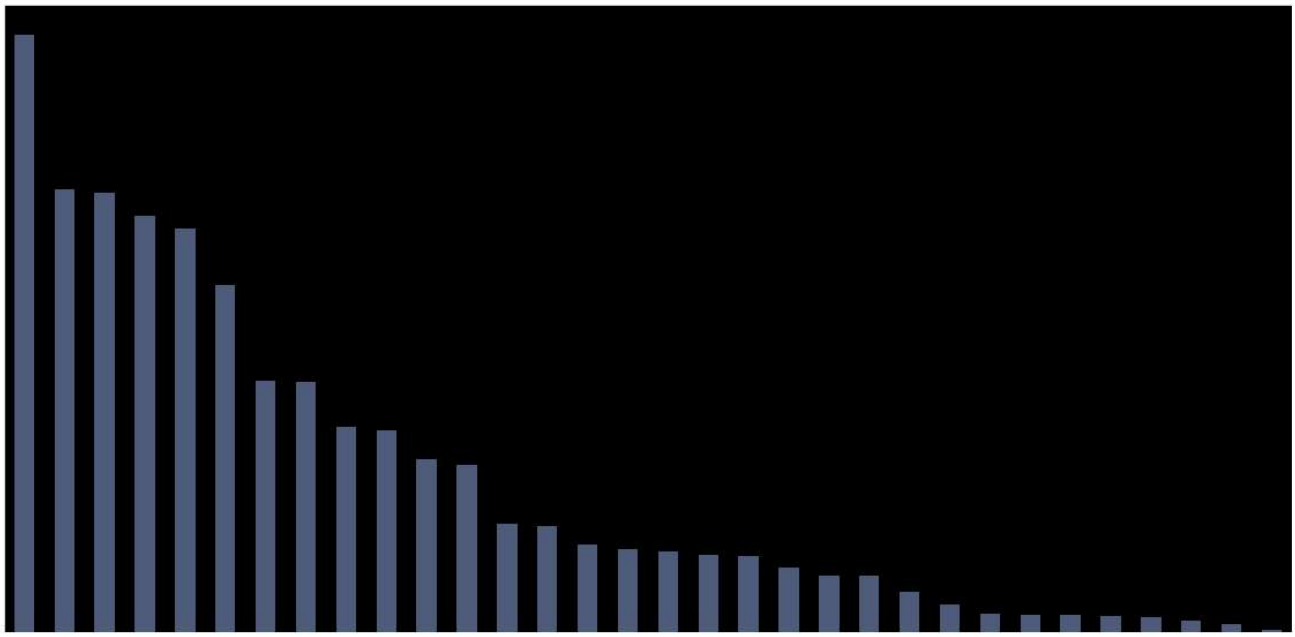
	id	host_id	latitude	longitude	room_type	pr
id	1.000000	0.588221	-0.003116	0.091076	0.066096	0.010
host_id	0.588221	1.000000	0.020193	0.127198	0.090372	0.015
latitude	-0.003116	0.020193	1.000000	0.084819	0.006270	0.033
longitude	0.091076	0.127198	0.084819	1.000000	0.184216	-0.149
room_type	0.066096	0.090372	0.006270	0.184216	1.000000	-0.249
price	0.010564	0.015328	0.033944	-0.149954	-0.249284	1.000
minimum_nights	-0.012038	-0.017027	0.025893	-0.062893	-0.070546	0.042
number_of_reviews	-0.320020	-0.140273	-0.015198	0.059151	0.002864	-0.047
last_review	-0.046102	-0.007194	-0.034839	0.071649	-0.015949	-0.085
reviews_per_month	0.180095	0.209644	-0.018702	0.138759	0.026637	-0.050
calculated_host_listings_count	0.133224	0.154954	0.019548	-0.114746	-0.106074	0.057
availability_365	0.085616	0.203743	-0.010775	0.082754	0.022628	0.081
Bronx	0.051238	0.073001	0.330517	0.221305	0.056223	-0.041
Brooklyn	-0.057816	-0.115910	-0.672890	0.017626	0.062873	-0.098
Queens	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

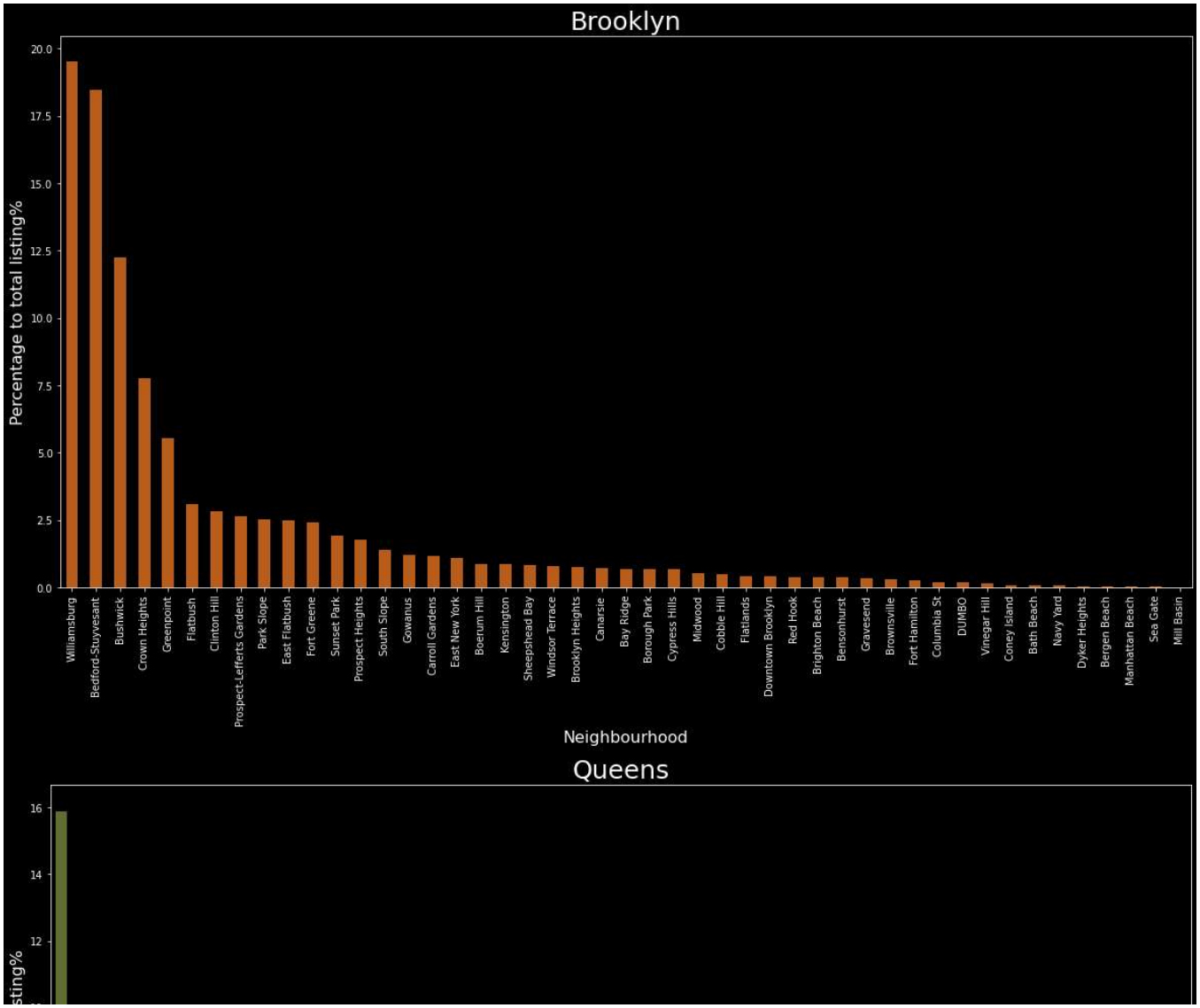
map_boroughs.gif

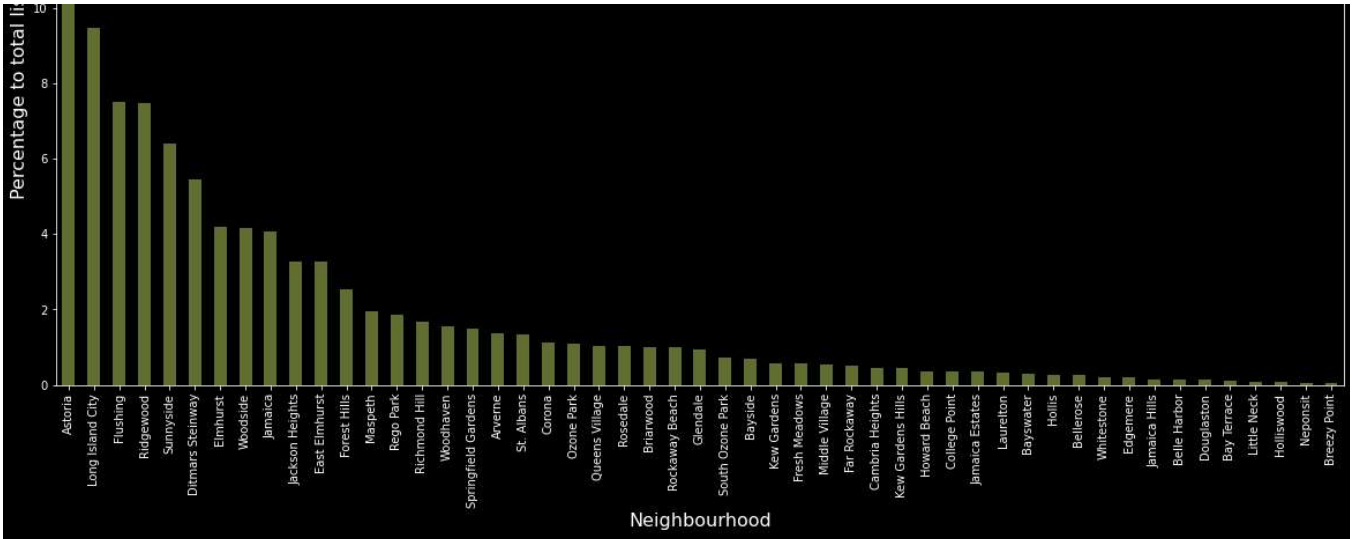
```
print(df['neighbourhood_group'].value_counts())
round(df['neighbourhood_group'].value_counts()/df.shape[0]*100,2).plot.pie(autopct='%1.2f%%',
```

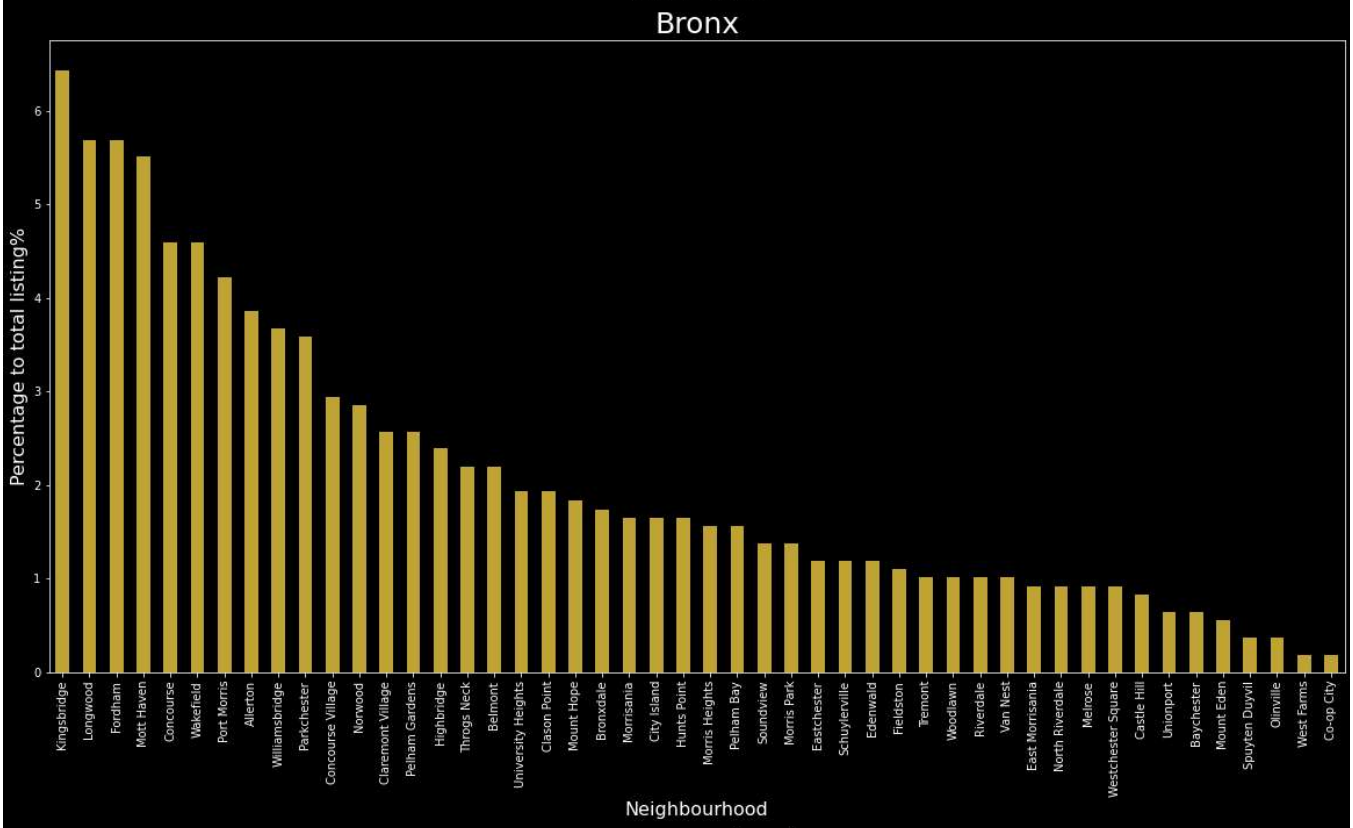
```
Manhattan        21643
Brooklyn         20089
Queens            5664
Bronx             1089
Staten Island      373
Name: neighbourhood_group, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f71ff469890>
```

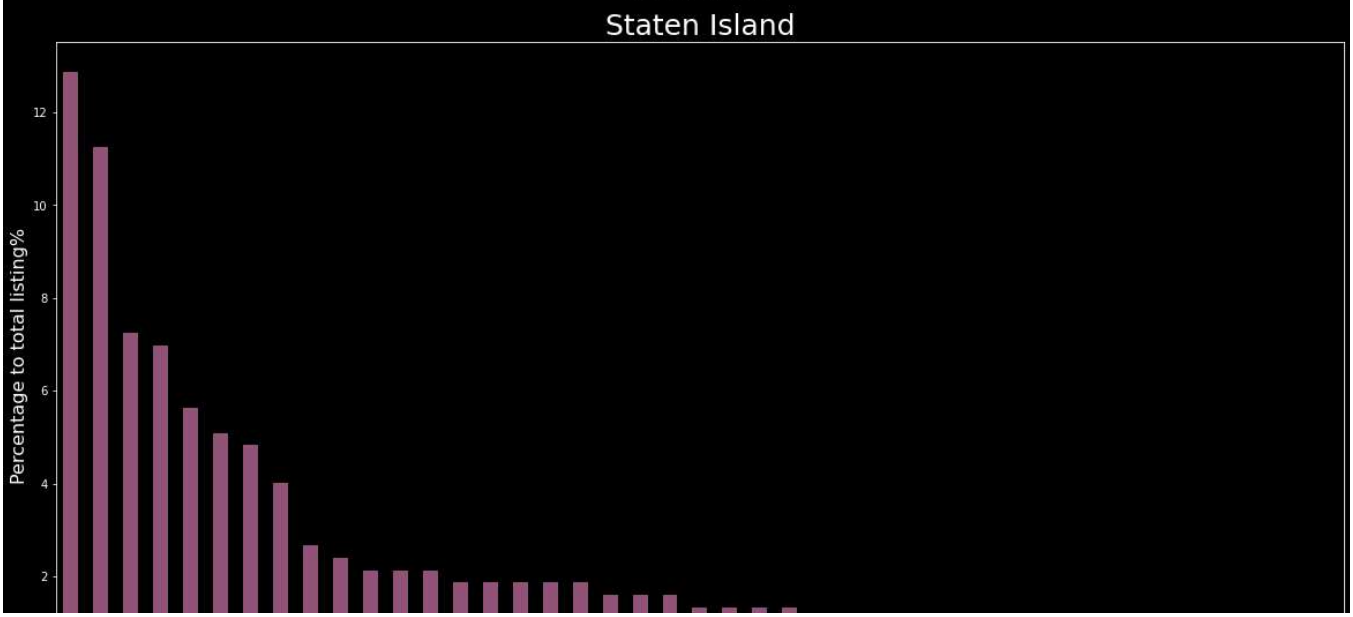
```
plt.figure(figsize=(20,10),facecolor='White')
plt.title('Manhattan',fontsize=25)
plt.xlabel('Neighbourhood', fontsize=16)
plt.ylabel('Percentage to total listing%', fontsize=16)
n1 = df[df['neighbourhood_group']=='Manhattan']
round(n1['neighbourhood'].value_counts()/n1.shape[0]*100,2).plot.bar(color='#4e5b7a')
plt.show()

plt.figure(figsize=(20,10))
plt.title('Brooklyn',fontsize=25)
plt.xlabel('Neighbourhood', fontsize=16)
plt.ylabel('Percentage to total listing%', fontsize=16)
n2 = df[df['neighbourhood_group']=='Brooklyn']
round(n2['neighbourhood'].value_counts()/n2.shape[0]*100,2).plot.bar(color='#b85d1c')
plt.show()

plt.figure(figsize=(20,10))
plt.title('Queens',fontsize=25)
plt.xlabel('Neighbourhood', fontsize=16)
plt.ylabel('Percentage to total listing%', fontsize=16)
n3 = df[df['neighbourhood_group']=='Queens']
round(n3['neighbourhood'].value_counts()/n3.shape[0]*100,2).plot.bar(color='#606e31')
plt.show()

plt.figure(figsize=(20,10))
plt.title('Bronx',fontsize=25)
plt.xlabel('Neighbourhood', fontsize=16)
plt.ylabel('Percentage to total listing%', fontsize=16)
n4 = df[df['neighbourhood_group']=='Bronx']
round(n4['neighbourhood'].value_counts()/n4.shape[0]*100,2).plot.bar(color='#bfa334')
plt.show()

plt.figure(figsize=(20,10))
plt.title('Staten Island',fontsize=25)
plt.xlabel('Neighbourhood', fontsize=16)
plt.ylabel('Percentage to total listing%', fontsize=16)
n5 = df[df['neighbourhood_group']=='Staten Island']
round(n5['neighbourhood'].value_counts()/n5.shape[0]*100,2).plot.bar(color='#915378')
plt.show()
```

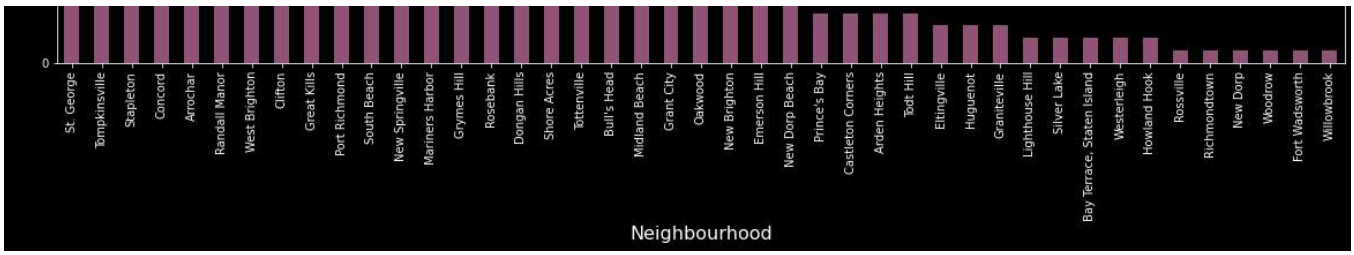



Brooklyn
Percentage to total listing%
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Williamsburg
Bedford-Stuyvesant
Bushwick
Crown Heights
Greenpoint
Flatbush
Clinton Hill
Prospect-Lefferts Gardens
Park Slope
East Flatbush
Fort Greene
Sunset Park
Prospect Heights
South Slope
Gowanus
Carroll Gardens
East New York
Boerum Hill
Kensington
Sheepshead Bay
Windsor Terrace
Brooklyn Heights
Canarsie
Bay Ridge
Borough Park
Cypress Hills
Midwood
Cobble Hill
Flatlands
Downtown Brooklyn
Red Hook
Brighton Beach
Bensonhurst
Gravesend
Brownsville
Fort Hamilton
Columbia St
DUMBO
Vinegar Hill
Coney Island
Bath Beach
Navy Yard
Dyker Heights
Bergen Beach
Manhattan Beach
Sea Gate
Mill Basin
Neighbourhood
Queens
16
14
12
sting%


Percentage to total li
Neighbourhood
Astoria
Long Island City
Flushing
Ridgewood
Sunnyside
Ditmars Steinway
Elmhurst
Woodside
Jamaica
Jackson Heights
East Elmhurst
Forest Hills
Maspeth
Rego Park
Richmond Hill
Woodhaven
Springfield Gardens
Arverne
St. Albans
Corona
Ozone Park
Queens Village
Rosedale
Briarwood
Rockaway Beach
Glendale
South Ozone Park
Bayside
Kew Gardens
Fresh Meadows
Middle Village
Far Rockaway
Cambria Heights
Kew Gardens Hills
Howard Beach
College Point
Jamaica Estates
Laurelton
Bayswater
Hollis
Bellerose
Whitestone
Edgemere
Jamaica Hills
Belle Harbor
Douglaston
Bay Terrace
Little Neck
Holliswood
Neponsit
Breezy Point


Bronx
Percentage to total listing%
Neighbourhood
Kingsbridge
Longwood
Fordham
Mott Haven
Concourse
Wakefield
Port Morris
Allerton
Williamsbridge
Parkchester
Concourse Village
Norwood
Claremont Village
Pelham Gardens
Highbridge
Throgs Neck
Belmont
University Heights
Clason Point
Mount Hope
Bronxdale
Morrisania
City Island
Hunts Point
Morris Heights
Pelham Bay
Soundview
Morris Park
Eastchester
Schuylerville
Edenwald
Fieldston
Tremont
Woodlawn
Riverdale
Van Nest
East Morrisania
North Riverdale
Melrose
Westchester Square
Castle Hill
Unionport
Baychester
Mount Eden
Spuyten Duyvil
Olinville
West Farms
Co-op City

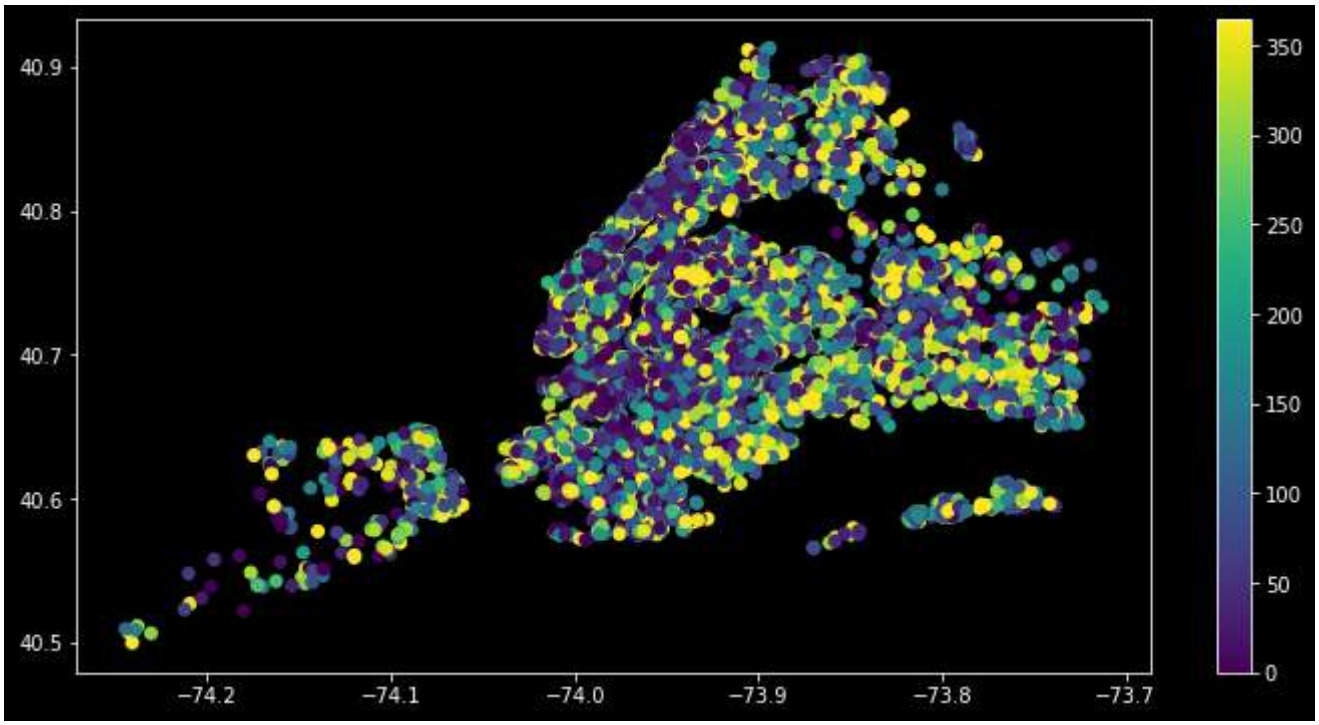

Staten Island
Percentage to total listing%


0
St. George
Tompkinsville
Stapleton
Concord
Arrochar
Randall Manor
West Brighton
Clifton
Great Kills
Port Richmond
South Beach
New Springville
Mariners Harbor
Grymes Hill
Rosebank
Dongan Hills
Shore Acres
Tottenville
Bull's Head
Midland Beach
Grant City
Oakwood
New Brighton
Emerson Hill
New Dorp Beach
Prince's Bay
Castleton Corners
Arden Heights
Todt Hill
Eltingville
Huguenot
Graniteville
Lighthouse Hill
Silver Lake
Bay Terrace, Staten Island
Westerleigh
Howland Hook
Rossville
Richmondtown
New Dorp
Woodrow
Fort Wadsworth
Willowbrook
Neighbourhood

```
print(df['room_type'].value_counts())
round(df['room_type'].value_counts()/df.shape[0]*100,2).plot.pie(autopct='%1.2f%%',shadow = T
```

```
Entire home/apt    25393
Private room       22306
Shared room         1159
Name: room_type, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f720158c650>
```


```
plt.scatter(data = df , x="longitude" , y="latitude" , c =df["availability_365"]  )
plt.colorbar ()
plt.show()
```



ML

```
df_encoded['Bronx'] = df_encoded['Bronx'].astype('int64')
df_encoded['Brooklyn'] = df_encoded['Brooklyn'].astype('int64')
df_encoded['Manhattan'] = df_encoded['Manhattan'].astype('int64')
df_encoded['Queens'] = df_encoded['Queens'].astype('int64')
df_encoded['Staten Island'] = df_encoded['Staten Island'].astype('int64')
df_encoded['price_log'] = df_encoded['price_log'].astype('float32')
```

```
df_encoded.drop('number_of_reviews' ,axis=1 , inplace = True)
```

```
X_features = df_encoded.drop(['price_log' , 'price' , 'id' , 'name' , 'host_id' ,'host_name'
X_features
```

```
Index(['latitude', 'longitude', 'room_type', 'minimum_nights', 'last_review',
```

```
            'reviews_per_month', 'calculated_host_listings_count',
            'availability_365', 'Bronx', 'Brooklyn', 'Manhattan', 'Queens',
            'Staten Island'],
           dtype='object')
```

```
df_encoded['finite'] = [math.isfinite(x) for x in df_encoded['price_log']]
```

```
df_encoded.drop(df_encoded[df_encoded['finite'] == False].index , inplace = True)
```

```
X = df_encoded.drop(['price_log' , 'price' , 'id' , 'name' , 'host_id' ,'host_name' , 'neighb
y = df_encoded['price_log']
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.25 , random_state=13)
```

```
lr = LinearRegression()
lr.fit(X_train,y_train)
```

```
     LinearRegression()
```

```
lr.score(X_train,y_train)
```

```
     0.4944277311758841
```

```
lr.score(X_test , y_test)
```

```
     0.48567505265868205
```

```
# RFR_Max_Features = [x for x in range(8 , 12)]
# RFR_Max_Depth = [x for x in range(7 , 11)]
# RMSEs = []
# train_score = []
# test_score = []
# R2s = []
```

```
# for max_fraures  in RFR_Max_Features :
#   for max_depth in RFR_Max_Depth:
#     RFR = RandomForestRegressor(max_depth=max_depth , max_features=max_fraures)
#     RFR.fit(X_train, y_train )
#     predictions = RFR.predict(X_test)
#     MSE = mean_squared_error(y_test, predictions)
#     RMSE = math.sqrt(MSE)
#     RMSEs.append(RMSE)
#     train_score.append(RFR.score(X_train , y_train) )
#     test_score.append(RFR.score(X_test , y_test))
#     R2s.append(r2_score(y_test, predictions))
```

```
# train_score


# test_score


# R2s









# x=np.array(RFR_Max_Features)
# y=np.array(RFR_Max_Depth)
# z = train_score
# X,Y = np.meshgrid(x,y)

# Z = np.reshape(z, (len(x), len(y)))

# plt.pcolormesh(X,Y,Z)

# plt.show()


# X


# sns.heatmap()


# fig , ax  = plt.subplots(1 , 3 , figsize = (6,6))
# ax[0].plot(train_score)




RFR = RandomForestRegressor(max_depth=10 , max_features=7)


RFR.fit(X_train, y_train )

     RandomForestRegressor(max_depth=10, max_features=7)


predictions = RFR.predict(X_test)
MSE = mean_squared_error(y_test, predictions)
RMSE = math.sqrt(MSE)
print('MAE: ' + str(mean_absolute_error(y_test, predictions)))
```

```
print('MSE: ' + str(MSE))
print('RMSE: ' + str(RMSE))
print('Score: '+ str(r2_score(y_test, predictions)))
```

```
MAE: 0.32121283583868193
MSE: 0.20107471681453856
RMSE: 0.4484135555651039
Score: 0.58587210807116
```

```
print(RFR.score(X_train, y_train))
```

```
0.6701485853476357
```

```
print(RFR.score(X_test, y_test))
```

```
0.58587210807116
```

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error

data_dmatrix = xgb.DMatrix(data=X,label=y)

xg_reg = xgb.XGBRegressor(objective ='reg:linear')

xg_reg.fit(X_train,y_train)
preds = xg_reg.predict(X_test)
```

```
[17:15:44] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
```

```
rmse = np.sqrt(mean_squared_error(y_test, preds))
rmse
```

```
0.4546616
```

```
xg_reg.score(X_train , y_train)
```

```
0.5946375393751626
```

```
xg_reg.score(X_test , y_test)
```

```
0.5742510588778161
```