# jupyter-arquivo

February 1, 2025

**Importando Bibliotecas**

[1]:
```
# manipulação de dados
import pandas as pd
import numpy as np

# gráficos
import seaborn as sns
import matplotlib.pyplot as plt

# modelo de prvisão
import plotly.express as px
from scipy.stats import kruskal
```

## 0.1 Tratamento preliminar de dados

**Carrgando Dados**

[2]:
```
df_main = pd.read_csv("teste_indicium_precificacao.csv")
```

[3]:
```
df_main.head(2)
```

[3]:
```
     id                                  nome  host_id  host_name bairro_group  \
0  2595                Skylit Midtown Castle     2845   Jennifer    Manhattan
1  3647  THE VILLAGE OF HARLEM…NEW YORK !     4632  Elisabeth    Manhattan

    bairro  latitude  longitude        room_type  price  minimo_noites  \
0  Midtown  40.75362  -73.98377  Entire home/apt    225              1
1   Harlem  40.80902  -73.94190     Private room    150              3

   numero_de_reviews ultima_review  reviews_por_mes  \
0                 45    2019-05-21             0.38
1                  0           NaN              NaN

   calculado_host_listings_count  disponibilidade_365
0                              2                  355
1                              1                  365
```

[4]:
```
df_main.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48894 entries, 0 to 48893
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48894 non-null  int64
 1   nome                            48878 non-null  object
 2   host_id                         48894 non-null  int64
 3   host_name                       48873 non-null  object
 4   bairro_group                    48894 non-null  object
 5   bairro                          48894 non-null  object
 6   latitude                        48894 non-null  float64
 7   longitude                       48894 non-null  float64
 8   room_type                       48894 non-null  object
 9   price                           48894 non-null  int64
 10  minimo_noites                   48894 non-null  int64
 11  numero_de_reviews               48894 non-null  int64
 12  ultima_review                   38842 non-null  object
 13  reviews_por_mes                 38842 non-null  float64
 14  calculado_host_listings_count   48894 non-null  int64
 15  disponibilidade_365             48894 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

### Lidando com Valores Ausentes

[5]:
```
df_main.isnull().sum()[df_main.isnull().sum() > 0]/df_main.shape[0] * 100
```

[5]:
```
nome                 0.032724
host_name            0.042950
ultima_review       20.558760
reviews_por_mes     20.558760
dtype: float64
```

As colunas nome, host_name, ultima_review e reviews_por_mes possuem valores ausentes, a abordagem para cada coluna será a seguinte:

nome: substituir por unknown (são poucos valores)

host_name: substituir por unknown

ultima_review: remover, pois corresponde a 20% dos dados

reviews_por_mes: remover, pois corresponde a 20% dos dados

[6]:
```
df_main.drop(['ultima_review', 'reviews_por_mes'], inplace=True, axis=1)
```

[7]:
```
df_main['nome'].fillna('unknown', inplace=True)
df_main['host_name'].fillna('unknown', inplace=True)
```

### Lidando com Outliers

[8]:
```
df_main.describe()
```

[8]:

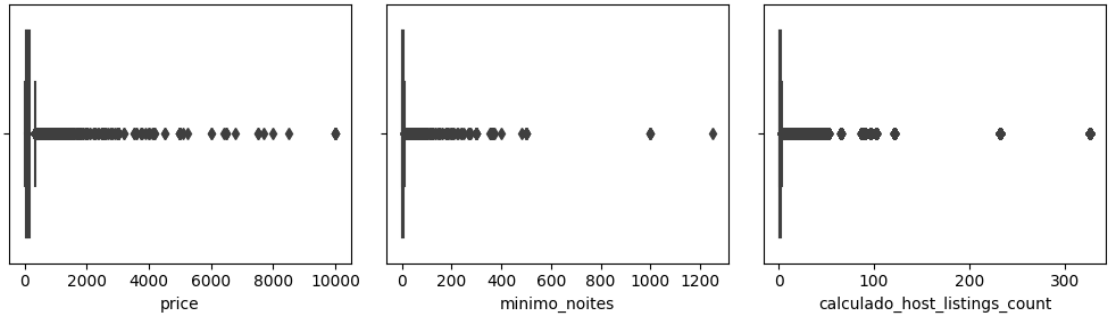
| | id | host_id | latitude | longitude | price |
|---|---|---|---|---|---|
| count | 4.889400e+04 | 4.889400e+04 | 48894.000000 | 48894.000000 | 48894.000000 |
| mean | 1.901753e+07 | 6.762139e+07 | 40.728951 | -73.952169 | 152.720763 |
| std | 1.098288e+07 | 7.861118e+07 | 0.054529 | 0.046157 | 240.156625 |
| min | 2.595000e+03 | 2.438000e+03 | 40.499790 | -74.244420 | 0.000000 |
| 25% | 9.472371e+06 | 7.822737e+06 | 40.690100 | -73.983070 | 69.000000 |
| 50% | 1.967743e+07 | 3.079553e+07 | 40.723075 | -73.955680 | 106.000000 |
| 75% | 2.915225e+07 | 1.074344e+08 | 40.763117 | -73.936273 | 175.000000 |
| max | 3.648724e+07 | 2.743213e+08 | 40.913060 | -73.712990 | 10000.000000 |

| | minimo_noites | numero_de_reviews | calculado_host_listings_count |
|---|---|---|---|
| count | 48894.000000 | 48894.000000 | 48894.000000 |
| mean | 7.030085 | 23.274758 | 7.144005 |
| std | 20.510741 | 44.550991 | 32.952855 |
| min | 1.000000 | 0.000000 | 1.000000 |
| 25% | 1.000000 | 1.000000 | 1.000000 |
| 50% | 3.000000 | 5.000000 | 1.000000 |
| 75% | 5.000000 | 24.000000 | 2.000000 |
| max | 1250.000000 | 629.000000 | 327.000000 |

| | disponibilidade_365 |
|---|---|
| count | 48894.000000 |
| mean | 112.776169 |
| std | 131.618692 |
| min | 0.000000 |
| 25% | 0.000000 |
| 50% | 45.000000 |
| 75% | 227.000000 |
| max | 365.000000 |

[9]:
```
max_linhas = 1
max_colunas = 3
linha = 0
coluna = 0

fig, ax = plt.subplots(max_linhas, max_colunas , figsize=(10, 3))

for coluna_n in ['price', 'minimo_noites', 'calculado_host_listings_count']:

    sns.boxplot(data=df_main, x=coluna_n, ax=ax[coluna]) # usar [linha, coluna]␣
 ↪quando tiver mais de 1 linha

    coluna += 1
```

```
    if coluna == max_colunas:
        linha += 1
        coluna = 0

plt.tight_layout()
plt.show()
```




Observando os conjunto de dados, é possível observar alguns outliers:

price: contem valores que vão até 10000

minimo_noites; contem valores que vão até 1250

calculado_host_listings_count: contem valores que vão até 327

embora esses dados possam ser reais, eles difcultam a análise, por isso serão retirados usando o **método do intervalo interquartil** . (posteriormente, caso necessário, lido com eles)

Antes de tratar os outliers, é possível observar que os outlier de price são porvenientes de uma determinado região de Manhattan, que é uma cidade cara e por isso o aluguél deve ser mais caro, observe o gráfico abaixo (pontos verde)

[10]:
```
Q1 = df_main['price'].quantile(0.25)
Q3 = df_main['price'].quantile(0.75)
IQR = Q3 - Q1

price_limite_superior = Q3 + (IQR * 1.5)

print(f'limite superior de price: {price_limite_superior}')

# este limite superior esta sendo definido para analisar os dados de forma mais␣
  ↪fácil
```

```
limite superior de price: 334.0
```

[11]:
```
# Localização de aluguéis sem outliers
```

```
fig = px.scatter_mapbox(df_main[df_main['price'] < price_limite_superior],␣
 ↪lat='latitude', lon='longitude', color="price")
fig.update_layout(mapbox_style="open-street-map")
fig.show()
```

[12]:
```
# Localização de aluguéis com outliers, aqui está a região que apresenta os␣
 ↪maiores valores price

fig = px.scatter_mapbox(df_main[df_main['price'] > price_limite_superior],␣
 ↪lat='latitude', lon='longitude', color="price")
fig.update_layout(mapbox_style="open-street-map")
fig.show()
```

Removendo outliers:

[13]:
```
# Só clique Run 1 vez, caso contrário altera os dados

limites = []

for coluna_n in ['price', 'minimo_noites', 'calculado_host_listings_count']:
    Q1 = df_main[coluna_n].quantile(0.25)
    Q3 = df_main[coluna_n].quantile(0.75)

    IQR = Q3 - Q1

    limite_inferior = Q1 - (IQR * 1.5)
    limite_superior = Q3 + (IQR * 1.5)



    dic = {'coluna_n': coluna_n, 'limite_inferior': limite_inferior,␣
 ↪'limite_superior': limite_superior, 'quantidade acima do limite superior':␣
 ↪df_main[df_main[coluna_n] > limite_superior][coluna_n].count()}
    limites.append(dic)

    df_main = df_main[df_main[coluna_n] < limite_superior]

pd.DataFrame(limites)
```

[13]:

| | coluna_n | limite_inferior | limite_superior | quantidade acima do limite superior |
|---|---|---|---|---|
| 0 | price | -90.0 | 334.0 | 2972 |
| 1 | minimo_noites | -5.0 | 11.0 | 6181 |
| 2 | calculado_host_listings_count | -0.5 | 3.5 | |

2 3755

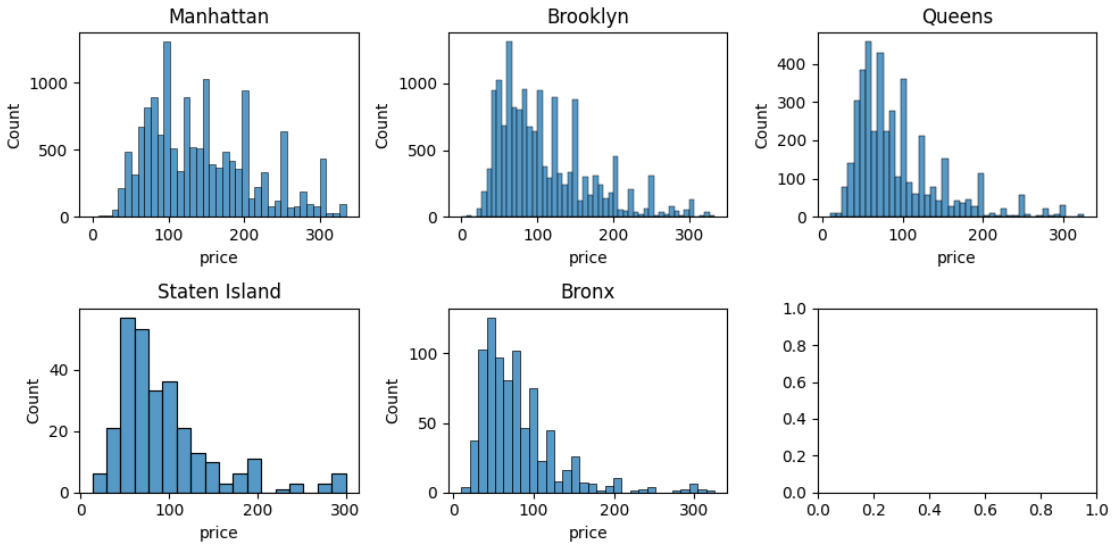
## 0.2 EDA

### Avaliando preço por bairro_group

[14]:
```
df_main['bairro_group'].value_counts()
```

[14]:
```
Brooklyn         15894
Manhattan        14769
Queens            4168
Bronx              837
Staten Island      283
Name: bairro_group, dtype: int64
```

[15]:
```
max_linhas = 2
max_colunas = 3
linha = 0
coluna = 0

coluna_de_analise = 'price'
coluna_de_segmentacao = 'bairro_group'

fig, ax = plt.subplots(max_linhas, max_colunas , figsize=(10, 5))

for coluna_n in df_main[coluna_de_segmentacao].unique():

    sns.histplot(data=df_main[df_main[coluna_de_segmentacao] == coluna_n],␣
 ↪x=coluna_de_analise, ax=ax[linha, coluna]) # usar [linha, coluna] quando␣
 ↪tiver mais de 1 linha
    ax[linha, coluna].set_title(coluna_n)

    coluna += 1

    if coluna == max_colunas:
        linha += 1
        coluna = 0

plt.tight_layout()
plt.show()
```




**Manhattan e Brooklyn** são os bairros com maior quantidade de imóveis/quartos para alugar, além disso apresentam maior diversidade de preço, possuindo mais imóveis/quartos de aluguél mais caros se comparado ao Queens, Staten Island e Bronx, Isso pode demosntrar que **Manhattan e Brookly** podem ser mais receptíveis a aluguéis mais caros se comparado aos outros bairros.

**preço por room_type** A maioria dos aluguéis são Entire home/apt e Private room

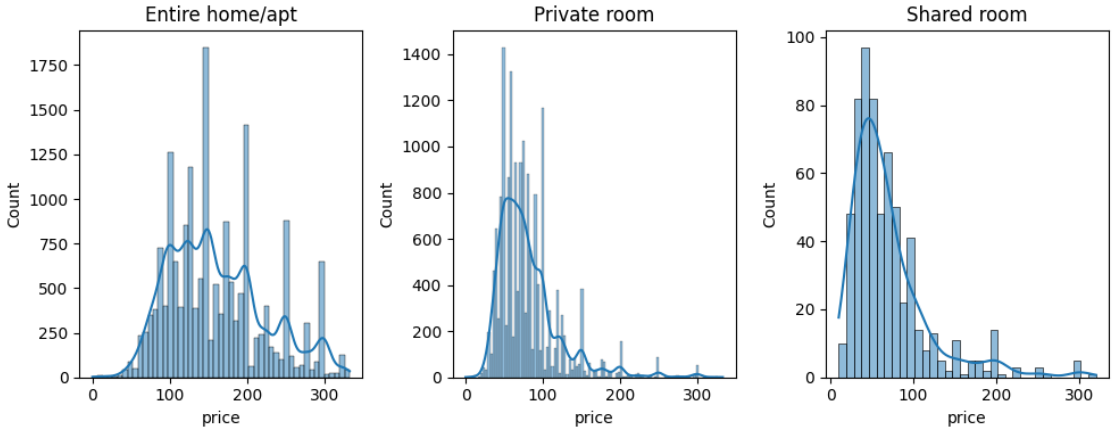
os **Entire home/apt** possuem boa diversificação de preço, porém quando se **trata de Private room e Shared room** observa-se que esses são a minoria quando o valor tende a ser maior que 200, isso pode dificultar a inclusão de aluguéis mais caros para esses tipos de quarto devido a concorrência dos aluguéis mais baratos

[16]:
```
df_main['room_type'].value_counts()
```

[16]:
```
Entire home/apt    18143
Private room       17168
Shared room          640
Name: room_type, dtype: int64
```

[17]:
```
max_linhas = 1
max_colunas = 3
linha = 0
coluna = 0

coluna_de_analise = 'price'
coluna_de_segmentacao = 'room_type'

fig, ax = plt.subplots(max_linhas, max_colunas , figsize=(10, 4))
```

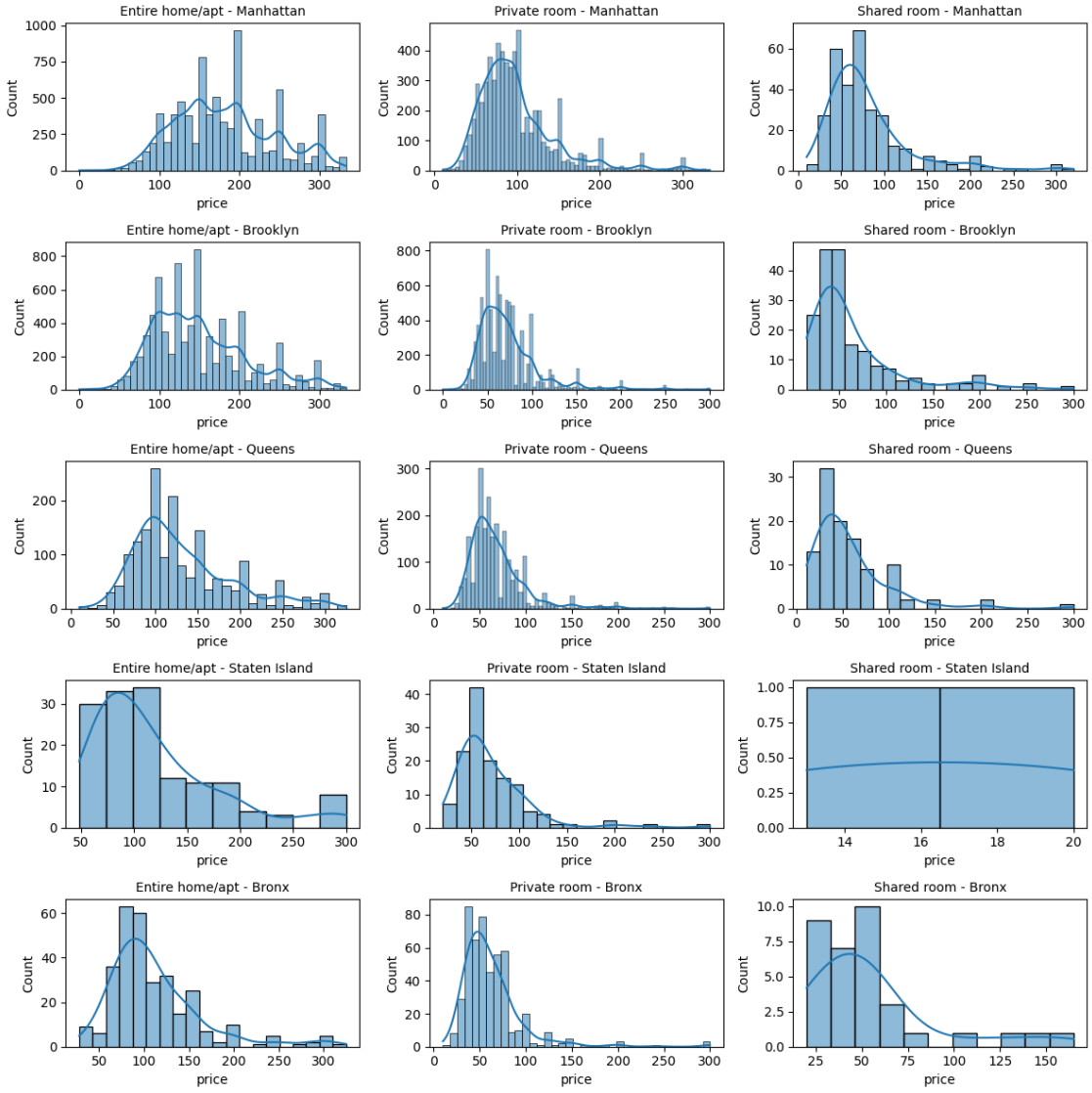
```
for coluna_n in df_main[coluna_de_segmentacao].unique():

    sns.histplot(data=df_main[df_main[coluna_de_segmentacao] == coluna_n],␣
 ↪x=coluna_de_analise, ax=ax[coluna], kde=True) # usar [linha, coluna] quando␣
 ↪tiver mais de 1 linha
    ax[coluna].set_title(coluna_n)

    coluna += 1

    if coluna == max_colunas:
        linha += 1
        coluna = 0

plt.tight_layout()
plt.show()
```




Além disso, quando observamos o preço dos aluguéis por **room_type** e **bairro_group**, observamos que Manhattan tem os aluguéis mais caros, independente do tipo de imóvel/quarto.

[18]:
```
linhas = 5
colunas = 3
coluna_de_plotagem = 0
coluna_de_análise = 'room_type'

fig, ax = plt.subplots(linhas, colunas , figsize=(12, 12))

for bairro_name in enumerate(df_main['bairro_group'].unique()):

    for bairro in enumerate(df_main[coluna_de_análise].unique()):
```

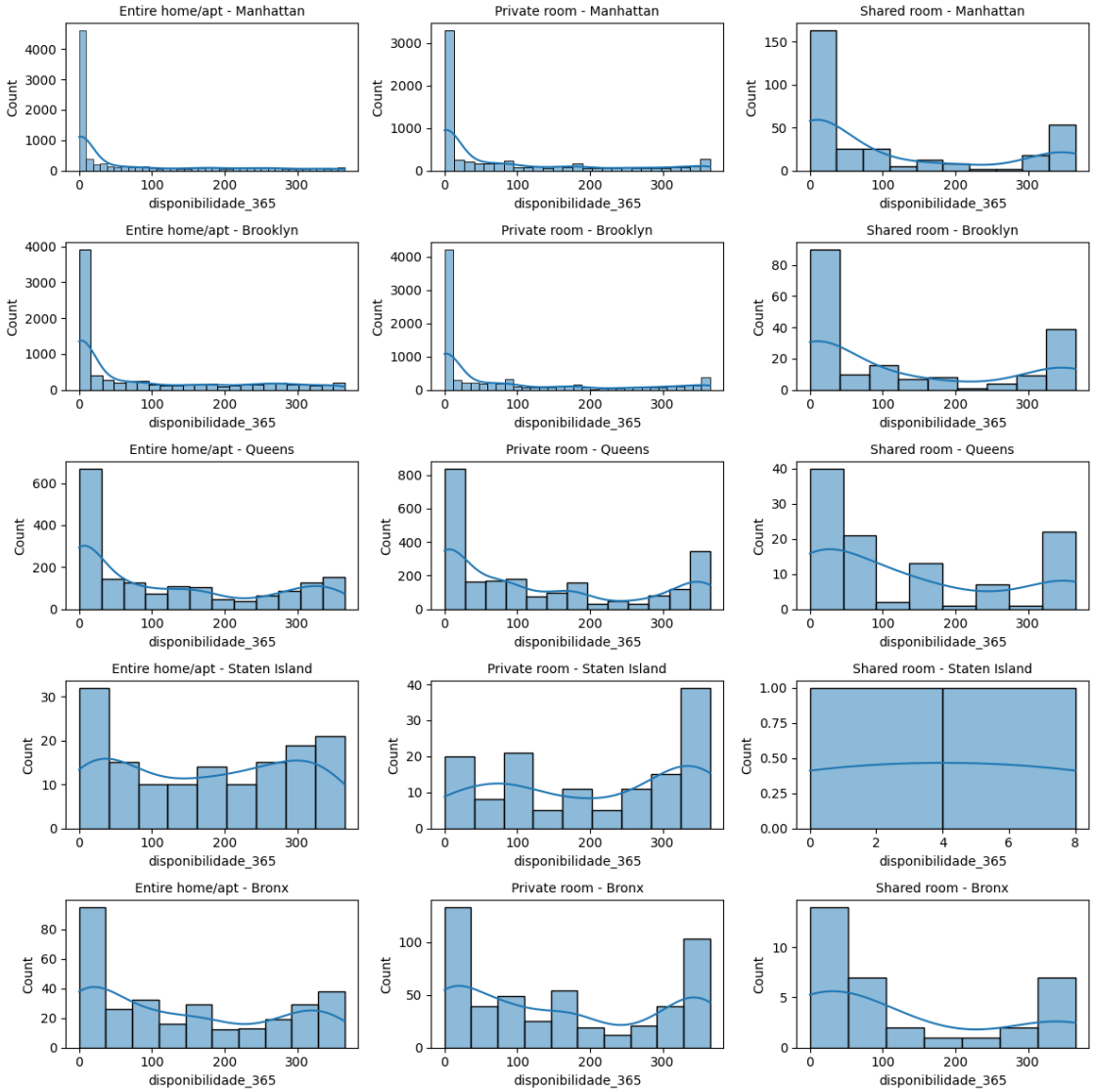
```
        sns.histplot(data=df_main[(df_main[coluna_de_análise] == bairro[1]) &␣
↪(df_main['bairro_group'] == bairro_name[1])]['price'], ax=ax[bairro_name[0],␣
↪coluna_de_plotagem], kde=True)
        ax[bairro_name[0], coluna_de_plotagem].set_title(bairro[1] + ' - ' +␣
↪bairro_name[1], fontsize=10)
        #mediana = df_main[(df_main[coluna_de_análise] == bairro[1])]['price'].
↪median()
        #ax[bairro_name[0], coluna_de_plotagem].axvline(mediana, color='red')

        coluna_de_plotagem += 1

        if coluna_de_plotagem == colunas:
            coluna_de_plotagem = 0


plt.tight_layout()
```




**Relação de Disponibilidade de Reserva com room_type e bairro_group** Avaliando quantos imóveis tem muita disponibilidade durante o ano, isso será feito para indicar locais e room_type que são pouco alugados

[19]:
```
linhas = 5
colunas = 3
coluna_de_plotagem = 0
coluna_de_análise = 'room_type'

fig, ax = plt.subplots(linhas, colunas , figsize=(12, 12))

for bairro_name in enumerate(df_main['bairro_group'].unique()):
```

```
    for bairro in enumerate(df_main[coluna_de_análise].unique()):

        sns.histplot(data=df_main[(df_main[coluna_de_análise] == bairro[1]) &␣
↪(df_main['bairro_group'] == bairro_name[1])]['disponibilidade_365'],␣
↪ax=ax[bairro_name[0], coluna_de_plotagem], kde=True)
        ax[bairro_name[0], coluna_de_plotagem].set_title(bairro[1] + ' - ' +␣
↪bairro_name[1], fontsize=10)
        #mediana = df_main[(df_main[coluna_de_análise] == bairro[1])]['price'].
↪median()
        #ax[bairro_name[0], coluna_de_plotagem].axvline(mediana, color='red')

        coluna_de_plotagem += 1

        if coluna_de_plotagem == colunas:
            coluna_de_plotagem = 0


plt.tight_layout()
```




porcentagem de imóveis disponíveis por mais de 100 dias e menos de 100

[20]:
```
threshold_de_disponibilidade = 100
```

[21]:
```
df_disponibilidade = pd.DataFrame(df_main.groupby(by=['bairro_group',␣
↪'room_type'])['disponibilidade_365'].agg(
    disponível_menos_de_100 = lambda x: (x < threshold_de_disponibilidade).
↪sum(),
    disponível_mais_de_100 = lambda x: (x >= threshold_de_disponibilidade).
↪sum()))
```

```
df_disponibilidade['Porcentagem_com_disponibilidade_acima_de_100'] =␣
 ↪(df_disponibilidade['disponível_mais_de_100']/
 ↪(df_disponibilidade['disponível_menos_de_100']+df_disponibilidade['disponível_mais_de_100']

df_disponibilidade
```

[21]:

| bairro_group | room_type | disponível_menos_de_100 | disponível_mais_de_100 | Porcentagem_com_disponibilidade_acima_de_100 |
|---|---|---|---|---|
| Bronx | Entire home/apt | 146 | 163 | 52.750809 |
| | Private room | 218 | 276 | 55.870445 |
| | Shared room | 21 | 13 | 38.235294 |
| Brooklyn | Entire home/apt | 5325 | 2414 | |
| | Private room | 5861 | 2110 | |
| | Shared room | 115 | 69 | |
| Manhattan | Entire home/apt | 6183 | 2038 | |
| | Private room | 4540 | 1695 | |
| | Shared room | 213 | 100 | |
| Queens | Entire home/apt | 953 | 775 | |
| | Private room | 1326 | 1007 | |
| | Shared room | 61 | 46 | |
| Staten Island | Entire home/apt | 54 | 92 | |
| | Private room | 47 | 88 | |
| | Shared room | 2 | 0 | |

```
Brooklyn      Entire home/apt                 31.192661
              Private room                    26.470957
              Shared room                     37.500000
Manhattan     Entire home/apt                 24.790172
              Private room                    27.185245
              Shared room                     31.948882
Queens        Entire home/apt                 44.849537
              Private room                    43.163309
              Shared room                     42.990654
Staten Island Entire home/apt                 63.013699
              Private room                    65.185185
              Shared room                      0.000000
```

avaliando as disponibilidade, é possível o bservar que Manhattan e Brooklyn tem os imóveis/quartos com a menor porcentagem de mais de 100 dias disponíveis, isso indica que comparado aos outros imóveis/quartos eles possuem uma recorência maior de pessoas alugando

**Quantidade de imóveis/quartos por room_type e bairro_group**

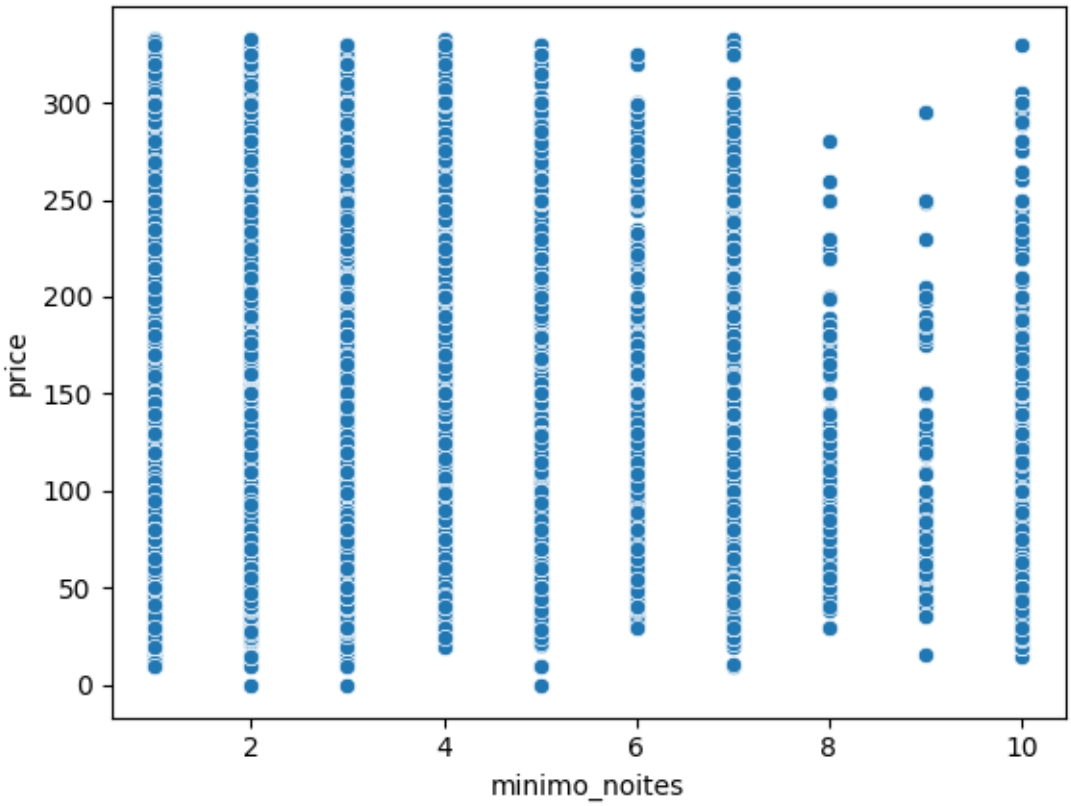
[22]:
```
pd.DataFrame(df_main.groupby(by=['bairro_group', 'room_type'])['room_type'].
↪count())
```

[22]:
```
                            room_type
bairro_group  room_type
Bronx         Entire home/apt        309
              Private room           494
              Shared room             34
Brooklyn      Entire home/apt       7739
              Private room          7971
              Shared room            184
Manhattan     Entire home/apt       8221
              Private room          6235
              Shared room            313
Queens        Entire home/apt       1728
              Private room          2333
              Shared room            107
Staten Island Entire home/apt        146
              Private room           135
              Shared room              2
```

**Relação entre minimo_noites e price** Não parece haver uma forte relação entre minimo_noites e price, o price tende a variar independente do número de noites

[23]:
```
sns.scatterplot(data=df_main, x='minimo_noites', y='price')
```

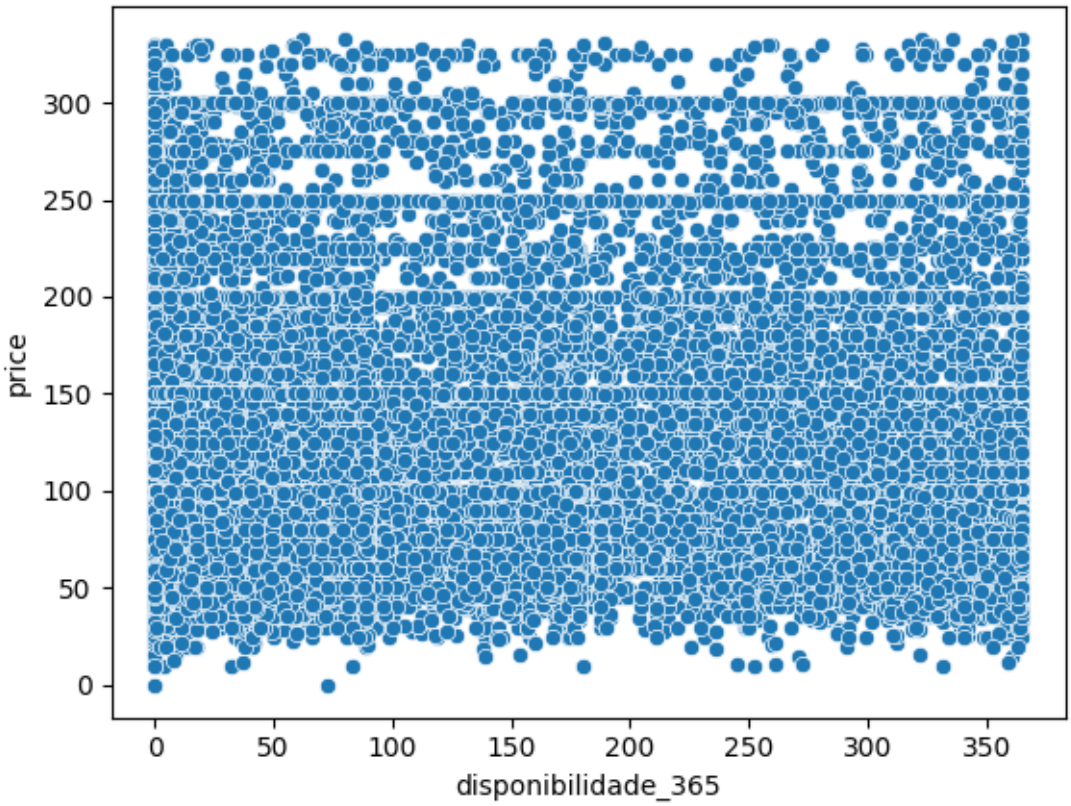
[23]:
```
<Axes: xlabel='minimo_noites', ylabel='price'>
```




**Relação entre disponibilidade_365 e price** Não parece haver uma forte relação

[24]:
```
sns.scatterplot(data=df_main, x='disponibilidade_365', y='price')
```

[24]:
```
<Axes: xlabel='disponibilidade_365', ylabel='price'>
```




### 0.2.1 Análise de Texto no anúncio

[25]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\pfesc\anaconda3\lib\site-
packages (3.7)
Requirement already satisfied: click in c:\users\pfesc\anaconda3\lib\site-
packages (from nltk) (8.0.4)
Requirement already satisfied: regex>=2021.8.3 in
c:\users\pfesc\anaconda3\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\users\pfesc\anaconda3\lib\site-
packages (from nltk) (4.64.1)
Requirement already satisfied: joblib in c:\users\pfesc\anaconda3\lib\site-
packages (from nltk) (1.1.0)
Requirement already satisfied: colorama in c:\users\pfesc\anaconda3\lib\site-
packages (from click->nltk) (0.4.5)
```

**Importando Bibliotecas**

[26]:
```
from collections import Counter
import nltk
```

```
from nltk.util import ngrams
from nltk.corpus import stopwords
```

### Carregando Dados

[27]:
```
df_text_anuncio = pd.read_csv('teste_indicium_precificacao.csv')
```

### Tratando Dados

[28]:
```
manter = ['nome', 'price']

for coluna in df_text_anuncio.columns:
    if coluna not in manter:
        df_text_anuncio.drop(coluna, axis=1, inplace=True)
```

[29]:
```
df_text_anuncio['nome'].fillna('unknown', inplace=True)
```

### Criando Bigramas e Trigramas

Baixando a lista de palavras para ignorar e o tokenizador

[30]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\pfesc\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[30]:
```
True
```

[31]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pfesc\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[31]:
```
True
```

[32]:
```
def create_ngram(nome, n):
    palavras = nltk.word_tokenize(nome.lower()) # transforma o texto em uma␣
 ↪lista de palavras
    palavras_p_ignorar = set(stopwords.words('english')) # desconsiderando␣
 ↪palavras

    # após o primeiro round percebi que | ? * + estavam atrapalhando os␣
 ↪trigramas e bigramas por isso:
    palavras_p_ignorar.update({'?', '*', '+', '|', "!"})

    palavras = [palavra for palavra in palavras if palavra not in␣
 ↪palavras_p_ignorar] # removendo algumas palavras
```

```
    return list(ngrams(palavras, n)) #gera brigramas e trigramas que são␣
 ↪adiconados as lista

df_text_anuncio['bigramas'] = df_text_anuncio['nome'].apply(lambda nome:␣
 ↪create_ngram(nome, 2))
df_text_anuncio['trigrama'] = df_text_anuncio['nome'].apply(lambda nome:␣
 ↪create_ngram(nome, 3))
```

[33]:
```
df_text_anuncio
```

[33]:
```
                                                  nome  price  \
0                                Skylit Midtown Castle    225
1                  THE VILLAGE OF HARLEM…NEW YORK !    150
2                      Cozy Entire Floor of Brownstone     89
3       Entire Apt: Spacious Studio/Loft by central park     80
4              Large Cozy 1 BR Apartment In Midtown East    200
…                                                    …    …
48889    Charming one bedroom - newly renovated rowhouse     70
48890      Affordable room in Bushwick/East Williamsburg     40
48891            Sunny Studio at Historical Neighborhood    115
48892               43rd St. Time Square-cozy single bed     55
48893  Trendy duplex in the very heart of Hell's Kitchen     90

                                               bigramas  \
0                 [(skylit, midtown), (midtown, castle)]
1      [(village, harlem), (harlem, …), (…, new…
2      [(cozy, entire), (entire, floor), (floor, brow…
3      [(entire, apt), (apt, :), (:, spacious), (spac…
4      [(large, cozy), (cozy, 1), (1, br), (br, apart…
…                                                    …
48889  [(charming, one), (one, bedroom), (bedroom, -)…
48890  [(affordable, room), (room, bushwick/east), (b…
48891  [(sunny, studio), (studio, historical), (histo…
48892  [(43rd, st.), (st., time), (time, square-cozy)…
48893  [(trendy, duplex), (duplex, heart), (heart, he…

                                               trigrama
0                          [(skylit, midtown, castle)]
1      [(village, harlem, …), (harlem, …, new),…
2      [(cozy, entire, floor), (entire, floor, browns…
3      [(entire, apt, :), (apt, :, spacious), (:, spa…
4      [(large, cozy, 1), (cozy, 1, br), (1, br, apar…
…                                                    …
48889  [(charming, one, bedroom), (one, bedroom, -), …
48890  [(affordable, room, bushwick/east), (room, bus…
48891  [(sunny, studio, historical), (studio, histori…
48892  [(43rd, st., time), (st., time, square-cozy), …
```

```
48893  [(trendy, duplex, heart), (duplex, heart, hell…

[48894 rows x 4 columns]
```

Selecionando os anuncios com maiores preços

[34]:
```
limite_superior = df_text_anuncio['price'].quantile(0.75)

df_text_anuncio_mais_caro = df_text_anuncio[df_text_anuncio['price'] >␣
 ↪limite_superior]
```

[35]:
```
df_text_anuncio_mais_caro
```

[35]:
```
                                                   nome  price  \
0                                 Skylit Midtown Castle    225
4            Large Cozy 1 BR Apartment In Midtown East    200
15                   Perfect for Your Parents + Garden    215
18                   Huge 2 BR Upper East  Cental Park    190
19                    Sweet and Spacious Brooklyn Loft    299
…                                                   …    …
48852    Sunny&quiet paradise in the WV with open views    202
48855  Large 3 bed, 2 bath , garden , bbq , all you need    345
48871                                  Nycity-MyHome    260
48883       Brooklyn Oasis in the heart of Williamsburg    190
48885                    Comfy 1 Bedroom in Midtown East    200

                                                bigramas  \
0                [(skylit, midtown), (midtown, castle)]
4      [(large, cozy), (cozy, 1), (1, br), (br, apart…
15               [(perfect, parents), (parents, garden)]
18     [(huge, 2), (2, br), (br, upper), (upper, east…
19     [(sweet, spacious), (spacious, brooklyn), (bro…
…                                                   …
48852  [(sunny, &), (&, quiet), (quiet, paradise), (p…
48855  [(large, 3), (3, bed), (bed, ,), (,, 2), (2, b…
48871                                                 []
48883  [(brooklyn, oasis), (oasis, heart), (heart, wi…
48885  [(comfy, 1), (1, bedroom), (bedroom, midtown),…

                                                trigrama
0                          [(skylit, midtown, castle)]
4      [(large, cozy, 1), (cozy, 1, br), (1, br, apar…
15                        [(perfect, parents, garden)]
18     [(huge, 2, br), (2, br, upper), (br, upper, ea…
19     [(sweet, spacious, brooklyn), (spacious, brook…
…                                                   …
48852  [(sunny, &, quiet), (&, quiet, paradise), (qui…
```

```
48855  [(large, 3, bed), (3, bed, ,), (bed, ,, 2), (,…
48871                                                 []
48883  [(brooklyn, oasis, heart), (oasis, heart, will…
48885  [(comfy, 1, bedroom), (1, bedroom, midtown), (…

[12177 rows x 4 columns]
```

[36]:
```
bigramas = [bigrama for bigrama_list in df_text_anuncio_mais_caro['bigramas']␣
 ↪for bigrama in bigrama_list]
trigramas = [trigrama for trigrama_list in␣
 ↪df_text_anuncio_mais_caro['trigrama'] for trigrama in trigrama_list]
```

Contando palavras mais comuns

[37]:
```
bigramas_mais_comuns = Counter(bigramas).most_common(20)
trigramas_mais_comuns = Counter(trigramas).most_common(20)
```

[38]:
```
bigramas_mais_comuns
```

[38]:
```
[(('2', 'bedroom'), 501),
 (('1', 'bedroom'), 444),
 (('central', 'park'), 438),
 (('east', 'village'), 419),
 (('west', 'village'), 379),
 (('east', 'side'), 281),
 (('one', 'bedroom'), 266),
 (('times', 'square'), 256),
 (('bedroom', 'apartment'), 252),
 (('3', 'bedroom'), 206),
 (('bedroom', 'apt'), 201),
 (('upper', 'east'), 191),
 (('new', 'york'), 175),
 (('apt', '.'), 172),
 (('2', 'bed'), 158),
 (('sonder', 'stock'), 158),
 (('stock', 'exchange'), 158),
 (('1', 'br'), 152),
 (('park', 'slope'), 150),
 (('2', 'br'), 148)]
```

[39]:
```
trigramas_mais_comuns
```

[39]:
```
[(('sonder', 'stock', 'exchange'), 158),
 (('upper', 'east', 'side'), 153),
 (('lower', 'east', 'side'), 114),
 (('upper', 'west', 'side'), 108),
 (('near', 'central', 'park'), 97),
```

```
 (('2', 'bedroom', 'apartment'), 82),
 (('1', 'bedroom', 'apt'), 73),
 (('2', 'bedroom', 'apt'), 68),
 (('new', 'york', 'city'), 65),
 (('1', 'bedroom', 'apartment'), 61),
 (('hell', "'s", 'kitchen'), 59),
 (('wyndham', 'midtown', '45'), 58),
 (('bed', '2', 'bath'), 53),
 (('near', 'times', 'square'), 50),
 (('guest', 'service', 'fee'), 49),
 (('2', 'bed', '2'), 44),
 (('one', 'bedroom', 'apartment'), 39),
 (('heart', 'east', 'village'), 38),
 (('east', 'village', 'apartment'), 37),
 (('heart', 'west', 'village'), 37)]
```

Essas são as palavras mais comuns para os anúncios de maior preço

## 0.3 Modelo (catboost)

O catboos será usado pois ele lida automaticamente com variáveis categóricas, como não tenho tantos dados também quero evitar ao máximo overfitting

### Importando Bibliotecas

[40]:
```
from catboost import CatBoostRegressor

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

### Carregando Dados

[41]:
```
df_model = pd.read_csv("teste_indicium_precificacao.csv")
```

[42]:
```
df_model.drop(['host_id', 'id', 'nome', 'host_name', 'ultima_review',␣
 ↪'reviews_por_mes'], inplace=True, axis=1)
```

### Limpando Dados

[43]:
```
df_model.isnull().sum()
```

[43]:
```
bairro_group                    0
bairro                          0
latitude                        0
longitude                       0
room_type                       0
price                           0
minimo_noites                   0
numero_de_reviews               0
```

```
calculado_host_listings_count    0
disponibilidade_365              0
dtype: int64
```

[44]:
```
features_categoricas = [feature for feature in df_model.columns if␣
 ↪df_model[feature].dtypes == 'object']
features_categoricas
```

[44]:
```
['bairro_group', 'bairro', 'room_type']
```

No início deste notebbok, foi explicado que a feature price possui em alguns momento valores muito altos, esses valores impedem o bom funcionanmento do modelo, por isso, será criado o seguinte modelo

modelo_1A = para treinar com os dados até o price 334

embora essa não seja a melhor forma de criar o modelo, por causa que ele pode acabar precificando de forma mais barata imóveis/quarto que poderiam ser mais caros, essa foi melhor forma que eu encontrei para se obter uma boa precição

**Treinando Modelo_1A**

[45]:
```
df_model_1A = df_model[df_model['price'] <= 337]
```

[46]:
```
x = df_model_1A.drop('price', axis=1)
y = df_model_1A['price']

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=42)
```

[47]:
```
# no momento não estou com tempo, se preicar aumentar a prenisão não esquece de␣
 ↪reduzir o learning_rate

modelo_1A = CatBoostRegressor(iterations=5000, depth=8, learning_rate=0.01,␣
 ↪verbose=20)
```

[48]:
```
# cuidado com o resultado, fica atento ao early_stopping_rounds pra evitar␣
 ↪overfitting

modelo_1A.fit(X_train, y_train, cat_features=features_categoricas,␣
 ↪eval_set=(X_test, y_test), early_stopping_rounds=1000)
```

```
0:      learn: 67.9114962       test: 67.7362375        best: 67.7362375 (0)
total: 187ms    remaining: 15m 33s
20:     learn: 62.0648444       test: 61.9641571        best: 61.9641571 (20)
total: 1.07s    remaining: 4m 14s
40:     learn: 57.6761889       test: 57.6502545        best: 57.6502545 (40)
total: 1.99s    remaining: 4m
60:     learn: 54.4343195       test: 54.4774401        best: 54.4774401 (60)
```

```
total: 3s       remaining: 4m 2s
80:     learn: 52.0228965       test: 52.1265685        best: 52.1265685 (80)
total: 4.03s    remaining: 4m 4s
100:    learn: 50.3002130       test: 50.4518974        best: 50.4518974 (100)
total: 5.06s    remaining: 4m 5s
120:    learn: 49.0260873       test: 49.2311768        best: 49.2311768 (120)
total: 6.11s    remaining: 4m 6s
140:    learn: 48.0464012       test: 48.3090031        best: 48.3090031 (140)
total: 7.13s    remaining: 4m 5s
160:    learn: 47.2836283       test: 47.5886914        best: 47.5886914 (160)
total: 8.11s    remaining: 4m 3s
180:    learn: 46.6742471       test: 47.0240888        best: 47.0240888 (180)
total: 9.07s    remaining: 4m 1s
200:    learn: 46.2194198       test: 46.6163459        best: 46.6163459 (200)
total: 10s      remaining: 3m 59s
220:    learn: 45.8810533       test: 46.3197220        best: 46.3197220 (220)
total: 11s      remaining: 3m 57s
240:    learn: 45.6016298       test: 46.0824089        best: 46.0824089 (240)
total: 12s      remaining: 3m 56s
260:    learn: 45.3760116       test: 45.8949953        best: 45.8949953 (260)
total: 13s      remaining: 3m 55s
280:    learn: 45.1733350       test: 45.7452244        best: 45.7452244 (280)
total: 14s      remaining: 3m 54s
300:    learn: 45.0084329       test: 45.6214488        best: 45.6214488 (300)
total: 15s      remaining: 3m 53s
320:    learn: 44.8676977       test: 45.5223839        best: 45.5223839 (320)
total: 16s      remaining: 3m 53s
340:    learn: 44.7511091       test: 45.4393367        best: 45.4393367 (340)
total: 17.1s    remaining: 3m 53s
360:    learn: 44.6464045       test: 45.3686535        best: 45.3686535 (360)
total: 18.1s    remaining: 3m 52s
380:    learn: 44.5533519       test: 45.3090519        best: 45.3090519 (380)
total: 19.1s    remaining: 3m 51s
400:    learn: 44.4678718       test: 45.2558408        best: 45.2558408 (400)
total: 20.2s    remaining: 3m 51s
420:    learn: 44.3902288       test: 45.2069294        best: 45.2069294 (420)
total: 21.2s    remaining: 3m 50s
440:    learn: 44.3137611       test: 45.1619552        best: 45.1619552 (440)
total: 22.3s    remaining: 3m 50s
460:    learn: 44.2490576       test: 45.1278773        best: 45.1278773 (460)
total: 23.4s    remaining: 3m 50s
480:    learn: 44.1889593       test: 45.1003061        best: 45.1003061 (480)
total: 24.5s    remaining: 3m 50s
500:    learn: 44.1359291       test: 45.0732818        best: 45.0732818 (500)
total: 25.5s    remaining: 3m 48s
520:    learn: 44.0822106       test: 45.0496987        best: 45.0496987 (520)
total: 26.6s    remaining: 3m 48s
540:    learn: 44.0284046       test: 45.0269770        best: 45.0269770 (540)
```

```
total: 27.7s    remaining: 3m 48s
560:    learn: 43.9765427       test: 45.0010584        best: 45.0010584 (560)
total: 28.8s    remaining: 3m 47s
580:    learn: 43.9257337       test: 44.9798271        best: 44.9798271 (580)
total: 30.1s    remaining: 3m 48s
600:    learn: 43.8798292       test: 44.9592973        best: 44.9592973 (600)
total: 31.3s    remaining: 3m 49s
620:    learn: 43.8335878       test: 44.9420561        best: 44.9420561 (620)
total: 32.4s    remaining: 3m 48s
640:    learn: 43.7950389       test: 44.9241713        best: 44.9241713 (640)
total: 33.5s    remaining: 3m 47s
660:    learn: 43.7558642       test: 44.9041652        best: 44.9041652 (660)
total: 34.5s    remaining: 3m 46s
680:    learn: 43.7221398       test: 44.8913115        best: 44.8913115 (680)
total: 35.6s    remaining: 3m 45s
700:    learn: 43.6866243       test: 44.8801860        best: 44.8801860 (700)
total: 36.7s    remaining: 3m 44s
720:    learn: 43.6489715       test: 44.8670447        best: 44.8670447 (720)
total: 37.7s    remaining: 3m 43s
740:    learn: 43.6219027       test: 44.8585704        best: 44.8585704 (740)
total: 38.7s    remaining: 3m 42s
760:    learn: 43.5853634       test: 44.8453478        best: 44.8453397 (759)
total: 39.7s    remaining: 3m 41s
780:    learn: 43.5551438       test: 44.8350625        best: 44.8350625 (780)
total: 40.7s    remaining: 3m 40s
800:    learn: 43.5246394       test: 44.8228125        best: 44.8228125 (800)
total: 41.7s    remaining: 3m 38s
820:    learn: 43.4926917       test: 44.8116845        best: 44.8114693 (819)
total: 42.7s    remaining: 3m 37s
840:    learn: 43.4689139       test: 44.8042075        best: 44.8042075 (840)
total: 43.8s    remaining: 3m 36s
860:    learn: 43.4434919       test: 44.7967273        best: 44.7967273 (860)
total: 44.8s    remaining: 3m 35s
880:    learn: 43.4125809       test: 44.7862551        best: 44.7862551 (880)
total: 45.8s    remaining: 3m 34s
900:    learn: 43.3858070       test: 44.7787877        best: 44.7787738 (899)
total: 46.8s    remaining: 3m 33s
920:    learn: 43.3572840       test: 44.7689110        best: 44.7689110 (920)
total: 47.9s    remaining: 3m 31s
940:    learn: 43.3265779       test: 44.7596207        best: 44.7595776 (939)
total: 48.9s    remaining: 3m 30s
960:    learn: 43.3011231       test: 44.7530380        best: 44.7530380 (960)
total: 50s      remaining: 3m 30s
980:    learn: 43.2687699       test: 44.7413184        best: 44.7413184 (980)
total: 51s      remaining: 3m 28s
1000:   learn: 43.2377882       test: 44.7305479        best: 44.7305479 (1000)
total: 52.1s    remaining: 3m 28s
1020:   learn: 43.2108134       test: 44.7214152        best: 44.7213170 (1019)
```

```
total: 53.1s    remaining: 3m 27s
1040:   learn: 43.1867234       test: 44.7137873        best: 44.7137873 (1040)
total: 54.1s    remaining: 3m 25s
1060:   learn: 43.1570112       test: 44.7049130        best: 44.7049130 (1060)
total: 55.2s    remaining: 3m 24s
1080:   learn: 43.1261536       test: 44.6951660        best: 44.6951660 (1080)
total: 56.2s    remaining: 3m 23s
1100:   learn: 43.0967439       test: 44.6893959        best: 44.6893959 (1100)
total: 57.3s    remaining: 3m 23s
1120:   learn: 43.0712449       test: 44.6818616        best: 44.6818616 (1120)
total: 58.4s    remaining: 3m 21s
1140:   learn: 43.0457375       test: 44.6685186        best: 44.6685186 (1140)
total: 59.4s    remaining: 3m 20s
1160:   learn: 43.0187861       test: 44.6579196        best: 44.6579196 (1160)
total: 1m       remaining: 3m 19s
1180:   learn: 42.9887015       test: 44.6520877        best: 44.6520877 (1180)
total: 1m 1s    remaining: 3m 19s
1200:   learn: 42.9583076       test: 44.6409732        best: 44.6409732 (1200)
total: 1m 2s    remaining: 3m 18s
1220:   learn: 42.9276491       test: 44.6295078        best: 44.6295078 (1220)
total: 1m 3s    remaining: 3m 16s
1240:   learn: 42.8942333       test: 44.6194192        best: 44.6194192 (1240)
total: 1m 4s    remaining: 3m 16s
1260:   learn: 42.8678957       test: 44.6081492        best: 44.6081492 (1260)
total: 1m 5s    remaining: 3m 15s
1280:   learn: 42.8377362       test: 44.6000781        best: 44.5999087 (1278)
total: 1m 6s    remaining: 3m 14s
1300:   learn: 42.8081667       test: 44.5914131        best: 44.5914131 (1300)
total: 1m 7s    remaining: 3m 12s
1320:   learn: 42.7818719       test: 44.5846135        best: 44.5846135 (1320)
total: 1m 8s    remaining: 3m 11s
1340:   learn: 42.7543745       test: 44.5753785        best: 44.5753785 (1340)
total: 1m 10s   remaining: 3m 11s
1360:   learn: 42.7313565       test: 44.5684194        best: 44.5684194 (1360)
total: 1m 11s   remaining: 3m 10s
1380:   learn: 42.7105164       test: 44.5630575        best: 44.5628828 (1378)
total: 1m 12s   remaining: 3m 9s
1400:   learn: 42.6851220       test: 44.5548276        best: 44.5548276 (1400)
total: 1m 13s   remaining: 3m 7s
1420:   learn: 42.6658842       test: 44.5502193        best: 44.5502193 (1420)
total: 1m 14s   remaining: 3m 6s
1440:   learn: 42.6449834       test: 44.5431200        best: 44.5431200 (1440)
total: 1m 15s   remaining: 3m 5s
1460:   learn: 42.6160074       test: 44.5333886        best: 44.5333342 (1459)
total: 1m 16s   remaining: 3m 4s
1480:   learn: 42.5944246       test: 44.5265989        best: 44.5265989 (1480)
total: 1m 17s   remaining: 3m 3s
1500:   learn: 42.5725283       test: 44.5213809        best: 44.5213809 (1500)
```

```
total: 1m 18s   remaining: 3m 2s
1520:   learn: 42.5493431       test: 44.5144879        best: 44.5144879 (1520)
total: 1m 19s   remaining: 3m 2s
1540:   learn: 42.5284235       test: 44.5090166        best: 44.5090166 (1540)
total: 1m 20s   remaining: 3m 1s
1560:   learn: 42.5059341       test: 44.5040926        best: 44.5039945 (1558)
total: 1m 21s   remaining: 3m
1580:   learn: 42.4841396       test: 44.4959140        best: 44.4958659 (1579)
total: 1m 22s   remaining: 2m 59s
1600:   learn: 42.4650593       test: 44.4893132        best: 44.4893132 (1600)
total: 1m 23s   remaining: 2m 58s
1620:   learn: 42.4445773       test: 44.4840702        best: 44.4838610 (1618)
total: 1m 24s   remaining: 2m 57s
1640:   learn: 42.4243713       test: 44.4785264        best: 44.4785264 (1640)
total: 1m 26s   remaining: 2m 56s
1660:   learn: 42.4072047       test: 44.4733864        best: 44.4733864 (1660)
total: 1m 27s   remaining: 2m 55s
1680:   learn: 42.3860660       test: 44.4672535        best: 44.4672535 (1680)
total: 1m 28s   remaining: 2m 54s
1700:   learn: 42.3617599       test: 44.4594673        best: 44.4594673 (1700)
total: 1m 30s   remaining: 2m 54s
1720:   learn: 42.3441468       test: 44.4570219        best: 44.4567561 (1719)
total: 1m 31s   remaining: 2m 54s
1740:   learn: 42.3247334       test: 44.4512515        best: 44.4512515 (1740)
total: 1m 32s   remaining: 2m 53s
1760:   learn: 42.3049781       test: 44.4468595        best: 44.4464928 (1752)
total: 1m 33s   remaining: 2m 52s
1780:   learn: 42.2832739       test: 44.4409128        best: 44.4409128 (1780)
total: 1m 35s   remaining: 2m 52s
1800:   learn: 42.2649903       test: 44.4377449        best: 44.4377371 (1799)
total: 1m 36s   remaining: 2m 51s
1820:   learn: 42.2429872       test: 44.4303385        best: 44.4303385 (1820)
total: 1m 38s   remaining: 2m 51s
1840:   learn: 42.2271560       test: 44.4281324        best: 44.4279034 (1838)
total: 1m 39s   remaining: 2m 50s
1860:   learn: 42.2095265       test: 44.4253663        best: 44.4253663 (1860)
total: 1m 40s   remaining: 2m 49s
1880:   learn: 42.1937446       test: 44.4215425        best: 44.4215406 (1879)
total: 1m 41s   remaining: 2m 49s
1900:   learn: 42.1775462       test: 44.4177487        best: 44.4175558 (1899)
total: 1m 43s   remaining: 2m 48s
1920:   learn: 42.1562686       test: 44.4115760        best: 44.4115760 (1920)
total: 1m 44s   remaining: 2m 47s
1940:   learn: 42.1420355       test: 44.4095383        best: 44.4091426 (1932)
total: 1m 45s   remaining: 2m 46s
1960:   learn: 42.1200024       test: 44.4046598        best: 44.4046598 (1960)
total: 1m 47s   remaining: 2m 45s
1980:   learn: 42.1055633       test: 44.4038644        best: 44.4037388 (1977)
```

```
total: 1m 48s   remaining: 2m 45s
2000:   learn: 42.0876765       test: 44.3985252        best: 44.3985252 (2000)
total: 1m 49s   remaining: 2m 44s
2020:   learn: 42.0717212       test: 44.3961084        best: 44.3961084 (2020)
total: 1m 50s   remaining: 2m 43s
2040:   learn: 42.0565782       test: 44.3921227        best: 44.3921227 (2040)
total: 1m 52s   remaining: 2m 42s
2060:   learn: 42.0422701       test: 44.3876235        best: 44.3874737 (2058)
total: 1m 53s   remaining: 2m 41s
2080:   learn: 42.0283618       test: 44.3849084        best: 44.3846183 (2079)
total: 1m 54s   remaining: 2m 40s
2100:   learn: 42.0098880       test: 44.3808360        best: 44.3808360 (2100)
total: 1m 55s   remaining: 2m 39s
2120:   learn: 41.9927386       test: 44.3799403        best: 44.3797877 (2119)
total: 1m 57s   remaining: 2m 38s
2140:   learn: 41.9753940       test: 44.3758762        best: 44.3758762 (2140)
total: 1m 58s   remaining: 2m 37s
2160:   learn: 41.9572450       test: 44.3725192        best: 44.3724920 (2157)
total: 1m 59s   remaining: 2m 36s
2180:   learn: 41.9447309       test: 44.3715309        best: 44.3711596 (2166)
total: 2m       remaining: 2m 35s
2200:   learn: 41.9277876       test: 44.3682193        best: 44.3681025 (2199)
total: 2m 1s    remaining: 2m 34s
2220:   learn: 41.9104951       test: 44.3637087        best: 44.3637087 (2220)
total: 2m 2s    remaining: 2m 33s
2240:   learn: 41.8936002       test: 44.3606306        best: 44.3606306 (2240)
total: 2m 4s    remaining: 2m 32s
2260:   learn: 41.8755849       test: 44.3589638        best: 44.3587463 (2258)
total: 2m 5s    remaining: 2m 31s
2280:   learn: 41.8607556       test: 44.3577177        best: 44.3572263 (2277)
total: 2m 6s    remaining: 2m 30s
2300:   learn: 41.8462140       test: 44.3552012        best: 44.3552012 (2300)
total: 2m 7s    remaining: 2m 30s
2320:   learn: 41.8334983       test: 44.3533361        best: 44.3533361 (2320)
total: 2m 9s    remaining: 2m 29s
2340:   learn: 41.8159213       test: 44.3506653        best: 44.3504383 (2339)
total: 2m 11s   remaining: 2m 28s
2360:   learn: 41.7995117       test: 44.3472013        best: 44.3472013 (2360)
total: 2m 12s   remaining: 2m 27s
2380:   learn: 41.7835452       test: 44.3428928        best: 44.3428928 (2380)
total: 2m 13s   remaining: 2m 27s
2400:   learn: 41.7693436       test: 44.3410537        best: 44.3410537 (2400)
total: 2m 14s   remaining: 2m 26s
2420:   learn: 41.7505939       test: 44.3369641        best: 44.3369641 (2420)
total: 2m 16s   remaining: 2m 25s
2440:   learn: 41.7358801       test: 44.3352085        best: 44.3350155 (2437)
total: 2m 17s   remaining: 2m 23s
2460:   learn: 41.7209051       test: 44.3346669        best: 44.3345020 (2446)
```

```
total: 2m 18s   remaining: 2m 22s
2480:   learn: 41.7048131       test: 44.3318941        best: 44.3317963 (2478)
total: 2m 19s   remaining: 2m 21s
2500:   learn: 41.6891352       test: 44.3310921        best: 44.3299746 (2492)
total: 2m 20s   remaining: 2m 20s
2520:   learn: 41.6742894       test: 44.3293226        best: 44.3289906 (2519)
total: 2m 21s   remaining: 2m 19s
2540:   learn: 41.6594053       test: 44.3273867        best: 44.3271453 (2538)
total: 2m 23s   remaining: 2m 18s
2560:   learn: 41.6435382       test: 44.3236325        best: 44.3236325 (2560)
total: 2m 24s   remaining: 2m 17s
2580:   learn: 41.6287106       test: 44.3219499        best: 44.3219499 (2580)
total: 2m 25s   remaining: 2m 16s
2600:   learn: 41.6138099       test: 44.3188291        best: 44.3184550 (2593)
total: 2m 26s   remaining: 2m 15s
2620:   learn: 41.5991716       test: 44.3183072        best: 44.3180583 (2619)
total: 2m 27s   remaining: 2m 13s
2640:   learn: 41.5832655       test: 44.3139168        best: 44.3139168 (2640)
total: 2m 28s   remaining: 2m 12s
2660:   learn: 41.5687257       test: 44.3120344        best: 44.3119562 (2659)
total: 2m 30s   remaining: 2m 11s
2680:   learn: 41.5543976       test: 44.3118437        best: 44.3114713 (2667)
total: 2m 31s   remaining: 2m 11s
2700:   learn: 41.5401858       test: 44.3097859        best: 44.3097859 (2700)
total: 2m 32s   remaining: 2m 9s
2720:   learn: 41.5243262       test: 44.3081666        best: 44.3081666 (2720)
total: 2m 33s   remaining: 2m 8s
2740:   learn: 41.5060610       test: 44.3066937        best: 44.3063610 (2734)
total: 2m 34s   remaining: 2m 7s
2760:   learn: 41.4909950       test: 44.3046703        best: 44.3046703 (2760)
total: 2m 36s   remaining: 2m 6s
2780:   learn: 41.4760580       test: 44.3032232        best: 44.3030712 (2777)
total: 2m 37s   remaining: 2m 5s
2800:   learn: 41.4613034       test: 44.3001262        best: 44.3000468 (2799)
total: 2m 38s   remaining: 2m 4s
2820:   learn: 41.4462302       test: 44.2975088        best: 44.2973542 (2816)
total: 2m 39s   remaining: 2m 3s
2840:   learn: 41.4321207       test: 44.2956647        best: 44.2956647 (2840)
total: 2m 41s   remaining: 2m 2s
2860:   learn: 41.4168748       test: 44.2936424        best: 44.2936424 (2860)
total: 2m 42s   remaining: 2m 1s
2880:   learn: 41.4037171       test: 44.2921318        best: 44.2920824 (2868)
total: 2m 43s   remaining: 2m
2900:   learn: 41.3884853       test: 44.2907592        best: 44.2905996 (2898)
total: 2m 45s   remaining: 1m 59s
2920:   learn: 41.3740271       test: 44.2898812        best: 44.2895979 (2917)
total: 2m 46s   remaining: 1m 58s
2940:   learn: 41.3575740       test: 44.2878664        best: 44.2874990 (2935)
```

```
total: 2m 47s   remaining: 1m 57s
2960:   learn: 41.3418850       test: 44.2852640        best: 44.2852640 (2960)
total: 2m 48s   remaining: 1m 56s
2980:   learn: 41.3282679       test: 44.2830093        best: 44.2830093 (2980)
total: 2m 49s   remaining: 1m 55s
3000:   learn: 41.3151600       test: 44.2798134        best: 44.2798076 (2998)
total: 2m 51s   remaining: 1m 54s
3020:   learn: 41.3021008       test: 44.2782866        best: 44.2782443 (3015)
total: 2m 53s   remaining: 1m 53s
3040:   learn: 41.2888726       test: 44.2758096        best: 44.2754316 (3034)
total: 2m 54s   remaining: 1m 52s
3060:   learn: 41.2751785       test: 44.2737857        best: 44.2734463 (3058)
total: 2m 56s   remaining: 1m 51s
3080:   learn: 41.2593980       test: 44.2740533        best: 44.2734463 (3058)
total: 2m 58s   remaining: 1m 51s
3100:   learn: 41.2440905       test: 44.2735182        best: 44.2734463 (3058)
total: 3m       remaining: 1m 50s
3120:   learn: 41.2290648       test: 44.2721450        best: 44.2721450 (3120)
total: 3m 1s    remaining: 1m 49s
3140:   learn: 41.2148729       test: 44.2701084        best: 44.2699520 (3139)
total: 3m 3s    remaining: 1m 48s
3160:   learn: 41.2013014       test: 44.2674626        best: 44.2674626 (3160)
total: 3m 4s    remaining: 1m 47s
3180:   learn: 41.1901589       test: 44.2660355        best: 44.2660355 (3180)
total: 3m 6s    remaining: 1m 46s
3200:   learn: 41.1767662       test: 44.2653763        best: 44.2652838 (3199)
total: 3m 7s    remaining: 1m 45s
3220:   learn: 41.1643907       test: 44.2641969        best: 44.2641969 (3220)
total: 3m 8s    remaining: 1m 44s
3240:   learn: 41.1534106       test: 44.2650698        best: 44.2641187 (3221)
total: 3m 10s   remaining: 1m 43s
3260:   learn: 41.1393143       test: 44.2637696        best: 44.2637696 (3260)
total: 3m 11s   remaining: 1m 41s
3280:   learn: 41.1237339       test: 44.2617836        best: 44.2617836 (3280)
total: 3m 12s   remaining: 1m 40s
3300:   learn: 41.1103842       test: 44.2587011        best: 44.2586354 (3299)
total: 3m 13s   remaining: 1m 39s
3320:   learn: 41.0968750       test: 44.2583612        best: 44.2581893 (3316)
total: 3m 14s   remaining: 1m 38s
3340:   learn: 41.0805630       test: 44.2565181        best: 44.2562526 (3338)
total: 3m 15s   remaining: 1m 37s
3360:   learn: 41.0695555       test: 44.2547960        best: 44.2547960 (3360)
total: 3m 16s   remaining: 1m 36s
3380:   learn: 41.0582373       test: 44.2543763        best: 44.2543763 (3380)
total: 3m 18s   remaining: 1m 34s
3400:   learn: 41.0442678       test: 44.2551542        best: 44.2542667 (3385)
total: 3m 19s   remaining: 1m 33s
3420:   learn: 41.0280059       test: 44.2520960        best: 44.2520960 (3420)
```

```
total: 3m 20s   remaining: 1m 32s
3440:   learn: 41.0124982       test: 44.2512683        best: 44.2512683 (3440)
total: 3m 21s   remaining: 1m 31s
3460:   learn: 40.9965945       test: 44.2497504        best: 44.2497504 (3460)
total: 3m 22s   remaining: 1m 30s
3480:   learn: 40.9836134       test: 44.2484860        best: 44.2484860 (3480)
total: 3m 23s   remaining: 1m 28s
3500:   learn: 40.9657451       test: 44.2477146        best: 44.2470786 (3493)
total: 3m 25s   remaining: 1m 27s
3520:   learn: 40.9533070       test: 44.2464160        best: 44.2464007 (3519)
total: 3m 26s   remaining: 1m 26s
3540:   learn: 40.9389064       test: 44.2437866        best: 44.2435255 (3537)
total: 3m 27s   remaining: 1m 25s
3560:   learn: 40.9236121       test: 44.2418339        best: 44.2418098 (3558)
total: 3m 28s   remaining: 1m 24s
3580:   learn: 40.9142685       test: 44.2411474        best: 44.2411474 (3580)
total: 3m 29s   remaining: 1m 23s
3600:   learn: 40.9020309       test: 44.2412961        best: 44.2402000 (3592)
total: 3m 30s   remaining: 1m 21s
3620:   learn: 40.8903635       test: 44.2407260        best: 44.2402000 (3592)
total: 3m 31s   remaining: 1m 20s
3640:   learn: 40.8768133       test: 44.2388803        best: 44.2382932 (3637)
total: 3m 33s   remaining: 1m 19s
3660:   learn: 40.8627787       test: 44.2381233        best: 44.2377821 (3647)
total: 3m 34s   remaining: 1m 18s
3680:   learn: 40.8478034       test: 44.2344418        best: 44.2344418 (3680)
total: 3m 35s   remaining: 1m 17s
3700:   learn: 40.8347665       test: 44.2334900        best: 44.2334900 (3700)
total: 3m 36s   remaining: 1m 16s
3720:   learn: 40.8239924       test: 44.2322592        best: 44.2320467 (3718)
total: 3m 37s   remaining: 1m 14s
3740:   learn: 40.8131122       test: 44.2310710        best: 44.2310710 (3740)
total: 3m 38s   remaining: 1m 13s
3760:   learn: 40.7999787       test: 44.2296737        best: 44.2296737 (3760)
total: 3m 40s   remaining: 1m 12s
3780:   learn: 40.7855998       test: 44.2289274        best: 44.2289274 (3780)
total: 3m 41s   remaining: 1m 11s
3800:   learn: 40.7736646       test: 44.2284382        best: 44.2284382 (3800)
total: 3m 42s   remaining: 1m 10s
3820:   learn: 40.7631179       test: 44.2268915        best: 44.2268915 (3820)
total: 3m 43s   remaining: 1m 8s
3840:   learn: 40.7483068       test: 44.2258436        best: 44.2256312 (3836)
total: 3m 44s   remaining: 1m 7s
3860:   learn: 40.7340830       test: 44.2241027        best: 44.2240202 (3859)
total: 3m 45s   remaining: 1m 6s
3880:   learn: 40.7207199       test: 44.2233374        best: 44.2231876 (3879)
total: 3m 46s   remaining: 1m 5s
3900:   learn: 40.7075068       test: 44.2224366        best: 44.2224108 (3897)
```

```
total: 3m 48s   remaining: 1m 4s
3920:   learn: 40.6947779       test: 44.2215398        best: 44.2210537 (3909)
total: 3m 49s   remaining: 1m 3s
3940:   learn: 40.6815079       test: 44.2188454        best: 44.2186781 (3937)
total: 3m 50s   remaining: 1m 1s
3960:   learn: 40.6653822       test: 44.2196345        best: 44.2186781 (3937)
total: 3m 51s   remaining: 1m
3980:   learn: 40.6512093       test: 44.2189942        best: 44.2186781 (3937)
total: 3m 52s   remaining: 59.6s
4000:   learn: 40.6394658       test: 44.2173852        best: 44.2171556 (3997)
total: 3m 53s   remaining: 58.4s
4020:   learn: 40.6273917       test: 44.2169010        best: 44.2168787 (4019)
total: 3m 55s   remaining: 57.2s
4040:   learn: 40.6134697       test: 44.2158988        best: 44.2155600 (4038)
total: 3m 56s   remaining: 56.1s
4060:   learn: 40.6003734       test: 44.2141074        best: 44.2140809 (4059)
total: 3m 57s   remaining: 54.9s
4080:   learn: 40.5851435       test: 44.2126594        best: 44.2126483 (4078)
total: 3m 58s   remaining: 53.7s
4100:   learn: 40.5733722       test: 44.2099532        best: 44.2099532 (4100)
total: 3m 59s   remaining: 52.5s
4120:   learn: 40.5634242       test: 44.2097966        best: 44.2093724 (4115)
total: 4m       remaining: 51.4s
4140:   learn: 40.5496842       test: 44.2083887        best: 44.2083887 (4140)
total: 4m 2s    remaining: 50.2s
4160:   learn: 40.5371845       test: 44.2073824        best: 44.2072434 (4157)
total: 4m 3s    remaining: 49s
4180:   learn: 40.5224843       test: 44.2063775        best: 44.2062874 (4177)
total: 4m 4s    remaining: 47.8s
4200:   learn: 40.5111162       test: 44.2052940        best: 44.2050242 (4191)
total: 4m 5s    remaining: 46.7s
4220:   learn: 40.4958237       test: 44.2032541        best: 44.2029678 (4218)
total: 4m 6s    remaining: 45.5s
4240:   learn: 40.4848251       test: 44.2017808        best: 44.2017808 (4240)
total: 4m 7s    remaining: 44.4s
4260:   learn: 40.4732650       test: 44.2001215        best: 44.2001215 (4260)
total: 4m 9s    remaining: 43.2s
4280:   learn: 40.4610174       test: 44.1989902        best: 44.1988676 (4278)
total: 4m 10s   remaining: 42.1s
4300:   learn: 40.4494424       test: 44.1979903        best: 44.1979903 (4300)
total: 4m 11s   remaining: 41s
4320:   learn: 40.4373828       test: 44.1973047        best: 44.1969796 (4313)
total: 4m 13s   remaining: 39.8s
4340:   learn: 40.4275115       test: 44.1966711        best: 44.1965263 (4339)
total: 4m 14s   remaining: 38.7s
4360:   learn: 40.4150963       test: 44.1954410        best: 44.1954410 (4360)
total: 4m 16s   remaining: 37.6s
4380:   learn: 40.4003116       test: 44.1940464        best: 44.1940464 (4380)
```

```
total: 4m 17s   remaining: 36.4s
4400:   learn: 40.3905966       test: 44.1938902        best: 44.1933664 (4389)
total: 4m 19s   remaining: 35.3s
4420:   learn: 40.3772066       test: 44.1924518        best: 44.1924518 (4420)
total: 4m 20s   remaining: 34.1s
4440:   learn: 40.3654505       test: 44.1910541        best: 44.1906518 (4437)
total: 4m 22s   remaining: 33s
4460:   learn: 40.3527926       test: 44.1909303        best: 44.1906518 (4437)
total: 4m 23s   remaining: 31.8s
4480:   learn: 40.3419987       test: 44.1892076        best: 44.1889873 (4479)
total: 4m 24s   remaining: 30.6s
4500:   learn: 40.3293024       test: 44.1891885        best: 44.1889414 (4483)
total: 4m 25s   remaining: 29.4s
4520:   learn: 40.3176907       test: 44.1872469        best: 44.1868492 (4518)
total: 4m 26s   remaining: 28.3s
4540:   learn: 40.3061804       test: 44.1847929        best: 44.1847929 (4540)
total: 4m 27s   remaining: 27.1s
4560:   learn: 40.2947548       test: 44.1838336        best: 44.1836644 (4559)
total: 4m 28s   remaining: 25.9s
4580:   learn: 40.2821679       test: 44.1837950        best: 44.1836644 (4559)
total: 4m 30s   remaining: 24.7s
4600:   learn: 40.2713843       test: 44.1831654        best: 44.1829055 (4597)
total: 4m 31s   remaining: 23.5s
4620:   learn: 40.2618260       test: 44.1817824        best: 44.1817417 (4619)
total: 4m 32s   remaining: 22.4s
4640:   learn: 40.2497034       test: 44.1823982        best: 44.1816456 (4623)
total: 4m 33s   remaining: 21.2s
4660:   learn: 40.2376945       test: 44.1833725        best: 44.1816456 (4623)
total: 4m 34s   remaining: 20s
4680:   learn: 40.2264297       test: 44.1819434        best: 44.1816456 (4623)
total: 4m 36s   remaining: 18.8s
4700:   learn: 40.2155814       test: 44.1814448        best: 44.1810549 (4689)
total: 4m 37s   remaining: 17.6s
4720:   learn: 40.2030119       test: 44.1797992        best: 44.1797992 (4720)
total: 4m 38s   remaining: 16.5s
4740:   learn: 40.1898732       test: 44.1788814        best: 44.1788814 (4740)
total: 4m 39s   remaining: 15.3s
4760:   learn: 40.1787041       test: 44.1779927        best: 44.1777757 (4759)
total: 4m 40s   remaining: 14.1s
4780:   learn: 40.1635290       test: 44.1774089        best: 44.1770510 (4776)
total: 4m 42s   remaining: 12.9s
4800:   learn: 40.1507816       test: 44.1771030        best: 44.1769984 (4799)
total: 4m 43s   remaining: 11.7s
4820:   learn: 40.1397487       test: 44.1759830        best: 44.1759830 (4820)
total: 4m 44s   remaining: 10.6s
4840:   learn: 40.1303382       test: 44.1760602        best: 44.1755118 (4831)
total: 4m 45s   remaining: 9.38s
4860:   learn: 40.1208961       test: 44.1756521        best: 44.1755118 (4831)
```

```
total: 4m 46s   remaining: 8.2s
4880:   learn: 40.1076191       test: 44.1732147        best: 44.1732147 (4880)
total: 4m 48s   remaining: 7.02s
4900:   learn: 40.0982405       test: 44.1731380        best: 44.1728612 (4892)
total: 4m 49s   remaining: 5.84s
4920:   learn: 40.0853027       test: 44.1733726        best: 44.1727218 (4907)
total: 4m 50s   remaining: 4.66s
4940:   learn: 40.0742732       test: 44.1721767        best: 44.1720213 (4938)
total: 4m 51s   remaining: 3.48s
4960:   learn: 40.0641967       test: 44.1721639        best: 44.1716079 (4956)
total: 4m 52s   remaining: 2.3s
4980:   learn: 40.0526534       test: 44.1716715        best: 44.1713943 (4970)
total: 4m 53s   remaining: 1.12s
4999:   learn: 40.0422340       test: 44.1713783        best: 44.1708933 (4991)
total: 4m 54s   remaining: 0us

bestTest = 44.17089325
bestIteration = 4991

Shrink model to first 4992 iterations.
```

[48]:
```
<catboost.core.CatBoostRegressor at 0x22b3a34aaf0>
```

[49]:
```
pred_model_1A = modelo_1A.predict(X_test)
```

[50]:
```
mean_absolute_error(y_test, pred_model_1A)
```

[50]:
```
31.583229345386183
```

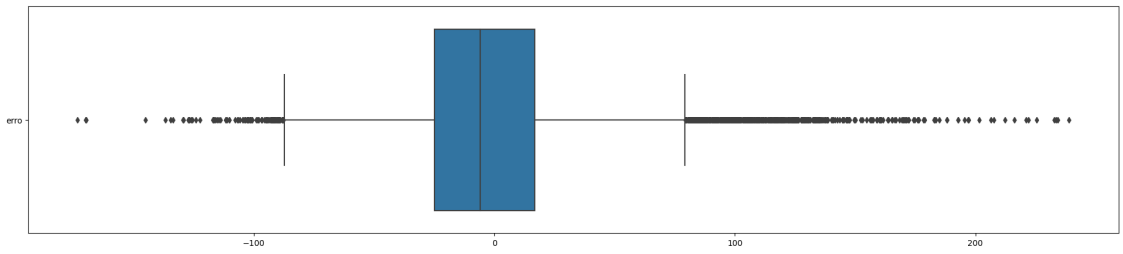
[51]:
```
def modelo(dataframe):
    if isinstance(dataframe, pd.DataFrame):
        dataframe = dataframe.drop(columns=['host_id', 'id', 'nome',␣
 ↪'host_name', 'ultima_review', 'reviews_por_mes'])
    elif isinstance(dataframe, dict):
        for key in ['host_id', 'id', 'nome', 'host_name', 'ultima_review',␣
 ↪'reviews_por_mes']:
            dataframe.pop(key)
        dataframe = pd.DataFrame([dataframe])

    valor = modelo_1A.predict(dataframe)
    return valor
```

**Verificando erro fora do intervalo interquatil** Para avaliar o erro do modelo, eu usei a erro médio absoluto, porém também quero haviar da seguinte maneira, considerando um valor, por exemplo 10 (10 sendo a unidade monetária como 10 reais ou dórlares), quanto aluguéis foram previtos com um erro maior ou menor que 10

[52]:
```
erros_model_1A = pd.DataFrame({'erro': y_test - pred_model_1A})
```

[53]:
```
fig = plt.figure(figsize=(24,5))

sns.boxplot(erros_model_1A, orient='h')
```

[53]: <Axes: >




[54]:
```
Q1 = erros_model_1A['erro'].quantile(0.25)
Q3 = erros_model_1A['erro'].quantile(0.75)

print('Q1:', Q1)
print('Q3:', Q3)
```

```
Q1: -24.915422479964732
Q3: 16.762602083806733
```

[55]:
```
pred_errado = erros_model_1A[(erros_model_1A['erro'] < -10) |␣
  ↪(erros_model_1A['erro'] > 10)].count()
qtde = erros_model_1A.shape[0]

print(f'Quantidade de linhas previstas: {qtde}')
print(f'Quantidade de linhas previstas com mais de 10: {pred_errado[0]}')
print(f'% previsto com mais de 10: {((pred_errado[0]/qtde)*100).round(2)}%')
```

```
Quantidade de linhas previstas: 9187
Quantidade de linhas previstas com mais de 10: 6867
% previsto com mais de 10: 74.75%
```

[56]:
```
pred_errado = erros_model_1A[(erros_model_1A['erro'] < -15) |␣
  ↪(erros_model_1A['erro'] > 15)].count()
qtde = erros_model_1A.shape[0]

print(f'Quantidade de linhas previtas: {qtde}')
print(f'Quantidade de linhas previtas com mais de 15: {pred_errado[0]}')
print(f'% previsto com mais de 15: {((pred_errado[0]/qtde)*100).round(2)}%')
```

```
Quantidade de linhas previtas: 9187
Quantidade de linhas previtas com mais de 15: 5842
% previsto com mais de 15: 63.59%
```

**Prevendo alguél com características informadas pela Indicium**

[57]:
```
prever_alug = pd.DataFrame([{'id': 2595,
 'nome': 'Skylit Midtown Castle',
 'host_id': 2845,
 'host_name': 'Jennifer',
 'bairro_group': 'Manhattan',
 'bairro': 'Midtown',
 'latitude': 40.75362,
 'longitude': -73.98377,
 'room_type': 'Entire home/apt',
 'minimo_noites': 1,
 'numero_de_reviews': 45,
 'ultima_review': '2019-05-21',
 'reviews_por_mes': 0.38,
 'calculado_host_listings_count': 2,
 'disponibilidade_365': 355}])
```

[58]:
```
sug_price = modelo(prever_alug)
```

[59]:
```
# preço sugerido

sug_price
```

[59]:
```
array([209.04105713])
```

### 0.3.1 Salvando o Modelo no Formato .pkl

**Importando biblioteca**

[60]:
```
import pickle
```

[61]:
```
with open('modelo_1A.pkl', 'wb') as file:
    pickle.dump(modelo, file)
```

**Testando o modelo_1A.pkl**

[62]:
```
with open('modelo_1A.pkl', 'rb') as file:
    modelo_para_prev = pickle.load(file)
```

[63]:
```
modelo_para_prev(prever_alug)
```

[63]:
```
array([209.04105713])
```