Perguntas

January 30, 2025

[2]:
```
#1.preparando as bibliotecas que serão usadoas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.colors as mcolors
import cartopy.crs as ccrs
import cartopy.feature as cfeature
```

[3]:
```
#carregar dados
df = pd.read_csv("teste_indicium_precificacao.csv")
```

[ ]:
```
"""
2a.Supondo que uma pessoa esteja pensando em investir em um apartamento para␣
 ↪alugar na plataforma, onde seria mais indicada a compra?
"""
```

[4]:
```
# Calcular o preço médio por bairro
preco_por_bairro = df.groupby("bairro")["price"].mean().
 ↪sort_values(ascending=False)

# Exibir os 10 bairros mais caros
print(preco_por_bairro.head(10))
```

```
bairro
Fort Wadsworth       800.000000
Woodrow              700.000000
Tribeca              490.638418
Sea Gate             487.857143
Riverdale            442.090909
Prince's Bay         409.500000
Battery Park City    367.557143
Flatiron District    341.925000
Randall Manor        336.000000
NoHo                 295.717949
Name: price, dtype: float64
```

[ ]:
```
'''
Agora sabemos os bairros mais caros, mas isso não significa necessariamente que␣
 ↪eles são os melhores investimentos
'''
```

[5]:
```
# Criar uma metrica para medir o possivel rendimento anual do estabelecimento.␣
 ↪Usaremos a quantidade de dias sisponiveis e o preço do estabeleciemnto
df['rendimento_anual'] = df['price'] * df['disponibilidade_365']

# Calcular o rendimento médio por bairro
retorno_por_bairro = df.groupby("bairro")["rendimento_anual"].mean().
 ↪sort_values(ascending=False)

# Exibir os 10 bairros com maior retorno anual médio
print(retorno_por_bairro.head(10))
```

```
bairro
Fort Wadsworth       292000.000000
Randall Manor        111905.421053
Riverdale             88673.454545
Willowbrook           87399.000000
Tribeca               83613.858757
Battery Park City     75414.257143
Neponsit              70670.000000
Sea Gate              58640.142857
Mill Basin            57636.500000
Flatiron District     53649.350000
Name: rendimento_anual, dtype: float64
```

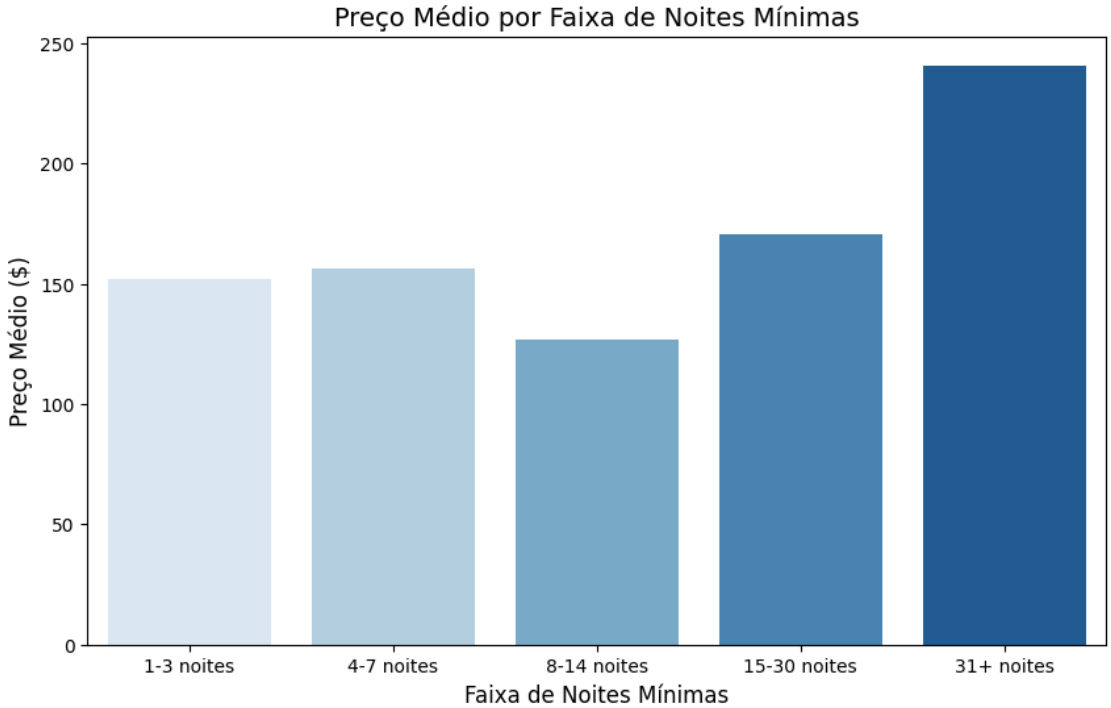
[ ]:
```
'''
É notavel que Fort Wadsworth tem o maior preço medio e o maior rendimento␣
 ↪anual, portanto ele é o melhor lugar para investir
'''
```

[ ]:
```
'''
2b.O número mínimo de noites e a disponibilidade ao longo do ano interferem no␣
 ↪preço?
'''
```

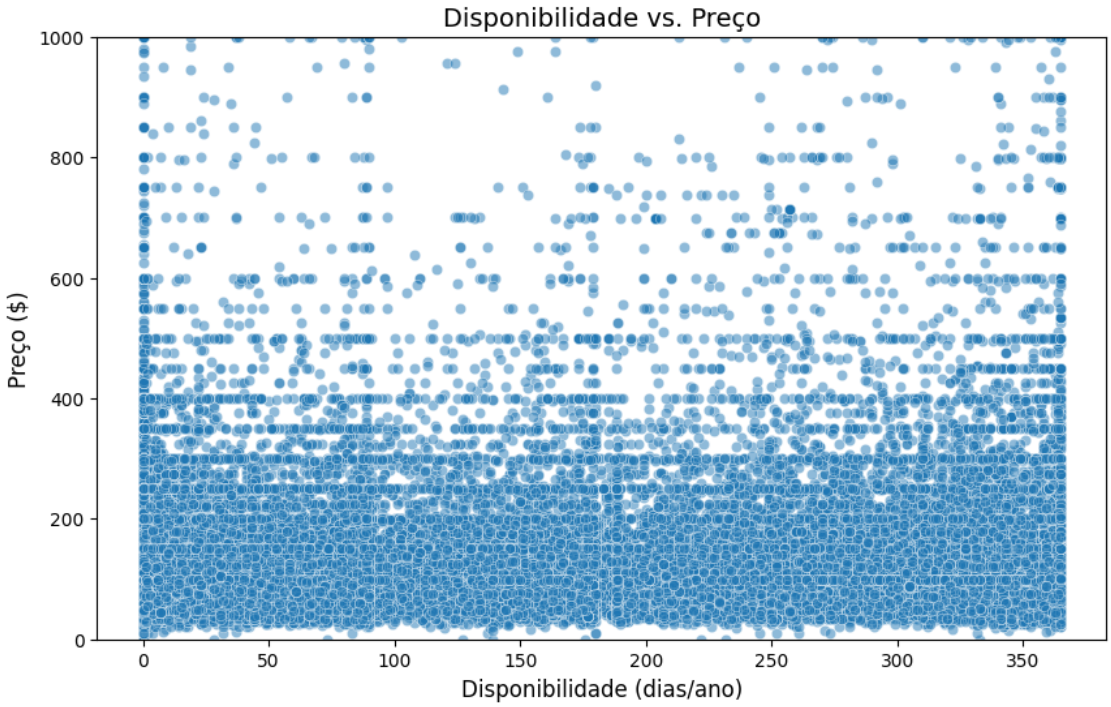
[6]:
```
# Calcular correlação entre preço, mínimo de noites e disponibilidade
correlacoes = df[['price', 'minimo_noites', 'disponibilidade_365']].corr()
print(correlacoes)
```

```
                     price  minimo_noites  disponibilidade_365
price                1.000000       0.042799             0.081833
minimo_noites        0.042799       1.000000             0.144320
disponibilidade_365  0.081833       0.144320             1.000000
```

[7]:
```
#Farei a analise grafica das correelações

# Criar categorias para noites mínimas
df['faixa_noites'] = pd.cut(df['minimo_noites'], bins=[1, 3, 7, 14, 30, 365],␣
 ↪labels=[
    "1-3 noites", "4-7 noites", "8-14 noites", "15-30 noites", "31+ noites"])

# Calcular o preço médio por faixa de noites mínimas
preco_medio_por_faixa = df.groupby("faixa_noites", observed=True)["price"].
 ↪mean()



plt.figure(figsize=(10, 6))
sns.barplot(x=preco_medio_por_faixa.index, y=preco_medio_por_faixa.values,␣
 ↪hue=preco_medio_por_faixa.index, palette="Blues", legend=False)



plt.title("Preço Médio por Faixa de Noites Mínimas", fontsize=14)
plt.xlabel("Faixa de Noites Mínimas", fontsize=12)
plt.ylabel("Preço Médio ($)", fontsize=12)
plt.show()
```



[8]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['disponibilidade_365'], y=df['price'], alpha=0.5)
plt.ylim(0, 1000)  # Evitar outliers extremos
plt.title("Disponibilidade vs. Preço", fontsize=14)
plt.xlabel("Disponibilidade (dias/ano)", fontsize=12)
plt.ylabel("Preço ($)", fontsize=12)
plt.show()
```



[ ]:
```
'''
Podemos perceber que pela analise numerica do calculo de correlação e pela␣
 ↪analise grafica que tanto a disponibilidade quanto a quantidade de noites␣
 ↪minimas tem pouca correlação com o preço. Apesar disso é notavel que quando␣
 ↪se tem de 8-14 noites o preço medio é o mais baixo e acima de 31 noites o␣
 ↪preço medio é maior.
'''
```

[ ]:
```
'''
2c.Existe algum padrão no texto do nome do local para lugares de mais alto␣
 ↪valor?
'''
```

[9]:
```
# Definir um limite superior para "lugares caros"
limite_superior = df['price'].quantile(0.75)
```

```
# Criar uma coluna indicando se o anúncio é caro
df['caro'] = df['price'] > limite_superior
```

[10]:
```
#usamos as sequintes bibliotecas para limpar os textos para analisar as␣
 ↪palavras mais comuns nos edificios caros
import nltk
from collections import Counter
from wordcloud import WordCloud


nltk.download('punkt_tab')
nltk.download('stopwords')
from nltk.corpus import stopwords

# Função para processar texto
def processar_texto(texto):
    palavras = nltk.word_tokenize(str(texto).lower())  # Converter para␣
 ↪minúsculas e tokenizar
    palavras = [palavra for palavra in palavras if palavra.isalpha()]  #␣
 ↪Remover pontuação e números
    palavras = [palavra for palavra in palavras if palavra not in stopwords.
 ↪words('english')]  # Remover stopwords
    return palavras

# Processar nomes dos anúncios caros
df_caro = df[df['caro']]
palavras_caro = df_caro['nome'].dropna().apply(processar_texto)

# Contar palavras mais comuns
todas_palavras = [palavra for lista in palavras_caro for palavra in lista]
contagem_palavras = Counter(todas_palavras)

# Exibir as 10 palavras mais comuns
print(contagem_palavras.most_common(10))
```

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

```
[('apartment', 1775), ('bedroom', 1715), ('apt', 1434), ('village', 1020),
('studio', 994), ('east', 962), ('luxury', 922), ('park', 874), ('spacious',
867), ('loft', 859)]
```

[11]:
```python
# Criar um word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(" ".join(todas_palavras))

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Palavras mais comuns em anúncios caros", fontsize=14)
plt.show()
```

Palavras mais comuns em anúncios caros

[ ]:
```
'''
Podemos ver algumas das palavras que mais aprecem nos apartamentos considerados␣
 ↪caros
'''
```

[ ]:
```
'''
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?
'''
```

[ ]:
```
'''
```

```
"A previsão do preço dos imóveis é um problema de regressão, pois estamos␣
 ↪lidando com um valor contínuo. Para construir um modelo preciso, escolhemos␣
 ↪variáveis que capturam informações relevantes, como localização (bairro,␣
 ↪latitude, longitude), tipo de acomodação, disponibilidade ao longo do ano e␣
 ↪número de avaliações.
Para o modelo preditivo, optamos pelo Gradient Boosting Regressor, pois ele␣
 ↪consegue capturar padrões complexos e não lineares, sendo mais robusto do␣
 ↪que uma regressão linear simples. Além disso, ele lida bem com dados␣
 ↪categóricos e permite ajuste fino de hiperparâmetros para melhorar a␣
 ↪performance.
A métrica utilizada para otimizar o modelo foi o R², escolhida por meio de uma␣
 ↪busca em grade (GridSearchCV). Entretanto, para melhor interpretar os erros,␣
 ↪podemos calcular também o RMSE (Root Mean Squared Error), pois ele mantém a␣
 ↪mesma unidade do preço e penaliza erros grandes de forma mais significativa."
'''
```