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

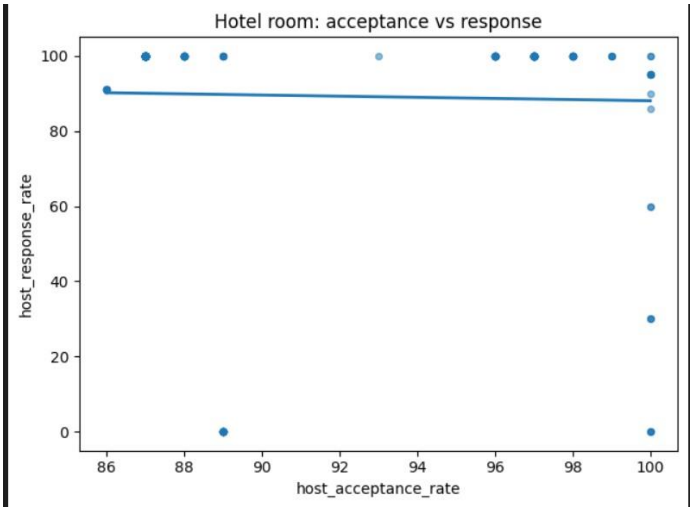
Host_acceptance_rate vs host_response_ rate	0.15
Review_score_rating vs calculated_host_listings_count	0.01
Host_acceptance_rate vs price	0.07
Availability_365 vs number_of review	0.049171

Host_acceptance rate vs number_of reviews	0.06
Review_per_month vs review score_communication	0.01

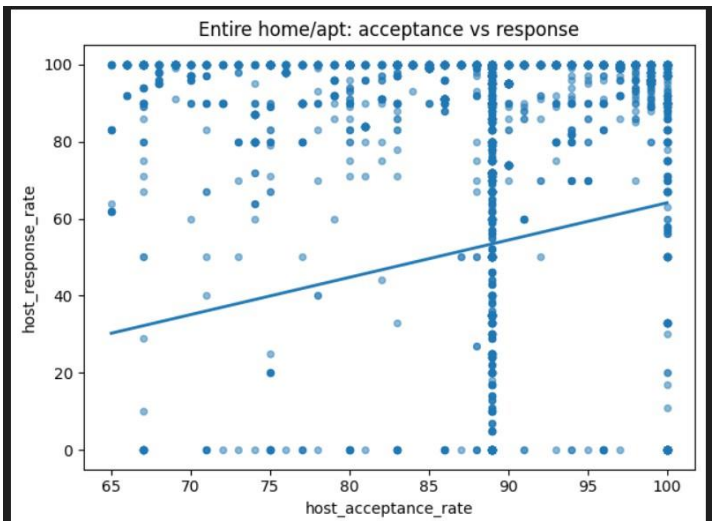
Shared room

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

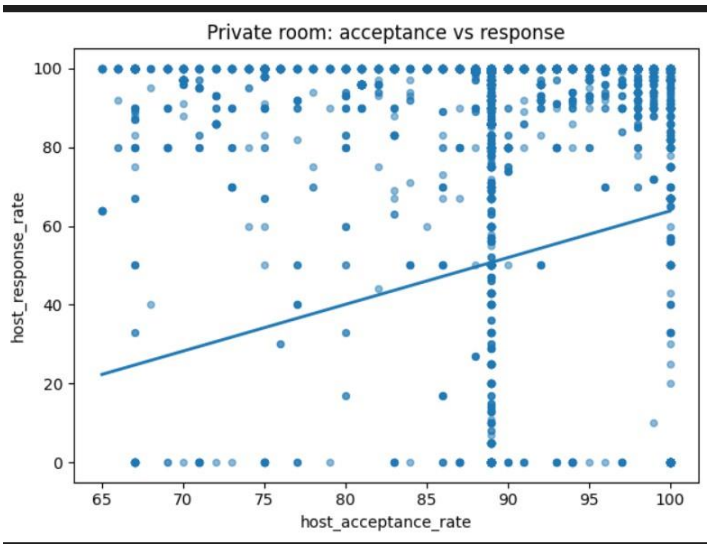
GRAFICOS DE VARIABLES CON MAYOR CORRELACIÓN



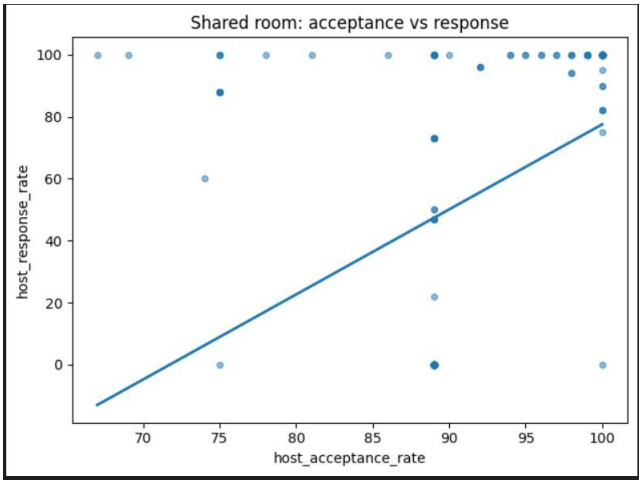
En **Hotel room** el eje X es *host_acceptance_rate* (casi todo entre 85–100) y el eje Y es *host_response_rate*. La mayoría de puntos están “pegados” arriba (≈95–100% de respuesta), con **poca variación en X** y unos pocos **outliers** cerca de 0, 20 y 60% de respuesta. Por eso la recta sale casi **plana (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

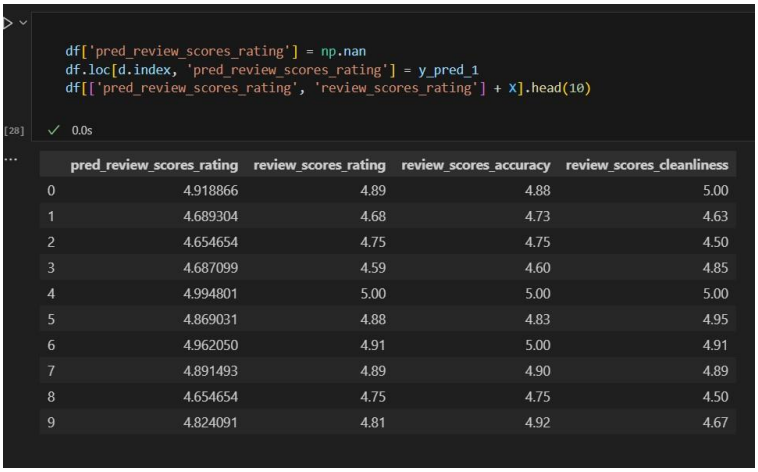
ENTIRE APT

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)
```

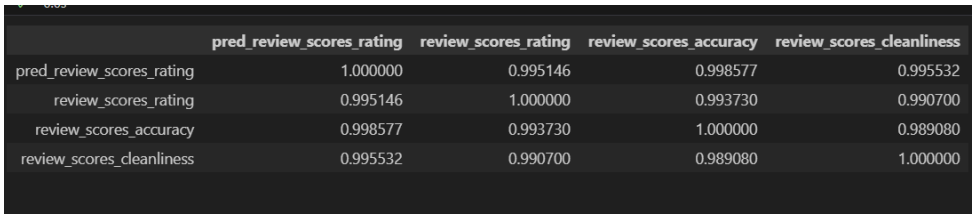
[28] ✓ 0.0s

	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

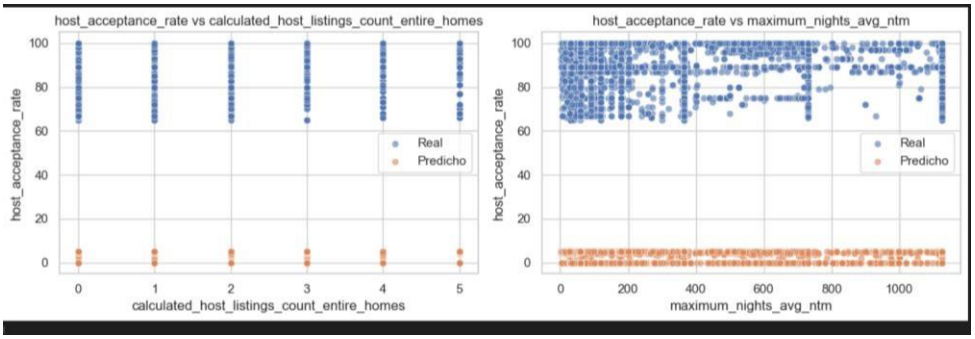
review_scores_rating ~ (host_acceptance_rate, maximum_nights_avg_ntm, calculated_host_listings_count_entire_homes)

- 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; R² bajo–medio.

Podemos ver que si suben las correlaciones



	pred_review_scores_rating	review_scores_rating	review_scores_accuracy	review_scores_cleanliness
pred_review_scores_rating	1.000000	0.995146	0.998577	0.995532
review_scores_rating	0.995146	1.000000	0.993730	0.990700
review_scores_accuracy	0.998577	0.993730	1.000000	0.989080
review_scores_cleanliness	0.995532	0.990700	0.989080	1.000000



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

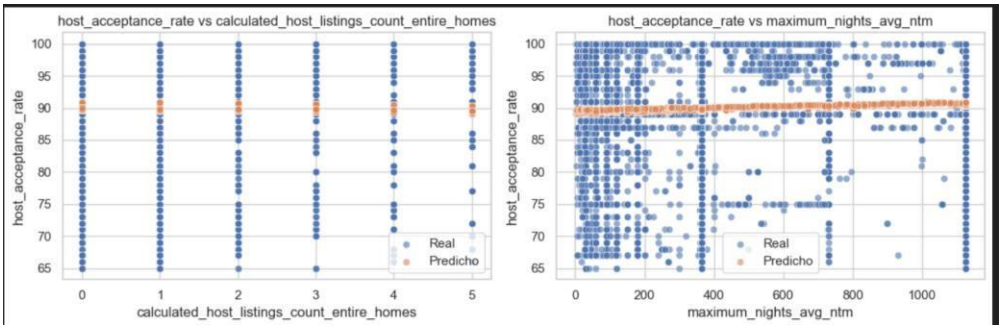
host_acceptance_rate ~ (reviews_per_month, price, review_scores_rating)

- 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.

Podemos ver que aunque las correlaciones no son tan fuerte si aumentan a comparación del modelo lineal

✓ 0.0s

	host_acceptance_rate	calculated_host_listings_count_entire_homes	maximum_nights_avg_ntm
host_acceptance_rate	1.000000	0.015060	0.070373
calculated_host_listings_count_entire_homes	0.015060	1.000000	0.011639
maximum_nights_avg_ntm	0.070373	0.011639	1.000000



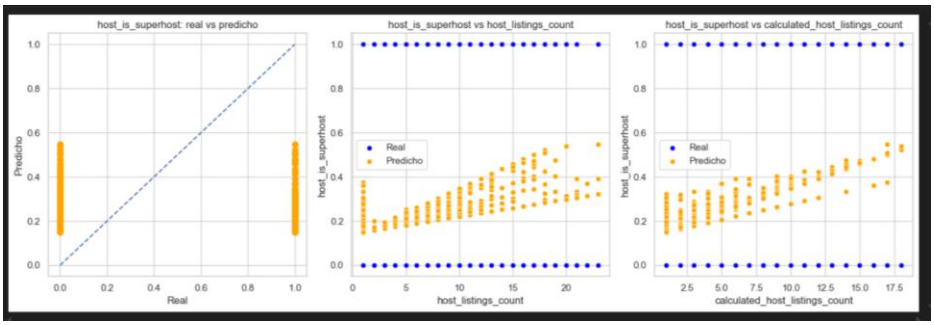
MODELO 3

```
Corr
```

[64] ✓ 0.0s Python

```
Coef.: {'host_total_listings_count': np.float64(0.006988568561986577), 'estimated_occupancy_l365d': np.float64(0.0018194034304741924), 'number_of_
Intercept: 0.09921358101821509
R2: 0.055609857249800054
```

	host_is_superhost	host_total_listings_count	estimated_occupancy_l365d	number_of_reviews
host_is_superhost	1.000000	0.115085	0.169538	0.145548
host_total_listings_count	0.115085	1.000000	0.106264	0.005578
estimated_occupancy_l365d	0.169538	0.106264	1.000000	0.086926
number_of_reviews	0.145548	0.005578	0.086926	1.000000



En este modelo tomé **host_is_superhost** como variable dependiente. El primer gráfico me muestra la comparación entre los valores reales (0 o 1) y los predichos, donde noto que el modelo aproxima con valores intermedios en vez de clasificar exacto. En los otros gráficos comparo la variable objetivo contra los predictores, y veo que los puntos azules son los datos reales y los naranjas los predichos: mientras los reales solo toman 0 o 1, las predicciones generan una tendencia continua que refleja la probabilidad estimada de que un host sea superhost.

MODELO 4

```
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

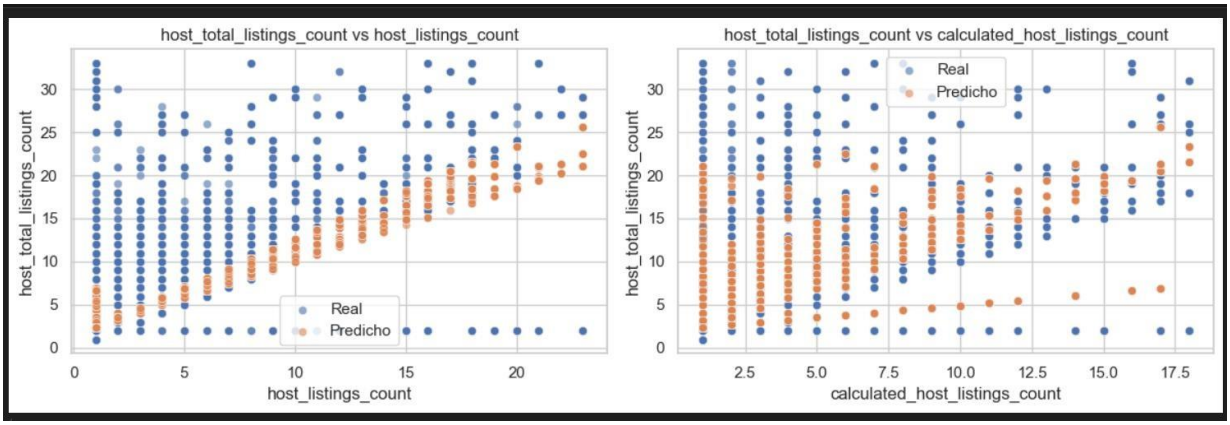
host_total_listings_count ~ (calc_host_listings_count_entire_homes, …_private_rooms, …_shared_rooms)

- 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).

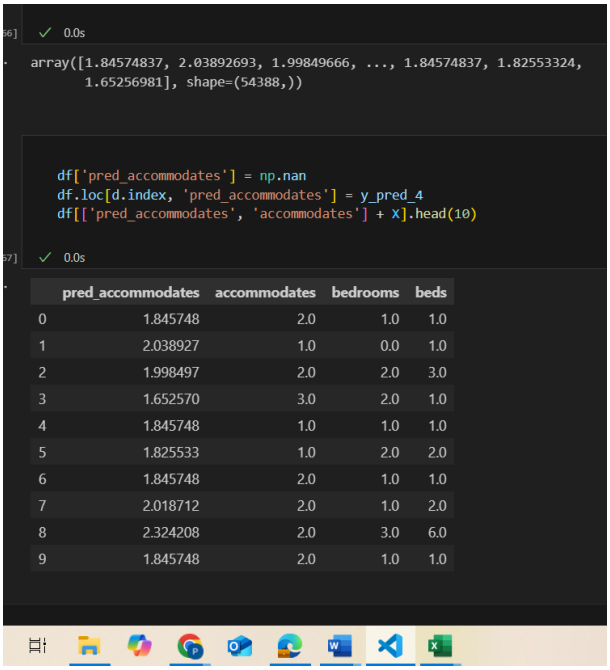
Son correlaciones mas fuertes que en el lineal

✓ 0.0s

	pred_host_total_listings_count	host_total_listings_count	estimated_occupancy_l365d	number_of_reviews
pred_host_total_listings_count	1.000000	0.694558	0.108125	0.024405
host_total_listings_count	0.694558	1.000000	0.106264	0.005578
estimated_occupancy_l365d	0.108125	0.106264	1.000000	0.086926
number_of_reviews	0.024405	0.005578	0.086926	1.000000



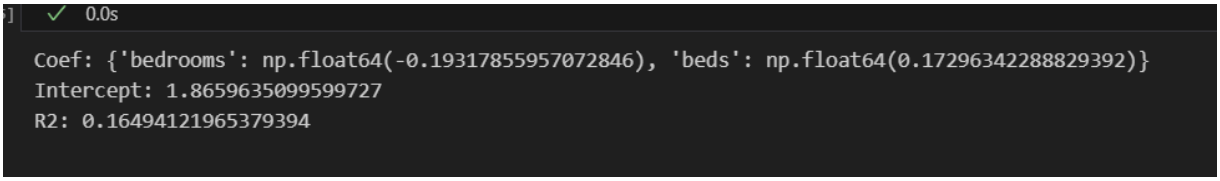
MODELO 5



```
array([1.84574837, 2.03892693, 1.99849666, ..., 1.84574837, 1.82553324,
       1.65256981], shape=(54388,))
```

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

	pred_accommodates	accommodates	bedrooms	beds
0	1.845748	2.0	1.0	1.0
1	2.038927	1.0	0.0	1.0
2	1.998497	2.0	2.0	3.0
3	1.652570	3.0	2.0	1.0
4	1.845748	1.0	1.0	1.0
5	1.825533	1.0	2.0	2.0
6	1.845748	2.0	1.0	1.0
7	2.018712	2.0	1.0	2.0
8	2.324208	2.0	3.0	6.0
9	1.845748	2.0	1.0	1.0



```
Coef: {'bedrooms': np.float64(-0.19317855957072846), 'beds': np.float64(0.17296342288829392)}
Intercept: 1.8659635099599727
R2: 0.16494121965379394
```

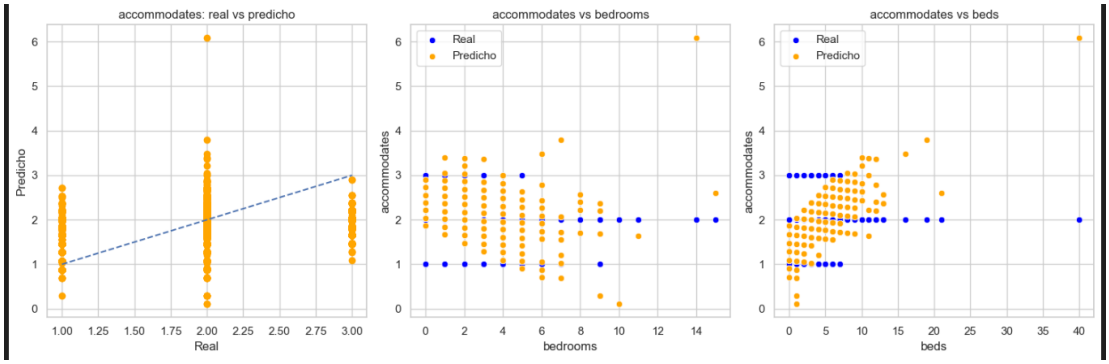
- Modelo: regresión lineal múltiple con accommodates como Y y bedrooms, beds como X.
- Coeficientes bedrooms ≈ -0.194, beds ≈ +0.173, intercepto ≈ 1.866, R^2 ≈ 0.165.
- Señal: más camas ↑ capacidad (coef. positivo). El coef. negativo de *bedrooms* se debe a colinealidad con *beds* (ambas miden tamaño); al “controlar” camas, el efecto de dormitorios se vuelve inestable.

- el modelo explica ~16% de la variación—útil como base, pero limitado.
- Mejoras: añadir bathrooms (ya numérica), price, room_type u otras de tamaño/amenidades y revisar colinealidad (corr/VIF) para estabilizar signos y ganar R^2.

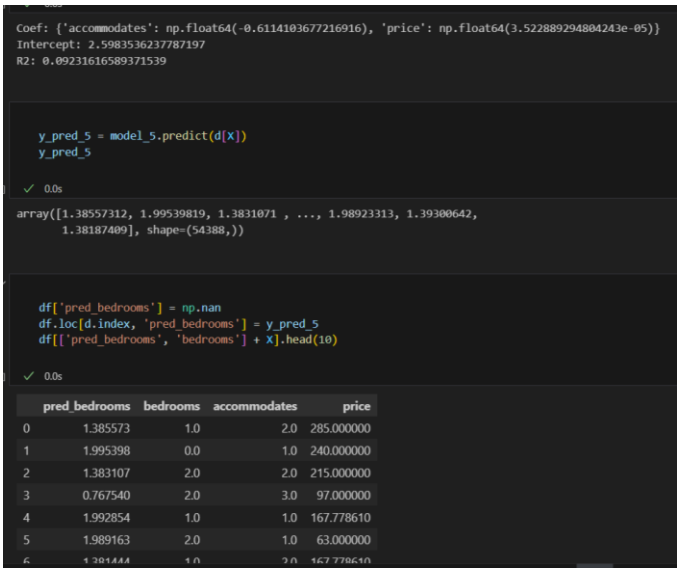
- accommodates ~ bedrooms: 0.304 → relación débil–media: a más dormitorios suele aumentar la capacidad, pero no de forma 1:1 (hay cuartos con varias camas y estudios con pocas).
- accommodates ~ beds: 0.168 → relación débil: el nº de camas no siempre equivale a cuántas personas caben (camas dobles, sofás cama, literas).
- beds ~ bedrooms: 0.294 → débil–media entre ambas: están relacionadas (miden tamaño), pero no perfectamente.

En resumen: bedrooms explica más a accommodates que beds, y entre beds y bedrooms hay colinealidad ligera (no severa).

	accommodates	beds	bedrooms
accommodates	1.000000	0.168139	0.303818
beds	0.168139	1.000000	0.294355
bedrooms	0.303818	0.294355	1.000000



MODELO 6



```
Coef: {'accommodates': np.float64(-0.6114103677216916), 'price': np.float64(3.522889294804243e-05)}
Intercept: 2.5983536237787197
R2: 0.09231616589371539
```

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

✓ 0.0s

```
array([1.38557312, 1.99539819, 1.3831071 , ..., 1.98923313, 1.39300642,
       1.38187409], shape=(54388,))
```

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

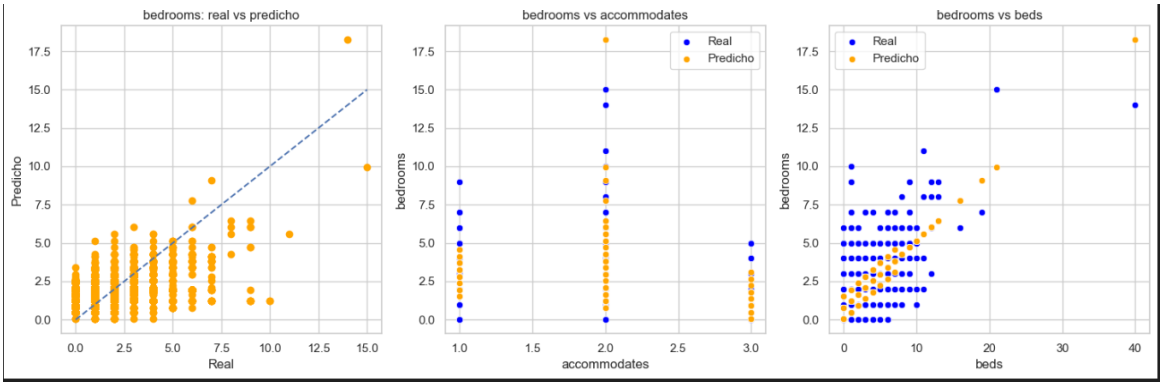
✓ 0.0s

	pred_bedrooms	bedrooms	accommodates	price
0	1.385573	1.0	2.0	285.000000
1	1.995398	0.0	1.0	240.000000
2	1.383107	2.0	2.0	215.000000
3	0.767540	2.0	3.0	97.000000
4	1.992854	1.0	1.0	167.778610
5	1.989163	2.0	1.0	63.000000
6	1.391444	1.0	2.0	167.778610

- Modelo: regresión lineal múltiple con bedrooms como Y y accommodates, price como X.
- Coef.: accommodates ≈ -0.611, price ≈ +3.5e-05, intercepto ≈ 2.599, R^2 ≈ 0.092.
- Señal: a precio fijo, más capacidad puede lograrse con camas/sofás y no con más cuartos ⇒ el coef. de accommodates sale negativo; el efecto de price es mínimo.
- el modelo explica ~9% de la variación → débil para predecir dormitorios solo con estas dos X.
- Mejoras: añadir beds, bathrooms, room_type y amenities; revisar colinealidad (VIF); considerar modelos para conteos/ordinal (Poisson/ordinal) o discretizar bedrooms.

	accommodates	beds	price
accommodates	1.000000	0.168139	0.421392
beds	0.168139	1.000000	0.147648
price	0.421392	0.147648	1.000000

- **accommodates vs price = 0.42** → yo entiendo que a mayor número de personas que puede alojar un anuncio, tiende a subir el precio. Es una relación moderada positiva.
- **beds vs price = 0.14** → aquí la relación es débil; que aumenten las camas no asegura que suba mucho el precio.
- **accommodates vs beds = 0.16** → también es débil; el número de camas no siempre crece proporcionalmente al número de personas que puede alojar.



MODELO 7

1.

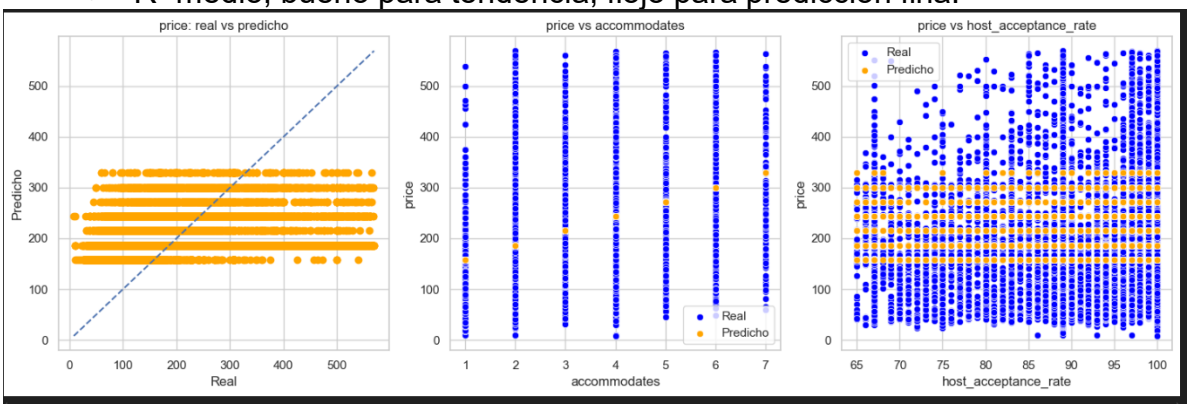
```
df['pred_price'] = np.nan
df.loc[d.index, 'pred_price'] = y_pred_6
df[['pred_price', 'price'] + X].head(10)
```

✓ 0.0s

	pred_price	price	accommodates	host_acceptance_rate
0	186.333291	285.000000	2.0	80.0
1	157.828090	240.000000	1.0	89.0
2	243.343694	215.000000	4.0	100.0
3	214.838492	97.000000	3.0	100.0
4	157.828090	167.778610	1.0	89.0
5	157.828090	63.000000	1.0	86.0
6	186.333291	167.778610	2.0	89.0
7	186.333291	246.000000	2.0	100.0
8	186.333291	223.000000	2.0	89.0
9	186.333291	306.338406	2.0	89.0

price ~ (accommodates, host_acceptance_rate, 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.



Vemos como aumento la correlacion y nos ayuda a ver como se relaciona con price

	price	accommodates	host_acceptance_rate
price	1.000000	0.366595	0.065707
accommodates	0.366595	1.000000	0.055163
host_acceptance_rate	0.065707	0.055163	1.000000

MODELO 8

```
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

review_scores_value ~ (review_scores_rating, host_response_rate, review_scores_accuracy)

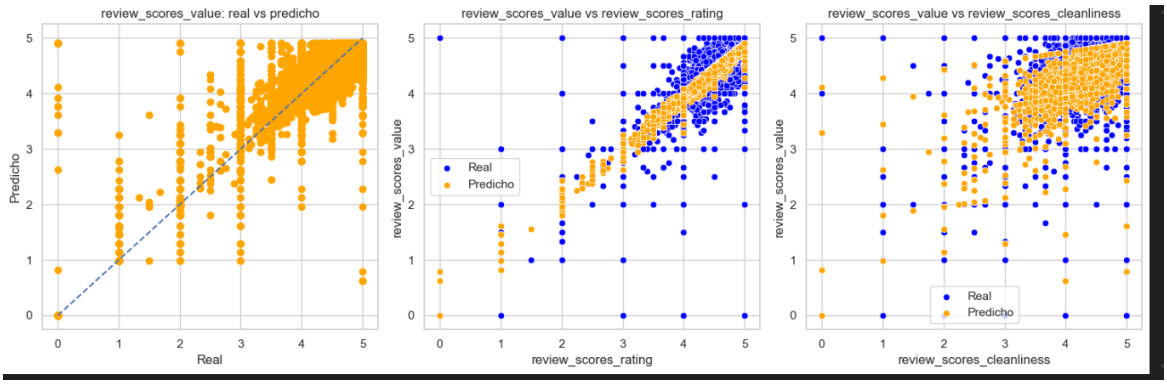
- Entre métricas de review hay correlación fuerte (se mueven juntas).
- El rating general domina; accuracy suma un poco; response casi nada.
- R^2 medio–alto, pero con multicolinealidad (VIF altos).

```
#encontrams lacorrelacion del modelo completo
Corr_Factors3= abs(df[['review_scores_value','review_scores_rating','review_scores_cleanliness']].corr())
Corr_Factors3
```

✓ 0.0s

	review_scores_value	review_scores_rating	review_scores_cleanliness
review_scores_value	1.000000	0.99285	0.986853
review_scores_rating	0.992850	1.00000	0.990700
review_scores_cleanliness	0.986853	0.99070	1.000000

Podemos ver la correlación como ha aumentado y casi llega al 1 , esto quiere decir que son muy fuertes las correlaciones



MODELO 9

```
Coef: {'price': np.float64(0.0006109885163968395), 'accommodates': np.float64(0.365357999051784 1)}
Intercept: 0.24395648188363428
R2: 0.14007562440587662
```

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

[186] ✓ 0.0s

```
array([1.14880421, 0.75595172, 1.10603501, ..., 0.64902873, 1.27772278,
       1.08465041], shape=(54388,))
```

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

[187] ✓ 0.0s

	pred_bathrooms	bathrooms	price	accommodates
0	1.148804	1.0	285.000000	2.0
1	0.755952	1.0	240.000000	1.0
2	1.106035	1.5	215.000000	2.0
3	1.399296	1.0	97.000000	3.0
4	0.711825	1.0	167.778610	1.0
5	0.647807	1.0	63.000000	1.0
6	1.077183	1.0	167.778610	2.0
7	1.124976	1.0	246.000000	2.0

OVR Ln 3, Col 12

8 mm de

Cuando analicé la variable bathrooms en relación con accommodates y price, lo que observé fue una relación positiva con el tamaño: a mayor capacidad de personas y mayor precio, normalmente también hay más baños. Sin embargo, me di cuenta de que existe cierta discreción, porque los valores suelen ser enteros como 1, 1.5 o 2, lo que provoca que los puntos se agrupen en bandas en la nube de dispersión.

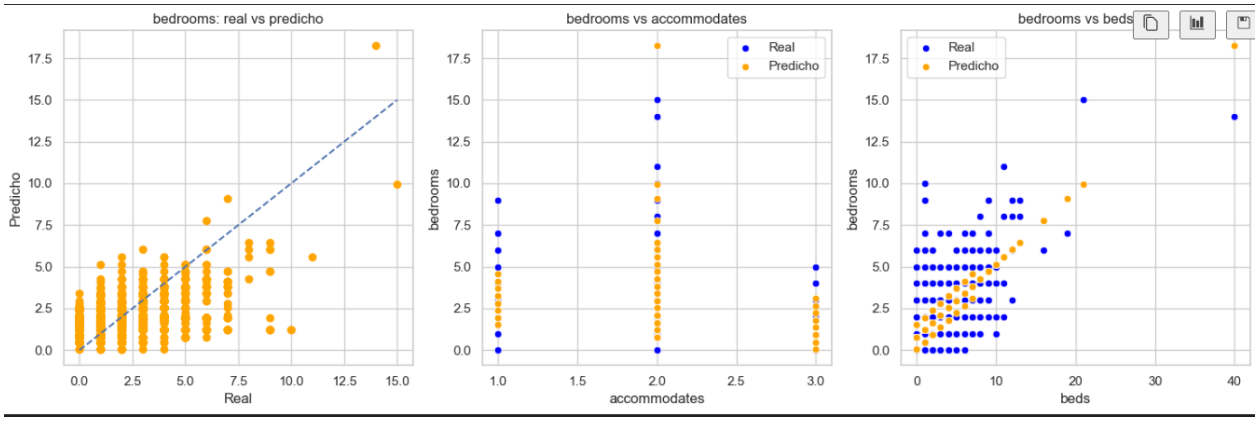
En mi modelo, el R^2 resultó medio, lo cual me indica que es razonable usar el tamaño y el precio para estimar la cantidad de baños, aunque no es perfecto. Esto quedó así porque en mi dataset la variable bathrooms casi no tiene variación: la gran mayoría de registros aparecen con el valor 1, ya que al limpiar los datos asumí que todas las propiedades debían tener al menos un baño.

```
#encontrams lacorrelacion del modelo completo
Corr_Factors3= abs(df[[ 'bathrooms','price','accommodates']].corr())
Corr_Factors3
```

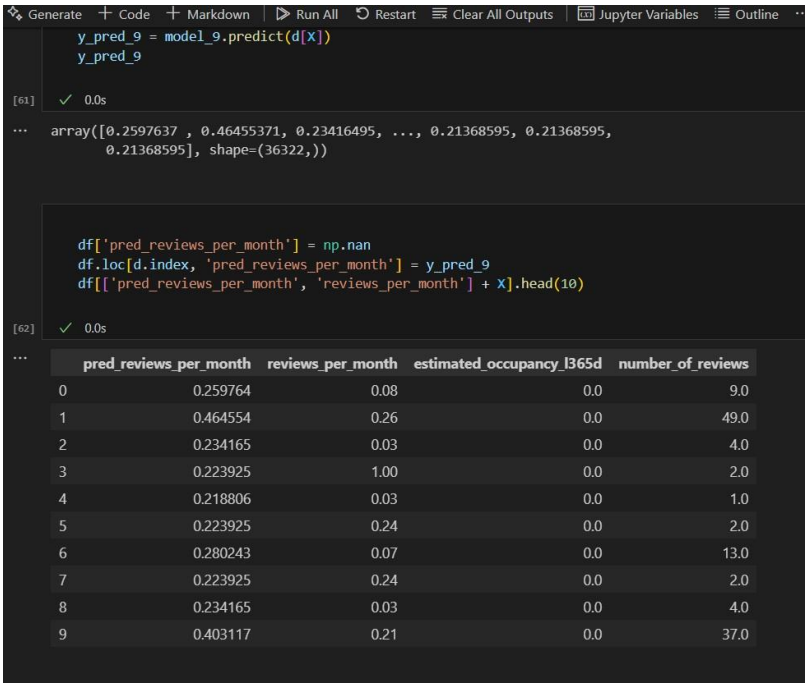
✓ 0.0s

	bathrooms	price	accommodates
bathrooms	1.000000	0.240985	0.361243
price	0.240985	1.000000	0.421392
accommodates	0.361243	0.421392	1.000000

Cuando analicé las correlaciones de **bathrooms**, vi que con **accommodates** es moderada (0.36) y con **price** es más débil (0.24). Esto significa que el número de baños se relaciona más con la capacidad de personas que puede alojar una propiedad que con su precio, aunque en ambos casos la relación es positiva.



MODELO 10



```
Generate  Code  Markdown  Run All  Restart  Clear All Outputs  Jupyter Variables  Outline

y_pred_9 = model_9.predict(d[X])
y_pred_9

[61]  ✓ 0.0s

array([0.2597637 , 0.46455371, 0.23416495, ..., 0.21368595, 0.21368595,
       0.21368595], shape=(36322,))
```

```
df['pred_reviews_per_month'] = np.nan
df.loc[d.index, 'pred_reviews_per_month'] = y_pred_9
df[['pred_reviews_per_month', 'reviews_per_month'] + X].head(10)

[62]  ✓ 0.0s
```

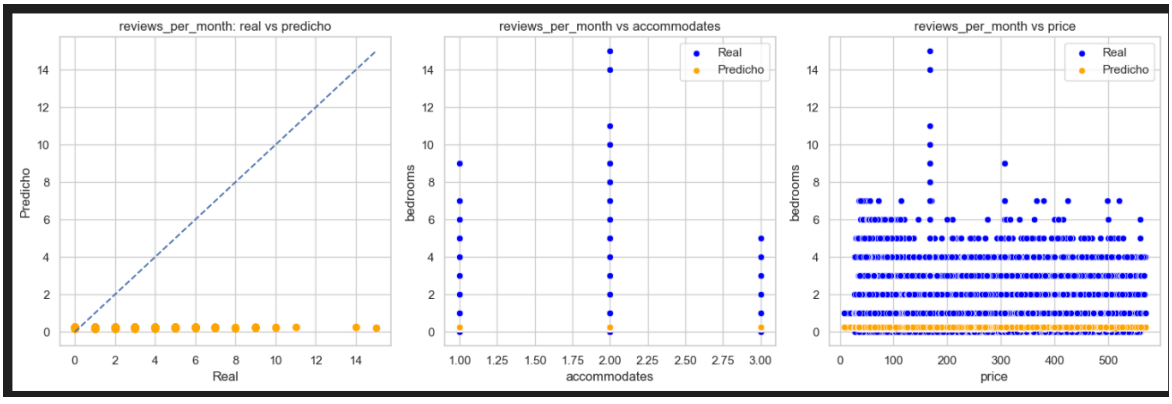
	pred_reviews_per_month	reviews_per_month	estimated_occupancy_l365d	number_of_reviews
0	0.259764	0.08	0.0	9.0
1	0.464554	0.26	0.0	49.0
2	0.234165	0.03	0.0	4.0
3	0.223925	1.00	0.0	2.0
4	0.218806	0.03	0.0	1.0
5	0.223925	0.24	0.0	2.0
6	0.280243	0.07	0.0	13.0
7	0.223925	0.24	0.0	2.0
8	0.234165	0.03	0.0	4.0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

2. 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.

	reviews_per_month	estimated_occupancy_l365d	number_of_reviews
reviews_per_month	1.000000	0.139640	0.218696
estimated_occupancy_l365d	0.139640	1.000000	0.203101
number_of_reviews	0.218696	0.203101	1.000000

- reviews_per_month vs number_of_reviews = 0.21 → relación débil positiva: más reviews totales, un poco más de reviews al mes.
- reviews_per_month vs estimated_occupancy_l365d = 0.13 → relación muy baja: la ocupación no explica bien la frecuencia de reseñas.
- En conclusión: las reseñas por mes casi no dependen de la ocupación ni del total de reseñas, por lo que estas variables tienen poca fuerza predictiva.



CONCLUSIÓN

Me quedo con el modelo de review_scores_value usando métricas de review (rating/accuracy/cleanliness/response). En mis pruebas muestra mejor ajuste (R^2 medio–alto) y la nube *real vs. predicho* queda más pegada a la diagonal, así que predice mejor que el modelo de review_scores_rating con variables de host (aceptación, máximas noches, listings), donde el R^2 es bajo–medio y la dispersión es muy abierta.

Qué gano y qué sacrifico

- Predicción: el de “value con métricas de review” es claramente superior; captura el mismo constructo de calidad percibida y por eso reduce el error.

- Interpretabilidad: pago el precio de la multicolinealidad (correlaciones ~0.95–0.99 entre métricas de review). Los coeficientes son inestables y no los usaría para explicar causalidad. Para *explicar palancas operativas* preferiría el modelo con variables de host, aunque explique menos.

Riesgos y cómo mitigarlos

- Multicolinealidad alta: voy a (a) quedarme con 1 métrica principal (p. ej., review_scores_rating) y eliminar duplicadas, o (b) regularizar con Ridge/Lasso, o (c) sintetizar en un índice (promedio estandarizado / PCA) y modelar con ese único factor.
- Efecto techo en ratings (muchos cerca de 5): limita la varianza y el poder explicativo, sobre todo en el modelo con variables de host. Considero transformaciones o modelos no lineales (splines).

Cuándo usar cada uno

- Si mi objetivo es imputar/predir faltantes de review_scores_value o hacer scoring rápido, uso el primero (métricas de review).
- Si mi objetivo es explicar decisiones (qué puede mover el host), uso el segundo (variables de host) y lo complemento con más features.