Light House Indicium Desafio

February 21, 2024

##**Notebook do desafio para o processo de formação da Lighthouse Indicium.**

Candidato : Ian Lucas Périgo Vianna

email :ian.perigo.v@gmail.com

github :https://github.com/IanPerigoVianna

Whatsapp :https://api.whatsapp.com/send?phone=5573999509423

Link do repositório : https://github.com/IanPerigoVianna/Lighthouse_DataScience

1 Visualizar Dados

- Análise geral das features
- Distinguir formato das features (int, float, char)
- Identificar anomalias, dados ausentes e erros

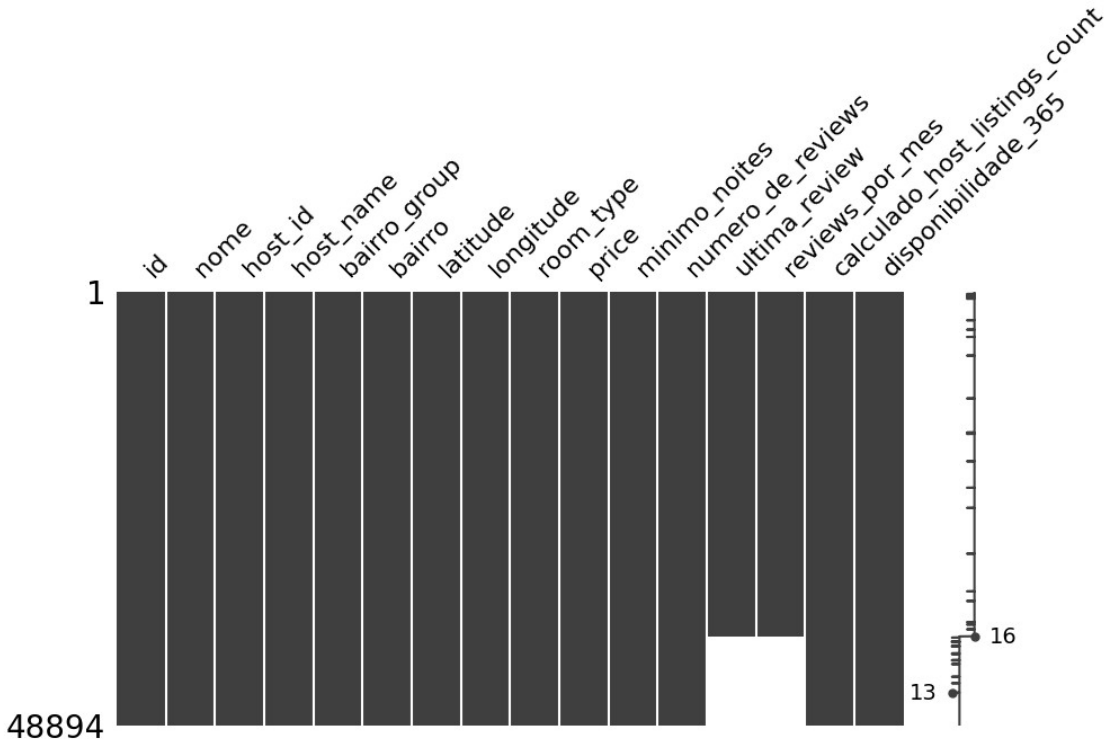
	id	host_id	latitude	longitude	price
count	4.889400e+04	4.889400e+04	48894.000000	48894.000000	48894.000000
mean	1.901753e+07	6.762139e+07	40.728951	-73.952169	152.720763
std	1.098288e+07	7.861118e+07	0.054529	0.046157	240.156625
min	2.595000e+03	2.438000e+03	40.499790	-74.244420	0.000000
25%	9.472371e+06	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

1.0.1 Análise Geral

- Notamos que algumas features apresentam valores máximos e mínimos discrepantes com a média
- Price apresenta valores mínimos igual a zero o que pode indicar um erro visto que cada imóve que será alugado tende a ter um preço.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48894 entries, 0 to 48893
Data columns (total 16 columns):
 #   Column                          Non-Null CountDtype  
---  ------                          --------------------
 0   id                              48894 non-nullint64  
 1   nome                            48878 non-nullobject 
 2   host_id                         48894 non-nullint64  
 3   host_name                       48873 non-nullobject 
 4   bairro_group                    48894 non-nullobject 
 5   bairro                          48894 non-nullobject 
 6   latitude                        48894 non-nullfloat64
 7   longitude                       48894 non-nullfloat64
 8   room_type                       48894 non-nullobject 
 9   price                           48894 non-nullint64  
 10  minimo_noites                   48894 non-nullint64  
 11  numero_de_reviews               48894 non-nullint64  
 12  ultima_review                   38842 non-nullobject 
 13  reviews_por_mes                 38842 non-nullfloat64
 14  calculado_host_listings_count   48894 non-nullint64  
 15  disponibilidade_365             48894 non-nullint64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

```
Tipo de Dados: int64
id, host_id, price, minimo_noites, numero_de_reviews,
calculado_host_listings_count, disponibilidade_365


Tipo de Dados: float64
latitude, longitude, reviews_por_mes


Tipo de Dados: object
nome, host_name, bairro_group, bairro, room_type, ultima_review




Valores únicos da feature 'room_type':
['Entire home/apt' 'Private room' 'Shared room']


Valores únicos da feature 'bairro_group':
['Manhattan' 'Brooklyn' 'Queens' 'Staten Island' 'Bronx']




Valores nulos dos dados categoricos em porcentagem
nome            0.032724
host_name       0.042950
bairro_group    0.000000
bairro          0.000000
room_type       0.000000
ultima_review  20.558760
dtype: float64
```



Observado uma correlação direta de valores nulos de reviews_por_mes e ultima_reviews.

```
Número de valores 0 em disponibilidade_365: 17533
Porcentagem de valores 0 em disponibilidade_365: 35.86%


Número de valores 0 em preço: 11
Porcentagem de valores 0 em price: 0.02%
```

Média de reviews_por_mes: 1.37
Mediana de reviews_por_mes: 0.72
Moda de reviews_por_mes: 0.02

Média de numero_de_reviews: 23.27
Mediana de numero_de_reviews: 5.00
Moda de numero_de_reviews: 0.00

Média de disponibilidade_dias_365: 112.78
Mediana de disponibilidade_dias_365: 45.00
Moda de disponibilidade_dias_365: 0.00

Média de preço: 152.72
Mediana de preço: 106.00
Moda de preço: 100.00

Média de reviews_por_mes: 1.37
Mediana de reviews_por_mes: 0.72
Moda de reviews_por_mes: 0.02

2 Manipulando e Preparando os dados

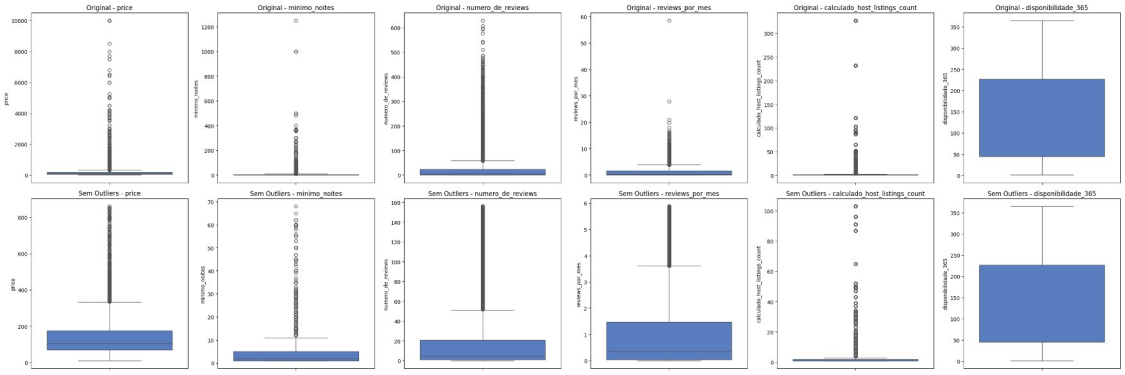
- Imputando valores ausentes e nulos
  - Features categóricas
  - Features não categóricas
- Diagnosticar e tratar Outliers
- Transformação de dados

[ ]:
```
# Inputar valores nulos da feature numero_de_review e reviews_por_mes que é␣
↪numérica com a mediana

# Verificar se os valores nulos foram inputados
```

```
id                              0
host_id                         0
latitude                        0
longitude                       0
price                           0
minimo_noites                   0
numero_de_reviews               0
reviews_por_mes                 0
calculado_host_listings_count   0
disponibilidade_365             0
dtype: int64
```

[ ]:

```
  sns.boxplot(data=df_without_outliers, y=feature, palette="muted",
ax=axes[1,i])
```

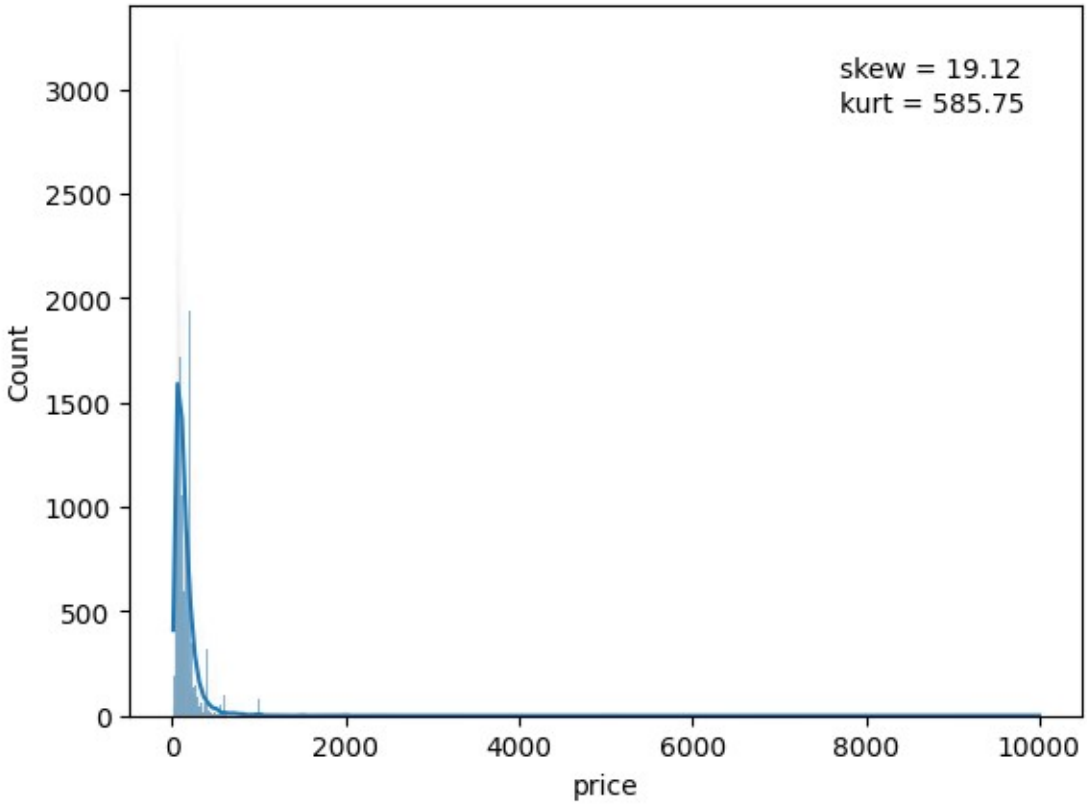


Identificado uma presença muito forte de outliers na maior parte das features.

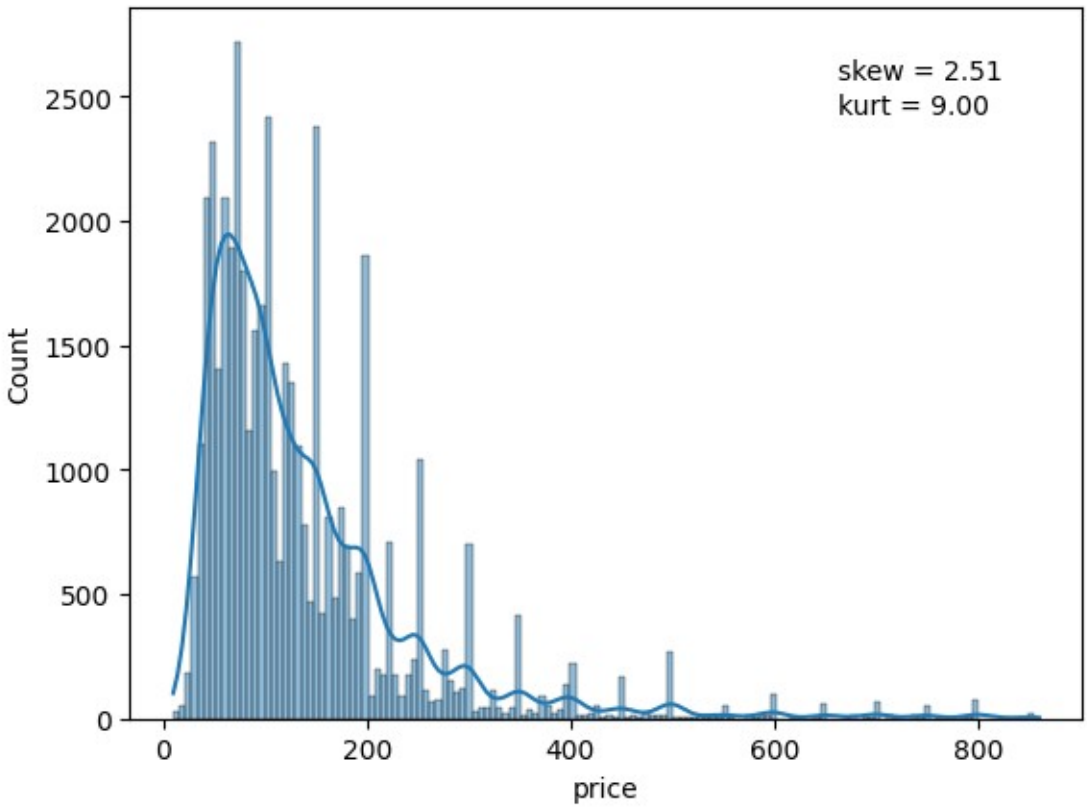
Mais de 3 mil linhas do nosso dataset foram dropadas devido aos outliers.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45483 entries, 0 to 48893
Data columns (total 16 columns):
 #   Column                          Non-Null CountDtype
---  ------                          -------------------
 0   nome                            45468 non-nullobject
 1   host_name                       45464 non-nullobject
 2   bairro_group                    45483 non-nullobject
 3   bairro                          45483 non-nullobject
 4   room_type                       45483 non-nullobject
 5   ultima_review                   45483 non-nullobject
 6   id                              45483 non-nullint64
 7   host_id                         45483 non-nullint64
 8   latitude                        45483 non-nullfloat64
 9   longitude                       45483 non-nullfloat64
 10  price                           45483 non-nullint64
 11  minimo_noites                   45483 non-nullint64
 12  numero_de_reviews               45483 non-nullint64
 13  reviews_por_mes                 45483 non-nullfloat64
 14  calculado_host_listings_count   45483 non-nullint64
```

Assimetria e curtose do preço original



Assimetria e curtose sem outliers



3 Análise Exploratória de Dados (EDA)

3.0.1 Objetivos:

- Definir os atributos mais adequados.
- Encontrar padrões
- Validar resultados.
- Refinar Features que serão utilizados para ML

[ ]:
```
price                           1.000000
disponibilidade_365             0.092545
calculado_host_listings_count   0.044119
minimo_noites                   0.011483
reviews_por_mes                -0.032093
numero_de_reviews              -0.037522
Name: price, dtype: float64
```

Observado correlações não muito fortes. Nas features numéricas

Abaixo vamos analisar as colunas categóricas.
Através de alguns processos vamos converter
Para números para poder gerar insights e melhorar
As correlações para treinar um modelo mais robusto

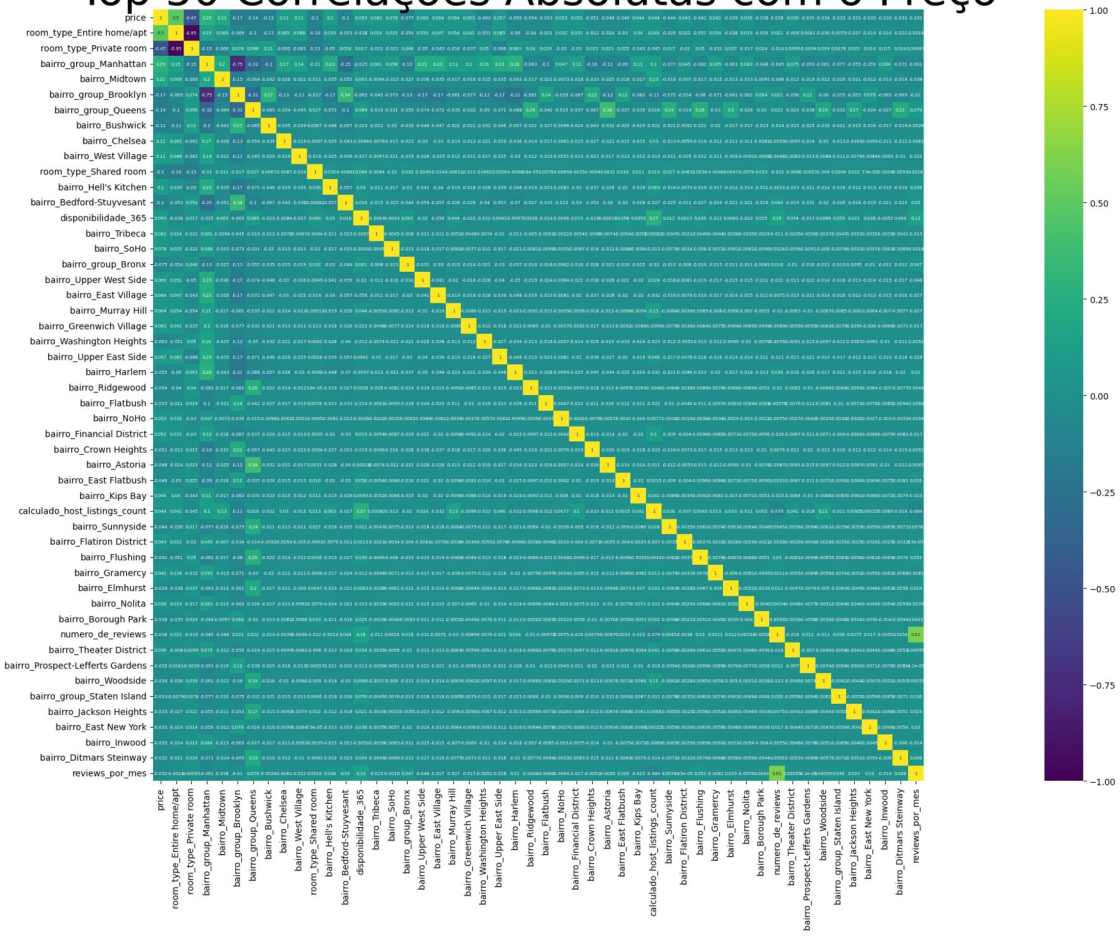
Vamos analisar os bairros que são as categorias mais presentes no nosso dataframe
O grupo do bairro que são 5 e o tipo de locação se é a casa completa, quarto compartilhado ou quarto privado.

```
Os 50 bairros mais frequentes:
Williamsburg             3733
Bedford-Stuyvesant       3498
Harlem                   2525
Bushwick                 2356
Upper West Side          1833
East Village             1735
Hell's Kitchen           1725
Upper East Side          1697
Crown Heights            1505
Midtown                  1393
Greenpoint               1086
East Harlem              1038
Chelsea                   978
Astoria                   861
Washington Heights        861
Lower East Side           856
West Village              708
Flatbush                  604
Clinton Hill              542
Long Island City          513
Prospect-Lefferts Gardens 508
Park Slope                476
Fort Greene               473
East Flatbush             472
Financial District        471
Kips Bay                  432
Murray Hill               412
Ridgewood                 408
Flushing                  394
Sunset Park               379
Greenwich Village         364
Sunnyside                 351
Chinatown                 350
Morningside Heights       333
SoHo                      330
Prospect Heights          326
Gramercy                  315
Ditmars Steinway          296
South Slope               263
Inwood                    245
Nolita                    240
Elmhurst                  232
Gowanus                   230
Carroll Gardens           229
Woodside                  221
East New York             203
Jamaica                   196
Theater District          195
Jackson Heights           177
Kensington                170
Name: bairro, dtype: int64
```

```
Correlações absolutas negativas:
Series([], Name: price, dtype: float64)
Correlações absolutas positivas:
price                     1.000000
room_type_Entire home/apt0.498956
room_type_Private room    0.467740
bairro_group_Manhattan    0.286208
bairro_Midtown            0.206084
                            …
bairro_Bay Terrace        0.000456
bairro_Eltingville        0.000297
bairro_Castleton Corners  0.000176
bairro_Holliswood         0.000173
bairro_Unionport          0.000068
Name: price, Length: 235, dtype: float64
```

Notamos várias features com melhor correlação
Com o preço do que as numéricas. Vamos
Selecionar as melhores e as que não possuem
Forte multicolinearidade para usar.

O mapa de calor gerado abaixo estará em formato
De imagem para melhor visualização ou pode ser
Visto no notebook dos códigos.



Foi observado uma forte multicolinearidade de Mahatan com Brooklyn, Entire home/apt com Private room e número de reviews por reviews por mês.

Disponibilidade de dias por ano é a primeira feature numérica original com maior correlação com o preço.

Optaremos por manter as features com maior correlação com o preço.

Análise:- Os tipos de locação total apresenta a correlação mais forte com o preço, e o tipo private room apresenta uma correlação negativa. Faz sentindo essa relação pois a locação de um imóvel inteiro tende a ser fator determinante do preço. O grupo de bairro de Manhattan apresenta a terceira maior correlação com o preço e a maior parte das correlações são pelas features dos bairr - Mínimo noites e host listening counts não apresentaram uma correlação forte com o preço.

3.0.3 Vou começar a aplicar alguns modelos de treinamento.

- O primeiro modelo testado foi de regressão linear usando o sklearn por se tratar de um problema onde a variável da resposta é contínua. O objetivo será modelar a relação entre as variáveis independentes (bairros,tipo de locação, etc..) e a variável de resposta (preço) para

fazermos previsões ou entender a relação entre essas variáveis.

- A regressão linear utilizando o sklearn obteve um Score 0.3545050872196557
- O segundo modelo foi MLPRegressor, porém obtive dificuldade na hora de inverter a normalização de y para gerar o valor real da predição do preço.
- O terceiro modelo testado foi MLP usando Keras. Só irei deixar no notebook o modelo com o melhor desempenho para não poluir de mais o notebook. O modelo que mais se adequou para o treinamento foi a MLP do Keras.

Vamos aplicar aprendizado profundo

[ ]:
```
#MLP

#!pip install keras scikit-learn


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, LeakyReLU
from keras.optimizers import Adam, RMSprop
```

[ ]:
```
X = df_top_50_dummies.drop(['price','reviews_por_mes','room_type_Private room','bairro_group_Brooklyn'], axis=1)
y = df_top_50_dummies['price']
```

[ ]:
```
# Inicializando o min-max
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_normalized = scaler_X.fit_transform(X)
y_normalized = scaler_y.fit_transform(y.values.reshape(-1, 1))



X_train, X_test, y_train, y_test = train_test_split(X_normalized, y_normalized, test_size=0.30, random_state=40)
```

[ ]:
```
# Modelo escolhido
model = Sequential()
#layer 1
model.add(Dense(300, input_dim=46, activation='relu'))
model.add(Dropout(0.1))

# layer 2
model.add(Dense(150))
model.add(Activation('tanh'))
model.add(Dropout(0.1))
```

```
#layer 3
model.add(Dense(100))
model.add(Activation('linear'))
model.add(Dropout(0.1))

#layer 4
model.add(Dense(10))
model.add(LeakyReLU(alpha = 0.1))

#Output layer
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_76 (Dense)            (None, 300)               14100

 dropout_44 (Dropout)        (None, 300)               0

 dense_77 (Dense)            (None, 150)               45150

 activation_28 (Activation)  (None, 150)               0

 dropout_45 (Dropout)        (None, 150)               0

 dense_78 (Dense)            (None, 100)               15100

 activation_29 (Activation)  (None, 100)               0

 dropout_46 (Dropout)        (None, 100)               0

 dense_79 (Dense)            (None, 10)                1010

 leaky_re_lu_16 (LeakyReLU)  (None, 10)                0

 dense_80 (Dense)            (None, 1)                 11

=================================================================
Total params: 75371 (294.42 KB)
Trainable params: 75371 (294.42 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

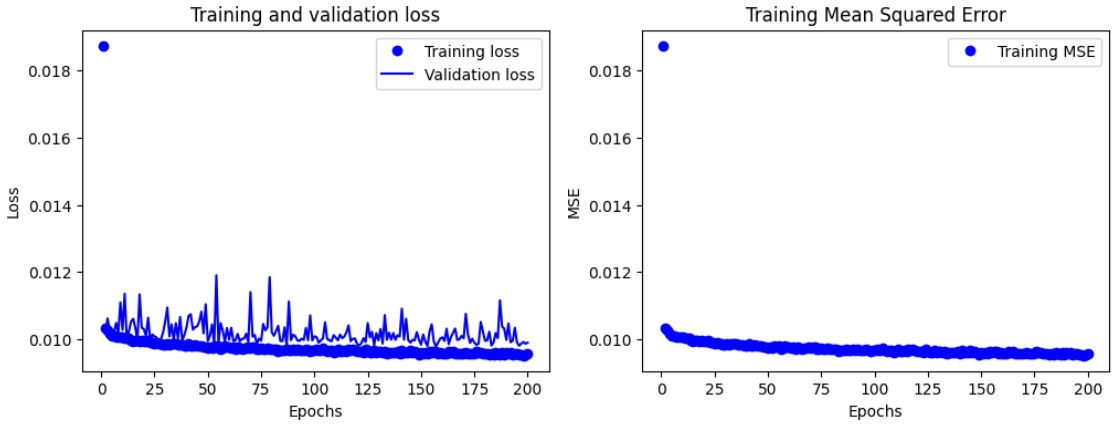
```
[ ]: model.compile(loss='mse',optimizer=RMSprop(0.005), metrics=['mse','mae'])
my_model = model.fit(X_train, y_train,␣
 ↪epochs=200,validation_data=(X_test,y_test))
```

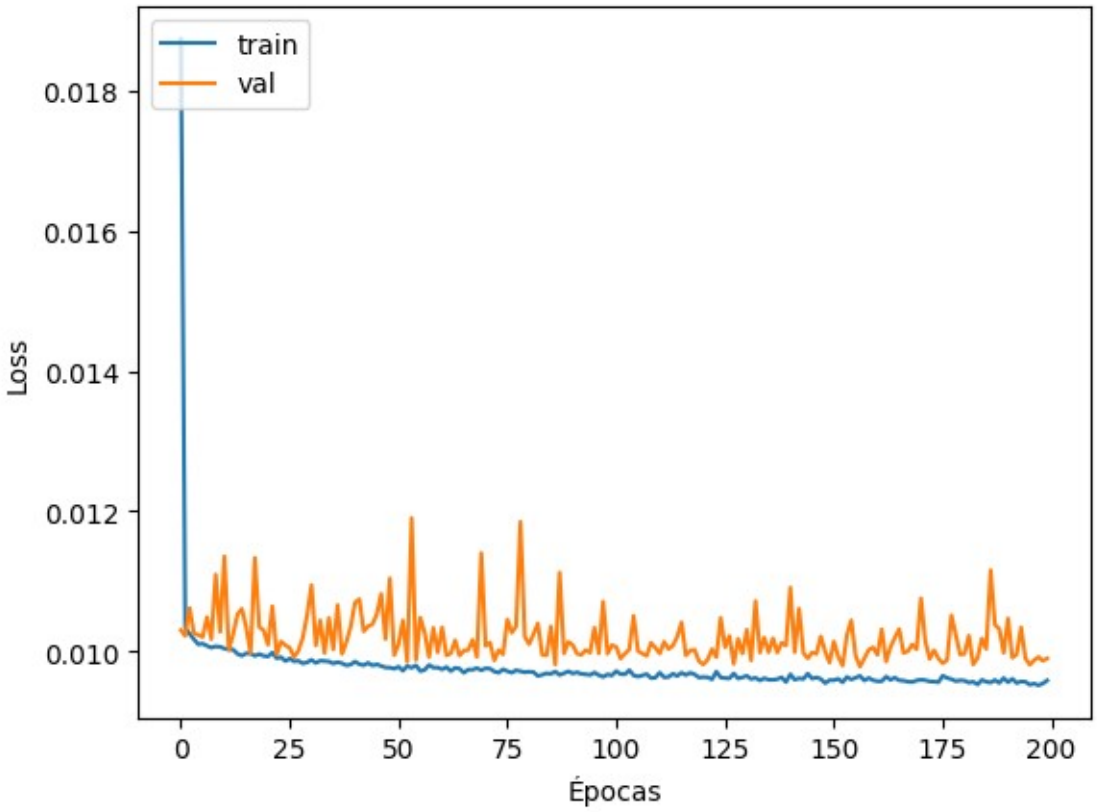
```
Epoch 1/200
995/995 [==============================] - 7s 6ms/step - loss: 0.0187 - mse:
0.0187 - mae: 0.0696 - val_loss: 0.0103 - val_mse: 0.0103 - val_mae: 0.0583
Epoch 2/200
995/995 [==============================] - 6s 6ms/step - loss: 0.0103 - mse:
0.0103 - mae: 0.0630 - val_loss: 0.0102 - val_mse: 0.0102 - val_mae: 0.0600
Epoch 3/200
995/995 [==============================] - 4s 4ms/step - loss: 0.0103 - mse:
0.0103 - mae: 0.0625 - val_loss: 0.0106 - val_mse: 0.0106 - val_mae: 0.0582
Epoch 4/200
995/995 [==============================] - 5s 5ms/step - loss: 0.0102 - mse:
0.0102 - mae: 0.0623 - val_loss: 0.0103 - val_mse: 0.0103 - val_mae: 0.0631
Epoch 5/200
995/995 [==============================] - 5s 5ms/step - loss: 0.0101 - mse:
0.0101 - mae: 0.0622 - val_loss: 0.0102 - val_mse: 0.0102 - val_mae: 0.0614
Epoch 6/200
995/995 [==============================] - 5s 5ms/step - loss: 0.0101 - mse:
0.0101 - mae: 0.0622 - val_loss: 0.0102 - val_mse: 0.0102 - val_mae: 0.0612
Epoch 7/200
995/995 [==============================] - 8s 8ms/step - loss: 0.0101 - mse:
0.0101 - mae: 0.0620 - val_loss: 0.0105 - val_mse: 0.0105 - val_mae: 0.0618
Epoch 8/200
995/995 [==============================] - 8s 8ms/step - loss: 0.0101 - mse:
0.0101 - mae: 0.0621 - val_loss: 0.0102 - val_mse: 0.0102 - val_mae: 0.0612
Epoch 9/200
995/995 [==============================] - 7s 7ms/step - loss: 0.0101 - mse:
0.0101 - mae: 0.0623 - val_loss: 0.0111 - val_mse: 0.0111 - val_mae: 0.0587
Epoch 10/200
995/995 [==============================] - 9s 9ms/step - loss: 0.0101 - mse:
0.0101 - mae: 0.0623 - val_loss: 0.0103 - val_mse: 0.0103 - val_mae: 0.0661
Epoch 11/200
995/995 [==============================] - 7s 7ms/step - loss: 0.0100 - mse:
0.0100 - mae: 0.0623 - val_loss: 0.0114 - val_mse: 0.0114 - val_mae: 0.0668
Epoch 12/200
995/995 [==============================] - 10s 10ms/step - loss: 0.0100 - mse:
0.0100 - mae: 0.0621 - val_loss: 0.0100 - val_mse: 0.0100 - val_mae: 0.0589
Epoch 13/200
995/995 [==============================] - 5s 5ms/step - loss: 0.0100 - mse:
0.0100 - mae: 0.0623 - val_loss: 0.0102 - val_mse: 0.0102 - val_mae: 0.0570
Epoch 14/200
995/995 [==============================] - 5s 5ms/step - loss: 0.0100 - mse:
0.0100 - mae: 0.0620 - val_loss: 0.0105 - val_mse: 0.0105 - val_mae: 0.0604
Epoch 15/200
```

```
995/995 [==============================] - 7s 7ms/step - loss: 0.0096 - mse:
0.0096 - mae: 0.0610 - val_loss: 0.0105 - val_mse: 0.0105 - val_mae: 0.0590
Epoch 192/200
995/995 [==============================] - 5s 5ms/step - loss: 0.0096 - mse:
0.0096 - mae: 0.0611 - val_loss: 0.0099 - val_mse: 0.0099 - val_mae: 0.0627
Epoch 193/200
995/995 [==============================] - 7s 7ms/step - loss: 0.0095 - mse:
0.0095 - mae: 0.0608 - val_loss: 0.0100 - val_mse: 0.0100 - val_mae: 0.0625
Epoch 194/200
995/995 [==============================] - 7s 7ms/step - loss: 0.0096 - mse:
0.0096 - mae: 0.0609 - val_loss: 0.0103 - val_mse: 0.0103 - val_mae: 0.0567
Epoch 195/200
995/995 [==============================] - 9s 9ms/step - loss: 0.0096 - mse:
0.0096 - mae: 0.0609 - val_loss: 0.0099 - val_mse: 0.0099 - val_mae: 0.0570
Epoch 196/200
995/995 [==============================] - 9s 9ms/step - loss: 0.0095 - mse:
0.0095 - mae: 0.0606 - val_loss: 0.0098 - val_mse: 0.0098 - val_mae: 0.0591
Epoch 197/200
995/995 [==============================] - 7s 7ms/step - loss: 0.0095 - mse:
0.0095 - mae: 0.0606 - val_loss: 0.0099 - val_mse: 0.0099 - val_mae: 0.0607
Epoch 198/200
995/995 [==============================] - 6s 6ms/step - loss: 0.0095 - mse:
0.0095 - mae: 0.0608 - val_loss: 0.0099 - val_mse: 0.0099 - val_mae: 0.0606
Epoch 199/200
995/995 [==============================] - 6s 6ms/step - loss: 0.0095 - mse:
0.0095 - mae: 0.0607 - val_loss: 0.0099 - val_mse: 0.0099 - val_mae: 0.0600
Epoch 200/200
995/995 [==============================] - 7s 7ms/step - loss: 0.0096 - mse:
0.0096 - mae: 0.0607 - val_loss: 0.0099 - val_mse: 0.0099 - val_mae: 0.0569
```

[ ]:
```python
# Treinamento do modelo
history = my_model.history

# Extraindo as métricas
loss = history['loss']
val_loss = history['val_loss']
mse = history['mse']
mae = history['mae']

# Número de épocas
epochs = range(1, len(loss) + 1)

# Visualizando a perda
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
```

```
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Visualizando o erro quadrático médio
plt.subplot(1, 2, 2)
plt.plot(epochs, mse, 'bo', label='Training MSE')
plt.title('Training Mean Squared Error')
plt.xlabel('Epochs')
plt.ylabel('MSE')
plt.legend()

plt.show()
```




train
val
0.018
0.016
0.014
0.012
0.010
Loss
0
25
50
75
100
125
150
175
200
Épocas

```
previsao_preco = scaler_y.inverse_transform(previsao_preco_scaled)


print(f"Sugestão de preço original: $ previsao_preco[0][0]:.2f} ")
```

```
1/1 [==============================] - 0s 77ms/step
Sugestão de preço original: $261.22
```

3.1.1 Sugestão de preço do nosso modelo:

- Sugestão de preço: $261.22

3.1.2 Avaliação da Mult-layer Perceptron

- Foi realizado o treino do alvo sem estar normalizado porém este apresentou um resultado muito negativo de perda, mse e mea.
- O alvo foi normalizado no treino, porém apresentou uma tendência a overfitting, para atenuar este problema foi diminuido a capacidade de aprendizado de 0.01 para 0.005
- Devido a disponibilidade do tempo não realizei treinos que passassem de 300 épocas.
- O último modelo ficou com 200 épocas
- Métricas (quanto mais próximo de 0 melhor) :
  - loss:0.0096 - mse:0.0096 - mae:0.0607 - val_loss:0.0099 - val_mse:0.0099 - val_mae: 0.0569
- Funções de ativação : relu,linear,tanh e leakrelu.

3.1.3 Nosso modelo de predição parou por aqui porém vamos investigar algumas possíveis correlações que no momento ainda não serão incorporadas ao nosso modelo. São elas:

- Nome do anúncio/imóvel
  - Iremos fazer processamento de texto, vetorização e dummies para verificar correlações.
- latitude e longitude
  - Iremos fazer o cálculo da distância de alguns imóveis por um de referência que será o mais caro.

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

[ ]:
```
#Importações necessárias
from sklearn.feature_extraction.text import CountVectorizer
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
```

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

def preprocess_text(text):
```

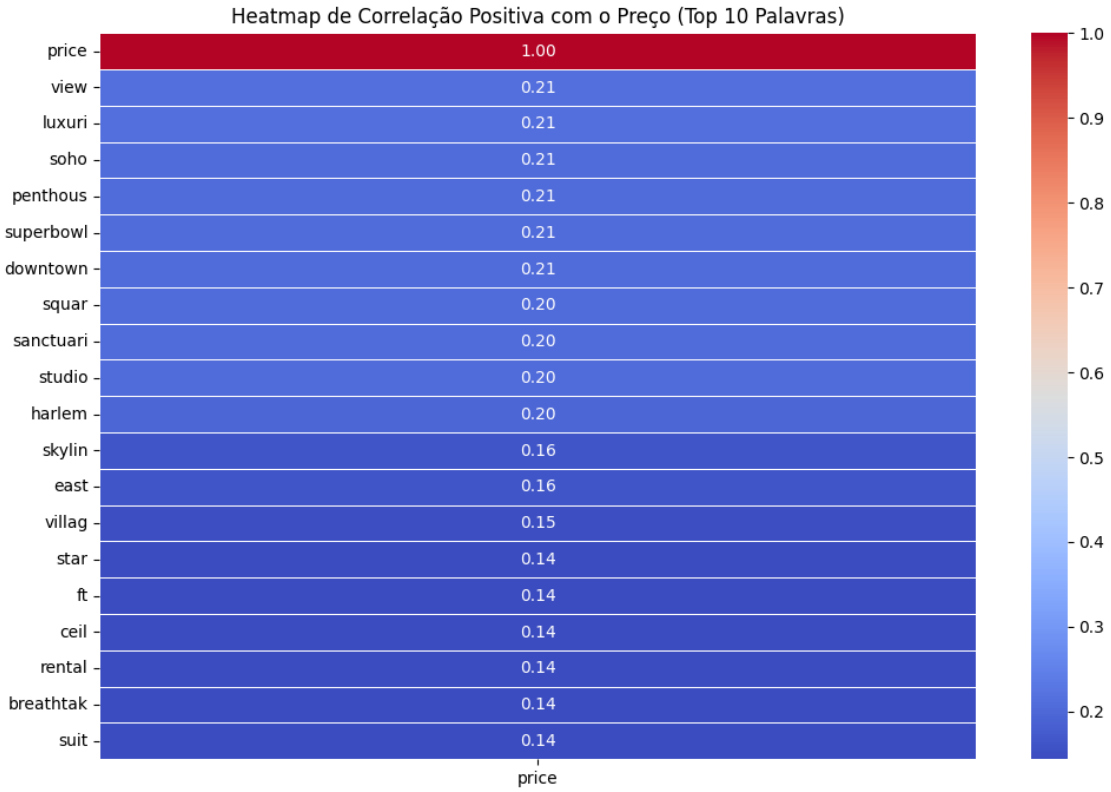
```
top_50_precos = top_50_precos.sort_values('price', ascending=False)

# Seleciona as 100 primeiras linhas do texto processado
primeiras_100_linhas = top_50_precos[['texto_processado', 'price']].head(100)

# Imprime as 100 primeiras linhas
print(primeiras_100_linhas)

# top 100 das palavras com maior frequência na feature 'nome' onde os preços
 são mais altos
frequencias_totais = df_frequencias.sum()

palavras_mais_frequentes = frequencias_totais.sort_values(ascending=False)

# Imprima as 100 palavras mais frequentes e suas contagens
top_palavras = palavras_mais_frequentes.head(100)
print(top_palavras)

# Selecione as top 100 palavras mais frequentes
top100_palavras = palavras_mais_frequentes.head(100).index.tolist()

# Cria variáveis dummy para as top 100 palavras-chave em 'nome'
top100_nome_dummies = top_50_precos['nome'].apply(preprocess_text).str.
 split(expand=True).stack().str.get_dummies().groupby(level=0).
 sum()[top100_palavras]

# Adiciona as variáveis dummy ao DataFrame principal
df_with_dummies = pd.concat([top_50_precos, top100_nome_dummies], axis=1)

# Calcula a matriz de correlação
correlation_matrix = df_with_dummies[['price'] + top100_palavras].corr()

# Obtém a correlação com a coluna 'price'
correlation_price = correlation_matrix['price']

# Filtra as correlações positivas maiores que 0
correlacoes_positivas = correlation_price[correlation_price > 0]

# Exibe as 10 palavras com maior correlação positiva
print("As 10 palavras com maior correlação positiva:")
print(correlacoes_positivas.sort_values(ascending=False).head(20))
```

```
                              texto_processado  price
37340                landmark limeston sanctuari   860
5350                   spaciou studio time squar   860
3634               superbowl rental hell kitchen   850
27599                    design townhous harlem   850
```

45765	flatiron loft star	850
36535	entir suit breathtak view nyc	850
22585	gim shelter	850
661	heart soul greenwich villag	850
12982	swing chandeli	850
19182	luxuri penthous hearth soho nyc	850
32937	soho penthous sq ft ft ceil	850
28140	brand new bedroom bath downtown gem	850
28149	luxuri skylin view best panarom view nyc	850
3596	superbowl bd ba w roof deck	850
29993	luxuri privat west villag townhom	850
45840	beauti renov modern townhous	850
20406	extrem rare townhous east villag	850
3364	w villag luxuri apt super bowl	850
16139	sqf spectacular luxuri downtown loft	850
46879	triplex patio empir state build	850
35414	respect beauti harlem room	848
47062	javit beauti stay nyc vacat	843
42103	new year eve manhattan club	840
26809	upper east side bath skylin view	840
45626	new bed full bath sf spaciou apart	830
7355	manhattan hous larg group	825
23148	brownston clinton hill brooklyn	825
39891	ave luxuri regi nyc king room manhattan	820
14242	west villag bed privat roof	814
38919	nyc design privat patio	805
18751	uniqu creativ artist space brooklyn	800
23648	nyc home away home triplex	800
48838	massag spa stay overnight author artist dream	800
1200	townhous bklyn height	800
25421	new york hidden secret luxuri live	800
43061	sustain event space rooftop photoshoot	800
9055	ave flatiron loft	800
46802	beauti townhous center nyc	800
46396	javitscent privat oasi	800
18739	fantast artist loft tribeca	800
25385	beach hous retreat minut manhattan	800
1909	littl palac brooklyn shoot event	800
25317	modern townhous williamsburg	800
23335	spaciou cobbl hill townhous	800
18419	sun soak privat roof deck	800
43652	fabul nomad midtown loft	800
30721	thebluehous entirehom babygrand freepark	800
15014	artist new loft bath time sq	800
3434	west villag apt step path	800
22068	manhattan hotel room	800
nyc	8	
luxuri	7	

```
townhous    7
villag      6
loft        6
           ..
ceil        1
brownston   1
overnight   1
breathtak   1
brand       1
Length: 100, dtype: int64
As 10 palavras com maior correlação positiva:
price       1.000000
view        0.213634
luxuri      0.210924
soho        0.205287
penthous    0.205287
superbowl   0.205287
downtown    0.205287
squar       0.204037
sanctuari   0.204037
studio      0.204037
harlem      0.196662
skylin      0.162160
east        0.162160
villag      0.147719
star        0.143671
ft          0.143671
ceil        0.143671
rental      0.143671
breathtak   0.143671
suit        0.143671
Name: price, dtype: float64
```

[ ]:
```
# Filtra as correlações positivas maiores que 0
correlacoes_positivas = correlation_price[correlation_price > 0]

# Seleciona as top 10 palavras positivas
top10_palavras_positivas = correlacoes_positivas.sort_values(ascending=False)
 .head(20).index.tolist()

# Cria o heatmap apenas com as top 10 palavras positivas
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix.loc[top10_palavras_positivas, ['price']],
 annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Heatmap de Correlação Positiva com o Preço (Top 10 Palavras)')
plt.show()
```

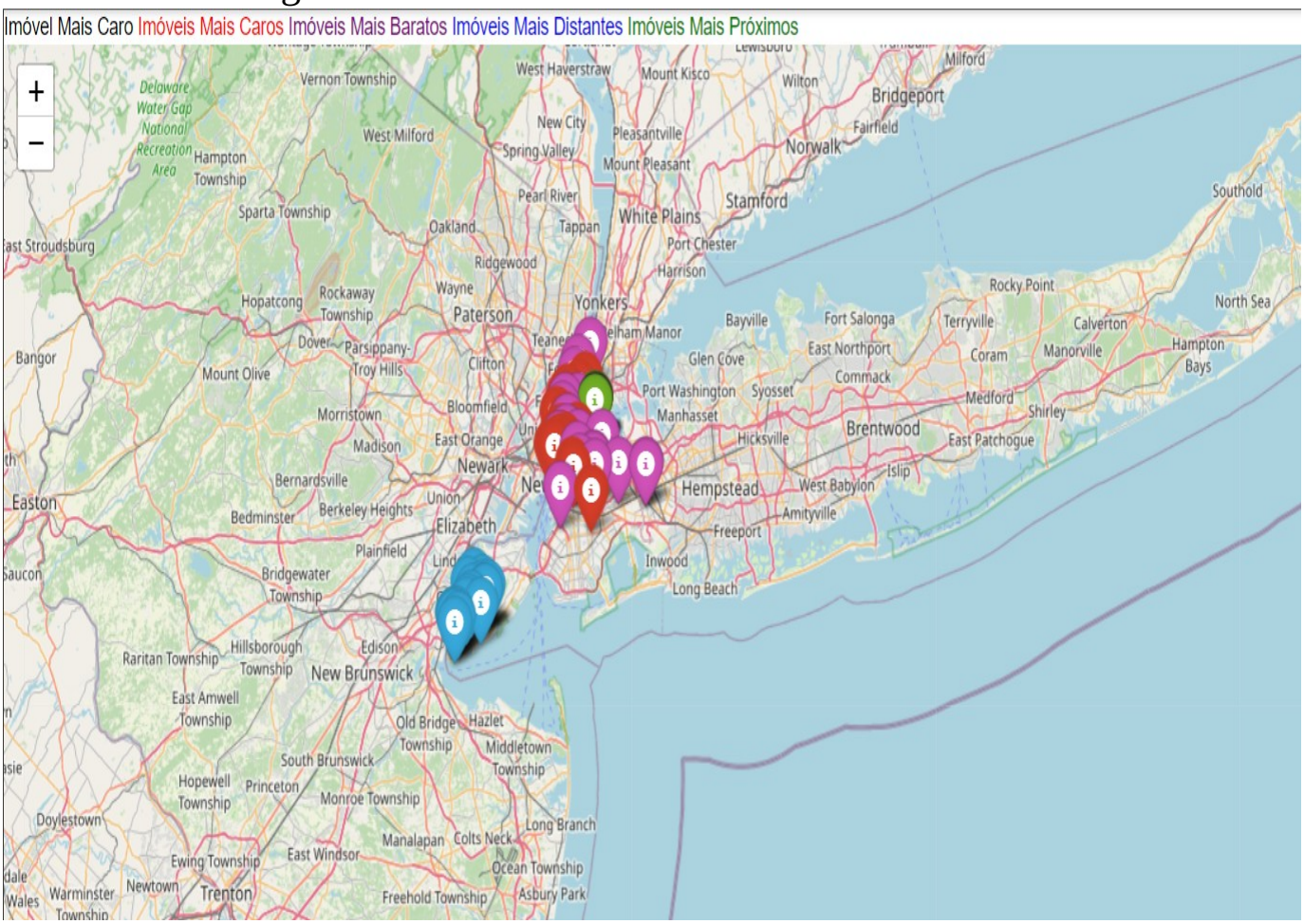


Observando as correlações com o preço dos textos mais frequentes nos anuncios de maior valor. Observamos que a maior parte das palavras são substantivos.

3.1.4 Latitude e Longitude.

[ ]:
```python
from math import radians, sin, cos, sqrt, atan2
from scipy.spatial import distance
# Cálculo da distância das latitudes
def calcular_distancia_geodesica(coord1, coord2):
    # Raio médio da Terra em quilômetros
    raio_terra_km = 6371.0

    # Converter coordenadas de graus para radianos
    lat1, lon1 = map(radians, coord1)
    lat2, lon2 = map(radians, coord2)

    # Diferenças de coordenadas
    dlat = lat2 - lat1
    dlon = lon2 - lon1

    # Fórmula de Haversine
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
```

Plot do mapa feito com o cálculo das distâncias de Latitude e longitude.



Análise das latitudes e longitudes

Foi utilizando as latitudes e longitudes fiz o cálculo da distância geodésica utilizando a fórmula de Haversine. anteriormente tinha feito o cálculo da distância euclidiana, mas está não poderia ser aplicada devido a curvatura da terra.

Não há uma aparente correlação das locais dos imóveis e o preço.

Acredito que se houvesse mais tempo poderia ser avaliado essas distâncias, feito correlações aplicado essas nos modelos.Notamos que o imóvel mais caro está cercado de imóveis muito baratos, há a possibilidade dele ser um erro talvez no preenchimento do campo do preço do preço. Para se ter uma melhor compreensão do caso deve ser investigado a distribuição dos bairros em NY, visto que a latitude e longitude muito próximas indica maior possibilidade de estar no mesmo bairro. Devido maior quantidade de features utilizadas na MLP foram de bairros deve ser analisado com mais cautela a como tratar os dados, fazer análise exploratória e melhorar o modelo.

Conclusão

No tratamento dos dados foi realizado inputs de dados nulos categóricos e numéricos cada um seguindo a recomendação para o caso em específico

Foi observado que alguns dados estavam foram de ordem no preenchimento o que pode indicar erro do operador ou usuário na hora de preencher os dados

Poderia ter realizado alguma função para correção dessas dados ao invés de remover como com a função de dropar outliers Além dos inputs de valores nulos foi realizado em valores iguais a zero que estavam errados Os dados não apresentaram no geral muitas correlações fortes, como a maior parte de correlações foram os bairros e estes como visto no mapa não é fator determinante de preço pois vários imóveis baratos estão no mesmo bairro dos imóveis mais caros. Foi destacado umas correlações não tão fortes de padrão textual com o preço do imóvel, essas features poderiam ter sido incorporadas ao nossos dados de treino para avaliação. Um feature que poderia ter sido tratada para ser usada foi a feature de ultima review, pois poderiamos com o tratamento correto verificar se os imóveis com avaliações atualizadas influem no preço do imóvel. A MLP do keras foi a que mais se adequou para fazer a previsão do nosso modelo.

Respostas do desafio

2. Responda também às seguintes perguntas:
   a. Supondo que uma pessoa esteja pensando em investir em um apartamento para alugar na plataforma, onde seria mais indicada a compra?
   - No Bairro de Midtown.
   b. O número mínimo de noites e a disponibilidade ao longo do ano interferem no preço?
   - A disponibilidade ao longo do ano sim, mas o número mínimo de noites não.
   c. Existe algum padrão no texto do nome do local para lugares de mais alto valor?
   - Sim, irei destacar alguns:
   - view que pode indicar lugares bem localizados com vista privilegiada.
   - soho que pode indicar que o imóvel pertence a Soho House que é uma rede hoteleira global.
   - downtown que é conhecida como baixa manhatthan e é conhecida por ser uma polo financeiro importante
   - penthouse que é o equivalente a cobertura em português.
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?
- Fiz a previsão dos escolhendo variáveis que apresentaram melhores coeficientes de correlação com o preço a maior parte delas foram o tipo de imóvel locado e os bairros, porém a disponibilidade , anúncios do imóvel e número de views contribuiram para nossa previsão.
- O problema enfrentado é de regressão pois queremos fazer a previsão de uma valor contínuo que é um preço e não apenas categorizar onde o mais indicado seria um classificação.
- O melhor modelo foi a MLP do keras e as medidas de performace foram a perda, mse e mae.
   - Perda: quantificando a discrepancia entre a previsão do modelo e o valor real.
   - MSE : Cálcula a média do quadrado das diferenças entre o previsto e o real é comumente utilizado em problemas de regressão.
   - MAE : calculá a média das diferenças absolutas entre o previsto e o real, é importante pois não trata todos os erros de maneira igual. é comumento usado em regressão.