Desafio Técnico - Caio Montenegro

import das bibliotecas necessárias

In [1]:
```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
import plotly.figure_factory as ff
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from scipy.spatial import KDTree
from sklearn.metrics import mean_squared_error
from geopy.distance import geodesic
import joblib
import os
```

Importando os dados

In [2]:
```
data = pd.read_csv('teste_indicium_precificacao.csv')
```

Visualização dos dados

In [3]:
```
data
```

Out[3]:

	id	nome	host_id	host_name	bairro_group	bairr
0	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtow
1	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harler
2	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hi
3	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	Eas Harler
4	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hi
...	...	...	...	...	...	.
48889	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford Stuyvesar
48890	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwic
48891	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harler
48892	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell' Kitche
48893	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell' Kitche

48894 rows × 16 columns

Visão geral dos dados presentes no dataset

In [4]:
```
data.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
```

Estatísticas descritivas básicas

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
id                                0
nome                             16
host_id                           0
host_name                        21
bairro_group                      0
bairro                            0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimo_noites                     0
numero_de_reviews                 0
ultima_review                 10052
reviews_por_mes               10052
calculado_host_listings_count     0
disponibilidade_365               0
dtype: int64
```

In [6]:
```
data.describe()
```

Out[6]:

	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

Aqui é possível analisar algumas coisas:

- O preço médio de diária é $152,72,
- Métade dos anúncios cobra menos de $106 a diária,
- O preço mínimo e máximo me chamou atenção pela discrepância tão alta, principalmente o mínimo por se tratar de $0
- Outra coisa que me estranhou foi o méximo de minimo_noites, está muito alto, aproximadamente 3 anos. Pode ser um erro
- Vamos analisar melhor no futuro

Histograma Preço

In [7]:
```
fig = make_subplots(rows=2, cols=1,
                    shared_xaxes=True,
                    vertical_spacing=0.1,
                    subplot_titles=("Boxplot dos Preços", "Histograma dos

boxplot = go.Box(x=data['price'], name="Boxplot dos Preços", boxmean=True

histogram = go.Histogram(x=data['price'], name="Histograma dos Preços", n

fig.add_trace(boxplot, row=1, col=1)
fig.add_trace(histogram, row=2, col=1)

fig.update_layout(height=700, width=1200, title_text="Distribuição dos Pr

fig.show()
```

Nessa primeira visualização dos dois gráficos é possível observar uma grande variação nos valores mas com uma forte concentração entre $0 e 499 a diária, com intuito de entender melhor a variação nesse range de maior concentração, vamos tirar possíveis outliers para essa análise:

In [8]:
```
limite_superior = data["price"].quantile(0.99)
data_filter = data[data["price"] < limite_superior]

fig = make_subplots(rows=2, cols=1,
                    shared_xaxes=True,
```

```
                    vertical_spacing=0.1,
                    subplot_titles=("Boxplot dos Preços (Sem Outliers)",

boxplot = go.Box(x=data_filter['price'], name="Boxplot dos Preços", boxme

histogram = go.Histogram(x=data_filter['price'], name="Histograma dos Pre

fig.add_trace(boxplot, row=1, col=1)
fig.add_trace(histogram, row=2, col=1)

fig.update_layout(
    height=700,
    width=1200,
    title_text="Distribuição dos Preços (Sem Outliers no Gráfico)",
    xaxis_title="Preço",
    yaxis_title="Frequência",
)

fig.show()
```

Verificar a correlação da região com o preço

In [9]:
```
fig = make_subplots(rows=1, cols=1, subplot_titles=["Distribuição do preç

boxplot= go.Box(
    x=data["bairro_group"],
    y=data["price"],
    name="Preço por Região",
    boxmean=True
)

fig.add_trace(boxplot, row=1, col=1)

fig.update_layout(
    height=1000,
    width=1000,
    title_text="Distribuição de Preços por Região",
    xaxis_title="Região",
    yaxis_title="Preço"
)

fig.show()
```

Podemos perceber que a média de preço médio de diária é (aproximadamente): Manhanttan (197), Brooklyn (124), Queens (99), Staten Island (115) e Bronx (87)

In [10]:
```
bairros_valorizados = data.groupby(["bairro", "bairro_group"]).agg(
    preco_medio=("price", "mean"),
    quantidade_imoveis=("price", "count"),
    latitude=("latitude", "mean"),
    longitude=("longitude", "mean")
).reset_index()


bairros_valorizados = bairros_valorizados[bairros_valorizados["quantidade

bairros_valorizados = bairros_valorizados.sort_values(by="preco_medio", a
```

```
print(bairros_valorizados.head(10))
```

```
                bairro   bairro_group  preco_medio  quantidade_imoveis  \
197            Tribeca      Manhattan   490.638418                 177
174          Sea Gate       Brooklyn   487.857143                   7
167         Riverdale          Bronx   442.090909                  11
6    Battery Park City      Manhattan   367.557143                  70
75   Flatiron District      Manhattan   341.925000                  80
161      Randall Manor  Staten Island   336.000000                  19
144               NoHo      Manhattan   295.717949                  78
178               SoHo      Manhattan   287.103352                 358
127            Midtown      Manhattan   282.719094                1545
209       West Village      Manhattan   267.682292                 768

      latitude  longitude
197  40.717744 -74.007400
174  40.577119 -74.008720
167  40.887352 -73.912064
6    40.709964 -74.016585
75   40.741052 -73.988414
161  40.632749 -74.121504
144  40.727010 -73.993052
178  40.724380 -74.002024
127  40.754871 -73.975820
209  40.734351 -74.004078
```

Regiões com menos disponibilidade

In [11]:
```
disponibilidade_media = data.groupby("bairro_group")["disponibilidade_365

fig = px.bar(
    disponibilidade_media.sort_values(by="disponibilidade_365", ascending
    x="bairro_group",
    y="disponibilidade_365",
    title="Média de Disponibilidade ao Longo do Ano por Região",
    labels={"bairro_group": "Região", "disponibilidade_365": "Disponibili
    color="disponibilidade_365",
    color_continuous_scale="Reds"
)

fig.update_layout(height=500, width=800)

fig.show()
```

Podemos observar que os que tem mais demanda e fica menos tempo vazio (em média) é Brooklyn logo depois, com uma pequena diferença é Manhattan, seguido de Queens, Bronx e Staten Island.

In [12]:
```
localizacao = data.groupby(["latitude", "longitude"])["disponibilidade_36

fig = px.density_mapbox(
    localizacao,
    lat="latitude",
    lon="longitude",
    z="disponibilidade_365",
    radius=10,
    center={"lat": data["latitude"].mean(), "lon": data["longitude"].mean
```

```
    zoom=10,
    mapbox_style="open-street-map",
    title="Mapa de Calor das Áreas com Menor Disponibilidade Média"
)

fig.show()
```

Nessa mapa podemos observar melhor as áreas com maior e menor demanda

Gráfico para Preço por Tipo de Quarto

In [13]:
```
media_preco_tipo = data.groupby("room_type")["price"].mean().reset_index(

fig = px.bar(
    media_preco_tipo.sort_values(by="price", ascending=False),
    x="room_type",
    y="price",
    text="price",
    title="Preço Médio por Tipo de Acomodação",
    labels={"room_type": "Tipo de Acomodação", "price": "Preço Médio"},
    color="price",
    color_continuous_scale="greens"
)

fig.update_layout(height=500, width=800)

fig.show()
```

Como de se esperar, a ordem do mais caro ao mais barato é: Entire home/apt, Private room e por fim, Shared room

Matrix de Correlação

Verificar a correlação das variáveis

In [14]:
```
data_corr = data.copy()

data_corr = pd.get_dummies(data_corr, columns=["bairro_group", "room_type

coluna_interesse = ["price", "minimo_noites", "numero_de_reviews", "dispo
coluna_interesse += list(data_corr.filter(like="bairro_group"))
coluna_interesse += list(data_corr.filter(like="room_type"))

matrix_correlacao = data_corr[coluna_interesse].corr()

fig = ff.create_annotated_heatmap(
    z=matrix_correlacao.values,
    x=list(matrix_correlacao.columns),
    y=list(matrix_correlacao.index),
    colorscale="reds",
    annotation_text=np.round(matrix_correlacao.values, 2),
    showscale=True
)

fig.update_layout(height=800, width=1000)
```

```
fig.show()
```

Principais insights:

- room_type_Entire home/apt tem a maior correlação positiva com price (+0.26)
  - imóveis inteiros são mais caros
- bairro_group_Manhattan tem correlação positiva com price (+0.16)
  - Confirma que Manhattan é o mais caro
- disponibilidade_365 tem correlação baixa com price (+0.08)
  - Sugere que a disponilidade não tem uma influência forte no preço
- minimo_noites tem correlação fraca com price (+0.04)
  - O número mínimo de noites nao vai impactar tanto no valor da diária

1. Principais características entre as variáveis e apresentando algumas hipóteses de

negócio relacionadas

Hipóteses:

- Manhattan tem preços significativamente mais altos que outras regiões
  - obs: O preço médio da diária em Manhattan é $197, enquanto em Brooklyn é 124$, em Queens 99, $Staten Island 115$ e Bronx $87.
  - Como testar: t-test para comparar a diferença entre os preços por bairro_group.
- Imóveis inteiros (Entire home/apt) são os mais caros
  - obs: Entire home/apt (211.79), $Private room (89.77)$ e Shared room ($70.13).
  - Como testar: Teste ANOVA e boxplot.
- O número mínimo de noites não afeta significativamente o preço
  - obs: minimo_noites tem uma correlação muito fraca com price (+0.04).
  - Como testar: regressão linear para verificar se há relação estatística.
- Brooklyn tem uma disponibilidade menor ao longo do ano porque está próximo ao centro de Nova York e oferece preços mais acessíveis do que Manhattan
  - obs: Brooklyn possui menor disponibilidade média (uma demanda maior), mesmo não sendo no bairro principal.
  - Como testar: teste t de Student para ver se a diferença de preços é estatisticamente significativa.
- Acomodações compartilhadas têm maior rotatividade
  - obs: A acomodação é utilizada por mais pessoas.
  - Como testar: comparar reviews_por_mes e/ou numero_de_reviews para cada tipo de acomodação.
- Manhattan tem os imóveis mais caros para compra comparado a outras regiões de Nova York

- obs: O preço médio de aluguel diário em Manhattan é o mais alto do dataset ($197), indicando que a região é valorizada.
- Como testar: Fonte de dados externos e comparar preços de venda entre bairros e correlacionar com preços de aluguel.

2

a) Supondo que uma pessoa esteja pensando em investir em um

apartamento para alugar na plataforma, onde seria mais indicada a

compra?

Depende do foco principal do investidor:

- Caso queira uma receita média por diária maior, a melhor escolha é Manhattan ($197);
  - Ainda tem a 2 maior demanda, tendo boa rotatividade também
- Caso queira uma maior demanda e ocupação, o melhor é o Brooklyn, que possui a maior quantidade de dias ocupados por anos
  - Ainda tem o segundo maior preço médio de diária ($124)
- Mas podemos fazer uma estimativa da receita média por ano por bairro de acordo com a média de preço de diária e média de ocupação:
  - nesse caso, Manhattan tem uma ocupação média de 253 dias (365-112) e diária média de $197, multiplcandoosdois$:49841 (aproximadamente)
  - Já o Brooklyn tem uma ocupação média de 265 dias (365-100) e diária média de $124, multiplcandoosdois$:32860 (aproximadamente)
    - Então, visando apenas a estimativa de rentabilidade média por ano, a escolha é Manhattan

b) O número mínimo de noites e a disponibilidade ao longo do ano

interferem no preço?

De acordo com a matriz de correlação, ambos tem uma correlação muito fraca com preço (minimo_noites: 0.04 e disponibilidade_365: 0.06) ou seja, interferem muito pouco

C) Existe algum padrão no texto do nome do local para lugares de mais

alto valor?

Para fazer isso vou filtrar os 1% mais caros e ver palavras que mais se repetem. Com isso, chegamos nisso: - luxury: 60 ocorrências - bedroom: 54 ocorrências - loft: 52 ocorrências - private: 50 ocorrências - townhouse: 47 ocorrências - apartment: 38 ocorrências - village: 38 ocorrências - park: 34 ocorrências - with: 33 ocorrências - manhattan: 30 ocorrências - room: 26 ocorrências - brooklyn: 26 ocorrências - west: 24 ocorrências - east: 24 ocorrências - midtown: 22 ocorrências - bath: 22 ocorrências - views: 22 ocorrências - beautiful: 20 ocorrências - central: 20 ocorrências - penthouse: 20 ocorrências

Com isso, podemos tirar dessa conta o conectivo "with" do pensamento. Chegamos a localizações como "Manhattan", "Brooklyn", "Midtown", "West", "East". Esses três últimos podem ser coisas como "Midtown Manhattan", "West Brooklyn", "East Manhattan". Também termos que são comumente usados para acomodações de alto padrão como "Luxury", "Penthouse", "Loft", "Townhouse", "Private" e adjetivos positivos, sendo eles "Views" e "Beautiful".

In [15]:
```
data_words = data.copy()

maior_preco = data_words["price"].quantile(0.99)
data_maior_preco = data_words[data_words["price"] >= maior_preco]

anuncio = " ".join(data_maior_preco["nome"].dropna()).lower()

anuncio_limpo = re.sub(r'[^a-zA-Z\s]', '', anuncio)

palavras = anuncio_limpo.split()

df_palavras = pd.DataFrame(np.unique(palavras, return_counts=True)).T
df_palavras.columns = ["Palavra", "Frequência"]
df_palavras = df_palavras.sort_values(by="Frequência", ascending=False)

df_palavras = df_palavras[df_palavras["Palavra"].str.len() > 3]

fig = px.bar(
    df_palavras.head(20), x="Palavra", y="Frequência",
    title="Palavras mais comuns nos nomes dos locais mais caros",
    labels={"Palavra": "Palavra no Nome do Local", "Frequência": "Número
    text="Frequência",
    color="Frequência",
    color_continuous_scale="Blues"
)

fig.update_layout(xaxis_title="Palavra", yaxis_title="Frequência", height

fig.show()
```

3. Explique como você faria a previsão do preço a partir dos dados. Quais

variáveis e/ou suas transformações você utilizou e por quê? Qual tipo de

problema estamos resolvendo (regressão, classificação)? Qual modelo

melhor se aproxima dos dados e quais seus prós e contras? Qual medida de

performance do modelo foi escolhida e por quê?

Primeiramente é válido ressaltar que a previsão do preço das acomodações é um problema de regressão, pois o objetivo é estimar um valor numérico contínuo. Para isso, é necessário um processo estruturado que envolve pré-processamento, seleção de variáveis, escolha do modelo e definição da métrica de performance. No pré-processamento, o primeiro passo é tratar outliers, pois identifiquei em variáveis como price e minimo_noites valores discrepantes e que podem causar falsos resultados. Além disso, para usar as variáveis categóricas, como bairro_group e room_type, devem ser convertidas em variáveis numéricas. Para selecionar as variáveis foi levado em consideração as características mais relevantes para a previsão do preço: bairro_group, room_type, latitude, longitude, numero_de_reviews e disponibilidade_365, pois apresentam maior impacto nos valores das acomodações. Farei com o Gradient Boosting, em minha cabeça ele funcionará melhor por capturar asrelações não lineares e interações entre variáveis, e será usado RMSE (Root Mean Squared Error) como métrica de avaliação pois ela penalizará mais desvios grandes.

4 Qual seria a sua sugestão de preço?

In [16]:
```
data_model = data.copy()
```

copiar o dataset

In [17]:
```
data_model["lat_km"] = data_model["latitude"] * 111
data_model["lon_km"] = data_model["longitude"] * 85

tree = KDTree(data_model[["lat_km", "lon_km"]].values)

raio_km = 0.65
media_preco_vizinhos = []

for index, row in data_model.iterrows():
    indices_vizinhos = tree.query_ball_point([row["lat_km"], row["lon_km"
```

```
    vizinhos = data_model.iloc[indices_vizinhos]

    mesmo_room_type = vizinhos[vizinhos["room_type"] == row["room_type"]]
    if len(vizinhos) > 1:
        media_preco = vizinhos[vizinhos.index != index]["price"].mean()
    else:
        media_preco = row["price"]

    media_preco_vizinhos.append(media_preco)

data_model["media_preco_650m"] = media_preco_vizinhos
limite_inferior_price = data_model["media_preco_650m"] * 0.3
limite_superior_price = data_model["media_preco_650m"] * 2
limite_superior_min_noites = data_model["minimo_noites"].quantile(0.99)

data_model.drop(columns=["lat_km", "lon_km"], inplace=True)
data_model = data_model[
    (data_model["price"] < limite_superior_price) &
    (data_model["price"] > limite_inferior_price) &
    (data_model["minimo_noites"] < limite_superior_min_noites)
]
```

Aqui estou tirando alguns possíveis outliers e/ou valores que estejam muito acima do preço de sua vizinhança (2x o preço médio da vizinhança) ou abaixo (30% do valor médio), isso com intuito de que esses valores discrepantes não atrapalhem o resultado final. Basicamente converti as coordenadas para km e criei uma KDtree com elas. Defini um raio de busca de 0.65 km e fui analisando esses vizinhos nesse raio e seus preços. Para isso, criei a variável media_preco_650m que vai ser usada no modelo.

limpar e transformar alguns dados

In [18]:
```
data_model = pd.get_dummies(data_model, columns=["room_type", "bairro_gro
```

Transformar variáveis categóricas em numéricas

In [19]:
```
tribeca_loc = data_model.loc[data_model["bairro"] == "Tribeca", ["latitud
data_model["distancia_tribeca"] = data_model.apply(
    lambda row: geodesic((row["latitude"], row["longitude"]), tribeca_loc
)
```

Criei essa variável, ela basicamente pega as coordenadas e analisa a distância do bairro com a maior média de preço (Tribeca)

In [20]:
```
def media_preco_650m(lat, lon, bairro):
    index_vizinhos = tree.query_ball_point([lat, lon], 0.65)

    precos_vizinhos = data_model.iloc[index_vizinhos]["price"]

    if precos_vizinhos.empty:
        media_preco = precos_vizinhos.mean()
    else:
        media_preco = np.nan

    if np.isnan(media_preco):
```

```
        bairro_apartamento = bairro
        media_preco = data_model[data_model["bairro"] == bairro_apartamen

    return media_preco
```

Essa função é usada para calcular a média de preço dos apartamentos em um raio de 650 metros do apartamento que queremos prever. Está relacionado à variável media_preco_650m que foi criada mais acima.

In [21]:
```
manhattan_loc = (40.7831, -73.9712)
data_model["distancia_manhattan"] = data_model.apply(lambda row: geodesic
```

Essa variável guarda a distância de um apartamento ao centro de Manhattan (bairro_group mais valorizado), foi pensado mais para aqueles apartamentos que não ficam em Manhattan.

In [22]:
```
variaveis = [
    "room_type_Entire home/apt", "room_type_Private room",  "latitude", "
    "disponibilidade_365", "distancia_tribeca",   "media_preco_650m", "di
]

X = data_model[variaveis]
y = data_model["price"]
```

Seleção das variáveis: depois de algumas combinações diferentes cheguei a essa lista, foi a que me trouxe melhor resultado

In [23]:
```
train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2,
```

Divisão dos dados de teste (20%) e de Treinamento (80%)

In [ ]:
```
modelo_GBR = GradientBoostingRegressor(n_estimators=100, learning_rate=0.
modelo_GBR.fit(train_x, train_y)
```

Treinar o modelo: a combinação de n_estimators, learning_rate e max_depth foi testada de diversas maneiras e assim foi o que me trouxe melhor resltado sem precisar de tanto tempo para treinar.

In [101…
```
destino = os.path.join("modelo", "modelo_GBR.pkl")

joblib.dump(modelo_GBR, destino)
```

Out[101… `['modelo/modelo_GBR.pkl']`

In [101…
```
pred_y = modelo_GBR.predict(test_x)

avaliacao = np.sqrt(mean_squared_error(test_y, pred_y))
print(f"RMSE do modelo: {avaliacao:.2f}")
```

```
RMSE do modelo: 49.32
```

Root Mean Squared Error ficou na no valor de 49.32, foi o mínimo que consegui no tempo que tive. Esse valor significa que o modelo tem um erro médio de $49.32 para

prever os preços das acomodações, é mais alto do que eu gostaria mas devido à grande variação nos preços dos imóveis achei aceitável.

In [102…

```
apartamento = pd.DataFrame([{
    "room_type_Entire home/apt": 1,
    "room_type_Private room": 0,
    "latitude": 40.75362,
    "longitude": -73.98377,
    "numero_de_reviews": 45,
    "disponibilidade_365": 355,
    "distancia_tribeca": geodesic((40.75362, -73.98377), tribeca_loc).kil
    "media_preco_650m": media_preco_650m(40.75362, -73.98377, "Midtown"),
    "distancia_manhattan": geodesic((40.75362, -73.98377), manhattan_loc)
}])

previsao_preco = modelo_GBR.predict(apartamento)[0]

print(f"Preço previsto para o apartamento: ${previsao_preco:.2f}")
```

```
Preço previsto para o apartamento: $221.99
```

Por fim, cheguei a previsão acima, 221.99 dólares a diária de acordo com os dados deste apartamento