Análisis Predictivo TP2

Profesores:

- Ezequiel Martín Eliano Sombory
- Leonardo Andrés Caravaggio
- Francisco Valentin

Nicolás Rodrigues da Cruz

Score obtenido en Kaggle

8	—	NICOLAS RODRIGUES DA CR UZ		0.68842	22	5d

Índice de contenidos

ITBA

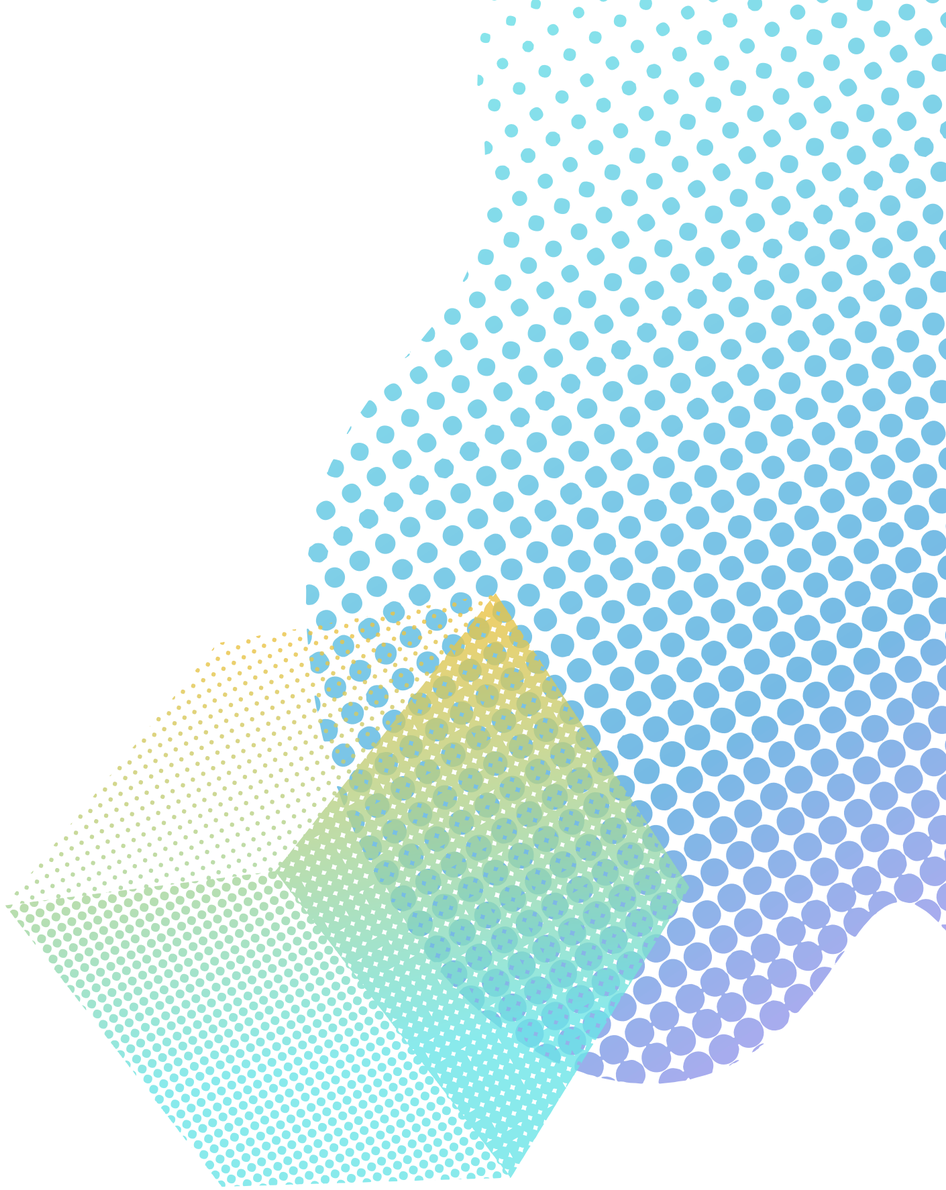
01. Vistazo general

Caracteristicas	Cantidad
Filas	4928
Columnas	68
Duplicados	0

01. Vistazo general

```
data_types = train.dtypes.value_counts()
print(data_types)
```

```
object      29
int64       22
float64     17
```



ITBA

02. Tratamiento de Nulos y Outliers

	Cantidad de Valores Nulos
neighbourhood_group_cleansed	4928
bathrooms	4928
calendar_updated	4928

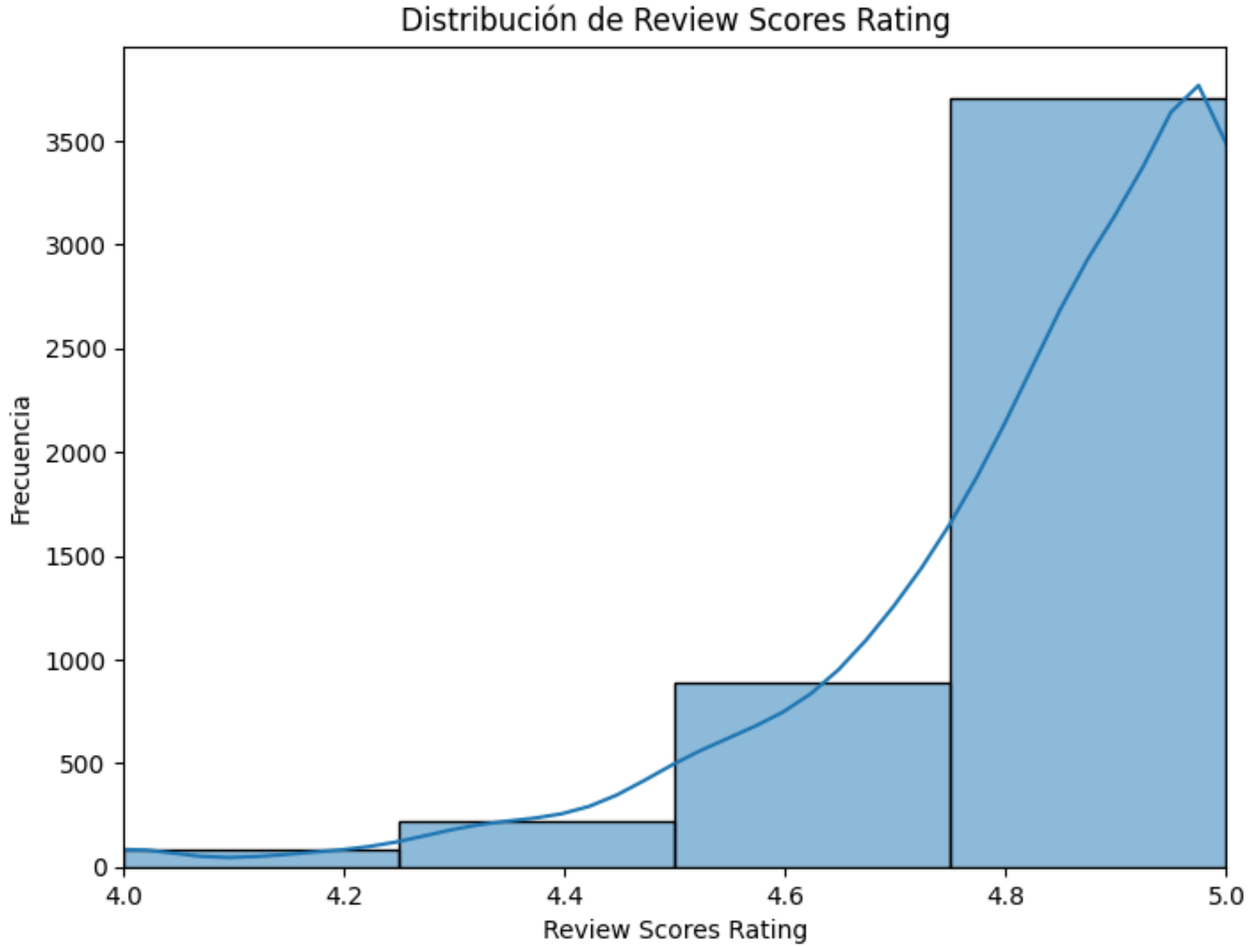
→ Se debía a un error de la base

- Para imputar los valores nulos se utilizó el método de imputación por k-vecinos más cercanos

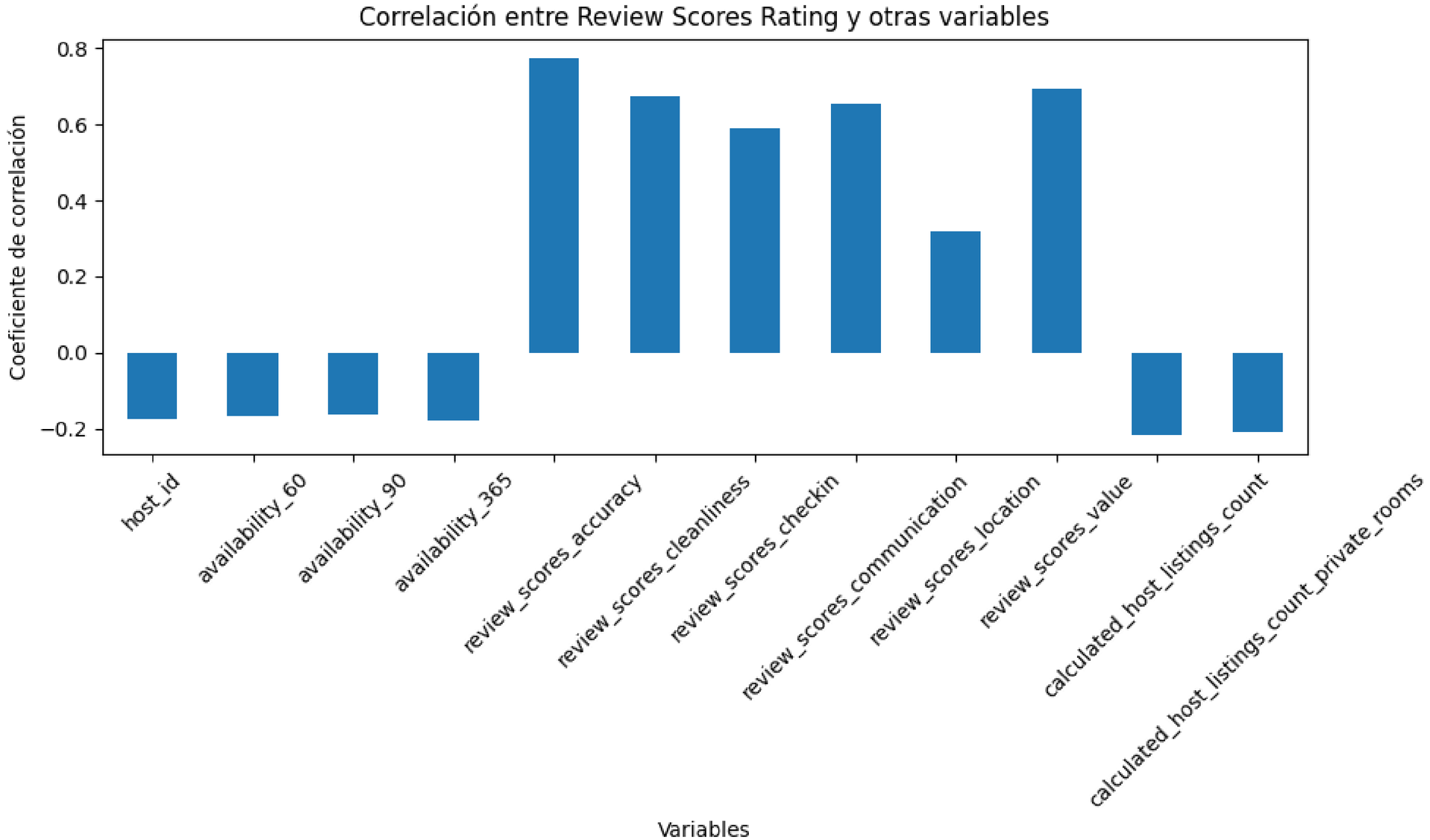
- En cuanto a los outliers no se encontraron valores que carezcan de sentido dentro de las distintas variables por lo que no fueron modificados

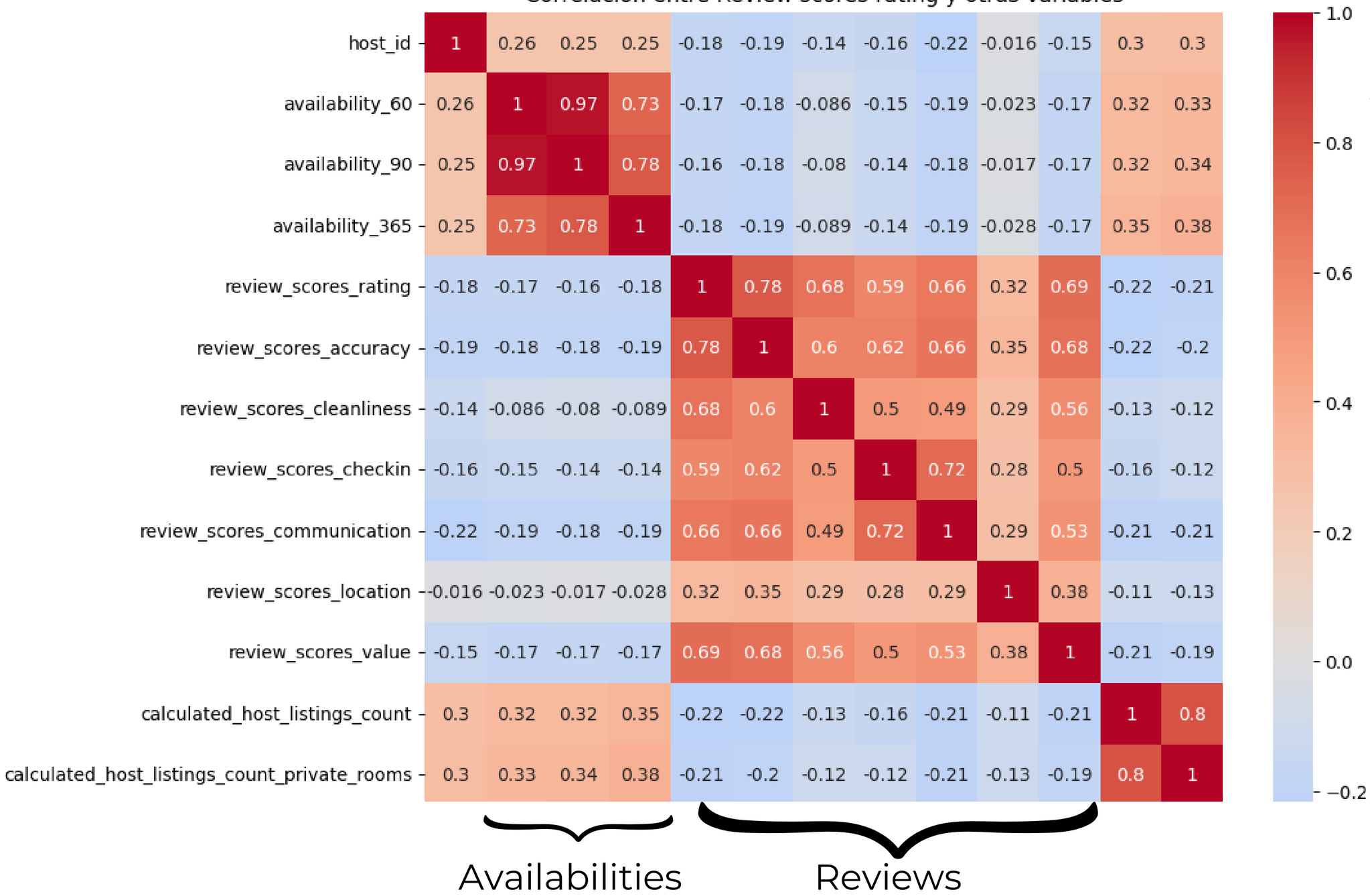
03. Variable objetivo

mean	4.812330
std	0.264727
min	0.000000
25%	4.750000
50%	4.880000
75%	4.990000
max	5.000000



03. Variable objetivo




Correlación entre Review scores rating y otras variables
host_id 1 0.26 0.25 0.25 -0.18 -0.19 -0.14 -0.16 -0.22 -0.016 -0.15 0.3 0.3
availability_60 0.26 1 0.97 0.73 -0.17 -0.18 -0.086 -0.15 -0.19 -0.023 -0.17 0.32 0.33
availability_90 0.25 0.97 1 0.78 -0.16 -0.18 -0.08 -0.14 -0.18 -0.017 -0.17 0.32 0.34
availability_365 0.25 0.73 0.78 1 -0.18 -0.19 -0.089 -0.14 -0.19 -0.028 -0.17 0.35 0.38
review_scores_rating -0.18 -0.17 -0.16 -0.18 1 0.78 0.68 0.59 0.66 0.32 0.69 -0.22 -0.21
review_scores_accuracy -0.19 -0.18 -0.18 -0.19 0.78 1 0.6 0.62 0.66 0.35 0.68 -0.22 -0.2
review_scores_cleanliness -0.14 -0.086 -0.08 -0.089 0.68 0.6 1 0.5 0.49 0.29 0.56 -0.13 -0.12
review_scores_checkin -0.16 -0.15 -0.14 -0.14 0.59 0.62 0.5 1 0.72 0.28 0.5 -0.16 -0.12
review_scores_communication -0.22 -0.19 -0.18 -0.19 0.66 0.66 0.49 0.72 1 0.29 0.53 -0.21 -0.21
review_scores_location -0.016 -0.023 -0.017 -0.028 0.32 0.35 0.29 0.28 0.29 1 0.38 -0.11 -0.13
review_scores_value -0.15 -0.17 -0.17 -0.17 0.69 0.68 0.56 0.5 0.53 0.38 1 -0.21 -0.19
calculated_host_listings_count 0.3 0.32 0.32 0.35 -0.22 -0.22 -0.13 -0.16 -0.21 -0.11 -0.21 1 0.8
calculated_host_listings_count_private_rooms 0.3 0.33 0.34 0.38 -0.21 -0.2 -0.12 -0.12 -0.21 -0.13 -0.19 0.8 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
Availabilities
Reviews

04. Creacíón de variables

Se crean nuevas variables binarias para:

1. Las variables true o false donde t=1 y f=0:

host_is_superhost_b, host_has_profile_pic_b, host_identity_verified_b, has_availability_b, instant_bookable_b

2. Aquellas que tienen dos unicos posibles valores::

- calendar_last_scraped_b donde '2022-12-05': 1, '2022-12-17': 0
- source_b donde 'city scrape': 1, 'previous scrape': 0

3. Las variables que estan compuestas por cadenas de texto largas si es null=0 o no=1:

has_description_b, has_neighborhood_b, has_neighbourhood_overview_b, has_host_neighbourhood_b, has_host_about_b

```
t_f_mapping = {
    't': 1, 'f': 0
    }
train['host_is_superhost_b'] = train['host_is_superhost'].replace(t_f_mapping)
train['host_has_profile_pic_b'] = train['host_has_profile_pic'].replace(t_f_mapping)
train['host_identity_verified_b'] = train['host_identity_verified'].replace(t_f_mapping)
train['has_availability_b'] = train['has_availability'].replace(t_f_mapping)
train['instant_bookable_b'] = train['instant_bookable'].replace(t_f_mapping)
```

```
[ ] train['source_b'] = train['source'].map({'city scrape': 1, 'previous scrape': 0})
    train['calendar_last_scraped_b'] = train['calendar_last_scraped'].map({'2022-12-05': 1, '2022-12-17': 0})
```

04. Modificación de variables

```
train['price'] = train['price'].str.replace('$', '')
train['price'] = train['price'].str.replace(',', '')
train['price'] = pd.to_numeric(train['price'])
```

```
[18] train['host_acceptance_rate'] = train['host_acceptance_rate'].str.rstrip('%').astype(float) / 100
     train['host_response_rate'] = train['host_response_rate'].str.rstrip('%').astype(float) / 100
```

04. Creacíón de variables

Se crearon tambien variables como:

host_since_year
cant_amenities
antiguedad
host_listings_ratio
host_response_ratio
log_price
minimum_maximum_nights_ratio
description_sentiment

04. Creación de variables

Y para las variables que tenian varios valores posibles se realizo frequency encoding:

- host_location
- host_neighbourhood
- neighbourhood
- neighbourhood_cleansed
- property_type
- room_type
- bathrooms_text
- host_city_or_state
- host_country

```
[ ]  train['room_type_num'] = train['room_type'].map({'Private room': 1, 'Entire home/apt': 2, 'Hotel room': 3, 'Shared room': 4})
     train['bathrooms_text_num'] = train['bathrooms_text'].map({'shared baths': 1, 'shared bath': 1, 'baths': 2, 'bath': 2, 'private bath': 3})
```

```
[ ]  from sklearn.preprocessing import LabelEncoder
     neighbourhood_encoder = LabelEncoder()
     train['neighbourhood_cleansed_encoded'] = neighbourhood_encoder.fit_transform(train['neighbourhood_cleansed'])
```

05. Modelos y predicciones

- Se utilizaron como tamaño de las particiones a 0.15 y 0.2, ya que se encontro que eran los mejores

- Se utilizó el metodo GridSearchCV de la libreria sklearn.model_selection como método de ajuste

  - Modelos utilizados: XGBoost, Random Forest y Regresion Ridge

05. Modelos y predicciones

Random Forest

```
Puntajes de validación cruzada: [0.76238567 0.74664083 0.75283926 0.7820809  0.52877466]
Puntuación promedio de la validación cruzada: 0.714544264652457
{'max_depth': None, 'min_samples_split': 10, 'n_estimators': 100}
```

05. Modelos y predicciones

XBoost

```
Puntajes de validación cruzada: [0.78971143 0.76931554 0.6581036  0.730894   0.77296159]
Score: 0.744197232974964 4
```

05. Modelos y predicciones

Regresion Ridge

RegRidgesubmission.csv
Complete · 13d ago

0.72846 0.69743

Gracias