Modelo

January 30, 2025

[9]:
```
 import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import OneHotEncoder, StandardScaler,␣
 ↪FunctionTransformer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.base import BaseEstimator, TransformerMixin
import pickle



# Carregar o dataset
df = pd.read_csv("teste_indicium_precificacao.csv")

# Remover colunas irrelevantes
df = df.drop(columns=['id', 'host_id', 'host_name', 'nome'])

# Tratar valores ausentes
df["reviews_por_mes"] = df["reviews_por_mes"].fillna(0)
df["ultima_review"] = pd.to_datetime(df["ultima_review"], errors="coerce")
df["dias_desde_ultima_review"] = (pd.Timestamp.now() - df["ultima_review"]).dt.
 ↪days
df["dias_desde_ultima_review"] = df["dias_desde_ultima_review"].
 ↪fillna(df["dias_desde_ultima_review"].median())

# Criar novas variáveis úteis
df["reviews_ajustados"] = df["numero_de_reviews"] * df["reviews_por_mes"]
df["preco_medio_bairro"] = df.groupby("bairro")["price"].transform("mean")
df["disponibilidade_relativa"] = df["disponibilidade_365"] / 365

# Definir features (X) e target (y)
X = df.drop(columns=["price"])
```

```
y = df["price"]

# Separar treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Identificar colunas categóricas e numéricas
categorical_features = ["room_type", "bairro_group", "bairro"]
numerical_features = ["latitude", "longitude", "numero_de_reviews",␣
 ↪"reviews_por_mes",
                      "calculado_host_listings_count", "preco_medio_bairro",␣
 ↪"reviews_ajustados",
                      "dias_desde_ultima_review", "disponibilidade_relativa"]

# Criar pré-processador
preprocessor = ColumnTransformer([
    ("num", Pipeline([
        ("imputer", SimpleImputer(strategy="median")),
        ("scaler", StandardScaler())
    ]), numerical_features),
    ("cat", OneHotEncoder(handle_unknown="ignore"), categorical_features)
])

# Usar Gradient Boosting Regressor
modelo = GradientBoostingRegressor(random_state=42)

# Pipeline completo
pipeline = Pipeline([
    ("preprocessador", preprocessor),
    ("feature_selection", SelectKBest(score_func=f_regression, k=15)),  #␣
 ↪Selecionar as melhores features
    ("regressor", modelo)
])

# Ajuste de hiperparâmetros com GridSearchCV
param_grid = {
    "regressor__n_estimators": [100, 200, 300],
    "regressor__learning_rate": [0.01, 0.1, 0.2],
    "regressor__max_depth": [3, 5, 7]
}

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring="r2", n_jobs=-1)
grid_search.fit(X_train, y_train)

# Melhor modelo encontrado
best_model = grid_search.best_estimator_
```

```
# Fazer previsões no conjunto de teste
y_pred = best_model.predict(X_test)


# Salvar o modelo
with open('modelo.pkl', 'wb') as file:
    pickle.dump(best_model, file)
```

[8]:
```
# Carregar o modelo
with open('modelo.pkl', 'rb') as file:
    modelo_carregado = pickle.load(file)

# Dados do novo apartamento
novo_apartamento = {
    "room_type": "Entire home/apt",
    "bairro_group": "Manhattan",
    "bairro": "Midtown",
    "latitude": 40.7539,
    "longitude": -73.9834,
    "numero_de_reviews": 25,
    "reviews_por_mes": 2.5,
    "calculado_host_listings_count": 1,
    "disponibilidade_365": 120,
    "ultima_review": "2023-10-01",
}

# Converter para DataFrame e pré-processar
df_novo = pd.DataFrame([novo_apartamento])
df_novo["reviews_por_mes"] = df_novo["reviews_por_mes"].fillna(0)
df_novo["ultima_review"] = pd.to_datetime(df_novo["ultima_review"],␣
 ↪errors="coerce")
df_novo["dias_desde_ultima_review"] = (pd.Timestamp.now() -␣
 ↪df_novo["ultima_review"]).dt.days.fillna(365)
df_novo["reviews_ajustados"] = df_novo["numero_de_reviews"] *␣
 ↪df_novo["reviews_por_mes"]
df_novo["preco_medio_bairro"] = 150  # Substitua pelo valor real do seu dataset
df_novo["disponibilidade_relativa"] = df_novo["disponibilidade_365"] / 365
df_novo = df_novo.drop(columns=["ultima_review"])

# Fazer a predição
preco_previsto = modelo_carregado.predict(df_novo)
print(f"Preço previsto: ${preco_previsto[0]:.2f}")
```

```
Preço previsto: $220.26
```

[ ]: