Tecnológico de Monterrey, Campus Puebla

Gestión de proyectos de plataformas tecnológicas (Gpo 201)

Profesor:
Alfredo Garcia Suarez

Reporte ejecutivo de regresiones logarítmicas

Equipo Skibidi:

Max Vidal Moreira	A01736949
Paula Simonetta Madrid Pérez	A01736976
José Manuel Morales Escalante	A01737201

ÍNDICE

INTRODUCCIÓN

El documento titulado **"M3 Actividad 3 (Regresión Logística)"** es un reporte ejecutivo realizado por el equipo **Skibidi**, conformado por Max Vidal Moreira, Paula Simonetta Madrid Pérez y José Manuel Morales Escalante, del curso de **Gestión de Proyectos de Plataformas Tecnológicas** en el **Tecnológico de Monterrey, Campus Puebla**. Bajo la guía del profesor Alfredo García Suárez, el equipo lleva a cabo un análisis de regresión logística utilizando diversas variables relacionadas con el mercado de plataformas de alquiler, evaluando su desempeño en ciudades como **México, Ámsterdam y Madrid**.

El reporte examina la precisión, exactitud y sensibilidad de distintos modelos predictivos en relación con características clave de los anfitriones, tales como la presencia de fotos de perfil, la verificación de identidad, y la disponibilidad de los listados. A lo largo del documento, se analizan los resultados obtenidos en cada ciudad, destacando las fortalezas y debilidades de cada modelo en función de sus predicciones y la calidad de sus métricas.

Este análisis proporciona una visión detallada del comportamiento de las plataformas de alquiler en diversas localidades, subrayando la importancia de ajustar los modelos predictivos para mejorar la precisión y optimizar la toma de decisiones basadas en datos.

México

Regresión logística

host is superhost

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host has profile pic como independiente, esta variable se utilizó el test size del 0.30 y la precisión del modelo fue baja en este caso ya que dio un 0.0551 y la exactitud fue del 0.062 y la sensibilidad del modelo fue de 0.26 al no ser cercana al 1 lo tenemos que recalcular para que nos de un mejor modelo.

```
precision del modelo:
0.5512987012987013
```

```
sensibilidad del modelo
0.26867088607594936
```

```
exactitud del modelo
0.6229116945107399
```

host has profile pic

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host has profile pic como independiente, en este caso tuvimos buenos resultados ya que la precisión del modelo es de 0.98 la exactitud fue de 0.98 y la sensibilidad fue 1.0 que significa es un gran modelo para usarse. por ende podemos predecir que las personas que son anfitrionas van a poner una foto de perfil

```
precision del modelo:
0.9831679437256626
```

```
exactitud del modelo
0.9831679437256626
```

```
sensibilidad del modelo
1.0
```

host identity verify

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host identity verify como independiente, en donde tuvimos una precisión del modelo 0.96 una exactitud del modelo del

0.96 y sensibilidad perfecta del 1.0 donde significa que podemos predecir que los anfitriones están registrados y verificados por la plataforma

```
sensibilidad del modelo
1.0
```

```
precision del modelo:
0.9609345559603065

exactitud del modelo
0.9609345559603065
```

Has availability

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y has availability como independiente, en donde tuvimos una precisión del modelo 0.99 una exactitud del modelo del 0.99 y sensibilidad perfecta del 1.0 esto predice que siempre tienen disponibilidad

```
[ ] #calcular precision del modelo
    from sklearn.metrics import precision_score
    precision=precision_score(y_test, y_pred, average="binary", pos_label="yes")
    print("precision del modelo:")
    print(precision)
```

```
precision del modelo:
0.998492651676925
```

```
#calcular exactitud
from sklearn.metrics import accuracy_score
exactitud=accuracy_score(y_test, y_pred)
print("exactitud del modelo")
print(exactitud)
```

```
exactitud del modelo
0.998492651676925
```

```
[ ] #calcular la sensibilidad
    from sklearn.metrics import recall_score

    sensibilidad=recall_score(y_test, y_pred, average="binary", pos_label="yes")
    print("sensibilidad del modelo")
    print(sensibilidad)
```

```
sensibilidad del modelo
1.0
```

categorías a partir de clase

```
exactitud del modelo
0.998492651676925
sensibilidad del modelo
1.0
```

Host acceptance rate

	host_acceptance_rate
0	Aceptacion alta
1	Aceptacion alta
2	Aceptacion alta
3	Aceptacion alta
4	Aceptacion alta
...	...
26531	Aceptacion alta
26532	Aceptacion alta
26533	Aceptacion alta
26534	Aceptacion alta
26535	Aceptacion alta

Para host acceptance rate se crearon dos categorías para volverla dicotómica (aceptación alta y aceptación baja)

Host response rate

	host_response_rate
0	Respuesta alta
1	NaN
2	Respuesta alta
3	Respuesta alta
4	Respuesta alta

Para host acceptance rate se crearon dos categorías para volverla dicotómica (respuesta alta y respuesta baja)

	host_total_listings_count
0	Aceptacion baja
1	NaN
2	Aceptacion baja
3	Aceptacion baja
4	Aceptacion baja
...	...
26531	Aceptacion baja

Host Total listing counts

Para Host Total listing counts se crearon dos categorías para volverla dicotómica (aceptación alta y aceptación baja)

	number_of_reviews
0	Reseñas bajas
1	NaN
2	Reseñas bajas
3	Reseñas altas
4	Reseñas bajas

Number of reviews

Para Number of reviews counts se crearon dos categorías para volverla dicotómica (reseñas alta y reseña baja)

	latitude
0	Latitud altas
1	Latitud bajas
2	Latitud altas
3	Latitud altas
4	Latitud altas

Latitud

Para Latitud se crearon dos categorías para volverla dicotómica (latitud alta y latitud baja)

Regresión logística

host acceptance rate

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host acceptance rate como independiente, en donde tuvimos una precisión del modelo 0.97 una exactitud del modelo del 0.97 y sensibilidad perfecta del 1.0 esto predice que tiene una aceptación alta en los establecimientos.

```
[ ] #calcular precision del modelo
    from sklearn.metrics import precision_score
    precision=precision_score(y_test, y_pred, average="binary", pos_label="Aceptacion alta")
    print("precision del modelo:")
    print(precision)
```

```
precision del modelo:
0.9741426737575647
```

```
#calcular exactitud
from sklearn.metrics import accuracy_score
exactitud=accuracy_score(y_test, y_pred)
print("exactitud del modelo")
print(exactitud)
```

```
exactitud del modelo
0.9741426737575647
```

```
[ ] from sklearn.metrics import recall_score

    sensibilidad=recall_score(y_test, y_pred, average="binary", pos_label="Aceptacion alta")
    print("sensibilidad del modelo")
    print(sensibilidad)
```

```
sensibilidad del modelo
1.0
```

host response rate

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host response rate como independiente, en donde tuvimos una precisión del modelo 0.98 una exactitud del modelo del 0.98 y sensibilidad perfecta del 1.0 esto predice que tiene una respuesta alta.

```
[ ] #calcular precision del modelo
    from sklearn.metrics import precision_score
    precision=precision_score(y_test, y_pred, average="binary", pos_label="Respuesta alta")
    print("precision del modelo:")
    print(precision)
```

```
precision del modelo:
0.9851457913075371
```

```
#calcular exactitud
from sklearn.metrics import accuracy_score
exactitud=accuracy_score(y_test, y_pred)
print("exactitud del modelo")
print(exactitud)
```

```
exactitud del modelo
0.9851457913075371
```

```
[ ] from sklearn.metrics import recall_score

    sensibilidad=recall_score(y_test, y_pred, average="binary", pos_label="Respuesta alta")
    print("sensibilidad del modelo")
    print(sensibilidad)
```

```
sensibilidad del modelo
1.0
```

host total listing counts

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host total listing counts como independiente, en donde tuvimos una precisión del modelo 0.96 una exactitud del modelo del 0.96 y sensibilidad perfecta del 1.0 esto predice que tiene una aceptación baja .

```
#calcular precision del modelo
from sklearn.metrics import precision_score
precision=precision_score(y_test, y_pred, average="binary", pos_label="Aceptacion baja")
print("precision del modelo:")
print(precision)
```

```
precision del modelo:
0.9677241885200807
```

```
#calcular exactitud
from sklearn.metrics import accuracy_score
exactitud=accuracy_score(y_test, y_pred)
print("exactitud del modelo")
print(exactitud)
```

```
exactitud del modelo
0.9677241885200807
```

```
from sklearn.metrics import recall_score

sensibilidad=recall_score(y_test, y_pred, average="binary", pos_label="Aceptacion baja")
print("sensibilidad del modelo")
print(sensibilidad)
```

```
sensibilidad del modelo
1.0
```

number of reviews

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y host total listing counts como independiente, en donde tuvimos una precisión del modelo 0.92 una exactitud del modelo del 0.92 y sensibilidad perfecta del 1.0 esto predice que tiene una reseñas bajas .

```
#calcular precision del modelo
from sklearn.metrics import precision_score
precision=precision_score(y_test, y_pred, average="binary", pos_label="Reseñas bajas")
print("precision del modelo:")
print(precision)
```

```
precision del modelo:
0.9216944801026957
```

```
#calcular exactitud
from sklearn.metrics import accuracy_score
exactitud=accuracy_score(y_test, y_pred)
print("exactitud del modelo")
print(exactitud)
```

```
exactitud del modelo
0.9216944801026957
```

```
from sklearn.metrics import recall_score

sensibilidad=recall_score(y_test, y_pred, average="binary", pos_label="Reseñas bajas")
print("sensibilidad del modelo")
print(sensibilidad)
```

```
sensibilidad del modelo
1.0
```

Latitud

Para el análisis y la regresión logística de este modelo se utilizaron las variables price, reviews per month y bedrooms como variables independientes y Latitud como independiente, en donde tuvimos una precisión del modelo 0.92 una exactitud del modelo del 0.92 y sensibilidad perfecta del 1.0 esto predice que tiene una latitud alta .

```
[ ] #calcular precision del modelo
    from sklearn.metrics import precision_score
    precision=precision_score(y_test, y_pred, average="binary", pos_label="Latitud altas")
    print("precision del modelo:")
    print(precision)
```

```
precision del modelo:
0.7177642501383509
```

```
#calcular exactitud
from sklearn.metrics import accuracy_score
exactitud=accuracy_score(y_test, y_pred)
print("exactitud del modelo")
print(exactitud)
```

```
exactitud del modelo
0.7155694113332111
```

```
[ ] from sklearn.metrics import recall_score

    sensibilidad=recall_score(y_test, y_pred, average="binary", pos_label="Latitud altas")
    print("sensibilidad del modelo")
    print(sensibilidad)
```

```
sensibilidad del modelo
0.9946319018404908
```

AMSTERDAM

Regresión logística

host is superhost

- Alto valor de exactitud: El modelo parece ser bastante bueno en general, ya que predice correctamente la clase de la mayoría de las instancias.
- Baja sensibilidad: A pesar de la alta exactitud, la sensibilidad es bastante baja. Esto sugiere que el modelo tiene dificultades para identificar las instancias positivas.
- Desequilibrio de clases: La baja sensibilidad podría indicar un desequilibrio en las clases de tu conjunto de datos. Si tiene muchas más instancias negativas que positivas, el modelo podría estar sesgado hacia la clase mayoritaria.

```
precision del modelo:
0.6906077348066298
```

```
exactitud del modelo
0.8471177944862155
```

```
sensibilidad del modelo
0.25201612903225806
```

host has profile pic

Este modelo muestra un rendimiento muy alto. Con una precisión y exactitud cercanas al 98.57%, y una sensibilidad perfecta (1.0), el modelo es muy efectivo para clasificar correctamente tanto las instancias positivas como las negativas, con especial énfasis en no perder ningún caso positivo.

```
precision del modelo:
0.9856784819190835
```

```
exactitud del modelo
0.9856784819190835
```

```
sensibilidad del modelo
1.0
```

host identity verify

Estos resultados sugieren que tu modelo está funcionando de manera excepcionalmente bien. La alta precisión, exactitud y sensibilidad indican que el modelo es muy confiable: Cuando el modelo hace una predicción, hay una alta probabilidad de que sea correcta. No está pasando por alto casos positivos: El modelo está identificando todas las instancias positivas, lo cual es crucial en muchos contextos.

```
precision del modelo:
0.9738632295023273
```

```
exactitud del modelo
0.9738632295023273
```

```
sensibilidad del modelo
1.0
```

Has availability

El modelo detectó todas las instancias donde había disponibilidad. Esto es un resultado excepcional y sugiere que el modelo es extremadamente bueno en encontrar los casos positivos.

```
precision del modelo:
0.9960615825277479
```

```
exactitud del modelo
0.9960615825277479
```

```
sensibilidad del modelo
1.0
```

categorías a partir de clase

Host acceptance rate

```
0       Aceptacion alta
1       Aceptacion alta
2       Aceptacion alta
3       Aceptacion alta
4       Aceptacion alta
            ...
9305    Aceptacion alta
9306    Aceptacion alta
9307    Aceptacion alta
9308    Aceptacion alta
9309    Aceptacion alta
```

Host response rate

```
0       Respuesta alta
1       Respuesta baja
2       Respuesta alta
3       Respuesta alta
4       Respuesta alta
            ...
9305    Respuesta alta
9306    Respuesta alta
9307    Respuesta alta
9308    Respuesta alta
9309    Respuesta alta
```

Host Total listing counts

```
0       Aceptacion baja
1       Aceptacion baja
2       Aceptacion baja
3       Aceptacion baja
4       Aceptacion baja
            ...
9305    Aceptacion baja
9306    Aceptacion baja
9307    Aceptacion baja
9308    Aceptacion baja
9309    Aceptacion baja
```

Number of reviews

```
  0       Reseñas bajas
  1       Reseñas bajas
  2       Reseñas bajas
  3       Reseñas bajas
  4       Reseñas bajas
              ...
  9305    Reseñas bajas
  9306    Reseñas bajas
  9307    Reseñas bajas
  9308    Reseñas bajas
  9309    Reseñas bajas
```

Latitud

```
0       Latitud bajas
1       Latitud bajas
2       Latitud bajas
3       Latitud bajas
4       Latitud bajas
            ...
9305    Latitud bajas
9306    Latitud bajas
9307    Latitud altas
9308    Latitud bajas
9309    Latitud bajas
```

Regresión logística

host acceptance rate

El modelo predice correctamente si un host aceptará una solicitud en el 77.8% de los casos. Además, identifica el 100% de las solicitudes que son aceptadas, lo que significa que no está pasando por alto ninguna oportunidad. Estos resultados indican que el modelo es muy confiable para predecir la aceptación de hosts.

```
precision del modelo:
0.7780164697457931
```

```
exactitud del modelo
0.7780164697457931
```

```
sensibilidad del modelo
1.0
```

host response rate

El modelo predice correctamente si un host responderá a una solicitud en el 96.8% de los casos. Además, identifica el 100% de las solicitudes que reciben respuesta, lo que significa que no está pasando por alto ninguna oportunidad. Estos resultados indican que el modelo es muy confiable para predecir la respuesta de los hosts.

```
precision del modelo:
0.9684926602219835
```

```
exactitud del modelo
0.9684926602219835
```

```
sensibilidad del modelo
1.0
```

host total listing counts

El modelo está fallando en predecir el conteo total de listados. A pesar de una alta exactitud general (99.7%), la precisión y sensibilidad son nulas, lo que indica que el modelo probablemente está haciendo predicciones constantes y erróneas. Esto sugiere que el modelo no ha aprendido la relación entre los datos de entrada y el conteo de listados, y por lo tanto, no es útil para este propósito. Es necesario revisar la calidad de los datos, la complejidad del modelo y las métricas de evaluación para mejorar su desempeño.

```
precision del modelo:
0.0
```

```
exactitud del modelo
0.9974937343358395
```

```
sensibilidad del modelo
0.0
```

number of reviews

El modelo ha fallado en predecir el número de reseñas. A pesar de mostrar una alta exactitud general (95.7%), la precisión y sensibilidad son nulas, indicando que el modelo probablemente está haciendo predicciones constantes y erróneas. Esto sugiere que el modelo no ha aprendido la relación entre los datos de entrada y el número de reseñas, lo que lo vuelve inefectivo para esta tarea. Es necesario revisar la calidad de los datos, la complejidad del modelo y las métricas de evaluación para mejorar su desempeño.

```
precision del modelo:
0.0
```

```
exactitud del modelo
0.9570354457572503
```

```
sensibilidad del modelo
0.0
```

Latitude

El modelo está fallando en predecir la latitud. A pesar de mostrar una alta exactitud general (96.5%), la precisión y sensibilidad son nulas, lo que indica que el modelo probablemente está haciendo predicciones constantes y erróneas. Esto sugiere que el modelo no ha aprendido la relación entre los datos de entrada y la latitud, volviéndolo inútil para esta tarea. Es necesario revisar la calidad de los datos, la complejidad del modelo y las métricas de evaluación para mejorar su desempeño.

```
precision del modelo:
0.0
```

```
exactitud del modelo
0.9659863945578231
```

```
sensibilidad del modelo
0.0
```

Madrid

Modelo 1 host is superhost

```
#Declarar las variables independientes y la dependiente
Vars_Indep=df[["price", "reviews_per_month", "bedrooms"]]
Var_Dep=df["host_is_superhost"]
X=Vars_Indep
y=Var_Dep

#Dividimos el conjunto de datos
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

#Se escalan los datos
escalar=StandardScaler()
X_train=escalar.fit_transform(X_train)
X_test=escalar.fit_transform(X_test)

#definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo=LogisticRegression()

#Entrenar el modelo
algoritmo.fit(X_train, y_train)

y_pred=algoritmo.predict(X_test)
from sklearn.metrics import confusion_matrix
matriz=confusion_matrix(y_test, y_pred)
print("Matriz de confusión")
print(matriz)
```

```
Matriz de confusión
[[6060  101]
 [1843   74]]
```

La **precisión** es de **0.4229**, lo que significa que cuando el modelo predijo un resultado positivo, solo el **42.29%** de esas predicciones fueron correctas. Con una exactitud de **0.7593**, el modelo predice correctamente el **75.93%** de las veces.Un valor de en sensibilidad **0.0386** significa que el modelo solo identifica correctamente el **3.86%** de los casos positivos.

Modelo 2 host has profile pic

```
#Declarar las variables independientes y la dependiente
Vars_Indep=df[["price", "reviews_per_month", "bedrooms"]]
Var_Dep=df["host_has_profile_pic"]
X=Vars_Indep
y=Var_Dep

#Dividimos el conjunto de datos
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

#Se escalan los datos
escalar=StandardScaler()
X_train=escalar.fit_transform(X_train)
X_test=escalar.fit_transform(X_test)

#definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo=LogisticRegression()

#Entrenar el modelo
algoritmo.fit(X_train, y_train)

y_pred=algoritmo.predict(X_test)
from sklearn.metrics import confusion_matrix
matriz=confusion_matrix(y_test, y_pred)
print("Matriz de confusión")
print(matriz)
```

```
Matriz de confusión
[[   0  221]
 [   0 7857]]
```

Un valor en precisión de **0.9726** significa que el **97.26%** de las predicciones positivas del modelo fueron correctas.La **exactitud** de **0.9726** significa que el modelo hizo predicciones correctas el **97.26%** de las veces.Esta es una exactitud bastante alta.La sensibilidad de **1.0** significa que el modelo identificó **todos los casos positivos** sin ningún error.

Modelo 3 host identity verified

```
[13] #Declarar las variables independientes y la dependiente
     Vars_Indep=df[["price", "reviews_per_month", "bedrooms"]]
     Var_Dep=df["host_identity_verified"]
     X=Vars_Indep
     y=Var_Dep

     #Dividimos el conjunto de datos
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

     #Se escalan los datos
     escalar=StandardScaler()
     X_train=escalar.fit_transform(X_train)
     X_test=escalar.fit_transform(X_test)

     #definimos el algoritmo a utilizar
     from sklearn.linear_model import LogisticRegression
     algoritmo=LogisticRegression()

     #Entrenar el modelo
     algoritmo.fit(X_train, y_train)

     y_pred=algoritmo.predict(X_test)
     from sklearn.metrics import confusion_matrix
     matriz=confusion_matrix(y_test, y_pred)
     print("Matriz de confusión")
     print(matriz)

     Matriz de confusión
     [[   0  660]
      [   0 7418]]
```

Un valor en precisión de 0.**9183** significa que el **91.83%** de las predicciones positivas del modelo fueron correctas. La exactitud de 0.**9183** significa que el modelo hizo predicciones correctas el **91.83%** de las veces, lo que indica un buen desempeño. La sensibilidad de **1.0** significa que el modelo identificó todos los casos positivos sin ningún error.

Modelo 4 has availability

```
[15] #Declarar las variables independientes y la dependiente
     Vars_Indep=df[["price", "reviews_per_month", "bedrooms"]]
     Var_Dep=df["has_availability"]
     X=Vars_Indep
     y=Var_Dep

     #Dividimos el conjunto de datos
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

     #Se escalan los datos
     escalar=StandardScaler()
     X_train=escalar.fit_transform(X_train)
     X_test=escalar.fit_transform(X_test)

     #definimos el algoritmo a utilizar
     from sklearn.linear_model import LogisticRegression
     algoritmo=LogisticRegression()

     #Entrenar el modelo
     algoritmo.fit(X_train, y_train)

     y_pred=algoritmo.predict(X_test)
     from sklearn.metrics import confusion_matrix
     matriz=confusion_matrix(y_test, y_pred)
     print("Matriz de confusión")
     print(matriz)
```

```
Matriz de confusión
[[   0   35]
 [   0 8043]]
```

Un valor en precisión de **0.9957** significa que el **99.57%** de las predicciones positivas del modelo fueron correctas. La exactitud de 0.**9957** indica que el modelo hizo predicciones correctas el **99.57**% de las veces, mostrando un desempeño excelente. La sensibilidad de **1.0** significa que el modelo identificó todos los casos positivos sin ningún error.

Variables convertidas a dicotómicas:

host acceptance rate

```
#Declarar las variables independientes y la dependiente
Vars_Indep=df[["price", "accommodates", "bedrooms"]]
Var_Dep=df["host_acceptance_rate"]
X=Vars_Indep
y=Var_Dep

#Dividimos el conjunto de datos
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

#Se escalan los datos
escalar=StandardScaler()
X_train=escalar.fit_transform(X_train)
X_test=escalar.fit_transform(X_test)

#definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo=LogisticRegression()

#Entrenar el modelo
algoritmo.fit(X_train, y_train)

y_pred=algoritmo.predict(X_test)
from sklearn.metrics import confusion_matrix
matriz=confusion_matrix(y_test, y_pred)
print("Matriz de confusión")
print(matriz)
```

```
Matriz de confusión
[[7123    0]
 [ 955    0]]
```

Un valor en precisión de **0.8818** significa que el **88.18%** de las predicciones positivas del modelo fueron correctas. La exactitud de **0.8818** indica que el modelo hizo predicciones correctas el **88.18%** de las veces, lo que muestra un buen rendimiento general. La sensibilidad de **1.0** significa que el modelo identificó todos los casos positivos sin errores.

host response rate

```
[36] #Declarar las variables independientes y la dependiente
     Vars_Indep=df[["price", "accommodates", "bedrooms"]]
     Var_Dep=df["host_response_rate"]
     X=Vars_Indep
     y=Var_Dep

     #Dividimos el conjunto de datos
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

     #Se escalan los datos
     escalar=StandardScaler()
     X_train=escalar.fit_transform(X_train)
     X_test=escalar.fit_transform(X_test)

     #definimos el algoritmo a utilizar
     from sklearn.linear_model import LogisticRegression
     algoritmo=LogisticRegression()

     #Entrenar el modelo
     algoritmo.fit(X_train, y_train)

     y_pred=algoritmo.predict(X_test)
     from sklearn.metrics import confusion_matrix
     matriz=confusion_matrix(y_test, y_pred)
     print("Matriz de confusión")
     print(matriz)

Matriz de confusión
[[7738    0]
 [ 340    0]]
```

Un valor en precisión de **0.9579** significa que el **95.79%** de las predicciones positivas del modelo fueron correctas. La exactitud de **0.9579** indica que el modelo hizo predicciones correctas el **95.79%** de las veces, mostrando un rendimiento alto. La sensibilidad de **1.0** significa que el modelo identificó todos los casos positivos sin ningún error.

host total listings count

```
[40] #Declarar las variables independientes y la dependiente
     Vars_Indep=df[["price", "accommodates", "bedrooms"]]
     Var_Dep=df["host_total_listings_count"]
     X=Vars_Indep
     y=Var_Dep

     #Dividimos el conjunto de datos
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

     #Se escalan los datos
     escalar=StandardScaler()
     X_train=escalar.fit_transform(X_train)
     X_test=escalar.fit_transform(X_test)

     #definimos el algoritmo a utilizar
     from sklearn.linear_model import LogisticRegression
     algoritmo=LogisticRegression()

     #Entrenar el modelo
     algoritmo.fit(X_train, y_train)

     y_pred=algoritmo.predict(X_test)
     from sklearn.metrics import confusion_matrix
     matriz=confusion_matrix(y_test, y_pred)
     print("Matriz de confusión")
     print(matriz)

Matriz de confusión
[[   0  336]
 [   0 7742]]
```

Un valor en precisión de **0.0** significa que ninguna de las predicciones positivas del modelo fue correcta, es decir, todas las predicciones positivas fueron erróneas. A pesar de esto, la exactitud de **0.9584** indica que el modelo hizo predicciones correctas el **95.84**% de las veces, lo que sugiere que el modelo está acertando mayormente en la clase negativa. La sensibilidad de **0.0** significa que el modelo no identificó correctamente ningún caso positivo. Esto indica un problema grave en la detección de la clase positiva.

number of reviews

```
[44] #Declarar las variables independientes y la dependiente
     Vars_Indep=df[["price", "accommodates", "bedrooms"]]
     Var_Dep=df["number_of_reviews"]
     X=Vars_Indep
     y=Var_Dep

     #Dividimos el conjunto de datos
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

     #Se escalan los datos
     escalar=StandardScaler()
     X_train=escalar.fit_transform(X_train)
     X_test=escalar.fit_transform(X_test)

     #definimos el algoritmo a utilizar
     from sklearn.linear_model import LogisticRegression
     algoritmo=LogisticRegression()

     #Entrenar el modelo
     algoritmo.fit(X_train, y_train)

     y_pred=algoritmo.predict(X_test)
     from sklearn.metrics import confusion_matrix
     matriz=confusion_matrix(y_test, y_pred)
     print("Matriz de confusión")
     print(matriz)

Matriz de confusión
[[   0  478]
 [   0 7600]]
```

Un valor en precisión de **0.0** significa que ninguna de las predicciones positivas del modelo fue correcta, es decir, todas las predicciones positivas fueron erróneas. A pesar de esto, la exactitud de **0.9408** indica que el modelo hizo predicciones correctas el **94.08%** de las veces, lo que sugiere que está acertando mayormente en la clase negativa. La sensibilidad de **0.0** significa que el modelo no identificó correctamente ningún caso positivo, lo que señala una incapacidad significativa para detectar la clase positiva. Esto indica que el modelo tiene un rendimiento deficiente en la identificación de casos positivos.

latitude

```
[48] #Declarar las variables independientes y la dependiente
     Vars_Indep=df[["price", "accommodates", "bedrooms"]]
     Var_Dep=df["latitude"]
     X=Vars_Indep
     y=Var_Dep

     #Dividimos el conjunto de datos
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state=None)

     #Se escalan los datos
     escalar=StandardScaler()
     X_train=escalar.fit_transform(X_train)
     X_test=escalar.fit_transform(X_test)

     #definimos el algoritmo a utilizar
     from sklearn.linear_model import LogisticRegression
     algoritmo=LogisticRegression()

     #Entrenar el modelo
     algoritmo.fit(X_train, y_train)

     y_pred=algoritmo.predict(X_test)
     from sklearn.metrics import confusion_matrix
     matriz=confusion_matrix(y_test, y_pred)
     print("Matriz de confusión")
     print(matriz)

     Matriz de confusión
     [[1663 2382]
      [1395 2638]]
```

Un valor en precisión de **0.5438** significa que el **54.38%** de las predicciones positivas del modelo fueron correctas. La exactitud de **0.5324** indica que el modelo hizo predicciones correctas el **53.24%** de las veces, lo que sugiere un rendimiento moderado. La sensibilidad de **0.4111** significa que el modelo identificó correctamente el **41.11%** de los casos positivos, lo que indica que tiene dificultades para detectar la clase positiva. En conjunto, estas métricas sugieren que el modelo podría beneficiarse de mejoras significativas para optimizar su capacidad de predicción.

.

	Precisión
	Exactitud
	Sensibilidad

	Madrid			México			Amsterdam		
	P	E	S	P	E	S	P	E	S
host_is_superhost	0.42	0.75	0.03	0.55	0.26	0.62	0.69	0.84	0.25
host_has_profile_pic	0.97	0.97	1.0	0.98	0.98	1.0	0.98	0.98	1.0
host_identity_verified	0.91	0.91	1.0	0.96	0.96	1.0	0.97	0.97	1.0
has_availability	0.99	0.99	1.0	0.99	0.99	1.0	0.97	0.97	1.0
instant_bookable	0.60	0.62	1.0	0.58	0.60	1.0	0.99	0.99	1.0
host_acceptance_rate	0.88	0.88	1.0	0.97	0.97	1.0	0.77	0.77	1.0
host_response_rate	0.95	0.95	1.0	0.98	0.98	1.0	0.96	0.96	1.0
host_total_listings_count	0	0.95	0	0.96	0.96	1.0	0.0	0.99	0.0
number_of_reviews	0	0.94	0	0.92	0.92	1.0	0.0	0.95	0.0
latitude	0.54	0.53	0.41	0.71	0.71	0.99	0.0	0.96	0.0

Conclusión:

La tabla presentada parece mostrar los resultados de un modelo de regresión logística aplicado a un conjunto de datos relacionado con plataformas de alquiler de propiedades como Airbnb. Las columnas "P", "E" y "S" probablemente corresponden a Precisión, Exactitud y Sensibilidad, respectivamente, métricas comúnmente utilizadas para evaluar el desempeño de modelos de clasificación. Las filas representan diferentes características de los anfitriones o propiedades (por ejemplo, si el anfitrión es un superhost, si tiene una foto de perfil, etc.) y los valores numéricos indican la capacidad del modelo para predecir correctamente estas características en las ciudades de Madrid, México y Amsterdam.

Aunque el desempeño general es alto, se observan algunas diferencias notables entre las ciudades. Por ejemplo, en la característica "host_is_superhost", el modelo parece ser menos preciso en Madrid que en México o Amsterdam. Esto podría indicar diferencias en los patrones de comportamiento de los anfitriones o en la calidad de los datos en cada ciudad.
Algunas características parecen ser mejores predictores que otras. Por ejemplo, "host_has_profile_pic", "host_identity_verified", "has_availability", e "instant_bookable" muestran valores muy altos de precisión, exactitud y sensibilidad en todas las ciudades, lo que sugiere que estas características son muy importantes para predecir el resultado.
Las características "host_total_listings_count" y "number_of_reviews" muestran valores bajos o nulos de precisión, exactitud y sensibilidad en algunas ciudades. Esto podría deberse a varias razones, como la falta de variabilidad en estos datos, la forma en que se han codificado o la naturaleza misma de estos datos en relación con las otras características.