Relatório do projeto: Classificação de preços de residências

Richard André Satilite Carvalho

29 de janeiro de 2025

1 Introdução

O presente relatório apresenta a análise de informações e o desenvolvimento de um modelo para classificação de preços de residências, utilizando dados fornecidos pelo *dataset* da indicium. O trabalho descreve a abordagem adotada, incluindo métodos de análise, extração e filtragem dos dados, além do processo de treinamento do modelo. Também são detalhados os critérios para avaliar a precisão do modelo desenvolvido.

2 Análise exploratória de dados (AED)

Primeiramente, é realizado a disposição tabular dos dados para visualização com o uso da biblioteca de manipulação de dados *Pandas*, disponível na linguagem de programação *Python*. O *dataset* extraído é chamado de `teste_indicium_precificacao.csv`. A visualização permite aferir o tamanho do *dataset* em questão e os campos de atributos que cada residência contém.

id	...	host_id	...	bairro_group	bairro	...	room_type	price	...	disponibilidade_365
2595	...	2845	...	Manhattan	Midtown	...	Entire home/apt	225	...	355
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
36485609	...	30985759	...	Manhattan	Hell's Kitchen	...	Shared room	55	...	2
36487245	...	68119814	...	Manhattan	Harlem	...	Private room	150	...	365

Tabela 1: *Dataset* de tamanho 48894 linhas por 16 colunas.

A biblioteca *Pandas* permite uma sintetização das informações da *dataframe* bruto.

#	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

Tabela 2: Informações gerais sobre o conteúdo do *dataset*.

O resultado indica a existência de valores nulos que não devem ser consideradas para o treinamento do modelo e, portanto, devem ser removidos.

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

Tabela 3: Contagem de valores nulos em cada coluna.

Com os elementos nulos removidos, tona-se possível buscar relações das variávies do *dataframe* com o atributo de referência *price*.

#	Column	Non-Null Count	Dtype
0	id	38820 non-null	int64
1	nome	38820 non-null	object
2	host_id	38820 non-null	int64
3	host_name	38820 non-null	object
4	bairro_group	38820 non-null	object
5	bairro	38820 non-null	object
6	latitude	38820 non-null	float64
7	longitude	38820 non-null	float64
8	room_type	38820 non-null	object
9	price	38820 non-null	int64
10	minimo_noites	38820 non-null	int64
11	numero_de_reviews	38820 non-null	int64
12	ultima_review	38820 non-null	object
13	reviews_por_mes	38820 non-null	float64
14	calculado_host_listings_count	38820 non-null	int64
15	disponibilidade_365	38820 non-null	int64

Tabela 4: Informações sobre o *dataframe* sem elementos nulos.

Desta forma, o próximo processo consiste na filtragem e manipulação dos dados do *dataframe*, em especial sobre os dados não numéricos, que impossibilitam a interpretação estatística do modelo e que contém relevância sobre a decisão final na predição do preço. Os campos de `nome`, `bairro`, `bairro_group` e `room_type` não são numéricos e, como possuem relevância para a interpretação do preço, devem receber um tratamento antes de se realizar a divisão de treino e de teste.

2.1 Filtragem dos dados

Inicia-se a filtragem com a remoção de colunas que não contribuem significativamente com a métrica alvo `price` e que podem prejudicar o entendimento do modelo. São removidos os campos `id`, `host_id`, `host_name` e `ultima_review`.

Em seguida, é realizado um tratamentos dos campos que possuem conteúdos não numéricos mas que são relevantes. No caso, verifica-se a distribuição dos possíveis valores de cada campo com o método `value_counts()` do *Pandas*.

Bairro	**Quantidade**
Williamsburg	3163
Bedford-Stuyvesant	3141
Harlem	2206
Bushwick	1943
Hell's Kitchen	1531
...	...
Eltingville	2
New Dorp Beach	2
Richmondtown	1
Rossville	1
Willowbrook	1
Total	218

Tabela 5: Distribuição de registros por bairro.

Bairro_group	**Quantidade**
Manhattan	16628
Brooklyn	16444
Queens	4574
Bronx	876
Staten Island	314
Total	5

Tabela 6: Distribuição de registros por grupo de bairros.

Room Type	**Quantidade**
Entire home/apt	20327
Private room	17663
Shared room	846
Total	3

Tabela 7: Distribuição de registros por tipo de quarto.

nome	**Quantidade**
Home away from home	12
Loft Suite @ The Box House Hotel	11
Private Room	10
Brooklyn Apartment	9
New york Multi-unit building	8
...	...
Sunny Duplex in Brooklyn's Best Area	1
Private room in Charming, Cozy and Sunny Apt	1
2 Bedroom next to Prospect Park!	1
Lovely apartment as home	1
Cozy Private Room in Bushwick, Brooklyn	1
Total	38268

Tabela 8: Distribuição de registros por nome/descrição.

Baseado nos valores, o processo de *encoding* dos campos para valores numéricos depende da quantidade e do tipo informação que o campo carrega. No caso dos campos `bairro_group` e `room_type`, por se tratar de variações pequenas, o agrupamento via *hot_encoding* é suficiente e acarretará em um acréscimo de uma coluna para cada grupo, onde há um valor binário indicando se aquele registro percente ou não a um grupo, se pertence a um, não pertence aos demais. Para o campo `bairro`, dado o número elevado de variações, o método adequado será o *target_encoding*, que realizará um agrupamento médio entre todas as residências daquele bairro em específico, resultando em uma única coluna `bairro_encoded`.

nome	...	price	...	bairro_group_Bronx	bairro_group_Brooklyn
Skylit Midtown...	...	225	...	0	0
⋮	⋮	⋮	⋮	⋮	⋮
1B-1B apartment...	...	100	...	1	0
Cozy Private Room...	...	30	...	0	1

...	room_type_Entire home/apt	room_type_Private_room	...	bairro_encoded
...	1	0	...	267.583164
⋮	⋮	⋮	⋮	⋮
...	1	0	...	80.716981
...	0	1	...	85.104478

Tabela 9: *Dataset* com *encoding* de tamanho 38836 linhas por 18 colunas dividido em duas partes.

Os dados tratados permitem uma visualização de relação entre os campos com o campo alvo `price`. Com o uso das bibliotecas *matplotlib* e *seaborn* é possível gerar um *heatmap* onde associa cada campo aos demais.

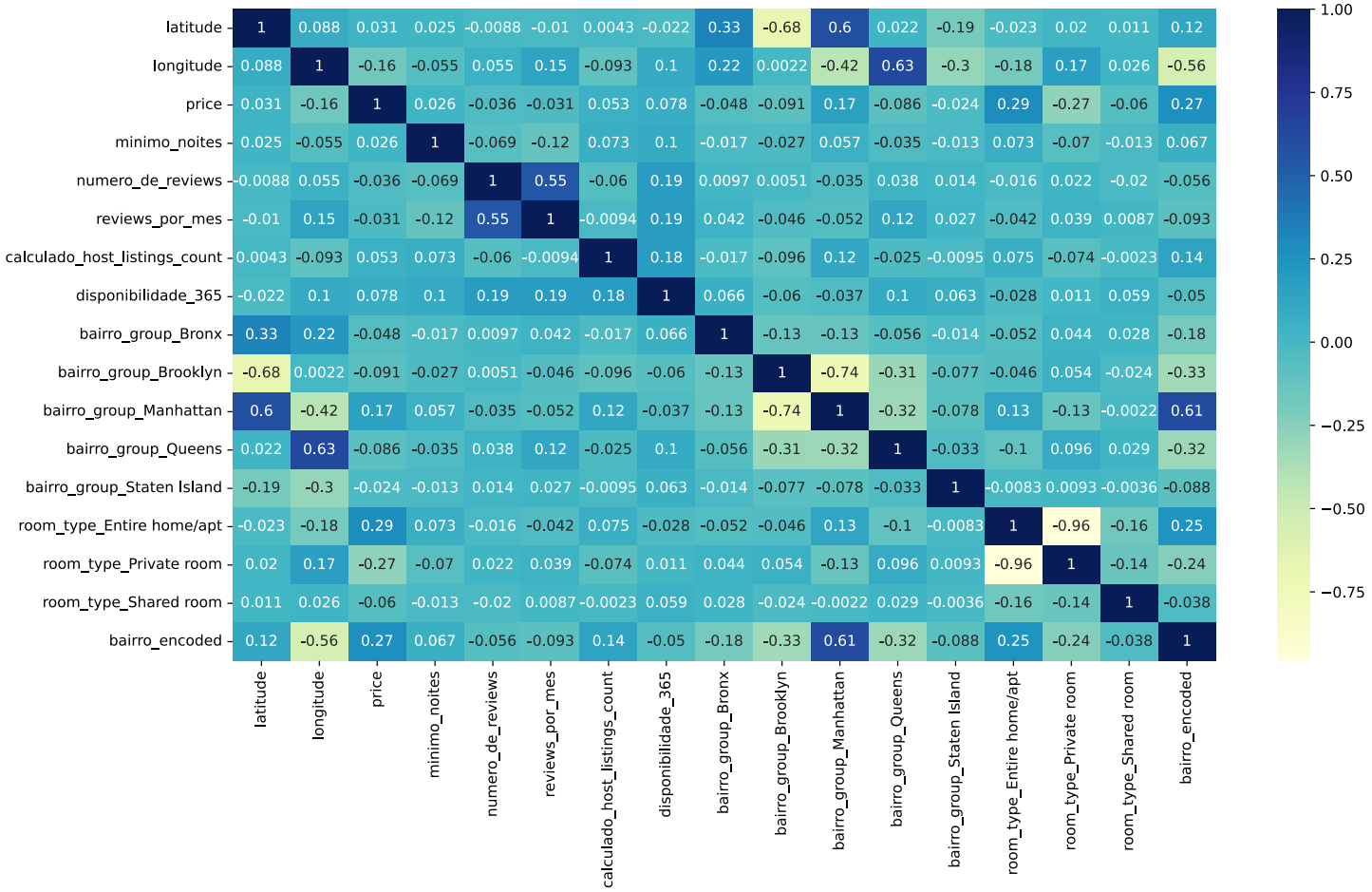

Figura 1: *Heatmap* da correlação entre os campos.

O gráfico indica que os campos `disponibilidade_365` e `minimo_noites` possuem uma relação proporcional ao campo de `price`, embora o impacto destes campos não sejam tão altos quanto os campos referentes ao grupo de bairro localizado.

O campo `nome` possui uma alta variação de um para cada registro e, com isso, o método mais adequado se trata da filtragem do texto, com remoção de caracteres especiais e palavras que não contribuem com o contexto descrito (*stopwords*), aliado à geração da matriz **TF-IDF**, capaz de indicar o termo mais importante de um conteúdo. A biblioteca *nltk* oferece uma gama de tratamentos em linguagem e filtros como *stopwords*, e é possível utilizar-se da matriz **TF-IDF** para correlacionar uma palavra do campo `nome` ao campo `price` em ordem descrescente.

nome	**correlação**
room	0.158119
luxury	0.099038
private	0.084328
loft	0.069488
cozy	0.069446
br	0.069092
village	0.061624
duplex	0.055309
midtown	0.054979
west	0.054524
bushwick	0.049341
chelsea	0.041398

Tabela 10: 12 palavras do campo `nome` com maior correlação com o campo `price`.

Os conteúdos de `nome` relcionados ao tipo de residência e localização possuem uma maior interferência no campo alvo. Ao todo, a matriz **TF-IDF** contém 77 palavras únicas que devem ser contabilizadas para o treinamento do modelo.

Com a biblioteca *seaborn*, é possível realizar a distribuição dos preços das residências em relação às posições geográficas de latitude e longitude. Previamente, dado que a concentração dos valores do campo `price` estão no intervarlo $(0, 300)$, um tratamento logarítmico, com uso da biblioteca *Numpy*, permite trazer os diferentes valores para uma faixa menor e com uma melhor visualização da distribuição.

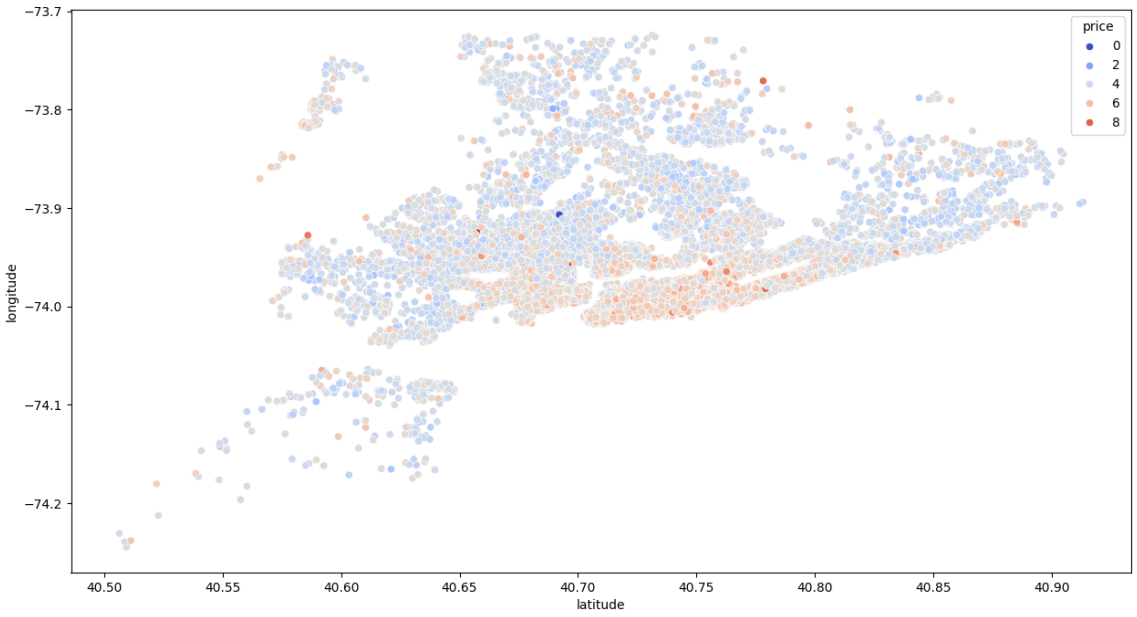

Figura 2: Distribuição dos preços em escala logarítmica com relação à latitude e à longitude.

A distibuição dos preços em relação à localização geográfica é um indicador de quais regiões alguém interessando em adquirir um imóvel deve considerar, no caso de uma decisão baseada no preço, as residências de maiores valores se encontram na faixa de $(40.65, 40.80)$ para latitude e $(-74.0, -73.9)$ para longitude.

3 Treinamento do modelo

Se tratando de um modelo preditivo para preços de residências, no caso, valores contínuos, o problema se trata de uma **regressão**. Os processos de análise exploratória mostraram que as informações de preço estão sujeitas a variações complexas não lineares como a latitude, longitude e o tipo de residência. Um dos métodos adequados à relacionar essas *features* com o campo `price` é o *Random Forest Regressor*, disponível na biblioteca *sklearn* com o método de *ensemble*.

Para avaliar o desempenho do modelo, é realizado uma disivão dos dados de treino e de teste sobre o *dataframe* já filtrado. A divisão é realizada com o médodo *train_test_split* da biblioteca *sklearn*. A razão da divisão seguirá a regra $\frac{80}{20}$, isto é, 80% do *dataframe* será destinado ao treinamento do modelo e 20% ao teste. O modelo não terá acesso ao *dataset* de teste durante o treinamento que será a usado para a avaliação de seu desempenho.

Para a correta normalização dos dados do *dataset* de treino e para evitar vazamento de dados, em especial em dados *outliers*, há um tratamento com o uso do método *StandardScaler* da biblioteca *sklearn*.

4 Resultados e validação cruzada

O resultado utilizando o método simples de *Random Forest Regressor* sobre o *dataset* de teste, com o número estimado de árvores de decisão igual a 100, foi de aproximadamente 65.29% de acurácia. Utilizando o método de valização cruzada *GridSearchCV* da biblioteca *sklearn* com os parâmetros `n_esimators` igual a $[100, 200, 300]$, `min_samples_split` igual a $[2, 4, 6, 8]$, `max_depth` igual a $[\texttt{None}, 4, 8]$ e com um `scoring` sendo do tipo erro quadrático médio, o melhor resultado quase não apresentou diferença, com uma acurácia de aproximadamente 65.40%.