Tecnológico de Monterrey

Campus Puebla

Curso:

Gestión de proyectos de plataformas tecnológicas (Gpo 201)

Actividad:

Actividad 3 (Regresión Logística)

Elaborado Por:

Samantha Mayrin Martínez Balbuena A01733837

Fecha:

11/10/2025

Introducción

Durante este reporte vamos a poder observar la metodología utilizada, los resultados y hallazgos respecto a la base de datos de airbnb en Creta, durante este análisis se utilizo la regresión logística, dos métodos de balanceo y 1 método para convertir variables de más de un tipo de resultado en variables dicotómicas.

Metodología

Librerías utilizadas:

```
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
```

Carga de archivo:

```
df= pd.read_csv("datoslimpioscreta.csv")
df.head()
```

Buscamos únicamente las columnas numéricas para usar en “x”:

```
df.select_dtypes(include=['number']).columns
```

Seleccionamos las columnas dicotómicas,debido a que solo contamos con 5 variables dicotómicas naturalmente, deberemos modificar algunas variables:

```
{col: df[col].unique() for col in dicotomicas}
```

Buscamos las 4 variables más correlacionadas con nuestra variable x, para seleccionar las 3 mejores, debido a que en algunas variables la variable que mejor se correlaciona es con la misma x :

```
dicotomicas = ['accommodates',
'host_acceptance_rate','reviews_per_month','availability_30','price','s
ource', 'host_has_profile_pic', 'host_identity_verified',
'instant_bookable', 'host_is_superhost']
```

Ejemplo:

	Variable Dicotómica	Variable Numérica	Correlación
0	accommodates	accommodates	1.000

Después de todos los pasos previos, llegamos al **primer caso,** donde declaramos nuestras variables:

```
Vars_Indep= df[['availability_eoy',
'availability_365','availability_90']]
Var_Dep= df['source']
```

La matriz de confusión nos muestra que hay una variable que no se está prediciendo de manera correcta una de las variables, lo que nos exige un balanceo

```
Matriz de Confusión:
[[7011    0]
 [ 839    0]]
```

Además el cálculo de la precisión del modelo para city scrape es: 0.893
Y la sensibilidad del modelo es: 1.0
Mientras que para previous scrape es: 0.0
Y la sensibilidad del modelo es: 0.0

Dicho balanceo nos lleva al **segundo caso,** donde utilizamos el método 1 para *source*, ponderación de clases, donde la matriz de confusión nos arroja los siguientes resultados

```
Matriz de Confusión:
[[5283 1728]
 [ 250  589]]
```

Ahora que está balanceado obtenemos diferentes resultados tanto para city scrape como para previous scrape en cuanto a precisión y sensibilidad del modelo:

Precisión del modelo previous scrape: 0.254
Sensibilidad del modelo previous scrape: 0.702
Precisión del modelo city scrape: 0.954
Sensibilidad del modelo city scrape: 0.753

city scrape (nuevos/recientes, más variación): el modelo los identifica con mucha precisión (0.954) y buen recall (0.753). Es decir, cuando predice “city” casi siempre acierta, aunque todavía se le escapan algunos casos.

previous scrape (continuidad/menor rotación): el modelo recupera varios (recall 0.702) pero se equivoca mucho al afirmarlos (precisión 0.254), lo que indica muchos falsos positivos.

En cuanto al **tercer caso,** las variables utilizadas fueron las siguientes:

```
Vars_Indep= df[['maximum_maximum_nights',
'maximum_nights_avg_ntm','maximum_nights']]
Var_Dep= df['host_has_profile_pic']
```

La matriz de confusión nos muestra que hay una variable que no se está prediciendo de manera correcta una de las variables, lo que nos exige un balanceo

```
Matriz de Confusión:
[[   0  335]
 [   0 7515]]
```

Precisión del modelo t:
0.9573248407643312
Precisión del modelo f:
0.0
Exactitud del modelo:
0.9573248407643312
Sensibilidad del modelo t:
1.0
Sensibilidad del modelo f:
0.0

Dicho balanceo nos lleva al **cuarto caso,** donde utilizamos el método 2 para *host has a profile pic*, sobremuestreo, donde la matriz de confusión nos arroja los siguientes resultados

```
Matriz de Confusión:
[[ 283   52]
 [3477 4038]]
```

Ahora que está balanceado obtenemos diferentes resultados tanto para t como para f en cuanto a precisión y sensibilidad del modelo:

Precisión del modelo t:
0.9872860635696822
Precisión del modelo f:
0.07526595744680852
Exactitud del modelo:
0.5504458598726115
Sensibilidad del modelo t:
0.5373253493013972
Sensibilidad del modelo f:

0.844776119402985
Para el **quinto caso** se usaron las siguientes variables:

```
Vars_Indep= df[['price', 'host_is_superhost','minimum_maximum_nights']]
Var_Dep= df['host_identity_verified']
```

Al igual que en los casos anteriores es necesario hacer un balance

```
Matriz de Confusión:
[[   0  166]
 [   7 7677]]
```

Precisión del modelo t:
0.9817741524343615
Precisión del modelo f:
0.0
Exactitud del modelo:
0.9812738853503185
Sensibilidad del modelo t:
0.9994809913066044
Sensibilidad del modelo f :
0.0

Dicho balance nos lleva al **sexto caso** para el que se usó el método 1 de balanceo, reponderación de clases, donde los nuevos resultados de la matriz de confusión son:

```
Matriz de Confusión:
[[  88   55]
 [2403 5304]]
```

Precisión del modelo:
0.9897368912110468
Precisión del modelo:
0.03532717784022481
Exactitud del modelo:
0.6868789808917197
Sensibilidad del modelo:
0.6882055274425847
Sensibilidad del modelo:
0.6153846153846154

A partir del **séptimo caso** no se necesitan balanceos, para este caso se utilizaron las siguientes variables:

```
Vars_Indep= df[['host_acceptance_rate',
'maximum_maximum_nights','maximum_nights_avg_ntm']]
Var_Dep= df['instant_bookable']
```

```
Matriz de Confusión:
[[ 775 2101]
 [ 170 4804]]
```

Precisión del modelo t:
0.6957277335264301
Precisión del modelo f:
0.8201058201058201
Exactitud del modelo:
0.7107006369426752
Sensibilidad del modelo t:
0.9658222758343386
Sensibilidad del modelo f:
0.269471488178025

A partir del **octavo caso** se presentaron diversas dificultades, entre ellas el tipo de caracter, pues al ser numérico no lo registraba en las líneas del código, sin embargo al cambiarlo a string funcionó correctamente y se pudieron obtener los modelos de precisión y sensibilidad.

Las variables usadas para dicho caso fueron las siguientes:

```
Vars_Indep= df[['estimated_occupancy_l365d',
'number_of_reviews_ly','number_of_reviews_ltm']]
Var_Dep= df['host_is_superhost']
```

```
Matriz de Confusión:
[[4236  438]
 [1807 1369]]
```

Precisión del modelo 0:
0.7009763362568261
Precisión del modelo 1:
0.7576092971776425
Exactitud del modelo:
0.7140127388535031
Sensibilidad del modelo 0:
0.9062901155327343
Sensibilidad del modelo 1:
0.4310453400503778

En el **noveno caso** se tuvo que comenzar la creación de categorías a partir de clases, en este caso fue necesario sacar la media para declarar los intervalos, debido a que había outliers y solo se registraba una de las categorías creadas. Para este caso las variables utilizadas fueron:

```
Vars_Indep= df[['accommodates', 'beds','bedrooms']]
Var_Dep= df['price']
```

```
Matriz de Confusión:
[[6258  191]
 [ 871  530]]
```

Precisión del modelo Cheap:
0.8778229765745547
Precisión del modelo Expensive:
0.7350901525658807
Exactitud del modelo:
0.8647133757961784
Sensibilidad del modelo Cheap:
0.9703830051170724
Sensibilidad del modelo Expensive:
0.3783012134189864

En cuanto al **décimo caso** se aplicó el método de creación de categorías a partir de clases, en el cual al darnos cuenta que el dataframe creado contenia ambas clases equilibradas, por lo que se pudo deducir que más adelante no tendríamos problemas de balance. La variables utilizadas para este caso fueron:

```
Vars_Indep= df[['availability_eoy',
'availability_60','availability_90']]
Var_Dep= df['availability_30']
```

```
Matriz de Confusión:
[[3707  339]
 [ 306 3498]]
```

Precisión del modelo Ocupación primera mitad del mes:
0.9237478195863443
Precisión del modelo Ocupación segunda mitad del mes:
0.9116497263487099
Exactitud del modelo:
0.9178343949044586
Sensibilidad del modelo Ocupación primera mitad del mes:
0.9162135442412259
Sensibilidad del modelo Ocupación segunda mitad del mes:
0.919558359621451

Para el **décimo primer caso** se realizó la creación de categorías por clases, aunque predominó una de las clases, el desarrollo para llegar a la matriz de confusión no fue complicado, pues la matriz tuvo balance. Las variables utilizadas para este caso fueron

```
Vars_Indep= df[['number_of_reviews_ltm',
'number_of_reviews_ly','estimated_occupancy_l365d']]
Var_Dep= df['reviews_per_month']
```

```
Matriz de Confusión:
[[5270  284]
 [ 494 1802]]
```

Precisión del modelo Más de 5:
0.8638542665388304
Precisión del modelo Menos de 5:
0.9142956280360861
Exactitud del modelo Más de 5:
0.900891719745223
Sensibilidad del modelo Menos de 5:
0.7848432055749129
Sensibilidad del modelo:
0.9488656823910695

En cuanto al **décimo segundo caso** de igual manera se realizó creación de categorías por clases, a simple vista solo se podía encontrar una categoría en el dataframe creado, sin embargo luego me di cuenta que la otra variable estaba presente, lo que significa que no se necesitaba de un balance; aunque también es cierto que no estaba muy equilibrada la presencia de una. Las variables usadas en este caso fueron:

```
Matriz de Confusión:
[[  36  466]
 [  30 7318]]
```

Precisión del modelo Más de la mitad:
0.9401336073997945
Precisión del modelo Menos de la mitad:
0.5454545454545454
Exactitud del modelo Más de la mitad:
0.9368152866242038
Sensibilidad del modelo Más de la mitad:
0.9959172563962984
Sensibilidad del modelo:
0.07171314741035857

Para el décimo tercer caso se realizó la creación de categorías por clases, en la cuál se observó que no sería necesario realizar un balanceo, sin embargo no estaría muy equilibradas la matriz de confusión:

```
Matriz de Confusión:
[[ 437  223]
 [  51 7139]]
```

Precisión del modelo Pocos huéspedes:
0.9697093181200761
Precisión del modelo Muchos huéspedes:
0.8954918032786885
Exactitud del modelo Pocos huéspedes:
0.9650955414012738
Sensibilidad del modelo Muchos huéspedes:
0.9929068150208623
Sensibilidad del modelo:
0.6621212121212121

Hallazgos

En general, los modelos de regresión logística mostraron buen desempeño en variables operativas (como disponibilidad, reseñas y capacidad del alojamiento), pero tuvieron dificultades con variables de tipo “estado del anfitrión” (como verificación, foto de perfil o tipo de scrape). Esto se debe principalmente al desbalance de clases, donde una categoría domina casi todos los registros y hace que el modelo tienda a predecir siempre la mayoría, obteniendo una exactitud alta pero poco útil.

Los mejores resultados se obtuvieron en las variables availability_30, reviews_per_month y accommodates, ya que el modelo logró distinguir correctamente entre las distintas categorías y mostró buena capacidad de generalización. En estos casos, los indicadores de rendimiento (como precisión y sensibilidad) fueron equilibrados, y las confusiones fueron pocas.
Esto significa que el modelo sí puede identificar patrones reales en la oferta y desempeño de los alojamientos: distingue bien entre propiedades con mayor disponibilidad, más actividad y mayor capacidad de huéspedes.

En cambio, en variables como source, host_has_profile_pic y host_identity_verified, el modelo falló al detectar la clase minoritaria. Aunque la exactitud fue alta, esto se debe a que la mayoría de los casos pertenecen a una sola categoría (por ejemplo, casi todos los hosts tienen foto o están verificados).
Cuando se aplicó reponderación de clases o sobremuestreo, se mejoró la detección de los casos menos frecuentes, aunque la precisión general bajó. Esto muestra que balancear los datos es necesario si se quiere que el modelo aprenda realmente las diferencias entre grupos.

En el caso de las variables derivadas como precio categorizado y disponibilidad anual (más/menos de la mitad), los resultados fueron intermedios: el modelo reconoce tendencias

generales (por ejemplo, más listados “baratos”), pero aún necesita cortes más representativos o variables complementarias (zona, amenities, vistas, etc.) para mejorar su desempeño.

y	x	P(1)	P(2)	E	S(1)	S(2)
source	availability_eoy,availability_365,availability_90	0.0	0.89	0.89	0.0	1.0
source(balance)	availability_eoy,availability_365,availability_90	0.26	0.95	0.75	0.69	0.76
host has profile pic	maximum_maximum_nights,maximum_nights_avg_ntm,maximum_nights	0.95	0.0	0.0	1.0	0.0
host has profile pic(balance)	maximum_maximum_nights,maximum_nights_avg_ntm,maximum_nights	0.98	0.08	0.60	0.60	0.74
host identity verified	price, host_is_superhost,minimum_maximum_nights	0.98	0.0	0.98	0.99	0.0
host identity verified(balance)	price, host_is_superhost,minimum_maximum_nights	0.98	0.03	0.68	0.68	0.61
instant bookable	host_acceptance_rate, maximum_maximum_nights,maximum_nights_avg_ntm	0.69	0.82	0.71	0.96	0,26
host is superhost	estimated_occupancy_l365d, number_of_reviews_ly,number_of_reviews_ltm	0.70	0.75	0.71	0.90	0.43
price	accommodates,beds,bedrooms	0.87	0.75	0.86	0.97	0.37
availability 30	availability_eoy, availability_60,availability_90	0.92	0.91	0.91	0.91	0.91
reviews per month	number_of_reviews_ltm, number_of_reviews_ly,estimated_occupancy_l365d	0.86	0.91	0.90	0.78	0.94
host acceptance rate	estimated_occupancy_l365d, host_response_rate,calculated_host_listings_count_private_rooms	0.94	0.54	0.93	0.99	0.07
accommodates	calculated_host_listings_count_entire_homes,bedrooms,beds	0.96	0.89	0.96	0.99	0.66