UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

PREDICCIÓN DE PRECIOS EN ALOJAMIENTOS DE AIRBNB MEDIANTE UN MODELO DE APRENDIZAJE MÁQUINA BASADO EN LIGHTGBM

TESIS

PARA OBTENER EL TÍTULO DE:

LICENCIADO EN MATEMÁTICAS APLICADAS

PRESENTA:

ABISAI JARQUÍN MARTÍNEZ

DIRECTOR DE TESIS:

Dr. TOMÁS PÉREZ BECERRA

HUAJUAPAN DE LEÓN, OAXACA *OCTUBRE DE 2024*

Índice general

Índice de figuras

lista de tabla

Introducción

Con el paso del tiempo, los lugares de hospedaje han tomado un papel importante en la sociedad, llegando así a convertirse actualmente en una necesidad. Con este apogeo, dicha actividad ha ido cambiando a lo largo del tiempo, generando que durante la época de vacaciones en la mayoría de hoteles no haya más espacio para hospedarse.

Ante esta problemática, en 2008, Brian Chesky, Joe Gebbia y Nathan Blecharczyk, decidieron rentar su habitación para obtener algo de ingresos durante el mes, dicha actividad logró que ambos salieran beneficiados; pues el coste de hospedaje era mucho menor que en un hotel y el arrendador tuvo ganancias extras. Dando así inicio a Airbnb; una compañía que ofrece una plataforma digital dedicada a la oferta de alojamientos a particulares y turísticos ([7]).

El modelo de negocio era sencillo: los propietarios o anfitriones publicaban un anuncio ofreciendo sus habitaciones disponibles; las personas que debían asistir a algún evento tendrían la posibilidad de encontrar alojamiento económico con desayuno cerca a su destino; y la web cobraría una comisión por cada reserva que se concretara. Además, como elemento diferencial con respecto al hospedaje tradicional, dieron un enfoque de comunidad al servicio, marcándose como objetivos lograr que desconocidos se hicieran amigos y contribuir a que la gente se sintiera como en casa en cualquier lugar ([3]).

Si bien, el uso de Airbnb resulta ser cómodo, económico y sencillo de usar, no es fácil deducir cuánto costará algún alojamiento en el futuro, si bajará o subirá de precio. En este caso, la ciencia de datos es una herramienta muy útil para determinar estas tendencias.

Las disciplinas directamente relacionadas con la ciencia de datos, que ha alcanzado su punto máximo en los últimos años debido a la diversidad de disciplinas que aglutina, son la estadística, inteligencia artificial y métodos numéricos. Esta se ha convertido en una herramienta para analizar y manipular datos para desarrollar

soluciones alternativas a los problemas de formas más sencillas, como por ejemplo mediante el aprendizaje profundo. Por lo tanto, la ciencia de datos resulta ser una muy útil herramienta para la predicción de costos a futuro, específicamente, en el incremento o disminución de lo precios en alojamientos Airbnb mediante el aprendizaje máquina.

La ciencia de datos es un campo interdisciplinario que incluye elementos de matemáticas, estadística y tecnología de la información, para extraer conocimiento y perspectivas valiosas de conjuntos de datos. Con el crecimiento exponencial de la cantidad de datos generados en la era digital, esta área se ha convertido en una disciplina crucial para comprender, analizar y tomar decisiones basadas en datos en diversos campos.

El objetivo principal de la ciencia de datos es descubrir patrones, tendencias y relaciones ocultas en los datos, lo que permite tomar decisiones informadas y obtener una ventaja competitiva en diferentes ámbitos. Se emplean una combinación de habilidades técnicas y conocimientos especializados en el tema para abordar problemas complejos, desde la predicción del comportamiento del consumidor hasta la optimización de procesos industriales.

El proceso de ciencia de datos generalmente sigue varios pasos, que incluyen la identificación del problema, la recopilación y preparación de los datos, el análisis exploratorio, la construcción de modelos y algoritmos, y la interpretación de los resultados. Estos pasos a menudo implican el uso de técnicas y herramientas como el aprendizaje automático (*machine learning*), la minería de datos, la visualización de datos y la programación.

Esta investigación se enmarca dentro del área de ciencia de datos y mercadotecnia, específicamente, se analiza la variación de los precios que los alojamientos de Airbnb presentan en la ciudad de Nueva York, mediante procesos de minería de datos y el aprendizaje máquina.

El objetivo general es predecir los precios futuros de algún alojamiento mediante un programa de aprendizaje máquina, la base de datos que se utilizará es la llamada New York Airbnb_4 dec 2021.csv, la cual describe las actividades en métricas de huéspedes y anfitriones que han utilizado Airbnb en forma de alojamiento. El archivo de datos incluye la información sobre los huéspedes y la disponibilidad geográfica y temporal para hacer predicciones y obtener conclusiones. Es de dominio público y la original proviene del sitio `http://insideairbnb.com/`, la utilizada en este trabajo es obtenida de

`https://www.kaggle.com/datasets/sirapatsam/airbnb-new-york-4dec2021`.

Airbnb se ha convertido en una plataforma líder para el alquiler de alojamientos en todo el mundo. La fijación de precios es una parte fundamental de la experiencia tanto para los anfitriones como para los huéspedes. La capacidad de establecer precios adecuados no solo afecta la rentabilidad de los anfitriones, sino también la elección de alojamiento para los huéspedes. En este contexto, la predicción de precios se convierte en un elemento clave para optimizar la satisfacción de ambas partes.

El problema de esta investigación se centra en la necesidad de desarrollar un modelo matemático de predicción de precio preciso y efectivos en Airbnb, ya que los existentes presentan errores relativamente altos y no consideran la base Airbnb propuesta en esta investigación[2]. El modelo propuesto es de tipo LigthGBM, el cual es un algoritmo propuesto por Guolin Ke, Qi Meng y sus colaboradores, aparece por primera vez en [5] en el año 2017. En síntesis, este problema se desglosa con las siguientes dimensiones planteadas en forma de interrogantes:

- La *Variabilidad de Precios:* ¿Cómo varían los precios de los alojamientos en Airbnb en función de factores como la ubicación, el tipo de propiedad, la temporada y la demanda? ¿Cuáles son las principales variables que influyen en la fijación de precios?

- *Modelos de Predicción:* ¿Qué enfoques de modelado y algoritmos son más efectivos para predecir los precios de los alojamientos en Airbnb? ¿Cómo se pueden combinar datos históricos, características del alojamiento y datos externos para mejorar la precisión de las predicciones? ¿El modelo LigthGBM tendrá un mejor desempeño que los existentes en la literatura?

A manera de respuesta, esta tesis se enfocará en la realización de un modelo de predicción de precios de tipo LigthGBM basado en la teoría matemática de los modelos de árboles de decisión y, además, se aplicaran diversas técnicas de ciencia de datos, con lo que se pretende proporcionar predicciones más efectivas que aquellas dadas por otros modelos existentes en la literatura. Este programa se realizará sólo hasta la fase de simulación y se espera una tasa de efectividad mayor del 80 % con un tiempo de ejecución relativamente corto, posteriormente a este trabajo, se podrá implementar su automatización en alguna interfaz de programa automático. Para lograrlo, en esta investigación se realizarán las siguientes etapas:

Capitulo 1 Se realizará una investigación sobre las actuales tendencias en lugares de renta de Airbnb. Además, se realizará un análisis exploratorio de datos a la base de datos (también llamada directorio) *Airbnb New York @ 4.DEC.2021* el cual consiste en limpiar el directorio, identificar entradas faltantes o, en su caso, realizar una reducción de dimensionalidad.

Capitulo 2 Se diseñará un programa de aprendizaje máquina de tipo LightGBM y se entrenará con el directorio *Airbnb New York @ 4.DEC.2021* para que sea capaz de identificar patrones y realizar estimaciones de costos de alojamientos en renta, así, posteriormente informar al usuario sobre el precio que tendrá un alojamiento en el futuro a corto plazo, específicamente, dentro de los cuatro meses posteriores.

A manera de justificación, el desarrollo de este estudio permitirá conectar el análisis exploratorio de datos y el aprendizaje máquina para desarrollar un modelo predictivo que informará sobre los precios de un alojamiento en un futuro. Esto mejorará la eficiencia en la predicción. Específicamente, se tienen los siguientes alcances:

1. Conveniencia: esta investigación tendrá una utilidad directa en la toma de decisiones del público en general, usuarios de Airbnb, sobre el precio que tendrá un lugar en el corto plazo.

2. Relevancia social: Actualmente, la población está recurriendo a medios digitales para realizar sus reservaciones con mucho tiempo de anticipación, sin saber el costo que esta tendrá para cuando les corresponda pagar el alquiler, por lo que la sociedad que hace uso de estos lugares de renta será beneficiada de los resultados de esta investigación al implementar un modelo matemático predictivo.

3. Implicaciones prácticas: Esta investigación impacta de forma directa en la perdición de precios de alojamientos del tipo Airbnb contribuyendo a proporcionar información sobre los costos de estos lugares en el futuro.

4. Valor teórico: Los resultados obtenidos brindarán la posibilidad de sugerir recomendaciones en cuanto al valor de renta de los alquileres mediante la implementación de algoritmos predictivos que favorezcan a los usuarios. Además, el modelo contará con las fundamentos que brinda la modelación matemática para ciencia de datos.

5. Utilidad metodológica: La investigación mejorará la implementación del análisis de datos y aprendizaje máquina en la predicción de precios.

Capítulo 1

Exploración y análisis de la base

El objetivo principal de esta sección es analizar a profundidad la base de datos mediante técnicas de visualización. Además, se realiza el pre-procesamiento que incluye la imputación y normalización de las variables, parte importante para estandarizar la base con la que se entrena el modelo matemático en el siguiente capítulo.

1.1. Exploración de los datos

En una exploración inicial a la base, se observa que contiene las variables mostradas en la Tabla 1.1, cuyas entradas son datos cuantitativos enteros, flotantes, cualitativos textuales y una variable contiene cadenas de texto, además, muestra registros de tiempo, por lo tanto se debe considerar de tipo temporal, específicamente, la variable *last_review*. Por lo tanto, la base es considerada mixta y, formalmente, se define a continuación:

Definición 1.1. *(Base de datos multidimensional) Un base de datos multidimensional D es un conjunto de n registros $\bar{X}_1, \ldots, \bar{X}_n$, tales que cada registro $\bar{X}_i$ contiene un conjunto de d características denotadas por $(x_i^1, \ldots, x_i^d)$.*

Para este caso, $n = 38,277$ y $d = 18$. En general, a la base D se le puede considerar como una matriz de dimensión $n \times d$. En la Tabla 1.2 se muestran los tipos de características de la base D junto con una descripción breve.

Una variable de interés es *last review*, que es una variable textual que contiene la fecha de la última revisión del alojamiento, por lo que debemos de convertirla a una variable de tipo estampa de tiempo, esto es, se transforma la fecha textual en

Tipos de datos	
Columna	Tipo
id	Entero
name	Texto
host id	Entero
host name	Texto
neighbourhood group	Texto
neighbourhood	Texto
latitude	Flotante
longitude	Flotante
room type	Texto
price	Entero
minimum nights	Entero
number of reviews	Entero
last review	Texto
reviews per month	Flotante
calculated host listings count	Entero
availability 365	Entero
number of reviews ltm	Entero
license	Texto

Tabla 1.1: Tipos de características.

una numérica, por lo que ahora la variable *last review* se convierte en *timestamp* o de tipo *datetime*.

Por otro lado, parte relevante en la estadística descriptiva son las medidas de tendencia central de las variables numéricas. Por ejemplo, la media de cada j-ésima característica, $j = 1, \ldots, d$, se calcula como

$$\mu_j = \frac{\sum_{k=1}^{n} x_k^j}{n};$$

la varianza se calcula como

$$\sigma_j^2 = \frac{\sum_{k=1}^{n} (x_k^j - \mu_j)^2}{n-1};$$

Descripción de columnas	
Columna	Descripción
id	Identificador de alojamiento de sitio web
name	Nombre del alojamiento
host id	Identificador del anfitrión
host name	Nombre del anfitrión
neighbourhood group	Ciudad en la que se encuentra el alojamiento
neighbourhood	Vecindario donde se encuentra el alojamiento
latitude	Distancia en grados, minutos y segundos con respecto al ecuador
longitude	Distancia en grados, minutos y segundos con respecto al meridiano
room type	Tipo de habitación o lugar que se desea rentar
price	Precio por una noche en el lugar
minimum nights	Noches mínimas para poder hospedarse en
number of reviews	Revisiones que recibe el lugar en la aplicación
last review	Fecha de la última revisión del lugar en la aplicación
reviews per month	Promedio de revisiones por mes del lugar en la aplicación
calculated host listings count	Recuento de listados de anfitriones calculado
availability 365	Días que está disponible el alojamiento durante el año
number of reviews ltm	Número de veces en que se revisó en los últimos 12 meses
license	Variable del tipo objeto, la cual no está bien definida

Tabla 1.2: Tipos de características.

y la desviación estándar es

$$\sigma = \sqrt{\sigma^2}.$$

A través de ellas, se derivan las siguientes conclusiones:

1. El precio en promedio de los alojamientos es de 170.85 dólares, con una desviación estándar de 305 dólares, esto es, los precios están dispersos de la

media.

2. Las medias de la longitud y la latitud son de 40.729206 y de -73.948967, respectivamente, este punto geográfico se puede considerar como el baricentro de las locaciones. Cabe mencionar que las desviaciones estándar respectivas son pequeñas, 0.055752 para la longitud y 0.050759 para la latitud. Lo que proporciona una idea de la ubicación con una mayor demanda.

3. El número mínimo de noches en los que se alquilan los alojamientos es de 21, aunque la desviación estándar es de 29.57 noches.

4. La variable *availability_365* muestra que en promedio cada anuncio estará disponible por 134 días con una desviación estándar de 143 días.

1.2. Imputación de datos perdidos y limpieza

Para ser específicos, los problemas de datos faltantes (o incompletos) están presentes en casi todos los estudios que involucran recopilación y análisis de datos, la falta de respuesta constituye una gran limitación por la pérdida de validez y poder estadístico que conlleva, bien cuando se produce en forma de participación parcial (el sujeto deja alguna pregunta sin contestar) o como ausencia de participación (el individuo no contesta ninguna pregunta).

Algunos problemas que se pueden presentar debido a datos faltantes son (véase [10], [4]):

- Los datos faltantes reducen el poder estadístico, que se refiere a la probabilidad de que la prueba rechace la hipótesis nula cuando es falsa.

- Los datos faltantes pueden causar sesgos en la estimación de los parámetros.

- Los datos faltantes pueden reducir la representatividad de las muestras.

Para solucionar estos problemas se recurre a una técnica llamada *imputación*, la cual se clasifica en dos tipos: imputación simple y múltiple. En ambos casos se sugiere, al igual que con el descarte, usarlas sólo si se está seguro de que los mecanismos son datos faltantes completamente aleatorios, es decir, cuando los datos perdidos no se encuentren consecutivamente [11]. La figura muestra un ejemplo de como deben estar los datos para poder hacer uso de esto.

Group	FAMINCI2	CURWRKN	FDIVDYN	FM_EDU1	Sample size
1	O	O	O	O	31957
2	O	O	O	M	14
3	O	O	M	O	112
4	O	M	O	O	357
5	O	M	O	M	1
6	O	M	M	O	3
7	M	O	O	O	7215
8	M	O	O	M	74
9	M	O	M	O	829
10	M	O	M	M	81
11	M	M	O	O	186
12	M	M	O	M	6
13	M	M	M	O	33
14	M	M	M	M	7

Figura 1.1: El patrón faltante de cuatro variables un conjunto de datos.

La Figura 1.2 es un diagrama de calor que muestra la distribución de los datos faltantes o perdidos (en amarillo) de la base de datos D, con lo que se observa que las variables *last_review*, *review_per_month* y *license* son las que más valores perdidos presentan, específicamente, el 24.82 %, 24.82 % y 99.9 % de datos faltantes, respectivamente.

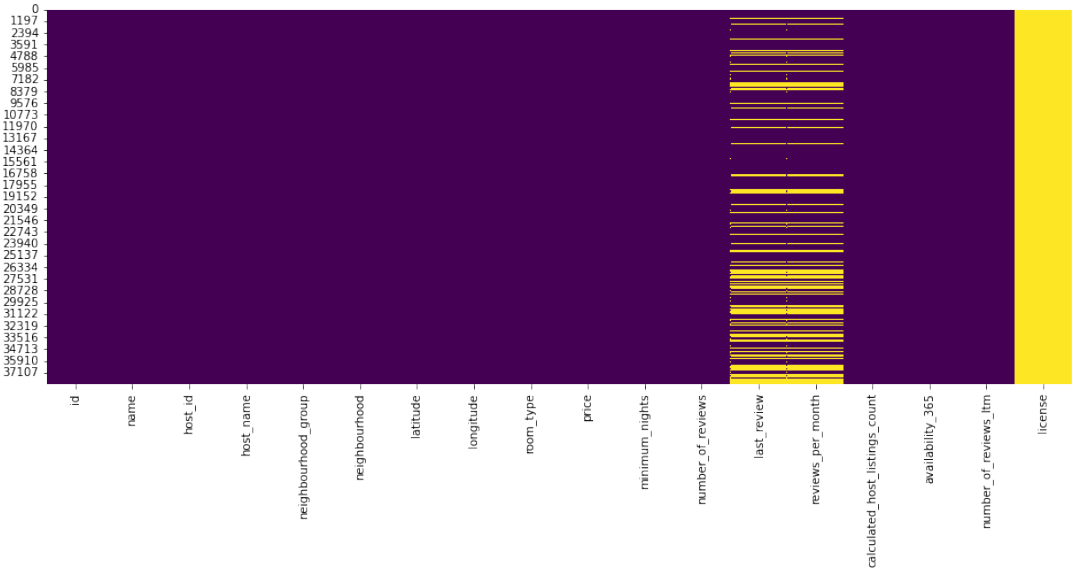

Figura 1.2: Diagrama de datos perdidos, en amarillo se muestra las celdas vacías.

En la imputación simple se usa un algoritmo para hacer una única estimación y el valor obtenido se usa para reemplazar el dato faltante correspondiente. Las técnicas más utilizadas son: la imputación por la media o la mediana y la imputación por regresión.

La imputación por la *media* o la *mediana* es la más sencilla de todas, tan solo se toman los valores conocidos en la variable donde se encuentran los datos faltantes, se calculan la media o la mediana y se reemplazan por los datos faltantes con cualquiera de estos dos valores. Aunque es muy fácil de implementar, este método tiene la desventaja de que al reemplazar muchos datos faltantes con un único valor estaremos cambiando la distribución de los datos.

Una alternativa es hacer la imputación por *regresión*. En este caso cada dato faltante es reemplazado con el valor predicho por un modelo de regresión. Para esto primero se combina la información de la columna con los datos faltantes con columnas en donde los datos están completos para así ajustar un modelo de regresión, luego se usa este modelo para predecir los datos faltantes. La desventaja es que para poder realizar cualquier tipo de regresión (regresión lineal, polinómica o regresión logística) debe haber algún tipo de correlación entre las variables que se utilizan para construir este modelo.

La imputación múltiple (como su nombre lo indica) hace múltiples estimaciones, que luego se combinan para producir un único valor, el cual será el utilizado para reemplazar el dato faltante correspondiente, con lo cual se puede disminuir el sesgo de la estimación. El método de imputación múltiple más usado es el algoritmo de Imputación Múltiple con Ecuaciones Encadenadas (o MICE por sus siglas en Inglés: Multiple Imputation by Chained Equations). En este algoritmo, se harán iteraciones de forma progresiva que cada vez se aproximarán mejor a los datos faltantes. Inicialmente la primera estimación no es muy buena, pues se hace con la imputación por la media que vimos anteriormente, pero en los pasos restantes se aplica una regresión lineal entre pares consecutivos de columnas, y el procedimiento se repite una y otra vez hasta completar un número pre-definido de iteraciones. La idea es que progresivamente las estimaciones sean cada vez más precisas y se acerquen más y más al valor real. Sin embargo, para el objetivo de esta tesis optaremos por eliminar la variable *license* debido a que una imputación no es posible por la cantidad de valores perdidos. Respecto a la variable *last_review* la imputaremos de forma simple con la fecha mínima de las que se conocen (Figura 1.3) y la característica *review_per_month* la imputamos de forma simple con

el valor cero (Figura 1.4), de esta forma el análisis se simplifica.

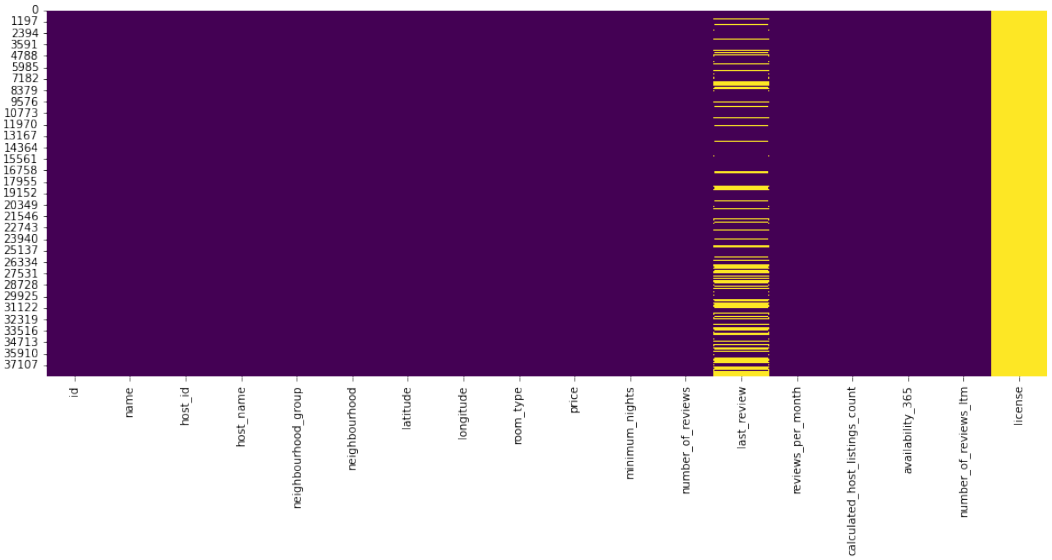

Figura 1.3: Diagrama de datos perdidos, en amarillo se muestra las celdas vacías.

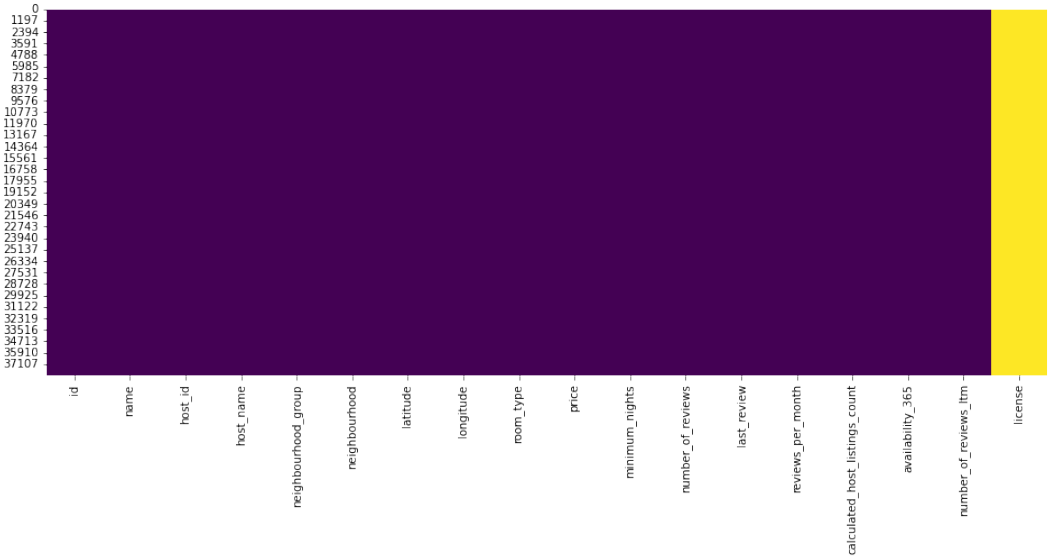

Figura 1.4: Diagrama de datos perdidos, en amarillo se muestra las celdas vacías.

El siguiente paso será eliminar las características que son consideradas no relevantes en el análisis, principalmente, debido a que son identificadores, estas variables son *host_name, name, host_id , id, license, number_of_reviews_ltm*. La

última variable es el número de veces que se revisó el alojamiento en los últimos 12 meses, debido a que las predicciones que se desea realizar son a corto plazo, esta variable se elimina junto con el resto. Con esto, se obtiene un nuevo valor $d_R = 12$, que es el número de características significativas. La variable *number_of_reviews* es el número de vistas que ha recibido en total cada alojamiento, sin embargo, existen 9,504 alojamientos sin ninguna vista. La variable *last_review* ahora es de forma numérica con valor 0 como la fecha mínima y el valor máximo es 4,002, siendo ahora una variable categórica y numérica a la vez.

1.3. Normalización de distribuciones sesgadas

Las variables que siguen distribuciones simétricas tienden a ser las ideales para entrenar los modelos de aprendizaje máquina, éstas muestran variaciones sustanciales, por lo que pueden ser utilizadas para discriminar características entre ellas. Pero las variables cuya distribución no es simétrica tienden a no ser efectivas a la hora de diferenciar sus características. En este sentido, una distribución sesgada hacia la derecha o hacia la izquierda, esto es, los valores se concentran a alguno de los lados, pueden centrarse y lograr que su distribución sea parecida a una normal, de ahí el verbo "normalizar".

Para lograr una normalización se necesita aplicar una transformación no lineal que mapee a todos los valores hacia un intervalo de longitud relativamente pequeña, por ejemplo, al $[0, 1]$. Un ejemplo de este tipo de funciones es la función logarítmica, la cual transforma la distribución en una campana simétrica. Tomando el logaritmo de una variable convierte su distribución en una distribución normal, con lo que al realizar las diferenciaciones entre los datos se logra con una mayor efectividad.

La variable *price*, que es la variable objetivo, tiene una distribución sesgada hacia la izquierda, como se observa en la Figura 1.5, sin embargo, si se aplica la transformación $f(x) = \log(1 + x)$, esta variable se normaliza. Además, existen algunos datos que se encuentran fuera de la campana, con el objetivo de que la variable tenga una distribución normal, se propone tomar sólo los datos dentro del rango $[3, 8]$, esto es, aquellos datos cuya normalización se encuentre en este intervalo.

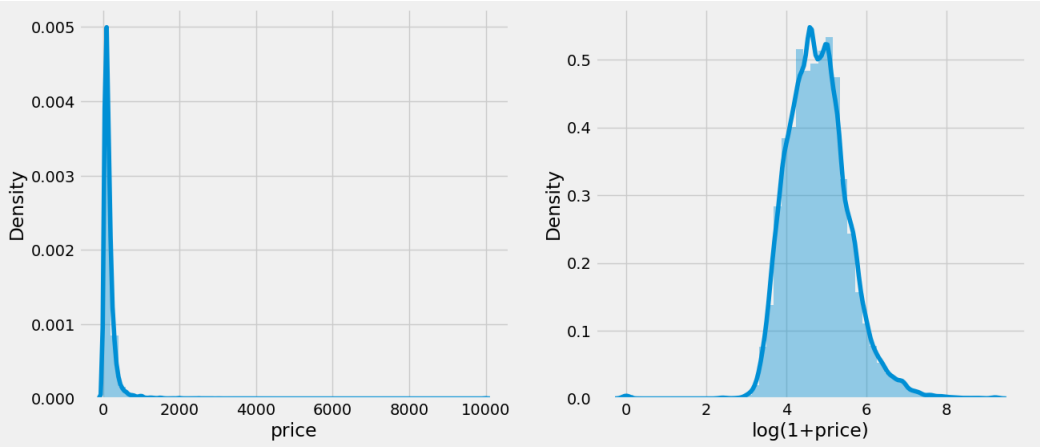

Figura 1.5: Normalización de la variable *price*.

En la Figura 1.5, que corresponde a la normalización de la variable *price*, se aprecia una mínima cantidad datos que quedan fuera de la gráfica, por lo que normalizar esta variable resulta favorable al análisis.

En el diagrama de normalización de la Figura 1.6, que corresponde a la variable *minimum_nights*, se contempla una cantidad de datos que quedan fuera de la gráfica, por lo que normalizar esta variable no resulta favorable al análisis.

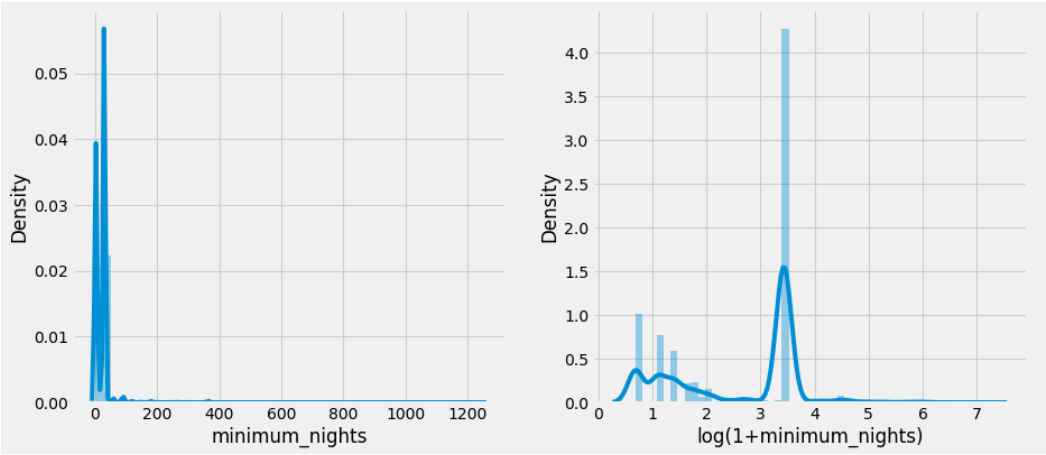

Figura 1.6: Normalización de la variable *minimum_nights*.

Mediante un diagrama de cajas y bigotes, se puede identificar si existen otras variables con distribuciones sesgadas, por ejemplo, las variables que ya se han normalizado muestran una tendencia de contar con los datos dentro de la ca-

ja (Figuras 1.7 y 1.8), a diferencia de las variables *number_of_reviews*, *calculated_host_listings_count* y *availability_365* (Figuras 1.9, 1.10 y 1.11). Ahora, si se normalizan éstas últimas variables, aún se preserva el sesgo en la distribución, por lo que no es conveniente realizarles este proceso (Figuras 1.12, 1.13 y 1.14).

Figura 1.7: Diagrama de caja y bigote de la variable *price*.

En el diagrama de caja y bigote de la Figura 1.7, que corresponde a la variable objetivo *price* previamente normalizada, se percibe una distribución simétrica con respecto a la media, esto resulta conveniente.

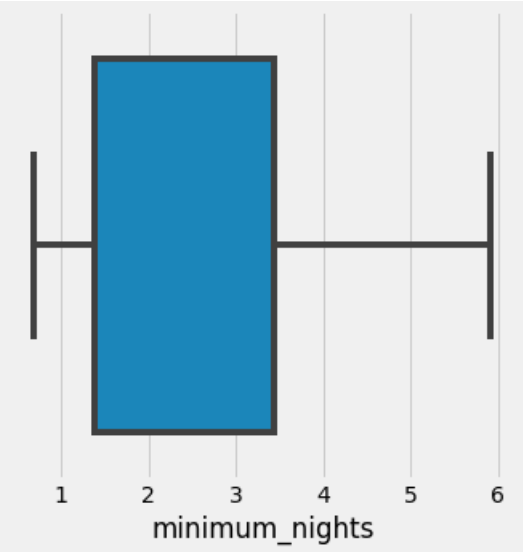

Figura 1.8: Diagrama de caja y bigote de la variable *minimun_nights*.

En el diagrama de caja y bigote de la Figura 1.8, que corresponde a la variable *minimun_nights*, se presenta un sesgo a la izquierda con respecto a la media, por lo cual no resulta favorable normalizarla, pues varios datos quedan fuera de la gráfica.

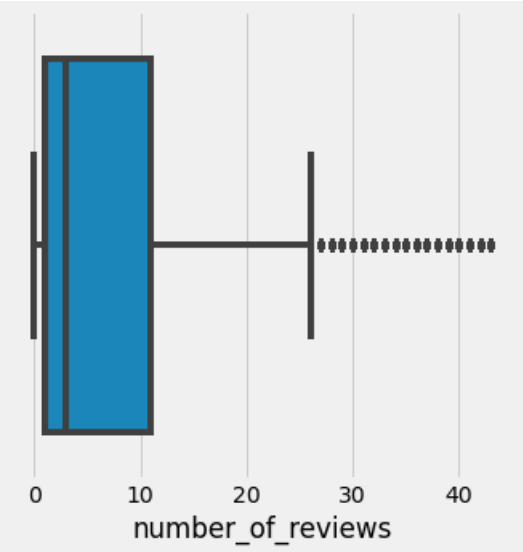

Figura 1.9: Diagrama de caja y bigote de la variable *number_of_reviews*.

En el diagrama de caja y bigote de la Figura 1.9, que corresponde a la variable

number_of_reviews, se expone un sesgo a la izquierda con respecto a la media, además se observa que muchos datos no entran en el diagrama, por lo cual no resulta beneficioso para el análisis.

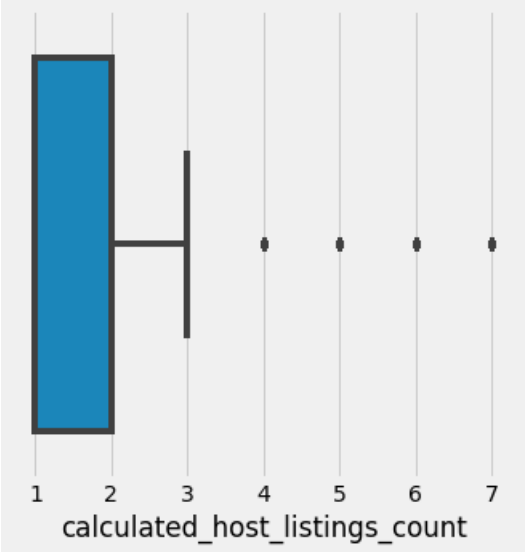

Figura 1.10: Diagrama de caja y bigote de la variable *calculated_host_listings_count*.

En el diagrama de caja y bigote de la Figura 1.10, que corresponde a la variable *calculated_host_listings_count*, se exhibe un sesgo a la izquierda con respecto a la media, además, varios datos quedan fuera del diagrama, por lo cual no resulta conveniente al análisis.

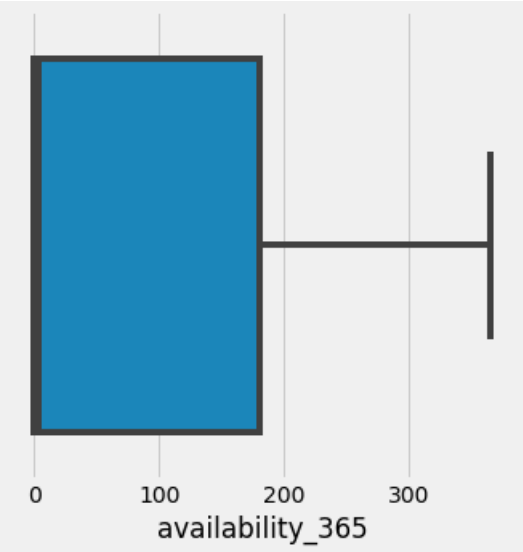

Figura 1.11: Diagrama de caja y bigote de la variable *availability_365*.

En el diagrama de caja y bigote de la Figura 1.11, que corresponde a la variable *availability_365*, se aprecia un sesgo a la izquierda con respecto a la media, por lo cual resulta desfavorable al análisis.

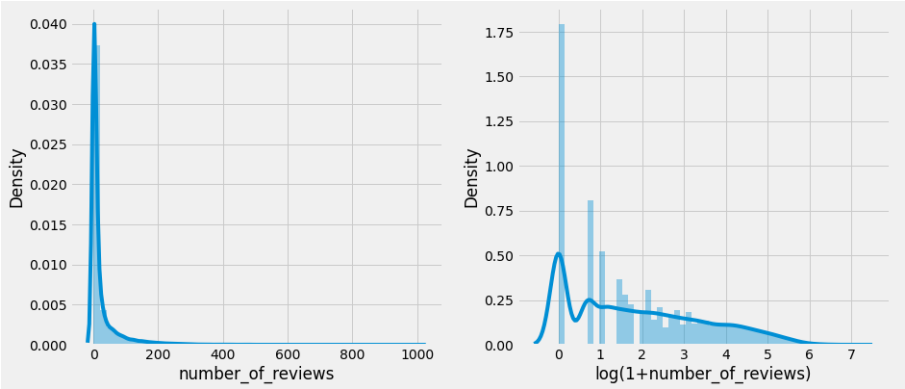

Figura 1.12: Normalización de la variable *number_of_reviews*.

En el diagrama de normalización de la Figura 1.12, que corresponde a la variable *number_of_reviews*, se muestran datos que quedan fuera de la gráfica, por lo que normalizar esta variable no resulta oportuno al análisis.

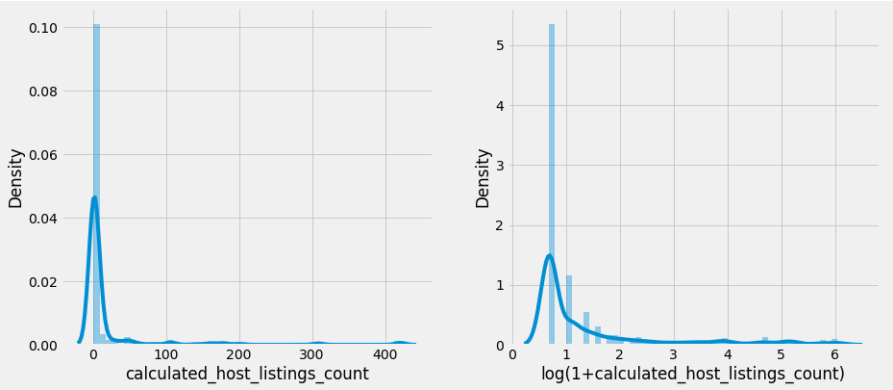

Figura 1.13: Normalización de la variable *calculated_host_listings_count*.

En el diagrama de normalización de la Figura 1.13, que corresponde a la variable *calculated_host_listings_count*, se muestran datos que quedan fuera de la gráfica, por lo que normalizar esta variable no resulta provechoso al análisis.

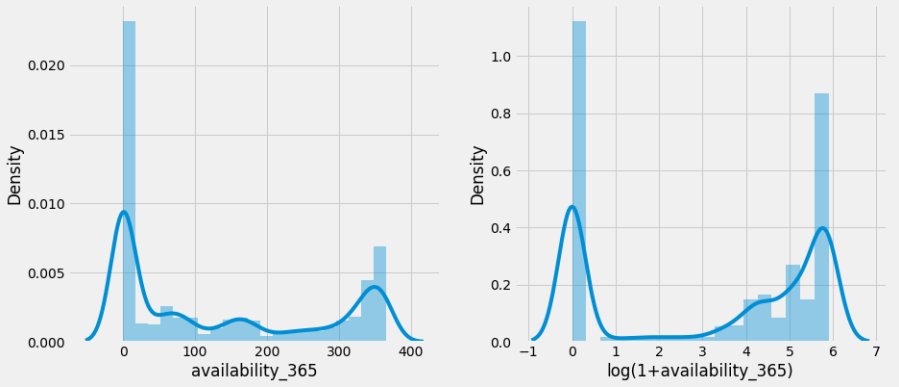

Figura 1.14: Normalización de la variable *availability_365*.

En el diagrama de normalización de la Figura 1.13, que corresponde a la variable *availability_365*, se evidencían datos que quedan fuera de la gráfica, por lo que normalizar esta variable no resulta propicio al análisis.

1.4. Visualización

Mediante representaciones gráficas de los datos se puede obtener una mejor interpretación de ellos. Un ejemplo es un gráfico de pastel, mediante el cual se puede visualizar que existen mayor número de alojamientos en los vecindarios de

Manhattan y Brooklyn, seguidos de Queens, Bronx y Staten Island (Figura 1.15). Ahora, si se usa un diagrama de barras, se logra visualizar que en el vecindario de Brooklyn las rentas de departamentos son más caras que cuartos privados y cuartos compartidos, además se puede observar que no hay hoteles. En el vecindario de Manhattan los hoteles son los más caros, seguidos de los departamentos. En Queens la renta de departamentos es más cara que la de los hoteles seguido de hoteles. En Bronx y Staten Island la renta de departamentos son más caras, además de que no hay hoteles (Figura 1.16).

La anterior información es lo que usualmente buscan las empresas o sectores para la toma de decisiones, por ejemplo, para el usuario, conocer que la renta más cara de hoteles se encuentra en Manhattan y que los más económicos están en Queens es muy valioso.

En la Figura 1.17 se muestra la distribución de los alojamientos por vecindario, obsérvese que en Staten Island los alojamientos se encuentras dispersos y en una cantidad menor que en el resto. También, se tiene una mayor concentración en Brooklyn; sin embargo, en la Figura 1.18 se observa la disponibilidad de éstos, donde se observa que en su mayoría están ocupados la mayor parte del tiempo, mostrando de color obscuro a aquellos que están disponibles la mayor parte del año.

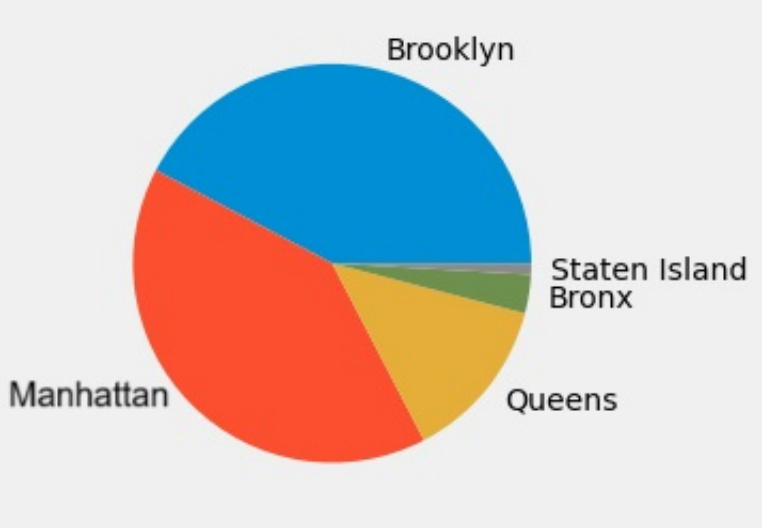

Figura 1.15: Gráfico de pastel de la variable *neighbourhood_group*.

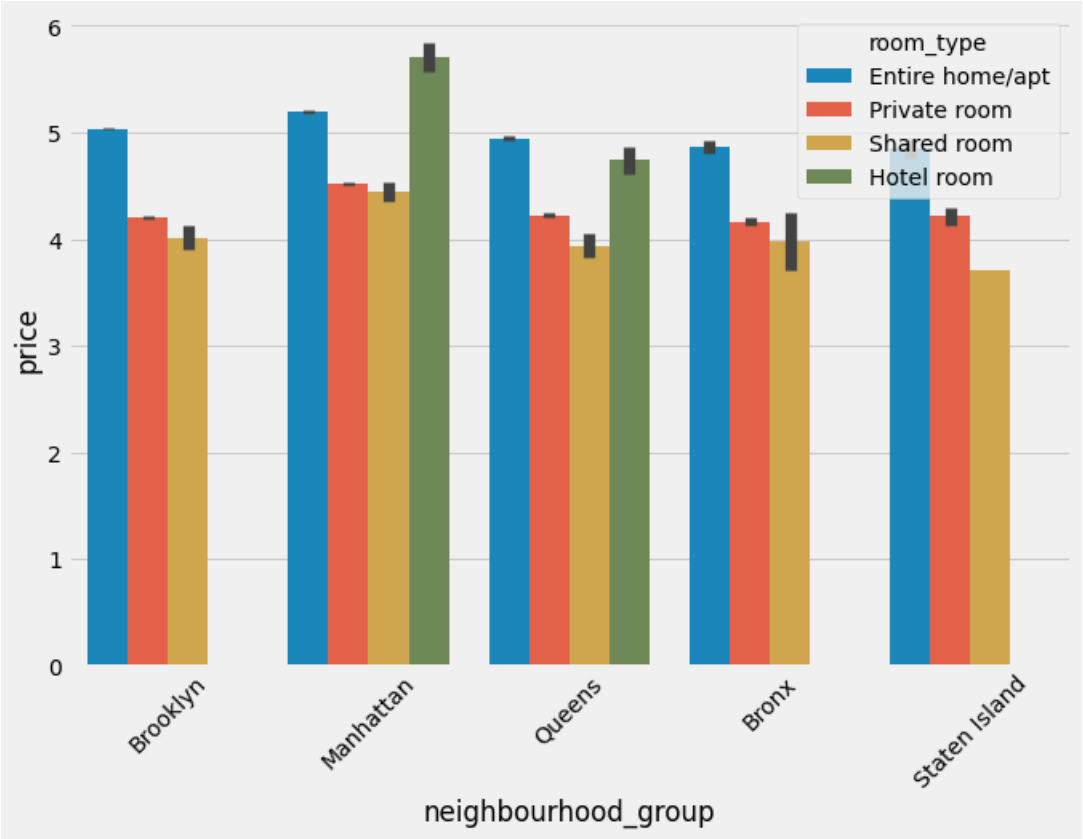

Figura 1.16: Gráfico de barras de la variable *neighbourhood_group*.

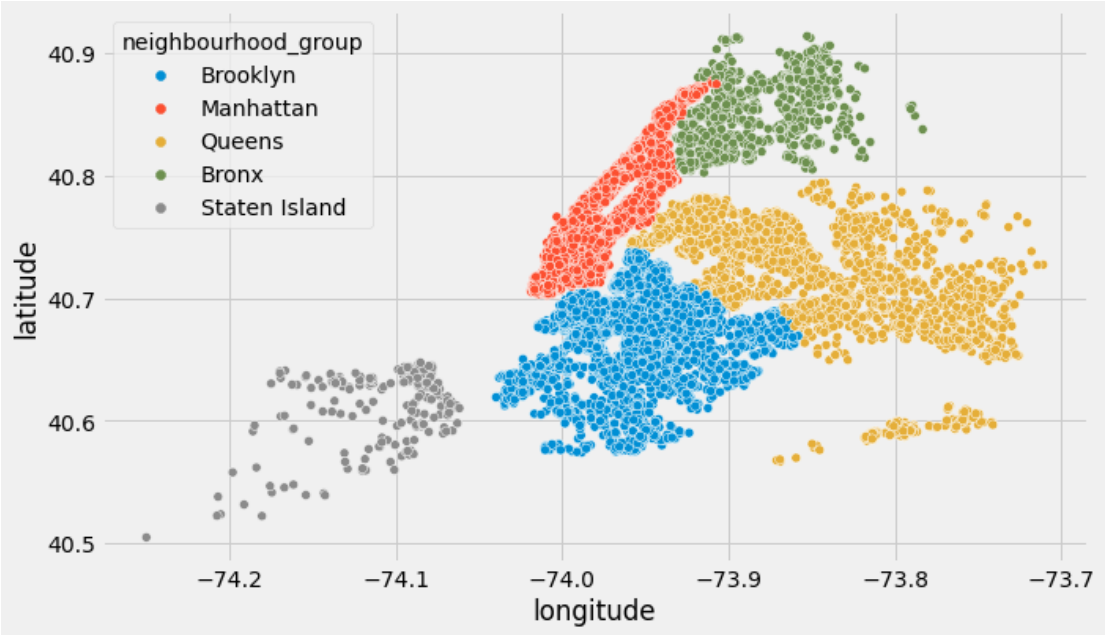

Figura 1.17: Mapa que muestra los alojamientos por vecindario.

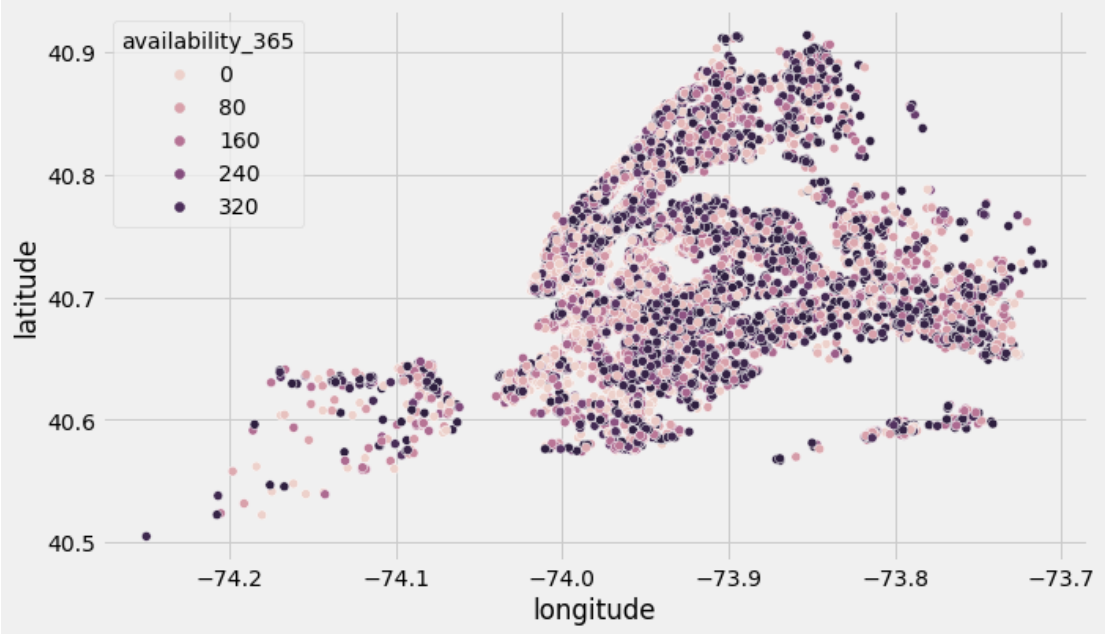

Figura 1.18: Mapa que muestra la disponibilidad de los alojamientos.

Figura 1.19: Ubicación geográfica.

En la Figura 1.19 se muestra la ubicación de los datos espaciales, debido a que parte importante en la ciencia de datos es saber utilizar este tipo de datos para extraer información. Por ejemplo, se puede ubicar exactamente a cada uno de los alojamientos en la base.

1.5. Distancias

Recordemos que la media de los datos espaciales latitud y longitud son 40.7292 y -73.9489, respectivamente. Este punto puede considerarse el centro de los alojamientos, una forma de manejar distancias es considerar a la base D como un conjunto de vectores d dimensionales dentro de un espacio métrico. De esta forma se puede trabajar con datos espaciales, para ello, se introduce una métrica de tipo Haversine [6] (o gran círculo) y es la distancia angular entre dos puntos en la superficie de una esfera. Se supone que la primera coordenada de cada punto es la latitud, la segunda es la longitud x expresadas en radianes y la dimensión de los datos debe ser dos. Esta métrica se define como

$$D(x,y) = 2\arcsin\left[\sqrt{\sin^2\left(\frac{x_1-y_1}{2}\right) + \cos(x_1)\cos(y_1)\sin^2\left(\frac{x_2-y_2}{2}\right)}\right],$$

con x_1, y_1, x_2 y y_2 en radianes.

Utilizando esta métrica es posible calcular la distancia entre cada alojamiento al centro, por ejemplo, la distancia del primer alojamiento al centro de ellos es de 3.75 kilómetros.

1.6. Correlaciones

El coeficiente de correlación $D(x, y)$ es un estadístico que mide el grado en que y es una función de x y viceversa, el valor del coeficiente de correlación varía de -1 a 1, donde 1 significa que están totalmente correlacionados y 0 implica ninguna relación o que son variables independientes. Las correlaciones negativas implican que las variables están anti-correlacionadas, lo que significa que cuando x sube, y baja. Las variables perfectamente anti-correlacionadas tienen una correlación de -1.

En esta investigación se propone usar el coeficiente de correlación de rangos de Spearman[10], para calcularlo, a cada valor de la variable x, se le asigna un valor, mediante la regla: al valor más pequeño se le asigna 1; 2 al siguiente; 3 al subsecuente y así sucesivamente, por lo que el rango del valor más pequeño es 1 y el valor más grande es n, este número se le conoce como rango y se denota como $rango(x_i)$. El coeficiente de Spearman es la suma de la diferencia de los rangos de todos los pares de puntos dividida entre un coeficiente dependiente del número de datos para lograr una normalización adecuada, específicamente, sea $rango(x_i)$ la posición de rango de x_i, entonces, el coeficiente se define como

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)},$$

donde $d_i = rango(x_i) - rango(y_i)$.

Los resultados del análisis de correlación entre las parejas de variables se muestra en la Figura 1.20. Se puede observar que ninguna variable está fuertemente relacionada con la variable *price*, sin embargo, sí están relacionadas las variables *reviews_per_month* y *last_review*, así como *reviews_per_month* con *number_of_reviews*, que son variables que muestran las revisiones de usuarios a cada alojamiento, estas correlaciones son las esperadas.

También se observa que no existen variables correlacionadas negativamente, salvo *last_review* con *minimum_nights* con una correlación de -0.44, pero con este valor no se pueden sacar conclusiones de ellas.

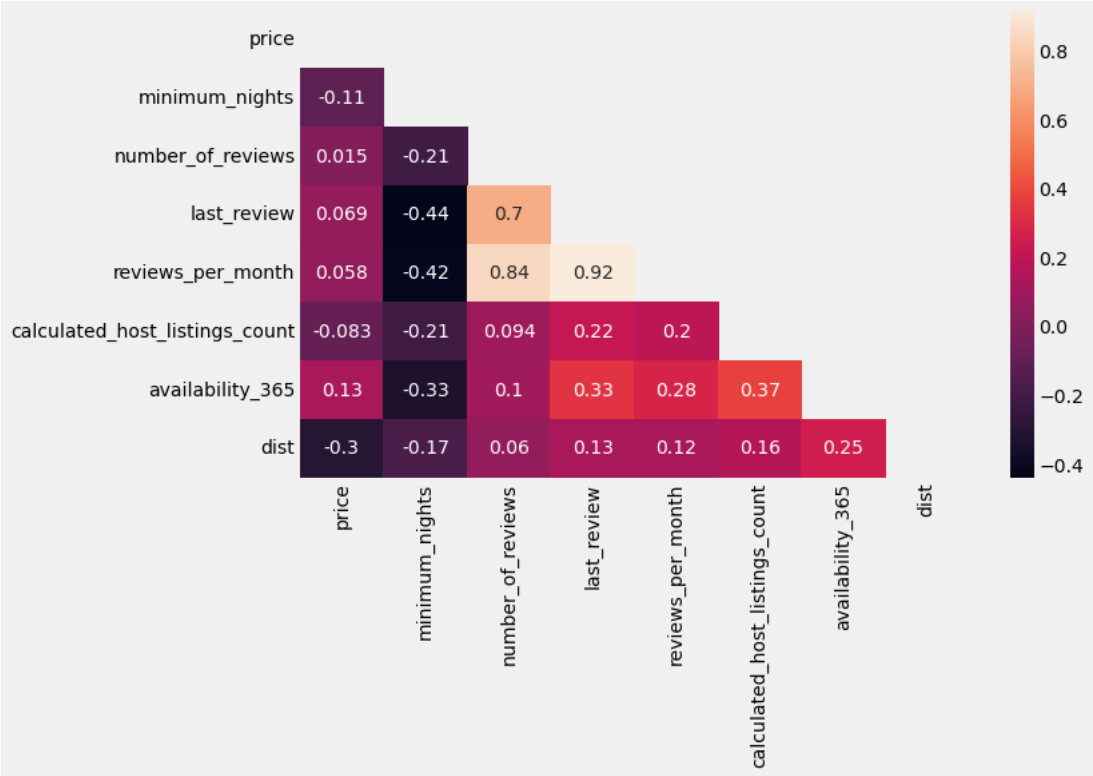

Figura 1.20: Matriz de correlaciones de las variables

Capítulo 2

Modelo matemático

2.1. Árboles de decisión

Los árboles de decisión son una metodología de clasificación, en la que el proceso de clasificación se modela con el uso de un conjunto de decisiones jerárquicas sobre las variables características, dispuestas en una estructura similar a un árbol. La decisión en un nodo particular del árbol, que se conoce como criterio de división, suele ser una condición para una o más variables de características en los datos de entrenamiento.

El criterio de división divide los datos de entrenamiento en dos o más partes. Por ejemplo, se considera el caso en el que Edad es un atributo y el criterio de división es Edad ≤ 30. El objetivo es identificar un criterio de división para que el nivel de "mezcla" de las variables de clase en cada rama del árbol se reduzca tanto como sea posible. Cada nodo en el árbol de decisión representa lógicamente un subconjunto del espacio de datos definido por la combinación de criterios de división en los nodos superiores.

El árbol de decisión generalmente se construye como una partición jerárquica de los ejemplos de entrenamiento, del mismo modo que un algoritmo de agrupamiento de arriba hacia abajo divide los datos jerárquicamente. Para ilustrar la idea básica de la construcción de un árbol de decisión, más adelante se muestra un ejemplo ilustrativo.

En la Tabla 2.1 se ilustra una tabla de un conjunto de datos hipotéticos sobre donaciones caritativas. Las dos variables de características representan los atributos de edad y salario. Ambos atributos están relacionados con la propensión a donar, que también es la etiqueta de clase. Específicamente, la probabilidad de

Name	Age	Salary	Donor?
Nancy	21	37000	N
Jim	27	41000	N
Allen	43	61000	Y
Jane	38	55000	N
Steve	44	30000	N
Peter	51	56000	Y
Sayani	53	70000	Y
Lata	56	74000	Y
Mary	59	25000	N
Victor	61	68000	Y
Dale	63	51000	Y

Tabla 2.1: Datos hipotéticos

que un individuo done se correlaciona positivamente con su edad y salario. Sin embargo, la mejor separación de clases sólo puede lograrse combinando los dos atributos. El objetivo en el proceso de construcción del árbol de decisión es realizar una secuencia de divisiones de arriba hacia abajo para crear nodos a nivel de hoja en los que los donantes y los no donantes estén bien separados. Una forma de lograr este objetivo se muestra en la Figura 2.1. La figura ilustra una disposición jerárquica de los ejemplos de entrenamiento en una estructura en forma de árbol. La división de primer nivel utiliza el atributo de edad, mientras que la división de segundo nivel para ambas ramas utiliza el atributo de salario. Se debe tener en cuenta que no es necesario que diferentes divisiones en el mismo nivel del árbol de decisión estén en el mismo atributo. Además, el árbol de decisión de la Figura 2.1 tiene dos ramas en cada nodo, pero no siempre tiene por qué ser así. En este caso, los ejemplos de entrenamiento en todos los nodos hoja pertenecen a la misma clase y, por lo tanto, no tiene sentido hacer crecer el árbol de decisión más allá de los nodos hoja. Las divisiones que se muestran en la Figura 2.1. se denominan divisiones univariadas porque utilizan un único atributo. Para clasificar una instancia de prueba, se recorre una única ruta relevante en el árbol de arriba hacia abajo utilizando los criterios de división para decidir qué rama seguir en cada nodo del árbol. La etiqueta de clase dominante en el nodo hoja se informa como la clase relevante. Por ejemplo, una instancia de prueba con una edad inferior a 50 años y un salario inferior a 60 000 recorrerá el camino más a la izquierda del árbol en

la Figura 2.1 debido a que el nodo hoja de esta ruta contiene solo ejemplos de entrenamiento que no son donantes, la instancia de prueba también se clasificará como no donante.

Las divisiones multivariadas utilizan más de un atributo en los criterios de división. Un ejemplo se ilustra en la Figura 2.2. En este caso particular, una única división conduce a la separación total de las clases. Esto sugiere que los criterios molturados son más poderosos porque conducen a árboles menos profundos, es decir, los criterios de partición que permiten construir árboles de decisión más compactos (es decir, menos profundos) son más eficaces o "poderosos.[en] el sentido de que pueden hacer divisiones más efectivas en los datos. Esto suele ser preferible en el diseño de árboles de decisión porque los árboles menos profundos son más manejables y menos propensos a sobreajustarse al ruido en los datos de entrenamiento..

Para el mismo nivel de separación de clases en los datos de entrenamiento, los árboles menos profundos generalmente son más deseables porque los nodos de hoja contienen más registros y, por lo tanto, es estadísticamente menos probable que sobreadapen el ruido en los datos de entrenamiento, esto es, dado el mismo nivel de separación de clases en los datos, es preferible construir árboles de decisión menos profundos. La razón es que, al ser menos profundos, los nodos de hoja contienen más registros. Esto ayuda a evitar el sobreajuste porque las decisiones se basan en más datos, lo cual hace que el modelo sea menos propenso a capturar el ruido específico del conjunto de entrenamiento y, por lo tanto, tiene una mejor capacidad de generalización a nuevos datos.

Un algoritmo de inducción de árbol de decisión tiene dos tipos de nodos, denominados nodos internos y nodos hoja. Cada nodo hoja está etiquetado con la clase dominante en ese nodo. Un nodo interno especial es el nodo raíz que corresponde a todo el espacio de características. El algoritmo de inducción del árbol de decisión genérico comienza con el conjunto de datos de entrenamiento completo en el nodo raíz y divide recursivamente los datos en nodos de nivel inferior según el criterio de división. Sólo es necesario dividir aún más los nodos que contienen una mezcla de diferentes clases. Finalmente, el algoritmo del árbol de decisión detiene el crecimiento del árbol según un criterio de parada. El criterio de parada más simple es aquel en el que todos los ejemplos de entrenamiento de la hoja pertenecen a la misma clase.

Un problema es que la construcción del árbol de decisión a este nivel puede

conducir a un sobreajuste, en el que el modelo se ajusta a los matices ruidosos de los datos de entrenamiento. Un árbol de este tipo no se generalizará muy bien a instancias de prueba invisibles. Para evitar la degradación de la precisión asociada con el sobreajuste, el clasificador utiliza un mecanismo de pospoda para eliminar los nodos sobreajustados. El algoritmo genérico de entrenamiento del árbol de decisión se ilustra en la Figura 2.2. Una vez construido un árbol de decisión, se utiliza para clasificar instancias de prueba invisibles mediante un recorrido de arriba hacia abajo desde la raíz hasta una hoja única. La condición de división en cada nodo interno se utiliza para seleccionar la rama correcta del árbol de decisión para su posterior recorrido. La etiqueta del nodo hoja alcanzado se informa para la instancia de prueba.

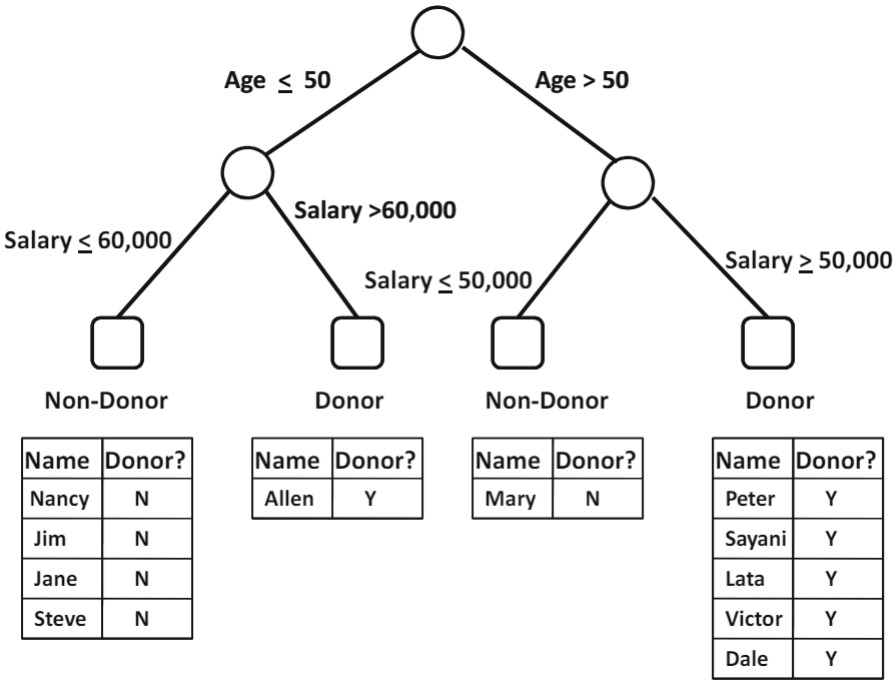

Figura 2.1: División univariada

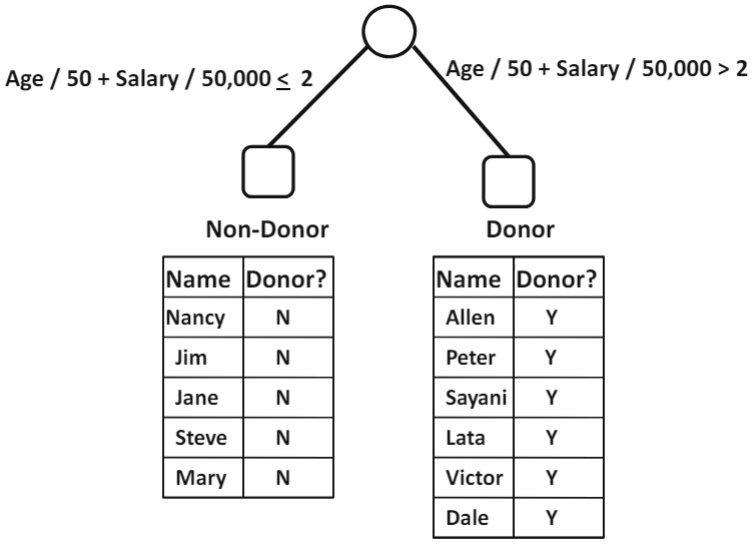

Figura 2.2: División multivariada

Criterios de división: El objetivo del criterio de división es maximizar la separación de las diferentes clases entre los nodos secundarios. A continuación, sólo se discutirán los criterios univariados. El diseño del criterio de división depende sobre la naturaleza del atributo subyacente:

1. *Atributo binario*: solo es posible un tipo de división y el árbol siempre es binario. Cada rama corresponde a uno de los valores binarios.
2. *Atributo categórico*: si un atributo categórico tiene r valores diferentes, hay múltiples formas de dividirlo. Una posibilidad es utilizar una división en forma de r, en la que cada rama de la separación corresponde a un valor de atributo particular. La otra posibilidad es utilizar una división binaria probando cada una de las $2^r - 1$ combinaciones (o agrupaciones) de categóricas atributos y seleccionando el mejor comparando las métricas de validación que se muestran adelante. Obviamente, esta no es una opción factible cuando el el valor de r es grande.
3. *Atributo numérico*: si el atributo numérico contiene un número pequeño r de valores ordenados es posible crear una división $r - recorrido$ para cada valor distinto. Sin embargo, para atributos numéricos continuos, la división normalmente se realiza mediante una condición binaria, como $x \leq a$, para el valor de atributo x y la constante a.

Algoritmo de árbol de decisión(conjunto de datos: D)

Empezar Cree un nodo raíz que contenga D.

repetir Seleccione un nodo elegible en el árbol;

Dividir el nodo seleccionado en dos o más nodos según un criterio de división predefinido;

hasta que no haya más nodos elegibles para dividir;

reducir los nodos sobre-ajustados del árbol;

Etiquete cada nodo hoja con su clase dominante;

fin

Las métricas para validar un árbol de decisión son las siguientes.

1. *Tasa de error*: Sea p la fracción de las instancias en un conjunto de puntos de datos S pertenecientes a la clase dominante. Entonces, la tasa de error es simplemente $1 - p$. Para una división de r vías del conjunto S en conjuntos $S_1, ..., S_r$, la tasa de error general de la división se puede cuantificar como el promedio ponderado de las tasas de error de los conjuntos individuales S_i, donde el peso de S_i es $|S_i|$. La división con la tasa de error más baja se selecciona de las alternativas.

2. *Índice de Gini*: el índice de Gini $G(S)$ para un conjunto S de puntos de datos se puede calcular de acuerdo con la distribución de clases $p_1, ..., p_k$ de los puntos de datos de entrenamiento en S. Se calcula mediante

$$G(S) = 1 - \sum_{j=1}^{k} p_j^2.$$

El índice de Gini general, para una división de r-vías del conjunto S en conjuntos $S_1, ..., S_r$, puede cuantificarse como el promedio ponderado de los valores del índice de Gini $G(S_i)$ de cada S_i, donde el peso de S_i es $|S_i|$:

$$\text{División de Gini}(S \Rightarrow S_1, ..., S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} G(S_i).$$

3. *Entropía*: la medida de entropía se utiliza en uno de los primeros algoritmos de clasificación, denominado $ID3$. La entropía $E(S)$ para un conjunto S se

puede calcular sobre la distribución de clases $p_1, ..., p_k$ de los puntos de datos de entrenamiento en el nodo de la siguiente forma:

$$E(S) = -\sum p_j \log_2(p_j).$$

Como en el caso del índice de Gini, la entropía general para una división del conjunto S en r subconjuntos $S_1...S_r$ puede calcularse como el promedio ponderado de los valores del índice de Gini $G(S_i)$ de cada S_i, donde el peso de S_i es $|S_i|$, de la siguiente forma:

$$\text{División de entropía}(S \Rightarrow S_1, ..., S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} E(S_i).$$

Valores más bajos de la entropía son los más deseables y la ganancia de información está estrechamente relacionada con la entropía y es igual a

$$E(S) - \text{División de entropía}(S \Rightarrow S_1, ..., S_r),$$

llamada reducción de entropía como resultado de la división. En este caso, también se desea obtener valores grandes de la reducción; a nivel conceptual, no hay diferencia entre usar cualquiera de los dos para una división, aunque es posible una normalización del grado de división en el caso de la ganancia de información. Se debe tener en cuenta que las medidas de entropía y ganancia de información deben usarse sólo para comparar dos divisiones del mismo grado, porque ambas medidas están sesgadas a favor de divisiones con mayor grado.

2.2. De árboles de decisión a árboles de regresión

Los árboles de regresión están diseñados para modelar relaciones no lineales entre las características y la variable de respuesta. Si el modelo de regresión se construye en cada nodo hoja en una partición jerárquica de los datos, se pueden obtener modelos de regresión lineal optimizados localmente dentro de cada partición. Incluso cuando la relación entre la variable de clase y las variable de característica no es lineal, una aproximación lineal local es bastante efectiva. Luego,

cada instancia de prueba se puede clasificar con su modelo de regresión lineal optimizado localmente determinando su partición apropiada. Esta partición jerárquica es esencialmente un árbol de decisión porque la partición asignada de una instancia de prueba está determinada por los criterios de división en los nodos internos. La estrategia general para construir un árbol de decisión sigue siendo la misma que en el caso de las variables de clase categóricas. De manera similar, las divisiones pueden utilizar divisiones univariadas (paralelas a los ejes) en las variables de características, como en un árbol de decisión tradicional. Sin embargo, es necesario realizar cambios en los criterios de división y poda debido a la variable de clase numérica:

Criterios de división: En el caso de clases categóricas, el criterio de división utiliza el índice de Gini o entropía de la variable de clase como medida cualitativa para decidir el atributo de división. Sin embargo, en el caso de clases numéricas, se utiliza una medida basada en errores. El enfoque de modelado de regresión de la sección anterior se aplica a cada hijo resultante de una posible división. Se calcula el error cuadrado agregado de predicción de todos los puntos de datos de entrenamiento en los diferentes nodos secundarios. La división con el error cuadrático agregado mínimo se selecciona entre todas las divisiones posibles en un nodo particular.

El principal problema computacional de este enfoque es que es necesario construir un modelo de regresión lineal para cada posible división. Una alternativa es no utilizar la regresión lineal en la fase de construcción del árbol. La varianza promedio de la variable de clase numérica en los nodos secundarios resultantes de una posible división se utiliza como criterio de calidad para la evaluación de la división. En otras palabras, el criterio de división del índice de Gini para la variable de clase categórica en la construcción tradicional de árboles de decisión se reemplaza por la varianza de la variable de clase numérica. Los modelos de regresión lineal se construyen en los nodos hojas para realizar predicciones solo después de que ya se haya construido todo el árbol. Si bien este enfoque dará como resultado árboles más grandes, es más práctico desde un punto de vista computacional.

Criterio de poda: El principal inconveniente de este enfoque es que el sobreajuste del modelo de regresión lineal es una posibilidad real cuando los nodos hoja no contienen suficientes datos. Por lo tanto, para empezar se

requiere una cantidad suficiente de datos de entrenamiento. En tales casos, los árboles de regresión pueden ser muy poderosos porque pueden modelar relaciones no lineales complejas.

Criterio de parada y poda: El criterio de parada para el crecimiento del árbol de decisión está íntimamente relacionado con la estrategia de poda subyacente. Cuando el árbol de decisión crece hasta el final hasta que cada nodo de hoja contiene sólo instancias que pertenecen a una clase particular, el árbol de decisión resultante muestra 100 % de precisión en instancias pertenecientes a los datos de entrenamiento. Sin embargo, a menudo se generaliza mal a instancias de prueba invisibles porque el árbol de decisión ahora se ha sobreajustado incluso a las instancias aleatorias características en las instancias de capacitación. La mayor parte de este ruido es aportado por el nivel inferior de los nodos, que contienen un número menor de puntos de datos. En general, los modelos más simples (árboles de decisión superficiales) son preferibles a modelos más complejos (árboles de decisión profundos) si producen el mismo error en los datos de entrenamiento.

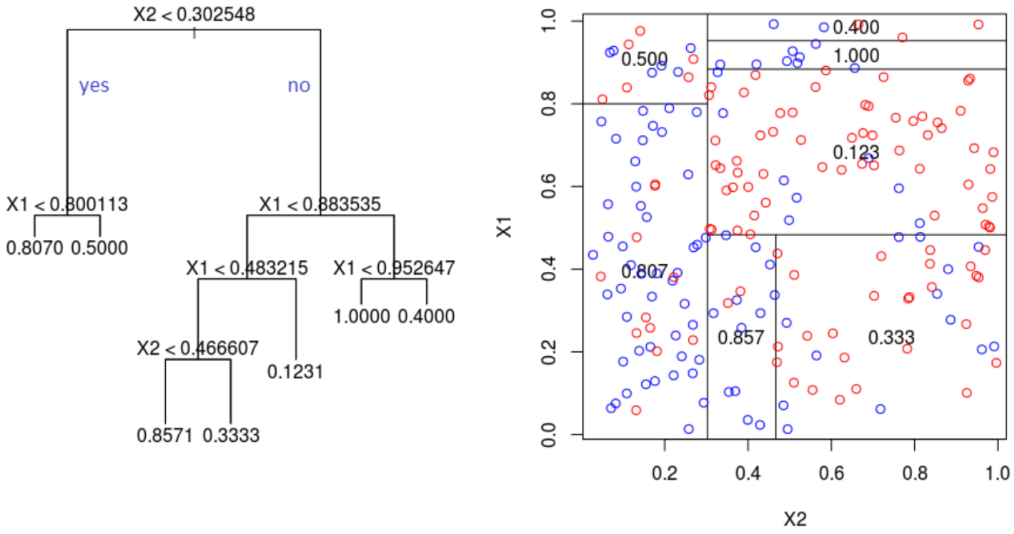

Figura 2.3: División multivariada

Para reducir el nivel de sobreajuste, una posibilidad es detener el crecimiento del árbol antes de tiempo. Lamentablemente, no hay forma de saber cuál es

el punto correcto en el que detener el crecimiento del árbol. Por lo tanto, una estrategia natural es podar las partes sobreajustadas del árbol de decisión y convertir los nodos internos en nodos hoja.

Existen muchos criterios diferentes disponibles para decidir si se debe podar un nodo. Una estrategia es penalizar explícitamente la complejidad del modelo con el uso del principio de longitud mínima de descripción (MDL). En este enfoque, el costo de un árbol se define por una suma ponderada de su error (datos de entrenamiento) y su complejidad (por ejemplo, el número de nodos). Los principios de la teoría de la información se utilizan para medir la complejidad de los árboles. Por lo tanto, el árbol se construye para optimizar el costo y no solo el error. El principal problema de este enfoque es que la función de costos es en sí misma una heurística que no funciona bien de manera consistente en diferentes conjuntos de datos. Una estrategia más simple e intuitiva es reservar una pequeña fracción (por ejemplo, el 20 %) de los datos de entrenamiento y construir el árbol de decisiones con los datos restantes.

El impacto de podar un nodo en la precisión de la clasificación se prueba en el conjunto de reservas. Si la poda mejora la precisión de la clasificación, entonces se poda el nodo. Los nodos de las hojas se podan de forma iterativa hasta que ya no es posible mejorar la precisión con la poda. Aunque este enfoque reduce la cantidad de datos de entrenamiento para construir el árbol, el impacto de la poda generalmente supera el impacto de la pérdida de datos de entrenamiento en la fase de construcción del árbol.

2.3. Gradient Boosting Decision Tree

El modelo utilizado para la realización de este análisis está basado en un modelo de tipo *árbol de decisión de gradiente creciente* (véase la sección siguiente) o también conocido por su nombre en inglés como *Gradient Boosting Desición Tree* (GBDT). Se inicia dividiendo los registros de la base D en dos subconjuntos: D_{train} y D_{test}, donde D_{train} será utilizada para ajustar el modelo, esto es, para entrenarlo. El subconjunto D_{test} será utilizado para la posterior validación. La base D_{train} será dos terceras partes de la base D y el resto se destinará para componer a la base D_{test}. Con ello, en esta sección se introducirá el modelo considerando sólo la base D_{train}.

Un registro x es un vector de d características, con una etiqueta destino y precio. Se denota el conjunto de características como

$$A = \{A_1, A_2, ..., A_n\}.$$

Se caracteriza a x como

$$X = \{x_1, x_2, ..., x_n\},$$

donde

$$x_1 \in A_1, x_2 \in A_2, ..., x_d \in A_d.$$

Sea

$$Y = \{y_1, y_2, ..., y_n\}^T,$$

el conjunto de la etiqueta de destino.

Se toma el promedio de la etiqueta de destino

$$\overline{y} = \sum \frac{y_i}{n}.$$

Se calcula una nueva característica A_r que serán los residuos obtenidos por restar el valor real de cada Y menos el vector cuyas entradas son el valor de $\overline{y}$, esto es

$$A_r = Y - \overline{y}.$$

Es decir

$$A_r = \{y_1 - \overline{y}, y_2 - \overline{y}, ..., y_n - \overline{y}\}^T,$$

luego se construye un árbol de decisión. Sea $\hat{A}_r$ el residuo resultante de la predicción que arroja el árbol de decisión y se introduce un hiper parámetro λ llamada tasa de aprendizaje,

Se plantea la cuestión de cómo se puede elegir la tasa de aprendizaje de λ. Un valor alto de λ dará como resultado tasas de aprendizaje rápidas, pero a veces puede dar como resultado soluciones subóptimas. Valores más pequeños de λ darán como resultado una convergencia hacia soluciones de mayor calidad, pero la convergencia será lenta. La tasa de aprendizaje reduce la contribución de cada árbol en función de la tasa de aprendizaje. Existe una compensación entre la tasa de aprendizaje y los estimadores n. Los valores deben estar en el rango [0.0, infinito]. para este caso, se tomo el valor predeterminado de 0.1

$$\hat{y} = \bar{y} + \lambda \hat{A}_r$$

donde $\hat{y}$ es el valor predicho. Se calculan los nuevos residuos:

$$\tilde{A}_r = y - \hat{y}.$$

Y se repite el proceso hasta que el número de iteraciones tenga como resultado que

$$\|\hat{y} - \bar{y}\| < \lambda \|\hat{A}_r\|.$$

Ya que está entrenado se introduce un nuevo registro $\bar{x}$ y se predice su etiqueta $\hat{y}$.

Los árboles de decisión son una metodología de clasificación, en la que se modela el proceso de clasificación con el uso de un conjunto de decisiones jerárquicas sobre las variables de características, estructura dispuesta en forma de árbol. La decisión en un nodo particular del árbol, que se conoce como *criterio de separación*, es típicamente una condición en una o más variables de características en los datos de entrenamiento.

El criterio de separación divide los datos de entrenamiento en dos o más partes. El objetivo es identificar un criterio de separación para que el nivel de "mezcla" de las variables de clase en cada rama del árbol sea reducido tanto como sea posible. Cada nodo en el árbol de decisión representa lógicamente un subconjunto del espacio de datos definido por la combinación de criterios de división en los nodos por encima de él. Una vez que se ha construido un árbol de decisión, se usa para la clasificación de bases de datos de prueba no vistas con el uso de un recorrido de arriba hacia abajo desde la raíz hasta una hoja única. La condición de división en cada nodo interno se utiliza para seleccionar la rama correcta del árbol de decisión para un recorrido posterior.

2.4. Algoritmo Gradient Boosting Decision Tree

Los pasos específicos para construir el árbol GBDT son los enumerados a continuación.

Paso 1 : **Calcular el promedio de la etiqueta objetivo**. Cuando se abordan problemas de regresión, se comienza con una hoja que es el valor promedio de la variable que se desea predecir. Esta hoja se utilizará como punto de partida para abordar la solución correcta en los pasos siguientes.

Paso 2 Calcular los residuos. Para cada muestra, se calcula el residual con la fórmula:

$$residual = valor\_real - valor\_previsto;$$

Luego se divide el nodo seleccionado en dos o más nodos según un criterio de división predefinido.

Paso 3 : **Construir un árbol de decisión**. Se construye un árbol con el objetivo de predecir los residuos. En otras palabras, cada hoja contendrá una predicción del valor del residual (no la etiqueta deseada).

En el caso de que haya más residuos que hojas, algunos residuos acabarán dentro de la misma hoja. Cuando esto sucede, se obtiene el promedio y se coloca dentro de la hoja.

Paso 4: Predecir la etiqueta objetivo utilizando todos los árboles dentro del conjunto. Cada muestra pasa a través de los nodos de decisión del árbol recién formado hasta llegar a un precio potencial determinado. El residual de dicha hoja se utiliza para predecir el parametro objetivo.

Se ha demostrado mediante experimentación [9] que al tomar pequeños pasos incrementales hacia la solución se logra un sesgo comparable con una varianza general más baja (una variación más baja conduce a una mejor precisión en muestras fuera de los datos de entrenamiento). Por tanto, para evitar el sobreajuste, se introduce un hiperparámetro llamado tasa de aprendizaje. Al realizar una predicción, cada residuo se multiplica por la tasa de aprendizaje. Esto obliga a utilizar más árboles de decisión, cada uno de los cuales da un pequeño paso hacia la solución final.

Paso 5 : **Calcular los nuevos residuos**. Se calcula un nuevo conjunto de residuos restando los precios reales de la vivienda de las predicciones realizadas en el paso anterior. Luego, los residuos se utilizarán para las hojas del siguiente árbol de decisión, como se describe en el paso 3.

Paso 6 : se repiten los pasos 3 a 5 hasta que el número de iteraciones coincida con el número especificado por el hiperparámetro (es decir, el número de estimadores).

Paso 7 : una vez entrenado, se utilizan todos los árboles del conjunto para hacer una predicción final sobre el valor de la variable objetivo.

La predicción final será igual a la media que se calculó en el primer paso, más todos los residuos predichos por los árboles que componen el bosque multiplicados por la tasa de aprendizaje.

2.5. Modelo LightGBM

En los algoritmos de aprendizaje automático que se utilizan para mejorar la precisión de los modelos de clasificación también conocidos como AdaBoost (Adaptive Boosting), el peso de la muestra sirve como un buen indicador de la importancia de las instancias de datos. Sin embargo, en GBDT no hay pesos de muestra nativos y, por lo tanto, los métodos de muestreo propuestos para AdaBoost no se pueden aplicar directamente. Afortunadamente, se observa que el gradiente para cada instancia de datos en GBDT nos proporciona información útil para el muestreo de datos. Es decir, si una instancia está asociada con un gradiente pequeño, el error de entrenamiento para esta instancia es pequeño y ya está bien entrenada. Una idea sencilla es descartar aquellas instancias de datos con gradientes pequeños. Sin embargo, la distribución de los datos se modificará al hacerlo, lo que dañará la precisión del modelo aprendido. Para evitar este problema, se propone un nuevo método llamado Muestreo unilateral basado en gradientes (GOSS).

GOSS conserva todas las instancias con gradientes grandes y realiza un muestreo aleatorio en las instancias con gradientes pequeños. Para compensar la influencia en la distribución de datos, al calcular la ganancia de información, GOSS introduce un multiplicador constante para las instancias de datos con gradientes pequeños. Específicamente, GOSS primero ordena las instancias de datos según el valor absoluto de sus gradientes y selecciona las instancias superiores a $\times$ 100 %. Luego, muestrea aleatoriamente las instancias b $\times$ 100 % del resto de los datos. Después de eso, GOSS amplifica los datos muestreados con gradientes pequeños mediante una constante $\frac{1-a}{b}$ al calcular la ganancia de información. Al hacerlo, se enfoca más en las instancias sub-entrenadas sin cambiar mucho la distribución de datos original.

El modelo LightGBM es un modelo altamente eficiente de tipo Gradient Boosting Decision Tree (GBDT), diseñado por Guolin Ke, Qi Meng, et al en [5], que utiliza un muestreo unilateral basado en gradiente (GOSS) y agrupación de ca-

racterísticas exclusivas (EFB). Con GOSS, se excluye una proporción significativa de instancias de datos con gradientes pequeños y sólo se usan el resto para estimar la ganancia de información). Con EFB, se agrupan funciones mutuamente excluyentes (es decir, rara vez toman valores distintos de cero simultáneamente), para reducir la cantidad de características. Algunos experimentos en múltiples conjuntos de datos públicos [8] muestran que LightGBM acelera el proceso de entrenamiento del GBDT convencional hasta más de 20 veces mientras logra casi la misma precisión.

Gradient Boosting Decision Tree (GBDT) usa árboles de decisión para entrenar una función del espacio de entrada X^d al espacio de gradiente G. Se supone que se tiene un conjunto de entrenamiento con n registros independientes idénticamente distribuidas $\{x_1, ..., x_n\}^T$ donde cada x_i es un vector con dimensión d en el espacio X^d. En cada iteración de aumento de gradiente, los gradientes negativos de la función de pérdida con respecto a la salida del modelo se denotan como $\{g_1, ..., g_n\}$. El modelo de árbol de decisión divide cada nodo en la característica más informativa (con mayor ganancia de información). Para GBDT, la ganancia de información generalmente se mide por la varianza después de la división, que se define a continuación.

Definición 2.1. *Sea O el conjunto de datos de entrenamiento en un nodo fijo del árbol de decisión. La ganancia de varianza de la característica de división j en el punto k para este nodo se define como*

$$V_{j|O}(k) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \in O : x_{ij} \leq k\}} g_i\right)^2}{n^j_{l|O}(k)} + \frac{\left(\sum_{\{x_i \in O : x_{ij} \geq k\}} g_i\right)^2}{n^j_{r|O}(k)}\right),$$

donde

$$n_O = \sum I[x_i \in O],$$

$$n^j_{l|O}(k) = \sum I[x_i \in O : x_{ij} \leq k],$$

y

$$n^j_{r|O}(k) = \sum I[x_i \in O : x_{ij} \geq k].$$

En lo anterior, I representa la función indicadora, la cual toma sólo los valores 0 y 1; 1 si el argumento es verdadero y 0 en caso contrario.

Para la característica j, el algoritmo del árbol de decisión selecciona

$$d_j^* = \arg\max_d V_j(d)$$

y calcula la mayor ganancia $V_j\left(d_j^*\right)$. Luego, los datos se dividen según la característica j^* en el punto d_{j*} en los nodos secundarios izquierdo y derecho.

En el método GOSS propuesto, primero, clasifica las instancias de entrenamiento según los valores absolutos de sus gradientes en orden descendente; segundo, se selecciona los a valores mas altos con los gradientes más grandes y una muestra de datos de tamaño b, se obtiene un subconjunto de instancias A; luego, para el conjunto restante A^c que consta de $(1-a)$ x 100 % de instancias con gradientes más pequeños, se realiza un muestreo aleatoriamente un subconjunto B con tamaño b x $|A^c|$ finalmente, se dividen las instancias de acuerdo con la ganancia de varianza estimada $\tilde{V}_j(d)$ sobre el subconjunto $A \cup B$, es decir,

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_l^j(d)}\right),$$

donde

$$\begin{aligned} A_l &= \{x_i \in A : x_{ij} \leq d\}, \\ A_r &= \{x_i \in A : x_{ij} > d\}, \\ B_l &= \{x_i \in B : x_{ij} \leq d\}, \\ B_r &= \{x_i \in B : x_{ij} > d\}, \end{aligned} \tag{2.1}$$

y el coeficiente $\frac{1-a}{b}$ se usa para normalizar la suma de los gradientes sobre B de nuevo al tamaño de A^c.

Por lo tanto, en GOSS, se usa el $\tilde{V}_j(d)$ estimado sobre un subconjunto de instancias más pequeño, en lugar del $V_j(d)$ preciso sobre todas las instancias para determinar el punto de división, y el costo de cálculo puede reducirse en gran medida. Más importante aún, en el artículo [5] indica que GOSS no perderá mucha precisión de entrenamiento y superará el muestreo aleatorio.

2.6. Entrenamiento

En esta fase, se aplican los métodos mencionados anteriormente. El modelo se construye a partir de la base D_{train} y será ajustado a través de estos registros, debido a que de entrada conoce los precios o la variable *price*. Este modelo es un clasificador que calcula el valor *price* de un nuevo registro ingresado sin esta característica. A través de las Figuras 2.4 y 2.5 se muestra el árbol de decisión resultado del entrenamiento, en el cual se presentan todas las variables y se clasifican dependiendo del error encontrado.

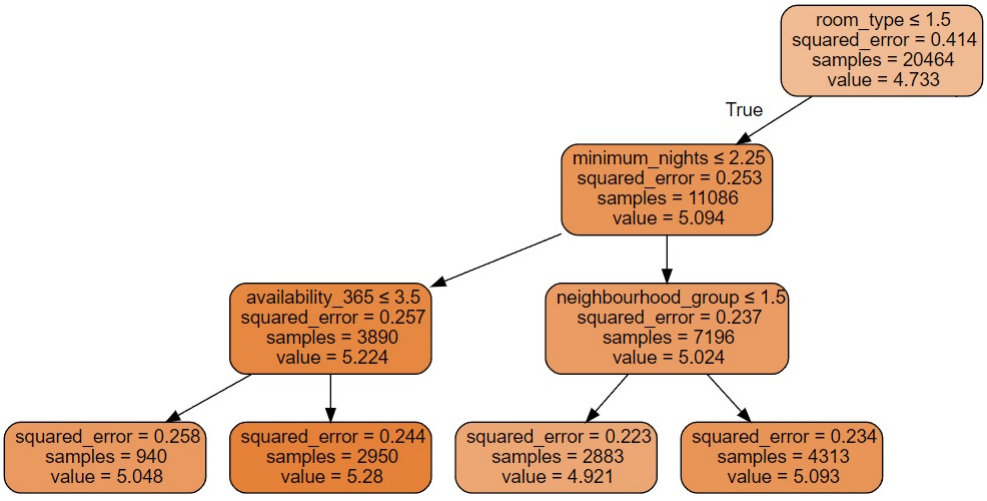

Figura 2.4: Árbol de decisión para *room_type*$\leq$ 1.5.

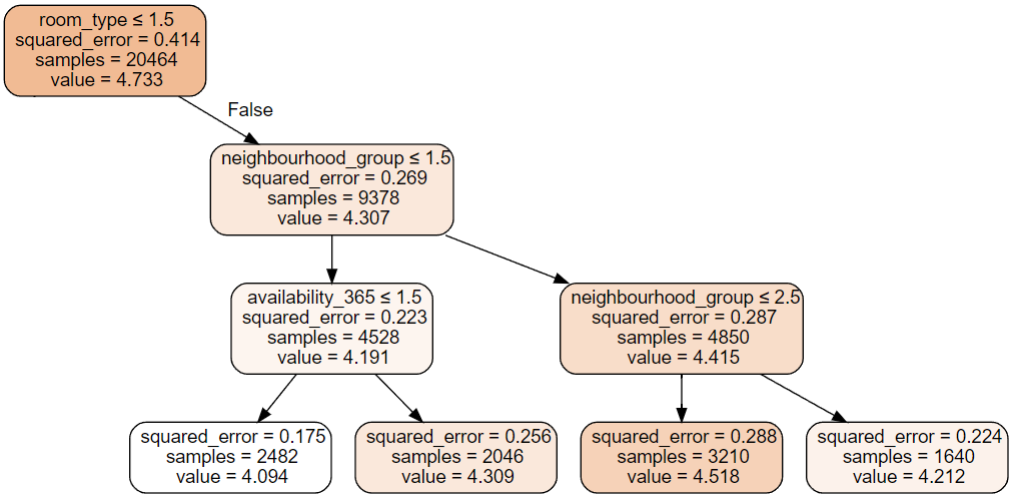

Figura 2.5: Árbol de decisión para *room_type* > 1.5.

2.7. Métricas de desempeño del modelo

Ahora, es conveniente medir el desempeño que presenta el modelo utilizado, para ello, se considera el conjunto de datos de prueba D_{test} y se le oculta la variable *price* al modelo, que se denota por y, luego se compara cuántos errores ha cometido en la predicción $\hat{y}$ mediante el cálculo de las siguientes métricas.

Error absoluto medio (MAE): Es el promedio de la diferencia absoluta entre el valor observado y los valores predichos. Esto es,

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|,$$

donde $y_j \in y$ son los elementos de la variable *price* del conjunto de datos de prueba D_{test} y $\hat{y}_i \in \hat{y}$ son las predicciones. El error absoluto medio o MAE es un puntaje lineal, lo que significa que todas las diferencias individuales se ponderan por igual en el promedio. Esta métrica varía de 0 en adelante, entre más cercano de cero, el modelo es más acertado. En este caso se tiene un MAE de 0.24, que se encuentra dentro de un rango aceptable.

R2: El coeficiente de determinación o R-cuadrado indica la bondad o la aptitud del modelo, a menudo se utiliza con fines descriptivos y muestra que también

las variables independientes seleccionadas explican la variabilidad en sus variables dependiente. $R2$ se define como:

$$R^2 = 1 - \frac{\sum(y_j - \hat{y_j})^2}{\sum(y_j - \bar{y})^2},$$

donde $y_j \in y$ son los elementos de la variable *price* del conjunto de datos de prueba D_{test} y $\hat{y}_i \in \hat{y}$ son las predicciones, y $\bar{y}$ es el promedio de todos los y_j. El rango de valores para $R2$ varía entre 0 y 1, donde entre más cercano de 1 el modelo es más acertado. Para el modelo propuesto, se alcanzó un valor $R2$ de 0.6.

2.8. Simulaciones

Recuerde que las variables significativas son *neighbourhood_group, neighbourhood, room_type, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count* y *availability_365*. Por lo que se introduce el nuevo registro, desconocido para el modelo entrenado y sin la etiqueta de clase, mostrado en la tabla 2.3.

neighbourhood_group	2
neighbourhood	20
latitude	40.86254
longitude	-73.89143
room_type	1
minimum_nights	5
number_of_reviews	15
last_review	2000
reviews_per_month	5
calculated_host_listings_count	3
availability_365	200

Tabla 2.2: Registro introducido al modelo.

Finalmente, el modelo nos arroja una predicción del precio del alojamiento que cuente con las características mostradas en la Tabla 2.3, en este caso, muestra que será de 99.21 dólares la noche para el mes próximo siguiente a la fecha de elaboración de la base, esto es, para el mes de enero del 2022.

Ahora bien, recordemos que se realizó una normalización con el logaritmo, por lo cual si el modelo arrojara un valor de 2 para la variable *price*, entonces al hacer la transformación con el antilogaritmo, se obtendría un valor de 100 dólares.

Valor real	Valor predicho
30	38.39375061
43	41.74676111
150	155.6501935
40	46.39221352
129	126.6507849
100	102.5222519
70	69.96624087
100	113.0368229
90	93.85609159
588	581.2319907
200	197.5773915
101	102.8624626

Tabla 2.3: Comparación de valores encontrados por el modelo

Conclusiones

En este trabajo, se realizó el análisis de la base de acceso libre New York Airbnb_4 dec 2021.csv, con el objetivo de diseñar un modelo matemático que realice predicciones de precios de alojamientos de Airbnb en NewYork. La dimensión de esta base es de $38,277 \times 18$.

En la Sección 1.1, se analizó el tipo de variables o columnas de la base y se extrajo información interesante a partir de las medidas de tendencia central.

En la Sección 1.2, se realizó un manejo de datos perdidos, se eligió eliminar una de las características (columna *license*) debido a su gran porcentaje de datos faltantes. Luego, se realizó la imputación de los registros que tuvieran datos perdidos en las columnas *reviews_per_month* y *last_review*, para esto se optó por llenar las columnas vacías con la fecha mínima debido a que el análisis de correlación no muestra una dependencia con el precio, por lo que una imputación que permitiera completar estos datos, no afecta en un incremento en la significancia con la variable *price*.

Además, se realizó una limpieza de la base eliminando las variables que no se consideraron relevantes, usualmente, existen métodos matemáticos para realizarlo, sin embargo, en su mayoría contenían valores constantes, y por experiencia, estas variables tienen un efecto mínimo en la variable objetivo que es despreciable.

En la Sección 1.3, se normalizó a la variable objetivo *price*, debido a que, en su forma original, provocó que el método tuviera un error relativamente grande. En este caso, se analizó la literatura sobre el tema y en [1] se muestra que este efecto se reduce si la variable tiene una distribución de frecuencia normal, con lo que se solucionó el problema y se disminuyó el error cometido por el modelo. Así mismo, se trató de normalizar algunas otras variables con distribución sesgada y se dedujo que no se reducía el sesgo por lo que no se lograba tener una distribución normal.

En la Sección 1.4, se realizaron algunas visualizaciones de la base, con lo que se obtuvieron algunas conclusiones interesantes:

- La mayoría de los alojamientos se encuentran en Manhattan y Brooklyn, seguido de Queens y Bronx y al final de Staten Island.
- En Manhattan se encuentran los hoteles con el mayor precio, seguidos de los de Queens. En Brooklyn, Bronx y Staten Island no existen hoteles en Airbnb.
- Los precios de los apartamentos o de las casas completas son similares en todos los barrios, sólo en Manhattan presentan un costo un poco más alto. Este mismo fenómeno se presenta con los cuartos privados y las habitaciones compartidas.
- El precio más bajo en todos los barrios se encuentran en los cuartos compartidos de Staten Island.
- En el barrio de Staten Island los alojamientos se encuentran conglomerados, a diferencia del resto.
- En su mayoría, los alojamientos se encuentran ocupados, aunque existe una poca cantidad con disponibilidad mayor a los 240 días al año.

En la Sección 1.5, se realizó el cálculo de la distancia entre cada uno de los alojamientos hacia el centro de ellos, que corresponde a 184-200 Diamond St, Brooklyn, NY 11222, EE. UU. con coordenadas de latitud y longitud 40.729206 y -73.948967, respectivamente. Estas distancias fueron calculadas utilizando la distancia Heversine, que es una distancia entre dos puntos de la esfera, con lo que se podría auxiliar al usuario a identificar distancias entre su alojamiento hacia algún otro punto de la ciudad.

En la Sección 1.6, se calcularon las correlaciones entre las variables con el objetivo de determinar si alguna de ellas es altamente dependiente de otra. En el Figura 1.20 se muestra que las variables *review_per_month* y *last_review* tienen una alta correlación positiva, por lo que tienen el mismo efecto en la variable *price*. Algunos autores, recomiendan, en este caso, eliminar alguna de ellas sin efectos adversos en el estudio o en el modelo matemático de clasificación, véase [1] y [10].

En la sección 2.1, se introdujo la teoría y metodología necesaria para comprender el modelo principal con el cual se trabajó, se introdujeron criterios de división y se ilustró un ejemplo de su funcionamiento.

En la sección 2.2, se plantea como mejorar un árbol de regresión para convertirlo en uno de decisión con el fin de adaptarlo a nuestras necesidades en la

aplicación del modelo matemático utilizado. Se muestran lo criterios de parada y de poda para evitar el sobreajuste en el modelo.

En la Sección 2.3, se describe el modelo llamado Gradient Boosting Decision Tree, el cual es la base principal del utilizado en esta investigación y en la Sección 2.4 se muestra el algoritmo. En estas secciones se muestra cómo se utiliza la función gradiente para la búsqueda óptima del criterio de división. Dando paso a la Sección 2.5, en la cual se planteó el modelo matemático del tipo de árbol de decisión de gradiente creciente, llamado LightGBM, que está basado en el muestreo unilateral de gradiente y en la agrupación de características exclusivas. Se realizó el entrenamiento y se calcularon las métricas de desempeño. Obteniendo un R2 de 0.6 por lo que el modelo se encuentra dentro de la categoría de un buen predictor.

En la Sección 2.6, se ajustó el modelo a través de los registros de la base de datos para que al final sea capaz de arrojar un valor predicho de los alojamientos.

En la Sección 2.7, se midió el desempeño que presenta el modelo, para esto se consideraron dos métricas, el Error Absoluto Medio, cuyos valores van desde cero en adelante, en donde más cerca del cero se encuentre el modelo es mejor. Para este modelo se obtuvo un Error Absoluto Medio de 0.24, el cual se encuentra dentro de un rango aceptable. Además, se midió con la métrica de coeficiente de determinación o R-Cuadrado(R2), el cual indica la bondad o la aptitud del modelo. El rango de valores para $R2$ varía entre 0 y 1, donde entre más cercano de 1 el modelo es más acertado. Para el modelo propuesto, se alcanzó un valor $R2$ de 0.6.

En la Sección 2.8, se introdujo un nuevo registro al modelo entrenado y éste arrojó el valor predictivo del precio en un mes siguiente.

Con todo lo anterior, se logra el objetivo de la investigación y el tesista logra adquirir los conocimientos sobre el proceso de ciencia de datos aplicada en la extracción de conocimiento y en la modelación matemática aplicados a problemas económicos.

Bibliografía

[1] Aggarwal C. C. *Data mining the textbook*. Springer, 1 edition.

[2] Dataquest. Python machine learning tutorial: Predicting airbnb prices, 2019. 23-01-2024.

[3] Javier Díaz. La historia de airbnb, la compañía que revolucionó la industria del turismo a nivel global, 2020. 19-01-2024.

[4] Yulei He, Guangyu Zhang, and Chiu-Hsieh Hsu. *Multiple Imputation of Missing Data in Practice: Basic Theory and Analysis Strategies*. Chapman and Hall/CRC, 1st edition, 2022.

[5] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[6] Scikit learn. Distancias haversine, scikit-learn machine learning in python, 2018. 17-01-2024.

[7] Redacción Smart Travel News. Todo sobre airbnb: historia, modelo de negocio y futuro, 2018. 19-01-2024.

[8] PriceLabs. Predictive price modeling for airbnb listings, 2015. 2-01-2024.

[9] Ruchi. Gradient boosted decision tree, 2023. 21-12-2023.

[10] S. S. Skiena. *Data Science Design manual*. Springer, 1 edition.

[11] Guangyu Zhang Yulei He. *Multiple Imputation of Missing Data in Practice Basic Theory and Analysis Strategies*. Taylor Francis Group, 1 edition.

Anexo 1: Validación

1. El proceso de exploración comienza cuando se cargan algunas de las librerías necesarias:

```
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt # plotting
import plotly
import seaborn as sns
```

2. Posteriormente se carga la base datos a analizar.

```
from google.colab import files
uploads = files.upload()
```

3. Se lee o se carga la base de datos en una variable cuando esta cargada en la memoria.

```
datos = pd.read_csv('New York Airbnb_4 dec 2021.csv')
```

4. Se transforma la variable *last_review* que es una fecha a una estampa de tiempo ya que se encuentra en tipo texto y no es posible manejarla.

```
datos['last_review'] =
pd.to_datetime(datos['last_review'],
infer_datetime_format=True)
```

5. Elimina las características que no aportan información

```
datos.drop(['host_name','name','host_id','id',
'license','number_of_reviews_ltm'], axis=1,
inplace=True)
```

6. Imputar la variable *reviews_per_month*

```
datos = pd.read_csv('New York Airbnb_4 dec 2021.csv')
```

7. Imputar la variable *last_review*

```
earliest = min(datos['last_review'])

datos['last_review'] =
datos['last_review'].fillna(earliest)

#normaliza las fechas, es decir, restamos la fecha
#menos la fecha mas temprana y eso nos da una variable
#numerica.

datos['last_review'] =
datos['last_review'].apply(lambda x: x.toordinal() -
earliest.toordinal())
```

8. Se cargan algunas librerías extras que nos servirán para graficar algunos datos.

```
import seaborn as sns
plt.style.use('fivethirtyeight')
plt.rcParams['font.family'] = "Arial"
import statsmodels
import statsmodels.api as sm
import scipy.stats as stats
from scipy.stats import norm
from scipy.special import boxcox1p
```

9. El siguiente código muestra la gráfica de la distribución para la variable *price*

```
fig, axes = plt.subplots(1,3, figsize=(21,6))
sns.distplot(datos['price'], ax=axes[0])
sns.distplot(np.log1p(datos['price']), ax=axes[1])
axes[1].set_xlabel('log(1+price)')
```

```
sm.qqplot(np.log1p(datos['price']), stats.norm,
fit=True, line='45', ax=axes[2]);
```

10. Eliminar los datos que se encuentran fuera de la campana.

```
datos = datos[np.log1p(datos['price']) < 8]
datos = datos[np.log1p(datos['price']) > 3]
```

11. Transformamos logaritmicamente la variable price

```
datos['price'] = np.log1p(datos['price'])
```

12. Normalizamos la variable *minimum_nights*

```
fig, axes = plt.subplots(1,3, figsize=(21,6))

sns.distplot(datos['minimum_nights'], ax=axes[0])

sns.distplot(np.log1p(datos['minimum_nights']),
ax=axes[1])

axes[1].set_xlabel('log(1+minimum_nights)')

sm.qqplot(np.log1p(datos['minimum_nights']),
stats.norm, fit=True, line='45', ax=axes[2]);
\begin{lstlisting}[frame=single]
datos['price'] = np.log1p(datos['price'])
```

13. Transformamos logaritmicamente la variable *minimum_nights*.

```
datos['minimum_nights'] =
np.log1p(datos['minimum_nights'])
```

14. Extracción de características.

```
feature_list = ['price', 'minimum_nights',
'number_of_reviews', 'calculated_host_listings_count',
'availability_365']
```

15.

```
for i in feature_list:
```

```
Q1 = np.percentile(datos[i], 25)
Q3 = np.percentile(datos[i], 75)
IQR = Q3 - Q1
upper = Q3 + 1.25 * IQR
lower = Q1 - 1.25 * IQR
anomaly = (datos[i] >= upper) | (datos[i] <= lower)
datos.drop(datos.index[anomaly],
axis = 0, inplace = True)
```

16. Calculamos los diagramas de caja y bigote

```
for i in range(0, 5):
plt.figure(figsize = (5, 5))
sns.boxplot(x=datos[feature_list[i]])
```

17. Gráfico de pastel de la variable neighbourhood group.

```
datos["neighbourhood_group"].value_counts().plot.pie()
```

18. Gráfico de barras de la variable neighbourhood group.

```
plt.figure(figsize=(10,7))
sns.barplot(x = "neighbourhood_group",
y = "price", hue = "room_type", data = datos)
plt.xticks(rotation=45)
plt.show()
```

19. Gráfico de dispersión que muestra los alojamientos por vecindario.

```
plt.figure(figsize=(10,6))
sns.scatterplot(x=datos.longitude, y=datos.latitude,
hue = datos.neighbourhood_group)
# Pasar x e y como argumentos de palabras clave
plt.ioff()
```

20.

```
plt.figure(figsize=(10,6))
sns.scatterplot(datos.longitude,
datos.latitude, hue = datos.availability_365)
plt.ioff()
```

21.

```
datos1 = datos[datos["neighbourhood_group"] == "Bronx"]
bronx_locations = datos1.drop(columns = ["neighbourhood",
"room_type", "minimum_nights", "number_of_reviews",
"calculated_host_listings_count", "availability_365"])
```

22. importamos una biblioteca permite la concepción de mapas interactivos

```
import folium
```

23.

```
map = folium.Map(location =
[bronx_locations["latitude"].mean(),
bronx_locations["longitude"].mean()],
zoom_start=12, control_scale=True)
for index, location_info in bronx_locations.iterrows():
folium.Marker([location_info["latitude"],
location_info["longitude"]]).add_to(map)
```

24.

```
list1 = ['price', 'minimum_nights',
'number_of_reviews',
'calculated_host_listings_count',
'availability_365']
for i in range(5):
print(list1[i])
print("Asimetria:", datos[list1[i]].skew())
print("Curtosis:", datos[list1[i]].kurtosis())
```

25. cálculo de distancia se ve a la base de datos como vectores

```
# definimos el centro
latitude=40.77213
longitude= -73.98665
```

26.

```
datos=datos.reset_index(drop= True)
```

27. importamos la métrica haversine

```
from math import radians
from sklearn.metrics.pairwise import haversine_distances
```

28. calculamos la distancia senoidal

```
def func(latitude, longitude, latitude1, longitude1):
cor_list = [latitude, longitude]
cor_list2 = [latitude1, longitude1]
radian1 = [radians(_) for _ in cor_list]
radian2 = [radians(_) for _ in cor_list2]
result = haversine_distances([radian1, radian2])
result = result * 6371000/1000
return result[0][1]
```

29. calcula la distancia del centro elegido hacia todos los puntos de la base

```
distance_list = [(func(datos["latitude"][i],
datos["longitude"][i], 40.71980, -73.98566))
for i in range(len(datos))]
```

30. Agregamos la distancia obtenida a nuestra base en una nueva variable llamada "dist"

```
datos["dist"] = distance_list
```

31. Matriz de correlaciones

```
datos1 = datos.drop(columns =
["latitude", "longitude"])
correlation_matrix =
datos1.corr(method = "spearman")
mask =
np.triu(np.ones_like(correlation_matrix, dtype=bool))
plt.figure(figsize = (10, 6))
sns.heatmap(correlation_matrix, annot =
True, mask = mask)
```

32. Modelo de aprendizaje maquina predictivo

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.linear_model import LinearRegression,
SGDRegressor, LogisticRegression
from sklearn.preprocessing import StandardScaler,
LabelEncoder, OneHotEncoder
from sklearn.ensemble import GradientBoostingRegressor,
RandomForestRegressor
from sklearn.svm import SVC
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score, r2_score,
mean_absolute_error, median_absolute_error
from scipy import stats
from collections import Counter
from nltk.corpus import stopwords
import re
from sklearn.metrics.pairwise import
haversine_distances
from math import radians
from statistics import median from lightgbm import
LGBMRegressor
```

33.

```
categorical_list =
["neighbourhood_group", "neighbourhood", "room_type"]
```

34. Conviértete datos categóricos en numéricos, codificador de etiquetas

```
label_encoder = LabelEncoder()
```

35. Codificamos etiquetas

```
for i in range(len(categorical_list)):
datos[categorical_list[i]] =
```

```
label_encoder.fit_transform(datos[categorical_list[i]])
```

36. Elegir la variable objetivo z"(precio) y las variables que están relacionadas con ella "x".

```
x = datos.drop(columns =
["price", "latitude", "longitude"])
y = datos.price
```

37. Separar la matriz "x.en dos conjuntos: entrenamiento del modelo y el de prueba.

```
X_train, X_test, y_train, y_test =
train_test_split(x, y, test_size=0.2,
random_state=105, shuffle=1)
```

38. Elegimos los modelos a utilizar

```
models = {
'Linear Regression' : LinearRegression(),
'Random Forest Regressor':
RandomForestRegressor(),
'Gradient Boosting Regressor' :
GradientBoostingRegressor(),
'LGBM Regressor' : LGBMRegressor()
}
```

39. Hacemos los entrenamientos y los validamos

```
for i in models:
model = models[i]
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print("Model Name", model)
print("")
print("R2 score of model is:",
r2_score(y_test, predictions))
print("Median of test input:", median(y_test))
print("Median absolute error of model is:",
median_absolute_error(y_test, predictions))
```

```
print("")
print("****************")
print("")
```

40. Pedimos las primeras filas del modelo entrenado

```
X_test.head()
```

41. importamos la librería Pipeline

```
from sklearn.pipeline import Pipeline
```

42. modelo Pipeline

```
modelo = Pipeline([
('LGBM Regressor', LGBMRegressor())
])
```

43. Entrenamos el modelo

```
modelo.fit(X_train, y_train)
```

44. Calculamos las metricas de error, error medio cuadrado, error absoluto medio

```
from sklearn.metrics import mean_squared_error,
mean_absolute_error
```

45. Ahora realiza predicciones con la base de prueba, para medir la eficiencia del modelo.

```
predictions = modelo.predict(X_test)
print(f"Error de media cuadratica:
{mean_squared_error(y_test, predictions)}")
print(f"Error medio absoluto:
{mean_absolute_error(y_test, predictions)}")
```

46.

```
output = pd.DataFrame(X_test)
output.to_csv('X_test.csv', index=False)
```

47.

```
predice=pd.read_csv=('datos_para_predicciones.csv')
```

48.

```
predice=pd.read_csv=('datos_para_predicciones.csv')
```