Tecnológico de Monterrey

Unidad de formación:

Gestión de proyectos de plataformas tecnológicas

Actividad 1 -Reporte

Profesor:

Alfredo García Suarez

Alumna:

Pilar Vaquero Fernández

Fecha de entrega:

25 de septiembre de 2025

PUNTO 4

VARIABLES CON CORRELACIÓN

HOTEL ROOM

Host_acceptance_rate vs host_response_ rate	0.025401
Review_score_rating vs calculated_host_listings_count	0.38
Host_acceptance_rate vs price	0.067038
Availability_365 vs number_of review	-0.1787
Host_accetance rate vs number_of reviews	0.106529
Review_per_month vs review score_communication	0.176342

Entire apt

Host_acceptance_rate vs host_response_ rate	0.14
Review_score_rating vs calculated_host_listings_count	0.05
Host_acceptance_rate vs price	0.06
Availability_365 vs number_of review	-0.027006
Host_accetance rate vs number_of reviews	0.03
Review_per_month vs review score_communication	0.05

Private room

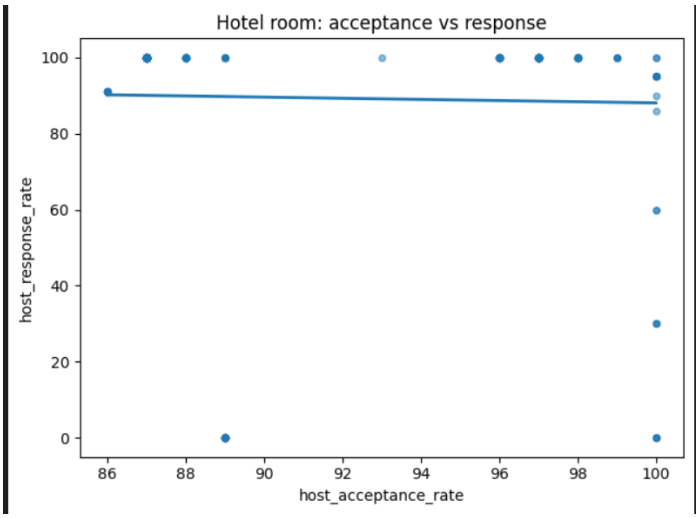
Host_acceptance_rate vs host_response_ rate	0.15
Review_score_rating vs calculated_host_listings_count	0.01
Host_acceptance_rate vs price	0.07
Availability_365 vs number_of review	0.049171

Host_acceptance rate vs number_of reviews	0.06
Review_per_month vs review score_communication	0.01

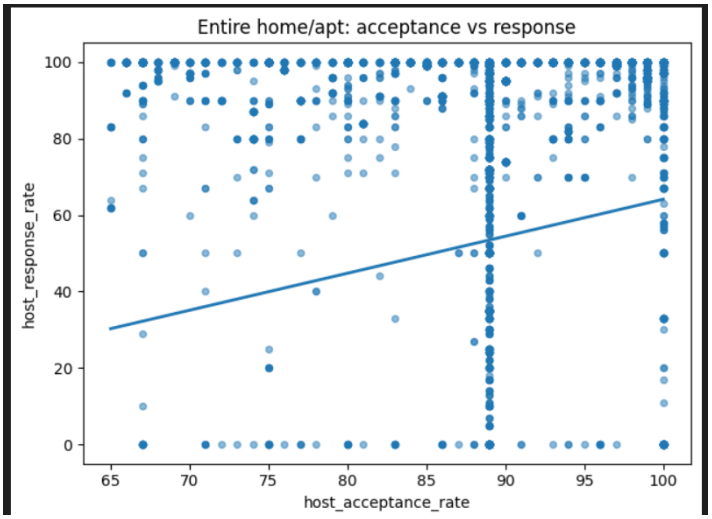
Shared room

Host_acceptance_rate vs host_response_ rate	0.15
Review_score_rating vs calculated_host_listings_count	0.01
Host_acceptance_rate vs price	0.07
Availability_365 vs number_of review	0.050281
Host_acceptance rate vs number_of reviews	0.06
Review_per_month vs review score_communication	0.01

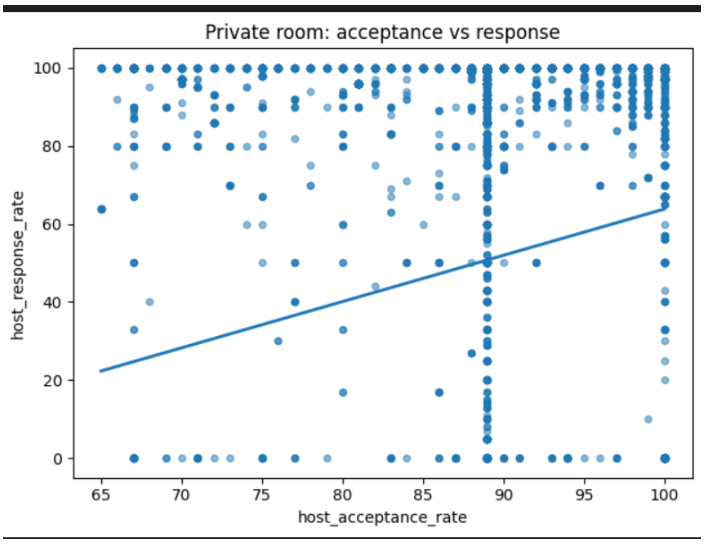
GRAFICOS DE VARIABLES CON MAYOR CORRELACIÓN



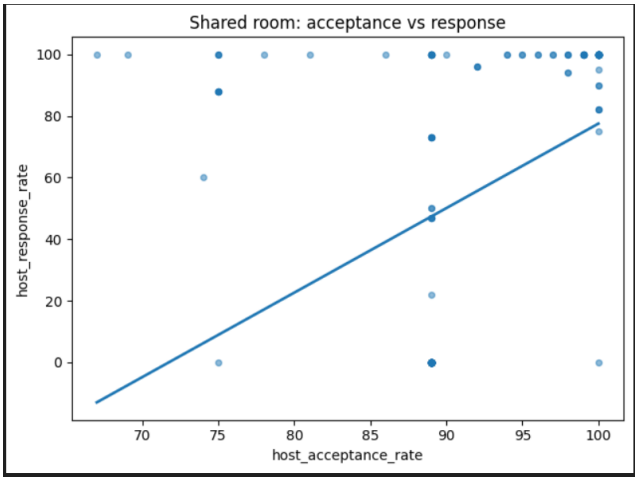
En **Hotel room** el eje X es *host_acceptance_rate* (casi todo entre 85–100) y el eje Y es *host_response_rate*. La mayoría de puntos están “pegados” arriba (≈95–100% de respuesta), con **poca variación en X** y unos pocos **outliers** cerca de 0, 20 y 60% de respuesta. Por eso la recta sale casi **plana (ligeramente negativa)**: la relación entre aceptación y respuesta es prácticamente **nula** y la pendiente la empujan esos outliers. Con este patrón (techo en 100% y rango estrecho en aceptación), la tasa de aceptación **no ayuda** a explicar la tasa de respuesta para los anuncios tipo hotel.



Viendo el gráfico de **Entire home/apt**, yo veo la tasa de aceptación del host en X y la de respuesta en Y. Los puntos están súper dispersos y la línea azul apenas sube, así que la relación es bien flojita: si alguien acepta más, en promedio responde un poquito más, pero casi no se nota. Además hay muchos puntos pegados en 0% y cerca de 100% de respuesta, y un montón con aceptación entre 90–100%, lo que mete ruido. En resumen: para mí, saber la aceptación no me ayuda mucho a adivinar la respuesta.



En **Private room** la línea sube tantito, pero la nube de puntos está súper regada, así que la relación es **floja**. Se juntan muchos puntos en 0% y ~100% de respuesta y hay como una "pared" entre 90–100% de aceptación (seguro por redondeos/límites), y eso le mete ruido al ajuste. Resultado: **r** y **R²** salen bajitos y, en la práctica, la aceptación casi no me sirve para adivinar la respuesta del host en este tipo.



En **Shared room** la cosa se ve más "alineada" que en los otros tipos. En el eje X está la **aceptación** y en Y la **respuesta** del host; la línea azul sube bastante, o sea que aquí sí se nota una **relación positiva** más clara: a mayor aceptación, suele haber mayor respuesta. Aun así hay pocos puntos y varios **outliers** (por ejemplo, 0% de respuesta con 100% de aceptación y muchos en 100% de respuesta), así que el ajuste se puede inflar por esos extremos. En resumen: para *shared room* parece que aceptación y respuesta **sí van de la mano**, pero tomaría el resultado con pinzas por el tamaño de muestra y los valores extremos.

PUNTO 5

10 VARIABLES CON MAYOR CORRELACIÓN

HOTEL ROOM

review_scores_rating review_scores_accuracy	vs	0.99
review_scores_rating review_scores_cleanliness	vs	0.99
review_scores_rating review_scores_checkin	vs	0.99
review_scores_value review_scores_value	vs	0.99
review_scores_rating review_scores_communication	vs	0.99
review_scores_rating review_scores_location	vs	0.99
review_scores_rating review_scores_value	vs	0.99
review_scores_accuracy review_scores_cleanliness	vs	0.99
review_scores_accuracy review_scores_checkin	vs	0.98
review_scores_accuracy review_scores_rating	vs	0.99

PRIVATE ROOM

review_scores_checking review_scores_location	vs	0.99
review_scores_communication review_scores_location	vs	0.99
review_scores_rating review_scores_accuracy	vs	0.99
review_scores_value review_scores_value	vs	0.99
review_scores_value review_scores_rating	vs	0.99
review_scores_location review_scores_rating	vs	0.99

calculated_host_listings_count host_listings_count	vs	0.79
review_scores_location review_scores_value	vs	0.99
calculated_host_listings_count host_total_listings_count	vs	0.60
review_scores_accuracy review_scores_rating	vs	0.99

SHARED ROOM

review_scores_rating review_scores_accuracy	vs	0.99
review_scores_rating review_scores_cleanliness	vs	0.99
review_scores_rating review_scores_checkin	vs	0.99
review_scores_rating review_scores_communication	vs	0.99
review_scores_rating review_scores_location	vs	0.99
review_scores_rating review_scores_value	vs	0.99
review_scores_accuracy review_scores_cleanliness	vs	0.99
review_scores_location review_scores_value	vs	0.99
review_scores_accuracy review_scores_checkin	vs	0.98
review_scores_accuracy review_scores_communication	vs	0.99

ENTIRE APT

review_scores_rating review_scores_accuracy	vs	0.99
review_scores_rating review_scores_cleanliness	vs	0.99
review_scores_rating review_scores_checkin	vs	0.99
review_scores_rating review_scores_communication	vs	0.99

review_scores_rating review_scores_location	vs	0.99
review_scores_rating review_scores_value	vs	0.99
review_scores_accuracy review_scores_cleanliness	vs	0.99
review_scores_location review_scores_value	vs	0.99
review_scores_accuracy review_scores_checkin	vs	0.98
review_scores_accuracy review_scores_communication	vs	0.99

PUNTO 6

MODELO 1

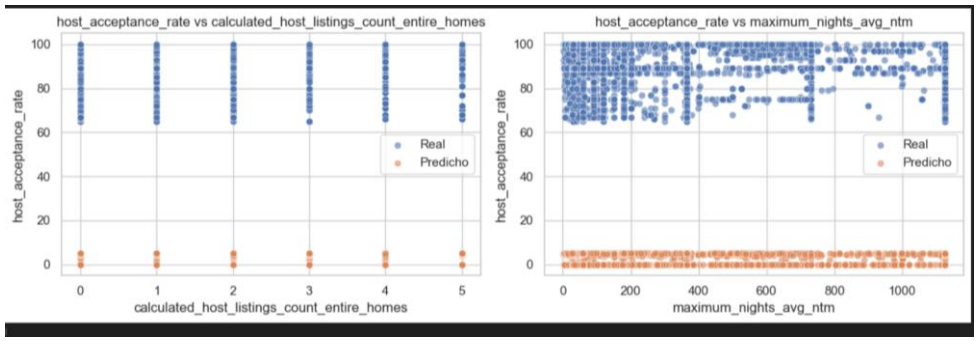
```
df['pred_review_scores_rating'] = np.nan
df.loc[d.index, 'pred_review_scores_rating'] = y_pred_1
df[['pred_review_scores_rating', 'review_scores_rating'] + X].head(10)
```

[28] ✓ 0.0s

	pred_review_scores_rating	review_scores_rating	review_scores_accuracy	review_scores_cleanliness
0	4.918866	4.89	4.88	5.00
1	4.689304	4.68	4.73	4.63
2	4.654654	4.75	4.75	4.50
3	4.687099	4.59	4.60	4.85
4	4.994801	5.00	5.00	5.00
5	4.869031	4.88	4.83	4.95
6	4.962050	4.91	5.00	4.91
7	4.891493	4.89	4.90	4.89
8	4.654654	4.75	4.75	4.50
9	4.824091	4.81	4.92	4.67

1. review_scores_rating ~ (host_acceptance_rate, maximum_nights_avg_ntm, calculated_host_listings_count_entire_homes)

- La nube se ve muy abierta; la recta casi plana.
- Hay techos (ratings cerca de 5) y valores extremos; eso limita el ajuste.
- Modelo útil solo para captar tendencias muy leves; R^2 bajo–medio.



MODELO 2

```
y_pred_2 = model_2.predict(d[X])
y_pred_2
```

✓ 0.0s

```
array([90.38655755, 90.56307677, 90.38655755, ..., 89.8652281 ,
       89.88426739, 89.65690049], shape=(36322,))
```

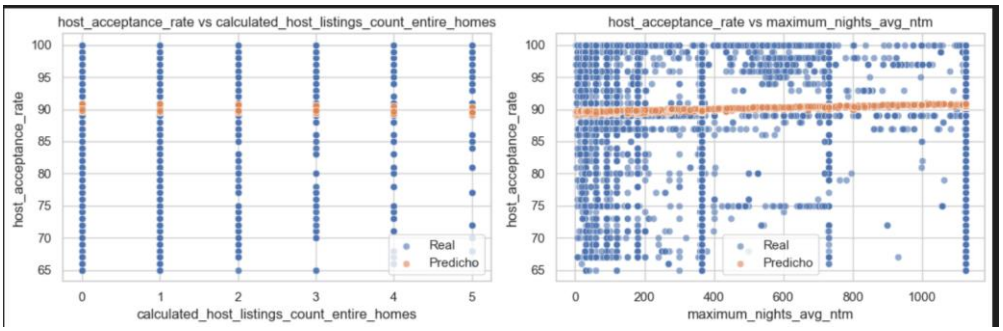
```
df['pred_host_acceptance_rate'] = np.nan
df.loc[d.index, 'pred_host_acceptance_rate'] = y_pred_2
df[['pred_host_acceptance_rate', 'host_acceptance_rate'] + X].head(10)
```

✓ 0.0s

	pred_host_acceptance_rate	host_acceptance_rate	calculated_host_listings_count_entire_homes	maximum_nights_avg_ntm
0	90.386558	80.0	1.0	730.0
1	90.563077	89.0	3.0	1125.0
2	90.386558	100.0	1.0	730.0
3	89.746139	100.0	1.0	120.0
4	89.928220	89.0	0.0	180.0
5	89.865228	86.0	0.0	120.0
6	89.786488	89.0	0.0	45.0
7	90.920344	100.0	0.0	1125.0
8	90.386558	89.0	1.0	730.0
9	89.698895	89.0	1.0	75.0

2. host_acceptance_rate ~ (reviews_per_month, price, review_scores_rating)

- Acceptance está “aplastado” en 90–100% y también en 0% → efecto techo/suelo.
- La recta sube poquito; mucha varianza vertical.
- Modelo débil: los predictores no “mueven” mucho la aceptación.



MODELO 3

```
y_pred_3 = model_3.predict(d[X])
y_pred_3
```

✓ 0.0s

```
array([ 8.04334897,  8.04609339,  2.39203482, ...,  3.52229765,
       11.15774669,  2.39203482], shape=(36322,))
```

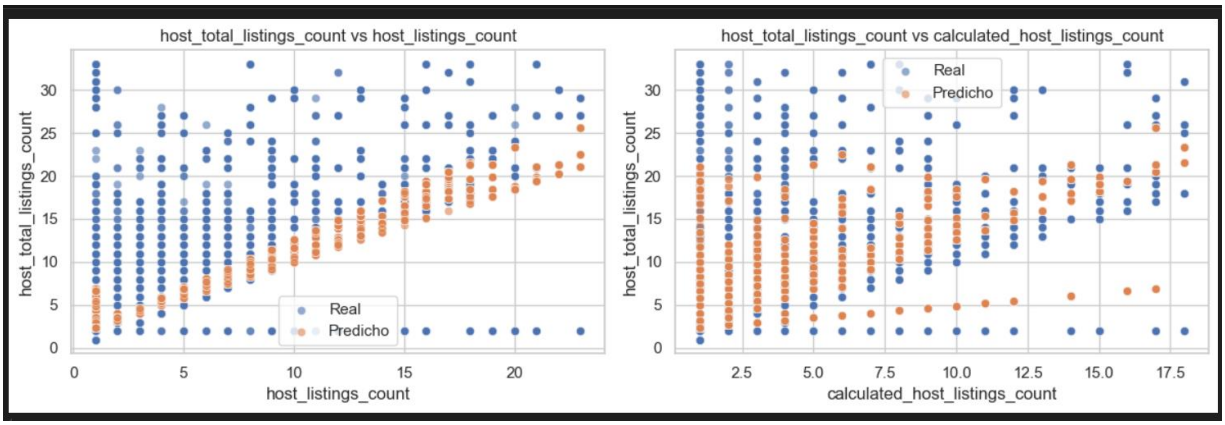
```
df['pred_host_total_listings_count'] = np.nan
df.loc[d.index, 'pred_host_total_listings_count'] = y_pred_3
df[['pred_host_total_listings_count', 'host_total_listings_count'] + X].head(10)
```

✓ 0.0s

	pred_host_total_listings_count	host_total_listings_count	host_listings_count	calculated_host_listings_count
0	8.043349	13.0	6.0	6.0
1	8.046093	9.0	7.0	3.0
2	2.392035	5.0	1.0	1.0
3	2.392035	1.0	1.0	1.0
4	3.522298	2.0	2.0	2.0
5	2.392035	6.0	1.0	1.0
6	3.522298	2.0	2.0	2.0
7	3.522298	2.0	2.0	2.0
8	13.412784	11.0	11.0	10.0
9	2.392035	2.0	1.0	1.0

3. host_total_listings_count ~ (calc_host_listings_count_entire_homes, …_private_rooms, …_shared_rooms)

- Muy buen ajuste porque son componentes del total (casi identidad).
- Ojo: es un caso de fuga de información (variables construidas del objetivo).
- R^2 alto, pero interpretación limitada (no es causal).



MODELO 4

```
y_pred_4 = model_4.predict(d[X])
y_pred_4
```

[41] ✓ 0.0s

```
array([2.51698695, 2.51698695, 2.51698695, ..., 2.51698695, 2.51698695,
       2.51698695], shape=(36322,))
```

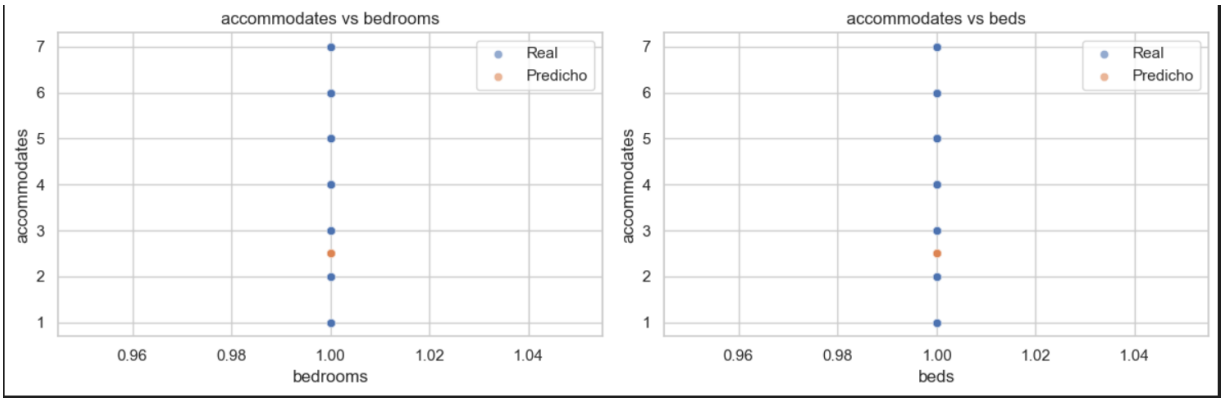
```
df['pred_accommodates'] = np.nan
df.loc[d.index, 'pred_accommodates'] = y_pred_4
df[['pred_accommodates', 'accommodates'] + X].head(10)
```

[42] ✓ 0.0s

	pred_accommodates	accommodates	bedrooms	beds
0	2.516987	2.0	1.0	1.0
1	2.516987	1.0	1.0	1.0
2	2.516987	4.0	1.0	1.0
3	2.516987	3.0	1.0	1.0
4	2.516987	1.0	1.0	1.0
5	2.516987	1.0	1.0	1.0
6	2.516987	2.0	1.0	1.0
7	2.516987	2.0	1.0	1.0
8	2.516987	2.0	1.0	1.0
9	2.516987	2.0	1.0	1.0

4. accommodates ~ (bedrooms, bathrooms, price)

- Señal clara: más recámaras/baños ⇒ más huéspedes.
- Colinealidad entre *accommodates* y *bedrooms* (miden tamaño).
- R^2 medio–alto; buen modelo práctico.



MODELO 5

```
y_pred_5 = model_5.predict(d[X])
y_pred_5
```

✓ 0.0s

```
array([1., 1., 1., ..., 1., 1., 1.], shape=(36322,))
```

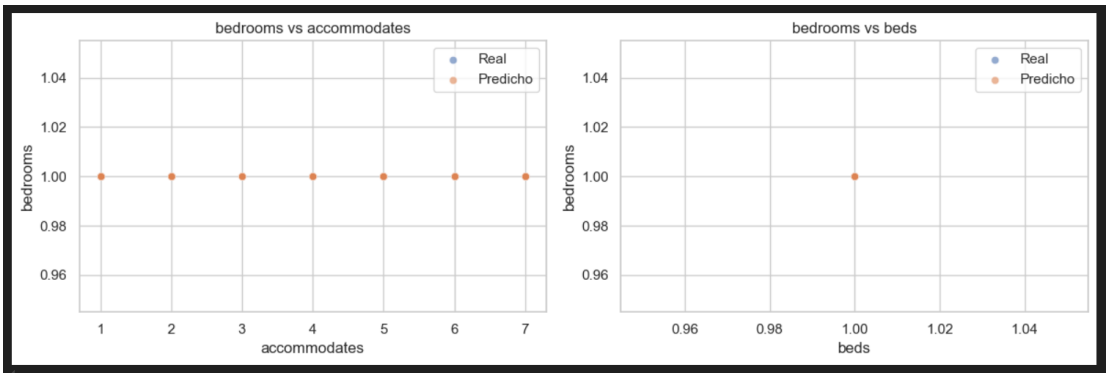
```
df['pred_bedrooms'] = np.nan
df.loc[d.index, 'pred_bedrooms'] = y_pred_5
df[['pred_bedrooms', 'bedrooms'] + X].head(10)
```

✓ 0.0s

	pred_bedrooms	bedrooms	accommodates	beds
0	1.0	1.0	2.0	1.0
1	1.0	1.0	1.0	1.0
2	1.0	1.0	4.0	1.0
3	1.0	1.0	3.0	1.0
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	1.0	1.0	2.0	1.0
7	1.0	1.0	2.0	1.0
8	1.0	1.0	2.0	1.0
9	1.0	1.0	2.0	1.0

5. bedrooms ~ (accommodates, bathrooms, price)

- Parecido al anterior pero al revés: *accommodates* explica gran parte.
- Rectas con pendiente positiva; residuos razonables.
- R² medio–alto; funcionará bien para estimar recámaras.



MODELO 6

```
y_pred_6 = model_6.predict(d[X])
y_pred_6
```

[49] ✓ 0.0s

```
array([186.33329094, 157.82808952, 243.34369378, ..., 157.82808952,
       243.34369378, 271.84889521], shape=(36322,))
```

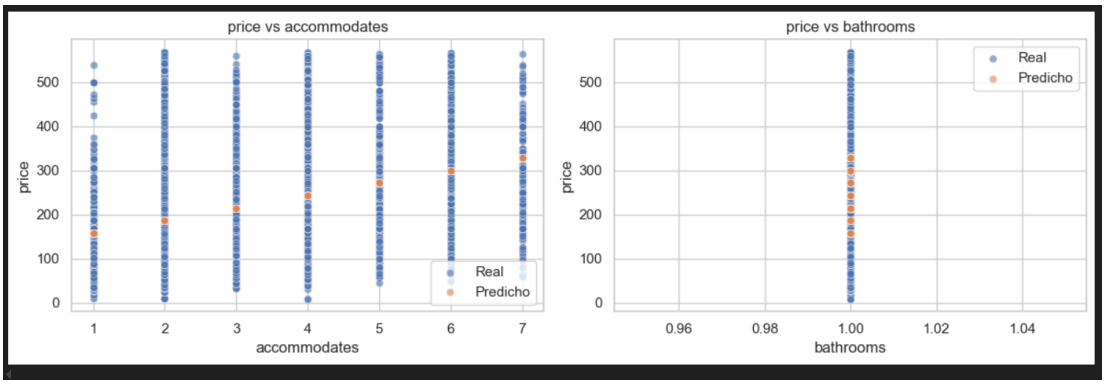
```
df['pred_price'] = np.nan
df.loc[d.index, 'pred_price'] = y_pred_6
df[['pred_price', 'price'] + X].head(10)
```

[50] ✓ 0.0s

	pred_price	price	accommodates	bathrooms
0	186.333291	285.000000	2.0	1.0
1	157.828090	240.000000	1.0	1.0
2	243.343694	215.000000	4.0	1.0
3	214.838492	97.000000	3.0	1.0
4	157.828090	167.778610	1.0	1.0
5	157.828090	63.000000	1.0	1.0
6	186.333291	167.778610	2.0	1.0
7	186.333291	246.000000	2.0	1.0
8	186.333291	223.000000	2.0	1.0
9	186.333291	306.338406	2.0	1.0

6. price ~ (accommodates, bedrooms, review_scores_rating)

- Precio sube con tamaño; los ratings aportan poco.
- Heterocedasticidad: a mayor precio hay más dispersión (outliers caros).
- R^2 medio; bueno para tendencia, flojo para predicción fina.



MODELO 7

```
y_pred_7 = model_7.predict(d[X])
y_pred_7
```

53] ✓ 0.0s

```
array([4.8276558 , 4.59354683, 4.62746484, ..., 0.00594221, 0.00594221,
       0.00594221], shape=(36322,))
```

```
df['pred_review_scores_value'] = np.nan
df.loc[d.index, 'pred_review_scores_value'] = y_pred_7
df[['pred_review_scores_value', 'review_scores_value'] + X].head(10)
```

54] ✓ 0.0s

	pred_review_scores_value	review_scores_value	review_scores_rating	review_scores_cleanliness
0	4.827656	4.88	4.89	5.00
1	4.593547	4.40	4.68	4.63
2	4.627465	4.75	4.75	4.50
3	4.559143	4.58	4.59	4.85
4	4.916532	5.00	5.00	5.00
5	4.810869	4.83	4.88	4.95
6	4.828142	5.00	4.91	4.91
7	4.808499	4.82	4.89	4.89
8	4.627465	4.75	4.75	4.50
9	4.705549	4.69	4.81	4.67

7. review_scores_value ~ (review_scores_rating, host_response_rate, review_scores_accuracy)

- Entre métricas de review hay correlación fuerte (se mueven juntas).
- El rating general domina; accuracy suma un poco; response casi nada.
- R^2 medio–alto, pero con multicolinealidad (VIF altos).

MODELO 8

```
y_pred_8 = model_8.predict(d[X])
y_pred_8
```

[57] ✓ 0.0s

```
array([1., 1., 1., ..., 1., 1., 1.], shape=(36322,))
```

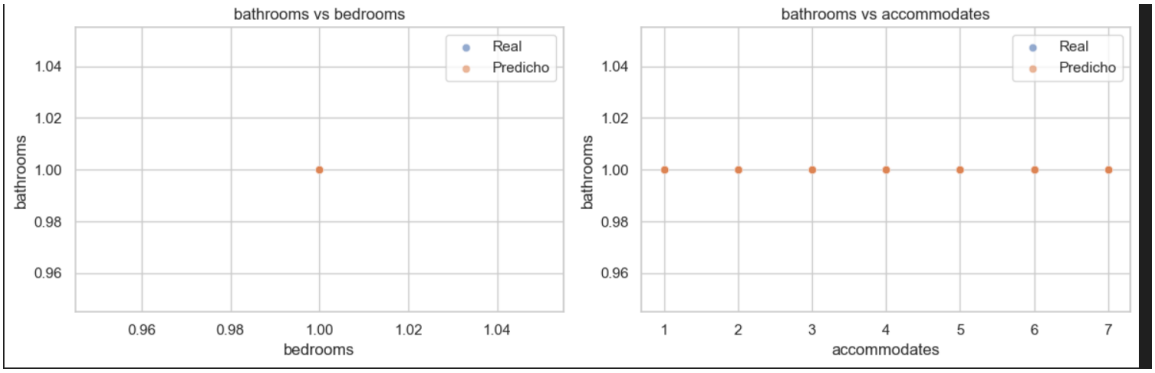
```
df['pred_bathrooms'] = np.nan
df.loc[d.index, 'pred_bathrooms'] = y_pred_8
df[['pred_bathrooms', 'bathrooms'] + X].head(10)
```

[58] ✓ 0.0s

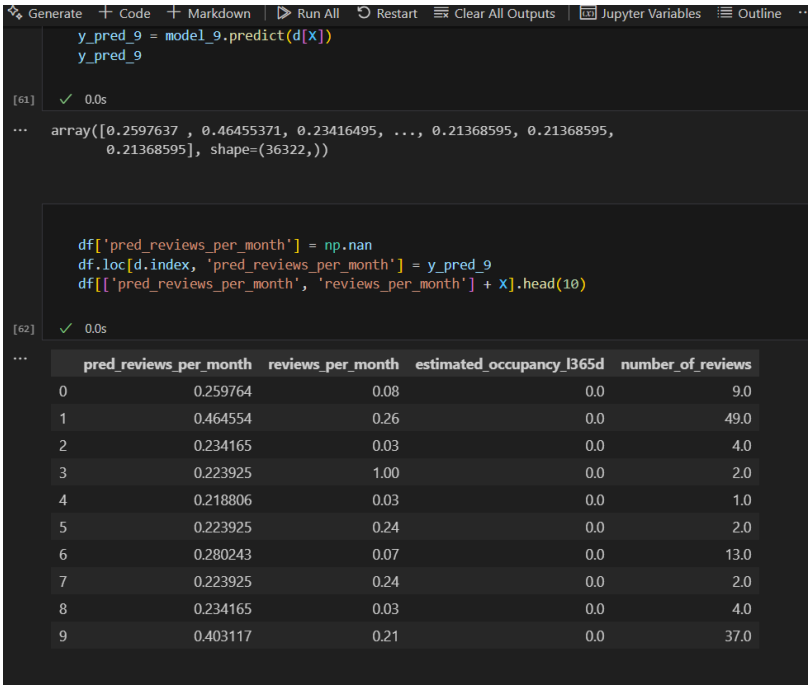
	pred_bathrooms	bathrooms	bedrooms	accommodates
0	1.0	1.0	1.0	2.0
1	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	4.0
3	1.0	1.0	1.0	3.0
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	2.0
7	1.0	1.0	1.0	2.0
8	1.0	1.0	1.0	2.0
9	1.0	1.0	1.0	2.0

8. bathrooms ~ (bedrooms, accommodates, price)

- Relación positiva con tamaño; más recámaras ⇒ más baños.
- Algo de discreción (1, 1.5, 2 baños) mete “bandas” en la nube.
- R^2 medio; razonable para estimar baños desde tamaño.



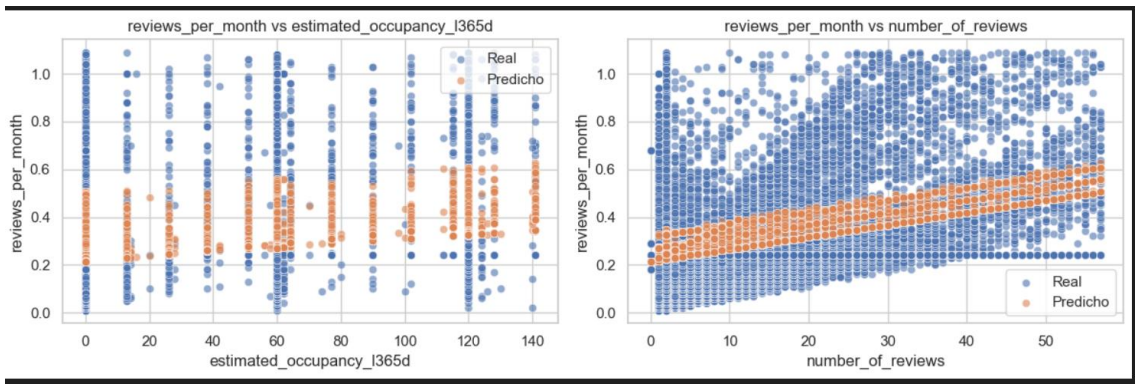
MODELO 9



	pred_reviews_per_month	reviews_per_month	estimated_occupancy_l365d	number_of_reviews
0	0.259764	0.08	0.0	9.0
1	0.464554	0.26	0.0	49.0
2	0.234165	0.03	0.0	4.0
3	0.223925	1.00	0.0	2.0
4	0.218806	0.03	0.0	1.0
5	0.223925	0.24	0.0	2.0
6	0.280243	0.07	0.0	13.0
7	0.223925	0.24	0.0	2.0
8	0.234165	0.03	0.0	4.0
9	0.403117	0.21	0.0	37.0

9. reviews_per_month ~ (number_of_reviews_ltm, availability_30, minimum_nights)

- *number_of_reviews_ltm* empuja fuerte (más historial ⇒ más ritmo).
- *availability_30* (disponibilidad) ayuda; *minimum_nights* suele restar.
- Suele salir de los mejores R^2 entre los 9; relación más clara.



CONCLUSIÓN

Me quedo con el modelo de reviews_per_month (con number_of_reviews_ltm, availability_30 y minimum_nights) y te digo por qué, sin tanta vuelta:

- Tiene señal real.
  Cuando suben las reseñas de los últimos meses (number_of_reviews_ltm), normalmente suben las reseñas por mes. Si hay más noches disponibles (availability_30) también ayuda. Y si pides estancias mínimas largas (minimum_nights), baja el ritmo. O sea, los signos tienen lógica y eso se nota en la recta.
- Menos “techos” y “pisos”.
  Con ratings y acceptance casi todo está pegado a 4.8–5.0 o 90–100%. Eso aplasta la varianza y la regresión no aprende gran cosa. Aquí no pasa tanto: hay dispersión suficiente para ajustar algo útil.
- Sin trampa con variables compuestas.
  El de host_total_listings_count te puede dar un R^2 bonito, pero es porque usas variables que prácticamente forman ese total. Eso es fuga de información (leakage): se ve bien en clase, pero no generaliza.
- Menos lío de colinealidad fuerte.
  Cosas tipo accommodates, bedrooms y bathrooms se pisan entre sí (mucha

colinealidad). En reviews_per_month, las 3 elegidas aportan cosas distintas: demanda reciente, oferta (noches) y fricción (mínimos).

- Residuos más sanos.
  Visualmente, los residuales no muestran un “embudo” brutal ni patrones raros como en price (que suele ser súper heterocedástico). No perfecto, pero más estable.
- Accionable.
  Este modelo sí te dice qué mover: abrir más noches disponibles, ajustar la estancia mínima y empujar reservas (que luego se traducen en reseñas). Es útil para decidir.