



Tecnológico
de Monterrey

Analítica de Datos y
Herramientas de Inteligencia Artificial

Reporte de actividad 4.1

Profesor: Alfredo García Suárez

Bernardo Quintana López | A01658064

Campus Puebla

27 de abril de 2025

Para comenzar, cargué mi base de datos de Airbnb de la ciudad de Londres que estoy analizando ya limpia, libre de valores nulos y atípicos con los métodos vistos anteriormente. Después importé todas las librerías necesarias para el análisis.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as special
from scipy.optimize import curve_fit
import seaborn as sns
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

```

✓ 0.0s Python

Filtré las variables de tipo int64 y float64 que son las más óptimas para trabajar con los modelos, ya que son de naturaleza numérica.

Unnamed: 0												review_scores_checkin
id	host_id	host_total_listings_count	accommodates	bathrooms	bedrooms	beds	price	minimum_nights
0	5.622900e+04	216660	11.0	2.0	1.5	1.0	89.0	5.0	4.880000
1	6.297000e+04	336059	4.5	6.0	1.2	3.0	3.0	220.0	1.0	4.800000
2	6.394800e+04	216660	11.0	2.0	1.0	1.0	88.0	3.0	4.870000
3	6.677200e+04	216660	11.0	2.0	1.0	1.0	75.0	3.0	4.870000
4	6.919800e+04	345886	1.0	1.0	1.0	1.0	53.0	3.0	4.620000
...
95139	95139	1.307780e+18	445228166	1.0	5.0	1.2	2.0	2.0	236.0	2.0	...	4.794925
95140	95140	1.307790e+18	50878819	2.0	2.0	1.0	1.0	2.0	88.0	5.0	...	4.794925
95141	95141	1.307790e+18	131418248	4.5	6.0	1.2	1.0	1.0	93.0	1.0	...	4.794925
95142	95142	1.307790e+18	252350161	1.0	2.0	1.0	1.0	1.0	79.0	1.0	...	4.794925
95143	95143	1.307800e+18	71174892	1.0	2.0	1.0	1.0	1.0	170.0	7.0	...	4.794925

95144 rows × 37 columns Python

De las 50 variables que tenía disponibles escogí las 12 que me parecían más interesantes e importantes a evaluar en esta actividad. Las cuales fueron las siguientes:

```
variables_seleccionadas = [
    'accommodates',          # Capacidad de personas (impacta en demanda)
    'bathrooms',              # Número de baños (comodidad)
    'bedrooms',               # Número de habitaciones
    'price',                  # Precio (variable clave)
    'minimum_nights',         # Política de estancia mínima
    'availability_365',       # Disponibilidad anual (inversa a la demanda)
    'number_of_reviews',       # Popularidad del alojamiento
    'review_scores_rating',   # Calificación global (confianza)
    'calculated_host_listings_count', # Experiencia del anfitrión
    'reviews_per_month',      # Actividad reciente
    'host_response_rate',     # Tasa de respuesta (servicio)
    'host_acceptance_rate'   # Tasa de aceptación (flexibilidad)
]

df = df[variables_seleccionadas]
df

```

✓ 0.0s Python

Para saber los criterios que escogería para hacer las variables que escogería como dependientes en dicotómicas, analicé sus valores mínimos, máximos y promedios para poder separarlas en dos categorías. Por ejemplo, con accommodates al tener 3 de promedio, hice dos categorías, menor o igual a 3 sería asignado un 0, mientras que mayor a 3, un 1.

```

stats_df = pd.DataFrame({
    'Mínimo': df[variables_seleccionadas].min(),
    'Máximo': df[variables_seleccionadas].max(),
    'Promedio': df[variables_seleccionadas].mean()
})
stats_df

```

[37] ✓ 0.0s Python

	Mínimo	Máximo	Promedio
accommodates	1.00	7.00	2.997036
bathrooms	0.50	1.50	1.179199
bedrooms	0.00	3.00	1.457476
price	1.00	360.00	163.003516
minimum_nights	1.00	8.00	2.489600
availability_365	0.00	365.00	135.791758
number_of_reviews	0.00	45.00	7.790749
review_scores_rating	4.29	5.00	4.786062
calculated_host_listings_count	1.00	16.00	2.825877
reviews_per_month	0.01	2.16	0.710909
host_response_rate	0.80	1.00	0.962403
host_acceptance_rate	0.57	1.00	0.895707

Procedí a calcular los coeficientes de correlación entre las variables para analizar sus relaciones entre variables.

```

corr_factors = df.corr()
corr_factors

```

[33] ✓ 0.0s Python

Outputs are collapsed ...

```

corr_factors1 = abs(corr_factors)
corr_factors1

```

[34] ✓ 0.0s Python

Outputs are collapsed ...

```

fig, ax = plt.subplots(figsize=(20, 15))
cax = ax.matshow(corr_factors1, cmap="Blues")
fig.colorbar(cax)

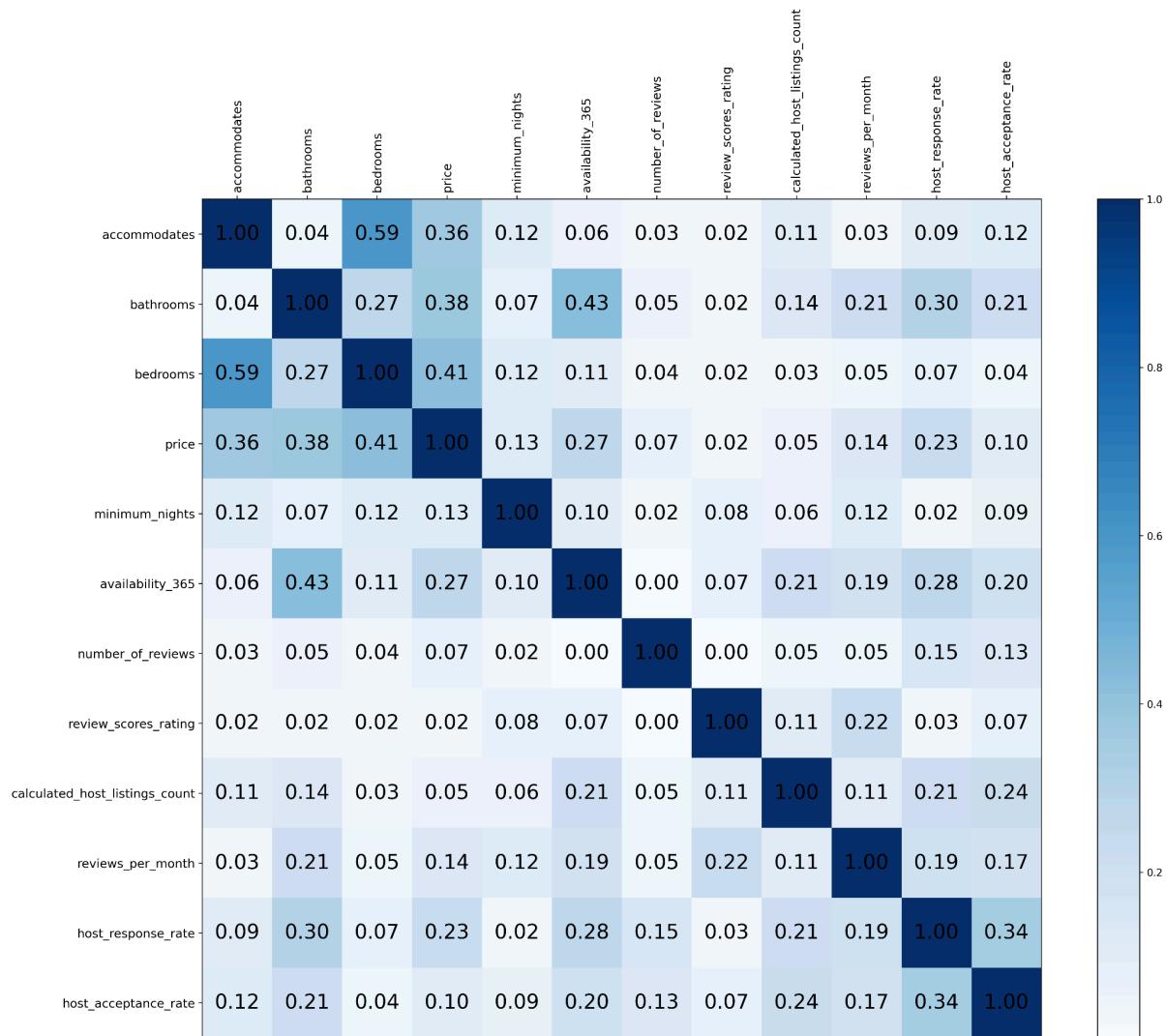
# Añadir anotaciones manualmente
for i in range(corr_factors1.shape[0]):
    for j in range(corr_factors1.shape[1]):
        ax.text(j, i, f'{corr_factors1.iloc[i, j]:.2f}',
                ha="center", va="center", fontsize=20)

plt.xticks(range(len(corr_factors1.columns)), corr_factors1.columns, rotation=90, fontsize=12)
plt.yticks(range(len(corr_factors1.index)), corr_factors1.index, fontsize=12)
plt.savefig('General.png', dpi=300, bbox_inches='tight')
plt.show()

```

✓ 1.3s Python

Decidí ilustrarlos en un heatmap para facilitar el análisis visual de las correlaciones. El heatmap resultante fue el siguiente:



Posteriormente, cargué las librerías y comandos necesarios para seguir con el modelaje y evaluación con las variables.

```

from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.linear_model import LogisticRegression

```

El procedimiento que realicé para cada modelo fue el siguiente:

1. Cree una copia del data frame original
2. En este nuevo data frame (df1) sustituía las variables que escogía como x con sus respectivos valores dicotómicos basados en los criterios que anteriormente he mencionado.
3. Aplicaba el código visto en clase para realizar el modelo de regresión logística
4. Definía las variables x y y, en y ponía las variables que mostraban una mayor relación con la variable objetivo.

5. Despu s, utilizaba los comandos importados para calcular la precisi n, exactitud y sensibilidad de cada modelo.

```
df1 = df.copy()
df1['accommodates'] = np.where(df1['accommodates'] <= 3, 0, 1)
✓ 0.0s Python

Vars_Indep = df1[['bedrooms','price','host_acceptance_rate']] #num rica
Var_Dep = df1['accommodates'] #dicot mica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predici n
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisi n del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

✓ 0.0s Python
```



```
df2 = df.copy()
df2['price'] = np.where(df2['price'] <= 163, 0, 1)
✓ 0.0s Python

Vars_Indep = df2[['accommodates','bathrooms','bedrooms']] #num rica
Var_Dep = df2['price'] #dicot mica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predici n
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisi n del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)
```

```

df3 = df.copy()
df3['bathrooms'] = np.where(df3['bathrooms'] <= 1, 0, 1)
✓ 0.0s Python

Vars_Indep = df3[['price','accommodates','bedrooms']] #numérica
Var_Dep = df3['bathrooms'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

```

```

df4 = df.copy()
df4['bedrooms'] = np.where(df4['bedrooms'] <= 1, 0, 1)

Vars_Indep = df4[['accommodates','bathrooms','price']] #numérica
Var_Dep = df4['bedrooms'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)
] ✓ 0.1s Python

```

```
df5 = df.copy()
df5['minimum_nights'] = np.where(df5['minimum_nights'] <= 2, 0, 1)

Vars_Indep = df5[['accommodates','bedrooms','price']] #numérica
Var_Dep = df5['minimum_nights'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

✓ 0.0s
```

Python

```
df6 = df.copy()
df6['availability_365'] = np.where(df6['availability_365'] <= 135, 0, 1)

Vars_Indep = df6[['bathrooms','price','host_response_rate']] #numérica
Var_Dep = df6['availability_365'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

✓ 0.0s
```

Python

```

df7 = df.copy()
df7['number_of_reviews'] = np.where(df7['number_of_reviews'] <= 5, 0, 1)

Vars_Indep = df7[['host_response_rate','host_acceptance_rate','price']] #numérica
Var_Dep = df7['number_of_reviews'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

✓ 0.0s

```

Python

```

df8 = df.copy()
df8['host_acceptance_rate'] = np.where(df8['host_acceptance_rate'] <= 0.75, 0, 1)

Vars_Indep = df8[['host_response_rate','calculated_host_listings_count','availability_365']] #numérica
Var_Dep = df8['host_acceptance_rate'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

✓ 0.0s

```

Python

```

df9 = df.copy()
df9['host_response_rate'] = np.where(df9['host_response_rate'] <= 0.95, 0, 1)

Vars_Indep = df9[['host_acceptance_rate','bathrooms','availability_365']] #numérica
Var_Dep = df9['host_response_rate'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

```



```

df10 = df.copy()
df10['reviews_per_month'] = np.where(df10['reviews_per_month'] <= 1, 0, 1)

Vars_Indep = df10[['review_scores_rating','bathrooms','availability_365']] #numérica
Var_Dep = df10['reviews_per_month'] #dicotómica

X = Vars_Indep
y = Var_Dep

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state =None)

escalar = StandardScaler()

X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train,y_train)
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

precision = precision_score(y_test,y_pred,average='binary')
print('Precisión del modelo:')
print(precision)

exactitud = accuracy_score(y_test,y_pred)
print('Exactitud del modelo:')
print(exactitud)

sensibilidad = recall_score(y_test,y_pred, average ='binary')
print('Sensibilidad del modelo:')
print(sensibilidad)

```

Python

Estos fueron los resultados obtenidos con los códigos y modelos realizados.

Modelo	Variable dicotómica (x)	Criterio dicotómica	Variables numéricas (y)	Precisión	Exactitud	Sensibilidad
1	accommodates	<= 3: 0 3>: 1	-bedrooms -price -host_acceptance_rate	0.80	0.83	0.69

2	price	<= 163: 0 163>: 1	-accommodates -bathrooms -bedrooms	0.79	0.78	0.81
3	bathrooms	<= 1: 0 1>: 1	-price -accommodates -bedrooms	0.78	0.77	0.82
4	bedrooms	<= 1: 0 1>: 1	-accommodates -bathrooms -price	0.74	0.76	0.79
5	minimum_nights	<= 2: 0 2>: 1	-accommodates -bedrooms -price	0.55	0.58	0.29
6	availability_365	<= 135: 0 135>: 1	-bathrooms -price -host_response_rate	0.63	0.67	0.60
7	number_of_reviews	<= 5: 0 5>: 1	-host_response_rate -host_acceptance_rate -price	0.57	0.61	0.58
8	hist_acceptance_rate	<= 0.75: 0 0.75>: 1	-host_acceptance_rate -calculated_host_listings_count -availability_365	0.93	0.94	1
9	host_respons e_rate	<= 0.95: 0 0.95>: 1	-host_acceptance_rate -bathrooms -availability_365	0.77	0.76	0.84
10	reviews_per_month	<= 1: 0 1>: 1	-review_scores_rating -bathrooms -availability_365	0.51	0.61	0.28

La tabla anterior presenta una comparativa de diez modelos de clasificación binaria, donde cada uno utiliza una variable dicotómica como predictor principal y un conjunto de variables

numéricas como soporte. Destaca que los modelos con variables como *host_acceptance_rate* y *accommodates* alcanzan precisiones superiores al 80%, lo que sugiere que estas características tienen un impacto significativo en la capacidad predictiva. Sin embargo, modelos basados en *minimum_nights* o *reviews_per_month* muestran precisiones inferiores al 60%, indicando que estas variables podrían no ser lo suficientemente discriminantes o estar afectadas por problemas como desbalance de clases o ruido en los datos. Esta disparidad en los resultados refleja la importancia de seleccionar variables con una relación clara con el fenómeno estudiado.

Un hallazgo relevante es el alto rendimiento del modelo 8, que utiliza *host_acceptance_rate* como variable dicotómica, alcanzando una precisión del 93% y una sensibilidad perfecta (1.0). Esto podría deberse a que la tasa de aceptación del anfitrión es un indicador directo de su disponibilidad y compromiso, factores críticos para predecir resultados en plataformas de hospedaje. Por el contrario, el modelo 5 (*minimum_nights*) y el modelo 10 (*reviews_per_month*) muestran sensibilidades particularmente bajas (0.29 y 0.28, respectivamente), lo que sugiere que estos modelos fallan en identificar correctamente los casos positivos, posiblemente por criterios de dicotomización poco óptimos o por la influencia de valores atípicos.

La exactitud y la precisión muestran tendencias similares en la mayoría de los modelos, lo que indica consistencia en las predicciones. No obstante, en casos como el modelo 1 (*accommodates*), la sensibilidad (0.69) es notablemente menor que la exactitud (0.83), revelando que el modelo tiene dificultades para captar instancias de la clase positiva. Esta divergencia podría apuntar a un sesgo hacia la clase mayoritaria, un problema común en conjuntos de datos desbalanceados.

Finalmente, los modelos evaluados evidencian la necesidad de un análisis más profundo sobre la interacción entre las variables predictoras y su relación con la variable objetivo. Por ejemplo, modelos que incluyen *price* o *bedrooms* como variables numéricas secundarias muestran resultados moderados. Además, la baja sensibilidad en algunos modelos sugiere que

la dicotomización de variables podría estar ocultando información relevante; en tales casos, enfoques alternativos, como regresión logística con variables continuas o modelos basados en árboles, podrían ofrecer mejores resultados. En conclusión, la actividad subraya la importancia de una selección rigurosa de variables y la evaluación integral de métricas para garantizar modelos robustos.