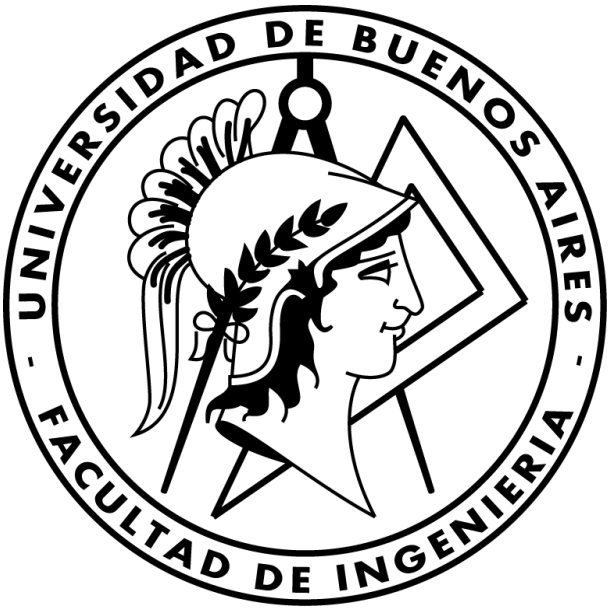
Universidad de Buenos Aires
Facultad de Ingeniería



Ciencia de Datos - TA047
Trabajo Práctico 1 - Cátedra Rodriguez
2do Cuatrimestre 2025

Grupo 07

Integrantes:

- Leandro Elias Brizuela
  - **Padrón:** 109842
  - **Mail:** lebrizuela@fi.uba.ar
- José Rafael Patty Morales
  - **Padrón:** 109843
  - **Mail:** jpatty@fi.uba.ar
- Jesabel Pugliese
  - **Padrón:** 110860
  - Mail: jpugliese@fi.uba.ar
- Candela Matelica
  - **Padrón:** 110641
  - Mail: cmatelica@fi.uba.ar

Ejercicio 1 - Análisis Exploratorio de Datos:

Descripción del Dataset:

Para este ejercicio se trabajó sobre subconjuntos de datos de viajes en taxi amarillos y limusinas de la ciudad de Nueva York, Estados Unidos. Los subconjuntos corresponden a los meses de enero, febrero y marzo del año 2024. Estos venían en archivos separados de tipo *.parquet* por lo que se optó por unificarlos en un mismo *Dataframe* concatenando las filas. El conjunto final resultante contiene 9554778 filas y 19 columnas, sobre el cuál se llevará a cabo el análisis.
Además, se usó el dataset *taxi_zone_lookup* el cuál proporciona la zona y la ciudad y un ID correspondiente. Esto nos será útil para poder identificar las zonas en las que empiezan y terminan los viajes.

Lo siguiente fue definir el tipo de variable que representa cada columna del dataset. Estas fueron divididas en **cuantitativas** y **categorías** que a su vez son subdivididas en *discretas* o *continuas* (en el caso de **variables cuantitativas**) y en *ordinales* o *nominales* (en el caso de **variables categóricas**). La forma de definir el tipo de variable fue basado en los apuntes brindados por la cátedra.

A continuación, se definirá el tipo de variable para cada columna:

Primero se definirán las categóricas:

- `VendorID`: Es un código que indica el proveedor de **TPEP** que suministró el registro. Esta variable puede tomar los valores:
  - $1 = Creative\ Mobile\ Technologies,\ LLC$
  - $2 = VeriFone\ Inc.$
  - ➢ **Tipo:** *Categórica numérica nominal, ya que* cada número representa una categoría de **TPEP** diferente, pero no sigue un orden ni una jerarquía específica.
  - ➢ **Tipo de dato:** $int32$

- `RatecodeID`: Es el código de tarifa final aplicado al terminar el viaje. Esta variable puede tomar los valores:
  - $1 = Tarifa\ estándar$
  - $2 = JFK$
  - $3 = Newark$
  - $4 = Nassau\ o\ Westchester$
  - $5 = Tarifa\ negociada$
  - $6 = Viaje\ compartido$
  - $99 = Nulo\ /\ Desconocido$
  - ➢ **Tipo:** *Categórica numérica nominal*, ya que cada número representa una categoría de tarifa final diferente, pero no sigue un orden ni una jerarquía específica.

➢ **Tipo de dato:** $float64$

- `store_and_fwd_flag`: Indica si el registro del viaje fue almacenado en la memoria del vehículo antes de enviarlo al proveedor, porque el vehículo no tenía conexión con el servidor. Esta variable puede tomar los valores:
  - $Y = viaje\ almacenado\ y\ reenviado$
  - $N = no\ almacenado\ ni\ reenviado$
  - ➢ **Tipo:** *Categórica texto nominal*, ya que cada letra representa una categoría diferente (Sí o No), pero no sigue un orden ni una jerarquía específica.
  - ➢ **Tipo de dato:** $object$

- `PULocationID`: Es la zona de taxi TLC en la que se activó el taxímetro.
  - ➢ **Tipo:** *Categórica numérica nominal*, ya que cada número representa una categoría diferente (los números identifican zonas diferentes) pero no sigue un orden ni una jerarquía específica.
  - ➢ **Tipo de dato:** $int32$

- `DOLocationID`: Es la zona de taxi TLC en la que se desactivó el taxímetro.
  - ➢ **Tipo:** *Categórica numérica nominal*, ya que cada número representa una categoría diferente (los números identifican zonas diferentes) pero no sigue un orden ni una jerarquía específica.
  - ➢ **Tipo de dato:** $int32$

- `payment_type`: Es el código numérico que indica cómo pagó el pasajero el viaje. Esta variable puede tomar los valores:
  - $1 = Tarjeta\ de\ crédito$
  - $2 = Efectivo$
  - $3 = Sin\ cargo$
  - $4 = Disputa$
  - $5 = Desconocido$
  - $6 = Viaje\ anulado$
  - ➢ **Tipo:** *Categórica numérica nominal*, ya que cada número representa una categoría de tipo de pago diferente, pero no sigue un orden ni una jerarquía específica.
  - ➢ **Tipo de dato:** $int64$

Ahora, se definirán las variables cuantitativas:

- `tpep_pickup_datetime`: Es la fecha y hora en que se activó el taxímetro.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que una fecha es una medición de tiempo que puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $datetime[64]$

- `tpep_dropoff_datetime`: Es la fecha y hora en que se desactivó el taxímetro.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que una fecha es una medición de tiempo que puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $datetime[64]$

- `passenger_count`: Es el número de pasajeros en el vehículo. Este es un valor ingresado por el conductor.
  - ➢ **Tipo:** *Cuantitativa discreta*, ya que cuenta la cantidad de pasajeros y solo puede tener valores enteros.
  - ➢ **Tipo de dato:** $float64$

- `trip_distance`: Es la distancia recorrida (en millas) reportada por el taxímetro.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición de distancia que puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $float64$

- `fare_amount`: Es el importe de la tarifa por tiempo y distancia calculada por el taxímetro.
  - ➢ **Tipo:** *Cuantitativa continua,* ya que representa un monto de dinero y puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $float64$

- `extra`: Son los cargos y recargos misceláneos. Actualmente, incluye solo los recargos de $\$0.50$ y $\$1$ aplicados en hora pico y nocturnos.
  - ➢ **Tipo:** *Cuantitativa discreta* ya que sólo puede tener los valores 0.50 y 1, si no se aplica 0.
  - ➢ **Tipo de dato:** $float64$

- `mta_tax`: Es el Impuesto **MTA** de $\$0.50$ que se aplica automáticamente según la tarifa usada.
  - ➢ **Tipo:** *Cuantitativa discreta* ya que sólo puede tener el valor 0.50, si no se aplica 0.
  - ➢ **Tipo de dato:** $float64$

- `tip_amount`: Es la propina. Este campo se completa automáticamente para propinas con tarjeta de crédito. Las propinas en efectivo no se incluyen.
  - ➢ **Tipo:** *Cuantitativa continua,* ya que representa un monto de dinero y puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $float64$

- `tolls_amount`: Es el monto total de peajes pagados en el viaje.
  - ➢ **Tipo:** *Cuantitativa continua,* ya que representa un monto de dinero y puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $float64$

- `improvement_surcharge`: Es el recargo de mejora de $\$0.30$ aplicado al inicio del viaje.
  - ➢ **Tipo:** *Cuantitativa discreta*, ya que sólo puede tener el valor 0.30, si no se aplica 0.
  - ➢ **Tipo de dato:** $float64$

- `total_amount`: Es el monto total cobrado al pasajero. No incluye propinas en efectivo.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que representa un monto de dinero y puede tomar infinitos valores dentro de un rango.
  - ➢ **Tipo de dato:** $float64$

- `congestion_surcharge`: Es el monto total cobrado en concepto de recargo por congestión del estado de Nueva York (NYS).
  - ➢ **Tipo:** *Cuantitativa discreta*, ya que solo puede tomar los valores 2.50, 1, 0.75 o sino se aplica 0.
  - ➢ **Tipo de dato:** $float64$

- `Airport_fee`: $\$1.75$ por recogida únicamente en los aeropuertos LaGuardia y John F. Kennedy.
  - ➢ **Tipo:** *Cuantitativa discreta* ya que sólo puede tener el valor 1.75, si no se aplica 0.
  - ➢ **Tipo de dato:** $float64$

Las **variables más importantes** para nuestro análisis son:

- `total_amount`, `passenger_count`, `tolls_amount`, `tip_amount`, `fare_amount`, `extra` y `trip_distance` para el análisis de la importancia de estas sobre el monto final a pagar por un viaje.
- `tpep_pickup_datetime`, `tpep_dropoff_datetime` y `PULocationID` para conocer la ubicación en la que comienzan y terminan los viajes, además de su hora de inicio y fin.
- `Airport_fee` la cual nos va a permitir conocer cuáles viajes se iniciaron desde alguno de los aeropuertos de La Guardia o John F. Kennedy.

A continuación se definirán las **hipótesis iniciales** y **supuestos**:

- A mayor cantidad de pasajeros, mayor monto a pagar.
- A mayor cantidad de pasajeros, mayor propina.
- A mayor distancia recorrida, mayor monto a pagar.
- A mayor distancia recorrida, mayor propina.
- A mayor monto de peajes, mayor propina.
- A mayor distancia, mayor es el monto de peajes.

Preprocesamiento:

Tratamiento de valores faltantes:

Para el tratamiento de los datos faltantes, lo primero que se hizo es ver cuáles variables/columnas tenían datos faltantes y qué porcentaje representan respecto de la cantidad total de observaciones en el Dataframe. Estas fueron:

- `passenger_count` con un 7.87% de datos faltantes sobre el total de observaciones.
- `RatecodeID` con un 7.87% de datos faltantes sobre el total de observaciones.

- `store_and_fwd_flag` con un 7.87% de datos faltantes sobre el total de observaciones.
- `congestion_surcharge` con un 7.87% de datos faltantes sobre el total de observaciones.
- `Airport_fee` con un 7.87% de datos faltantes sobre el total de observaciones.

A continuación se explicará el tratamiento individual que se aplicó sobre cada columna:

- Para la variable `passenger_count` lo que hicimos fue aprovechar que esta es de tipo *cuantitativa discreta* y se imputaron los datos faltantes con *sustitución por mediana*.
- Para la variable `RatecodeID` lo que hicimos fue analizar los valores que puede tomar esta columna y pudimos observar que hay tarifas que se aplican para ciertos lugares ($2 = JFK$, $3 = Newark$ y $4 = Nassau\,o\,Westchester$) por lo que se decidió que a todas las observaciones que tengan dato faltante en esta variable y que el viaje haya iniciado o terminado en alguna de estas localidades se le imputa el dato por el código de tarifa final correspondiente. En caso de no ser así, para no perder las observaciones, al resto de valores faltantes se los imputó con *sustitución por mediana*.
- Para la variable `store_and_fwd_flag` lo que hicimos fue aprovechar que esta es de tipo *categórica texto nominal* y se imputaron los datos faltantes con *sustitución por categoría más frecuente*.
- Para la variable `congestion_surcharge` lo que hicimos fue aprovechar que esta es de tipo *cuantitativa discreta* y se imputaron los datos faltantes con *sustitución por mediana*.
- Para la variable `Airport_fee` lo que hicimos fue aprovechar que esta es de tipo *cuantitativa discreta* y se imputaron los datos faltantes con *sustitución por mediana*.

Análisis de valores atípicos:

Una vez imputados los datos faltantes, se analizaron los **valores atípicos** de algunas variables de forma individual y, posteriormente, se realizó un análisis multivariado para aquellas que tenga sentido considerar en conjunto.

Primero se decidió usar *Boxplots* para observar la distribución de cada variable y visualizar *extremos univariados*, donde se pudo apreciar que en todos los *Boxplots* tenemos valores por encima de los *bigotes*, los cuales son potenciales candidatos a outliers.

Viendo los gráficos, se decidió filtrar todos aquellos valores que sean incoherentes respecto a su variable.

A continuación se explicará cuáles fueron los valores a filtrar de cada variable:

- En la variable `passenger_count`, se optó por considerar solamente aquellos viajes que hayan tenido como mínimo un pasajero y como máximo cuatro, ya que estamos trabajando sobre un dataset de taxis y es ilegal llevar a más de cuatro pasajeros.
- En la variable `trip_distance`, se optó por considerar solamente aquellos viajes que hayan tenido un recorrido mayor a 0 y menor a 62 millas, que son aproximadamente 100 kilómetros.
- En la variable `fare_amount`, se optó por considerar solamente aquellos viajes que tengan una tarifa por tiempo y distancia mayor a 0 y menor a 300 dólares.

- En la variable `extra`, se optó por considerar solamente aquellos viajes que tengan un extra mayor o igual a 0 .
- En la variable `tip_amount`, se optó por considerar solamente aquellas propinas que sean mayor o igual a 0 y menores a 100 dólares.
- En la variable `tolls_amount`, se optó por considerar solamente aquellos montos mayores a 0 y menores o iguales a 60 dólares, correspondiente al monto total de peajes pagados en cada viaje.
- En la variable `total_amount`, se optó por considerar solamente aquellos montos mayores a 0 y menores a 400 dólares, correspondientes al monto total cobrado al pasajero.

Después de filtrar las columnas con este tratamiento, se eliminaron 609.606 observaciones, que representa un 6.38% del dataset original.

Valores atípicos en `tpep_pickup_datetime`**:**
Como se mencionó anteriormente, estamos trabajando con un *Dataframe* que contiene la información de viajes de taxis amarillos de New York durante los meses de enero, febrero y marzo del año 2024. Por esta misma razón, todos aquellos viajes que iniciaran fuera de este rango, es decir, desde el $01/01/2024$ a las $00:00:00$ hasta el $31/03/2024$ a las $23:59:59$, no serán tomados en cuenta para el análisis y serán eliminados.
Después de filtrar las columnas con este tratamiento, se eliminaron 19 observaciones, que representan un 0.002% del dataset original.

Valores atípicos en `Airport_fee` **y** `PULocationID`**:**
En este caso, las variables están asociadas, ya que `Airport_fee` sólo se aplica si los taxis recogen personas, ya sea del aeropuerto *JFK* o de *LaGuardia*, en caso contrario no se aplicará esta tarifa. Esto implica que `PULocationID` debe tener como valor el número que identifique alguno de los dos aeropuertos.

- Analizando el Dataframe se detectaron 31.608 filas que tienen aplicada la tarifa de `Airport_fee` y el `PULocationID` no pertenece a ninguno de los aeropuertos.

Dada la cantidad considerable de registros afectados, se decidió adoptar el siguiente criterio para no eliminar tantas filas: Se conservarán aquellos viajes que hayan iniciado desde el área de los aeropuertos y tengan aplicada la tarifa.

Entonces:

- *LaGuardia Airport* está ubicado en *East Elmhurst, Queens*: Todos los viajes que tengan aplicada la tarifa y salgan de esta zona, serán reemplazados por el `PULocationID` del aeropuerto correspondiente.
- *John F. Kennedy* está ubicado en *Jamaica, Queens*: Todos los viajes que tengan aplicada la tarifa y salgan de esta zona, serán reemplazados por el `PULocationID` del aeropuerto correspondiente.

Después de filtrar las columnas, se eliminaron 1.806, que representa un 0.02% del dataset original.

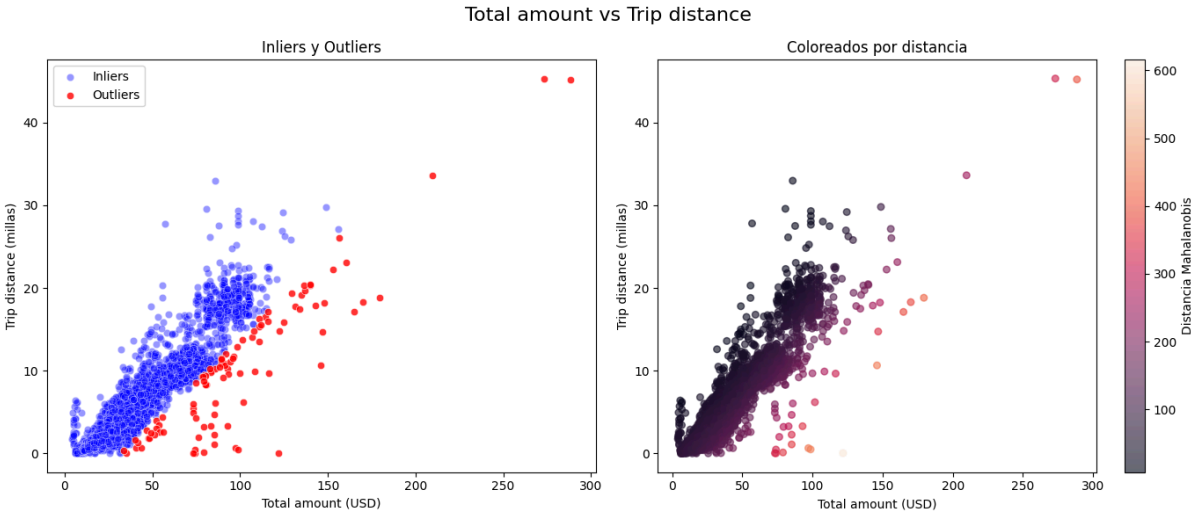
En resumen, la cantidad de observaciones eliminadas fue del: $6.38\% + 0.002\% + 0.02\% = 6.402\%$ del *Dataframe* original, lo cual es un porcentaje bastante aceptable.

Posteriormente a esto, se hicieron nuevamente *Boxplots* para ver cómo quedó la distribución de cada variable. Además, se contaron la cantidad de *outliers moderados* y *severos* respecto al *bigote superior e inferior* para cada feature. La decisión tomada respecto a este análisis fue no eliminarlos, ya que habían cantidades muy elevadas de *outliers* para algunas variables, lo que haría grandes pérdidas de información, por lo que se decidió conservarlos debido al valor que tienen los datos.

También se hizo un análisis con la métrica de *z-score modificado* para todas las variables exceptuando a la variable `tolls_amount` que utilizó la métrica *z-score*, ya que con *z-score modificado* daba valores nulos. Con estas métricas también se pudo observar una gran cantidad de *outliers* para cada feature por lo que se decidió no eliminarlos por la misma razón mencionada anteriormente.

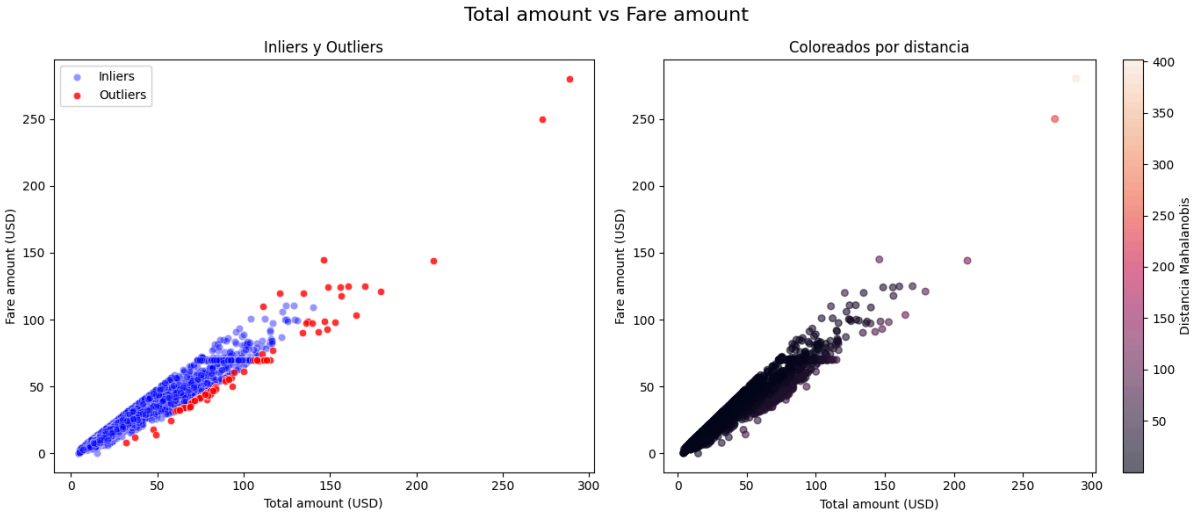
Por último, se hizo un análisis de outliers multivariados únicamente para enriquecer esta sección:

- `trip_distance` VS `total_amount`: Para este caso, se utilizó el método de detección *Mahalanobis distance*. Se calculó la *distancia de Mahalanobis* para cada punto y se hizo un gráfico de dispersión para mostrar cómo los outliers tienen una distancia más elevada respecto a los demás puntos.



  - Podemos observar algunos puntos muy distanciados respecto a los demás.

- `fare_amount` VS `total_amount`: Para este caso, se utilizó el método de detección *Mahalanobis distance*. Se calculó la *distancia de Mahalanobis* para cada punto y se hizo un gráfico de dispersión para mostrar cómo los outliers tienen una distancia más elevada respecto a los demás puntos.



- Podemos observar algunos puntos muy distanciados respecto a los demás.

- **Cálculo del umbral**: Para ambos casos, se decidió utilizar el percentil como criterio para fijar un umbral. Cuanto más alejados se encuentran los valores del centro de la elipse, mayor es la probabilidad de que sean outliers. Por esta razón, se considerarán outliers aquellos valores que se encuentren por encima del percentil 0.99.
- **Nota:** Se tomó la decisión de no eliminar los *outliers* de ambos casos, ya que están calculados respecto de una pequeña muestra (de 10.000 viajes para ser exactos) por lo que no son tan representativos respecto de los verdaderos *outliers*. Este análisis de *outliers multivariados* fue hecho como un complemento a esta sección.

Correlaciones destacables:

En está sección se mostrarán las *correlaciones* más importantes al analizar la *matriz de correlación* del *Dataframe* junto con un pequeño análisis de cada una:

- `total_amount` y `fare_amount` tienen una *correlación* muy alta: 0.98
  - **Análisis:** Esta *correlación* es bastante razonable, ya que a mayor tarifa base (mayor tiempo y distancia) del viaje, mayor monto a pagar se tendrá. El monto total está prácticamente determinado por la tarifa base.

- `total_amount` y `tip_amount` tienen una *correlación* mediana-alta: 0.69
  - **Análisis:** Esta *correlación* es razonable, ya que nos indica que la propina impacta en el monto final a pagar, pero de una manera menos significativa que la tarifa base.

- `total_amount` y `tolls_amount` tienen una *correlación* mediana-alta: 0.67
  - **Análisis:** Esta *correlación* es razonable, ya que nos indica que a mayor monto por peajes, mayor monto a pagar se tendrá.

- `total_amount` y `Airport_fee` tienen una *correlación* mediana-alta: 0.62
  - **Análisis:** Esta *correlación* es razonable, ya que nos indica que el recargo por recogida en los aeropuertos de *LaGuardia* o *John F. Kennedy* impacta en el monto final a pagar.

- `fare_amount` y `tip_amount` tienen una *correlación* mediana-alta: 0.59
  - **Análisis:** Esta *correlación* nos puede estar indicando que es más probable que se de una mayor propina en viajes con mayor duración, ya sea por distancia, tiempo o ambas.

- `fare_amount` y `tolls_amount` tienen una *correlación* mediana-alta: 0.56
  - **Análisis:** Esta *correlación* nos puede estar indicando que es más probable que en viajes largos se tengan más peajes.

- `fare_amount` y `Airport_fee` tienen una *correlación* mediana-alta: 0.58
  - **Análisis:** Esta *correlación* nos puede estar indicando que es más probable que los viajes iniciados desde los aeropuertos de *LaGuardia* o *John F. Kennedy* sean más largos.

- `mta_tax` y `improvement_surcharge` tienen una *correlación* muy alta: 0.91
  - **Análisis:** Esta *correlación* es razonable ya que son cargos fijos y obligatorios en la mayoría de los viajes.

- `tip_amount` y `tolls_amount` tienen una *correlación* considerable: 0.47
  - **Análisis:** Esto nos puede estar indicando que al tener un mayor monto de peajes, mayor propina se dará.

- `total_amount` y `trip_distance` tienen una *correlación* casi nula: 0.014
  - **Análisis:** Esto nos puede estar indicando que la distancia del viaje no tiene impacto alguno en el monto final a pagar lo que nos descarta uno de los supuestos iniciales.

- `total_amount` y `passenger_count` tienen una *correlación* casi nula: 0.036
  - **Análisis:** Esto nos puede estar indicando que la cantidad de pasajeros no tiene impacto alguno en el monto final a pagar lo que nos descarta uno de los supuestos iniciales.

- `tip_amount` y `passenger_count` tienen una *correlación* casi nula: 0.017
  - **Análisis:** Esto nos puede estar indicando que la cantidad de pasajeros no tiene impacto alguno en la propina lo que nos descarta uno de los supuestos iniciales.

Además, se realizó un análisis sobre la *matriz de covarianza* del *Dataframe* para fortificar el análisis de las correlaciones entre variables mencionadas. Sin embargo, si bien la matriz no está normalizada, no se encuentran valores significativos. Lo más destacable fue:

- `total_amount` y `fare_amount` tienen una *covarianza* muy alta, lo que fortifica mucho más la influencia directa que tiene la tarifa base sobre el monto final a pagar.
- `PULocationID` y `DOLocationID` presentan valores significativos de *covarianza* respecto a algunas variables, pero no aportan información útil, ya que son variables *categóricas numéricas nominales*.

Nuevos features generados:

Para un mejor análisis de los datos, se tomó la decisión de crear los siguientes features:

- `pickup_time_of_day`: Es una variable que nos dirá el horario del día en que fue iniciado el viaje. Esta variable puede tomar los valores:
  - $morning = Mañana$ ← Si el viaje empieza entre las $06:00$ y $11:59$.
  - $afternoon = Tarde$ ← Si el viaje empieza entre las $12:00$ y $17:59$.
  - $evening = Noche$ ← Si el viaje empieza entre las $18:00$ y $23:59$.
  - early_morning $= Madrugada$ ← Si el viaje empieza entre las $00:00$ y $05:59$.
  - ➢ **Tipo:** *Categórica texto ordinal*, ya que los periodos del día tienen un orden secuencial y forman parte del ciclo diario.

- `trip_distance_category`: Es una variable que nos dirá qué tipo de categoría de distancia tiene el viaje. Esta variable puede tomar los valores:
  - short_distance $= Distancia\ corta$ ← Si el viaje tiene una distancia menor a 5 millas.
  - medium_distance $= Distancia\ media$ ← Si el viaje tiene una distancia entre 5 y 10 millas.
  - large_distance $= Distancia\ larga$ ← Si el viaje tiene una distancia mayor a 10 millas.
  - ➢ **Tipo:** *Categórica texto ordinal*, ya que las categorías representan niveles de distancia con un orden jerárquico.

- `day_of_week`: Es una variable que nos dirá el día de la semana en el cual se inició el viaje. Esta variable puede tomar los valores:
  - $monday = Lunes.$
  - $tuesday = Martes.$
  - $wednesday = Miércoles.$
  - $thursday = Jueves.$
  - $friday = Viernes.$
  - $saturday = Sábado.$
  - $sunday = Domingo.$
  - ➢ **Tipo:** *Categórica texto ordinal*, ya que los días de la semana tienen un orden secuencial y forman parte de un ciclo natural.

- `holiday_or_weekend`: Es una variable que nos dirá si el día en que se inició el viaje era un feriado o un día del fin de semana. Esta variable puede tomar los valores:
  - $\mathrm{Y} = Si$ ← El viaje se inició en un feriado o un día del fin de semana.
  - $\mathrm{N} = No$ ← El viaje no se inició en un feriado o un día del fin de semana.

➢ **Tipo:** *Categórica texto nominal*, ya que cada número representa una categoría diferente (Sí o No) pero no sigue un orden ni una jerarquía específica.

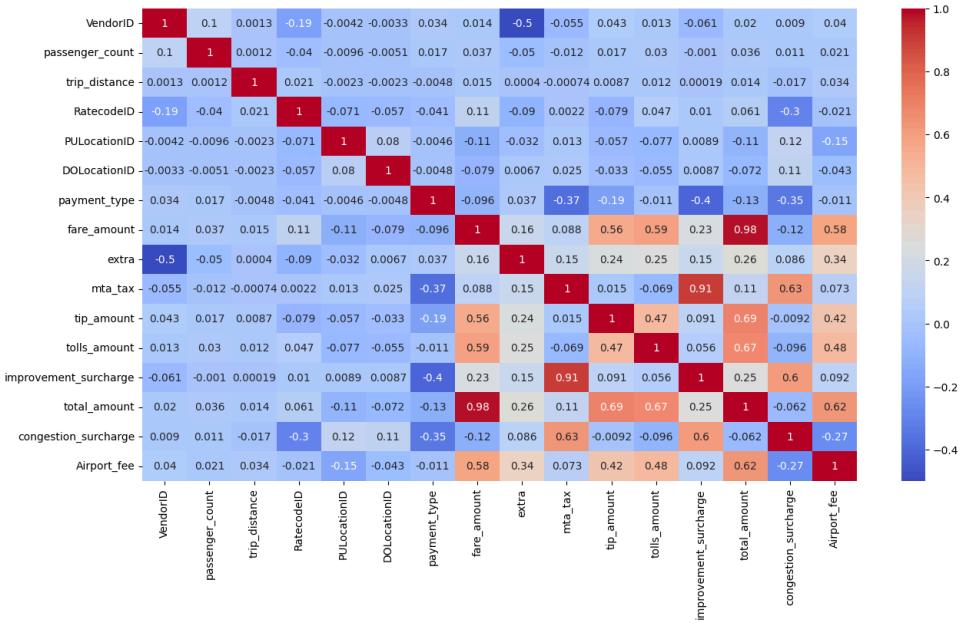
Columnas eliminadas:
Se tomó la decisión de eliminar del dataset las variables `store_and_fwd_flag`, `mta_tax`, `congestion_surcharge`, `improvement_surcharge` y `VendorID` ya que estas no son utilizadas en ningún momento para responder las preguntas de investigación planteadas en la siguiente sección.

Visualizaciones y preguntas de investigación:

En este apartado se mostrará la definición de cada pregunta de investigación y su desarrollo, estas preguntas fueron planteadas una vez terminado todo el preprocesamiento de datos mencionado en la sección anterior.

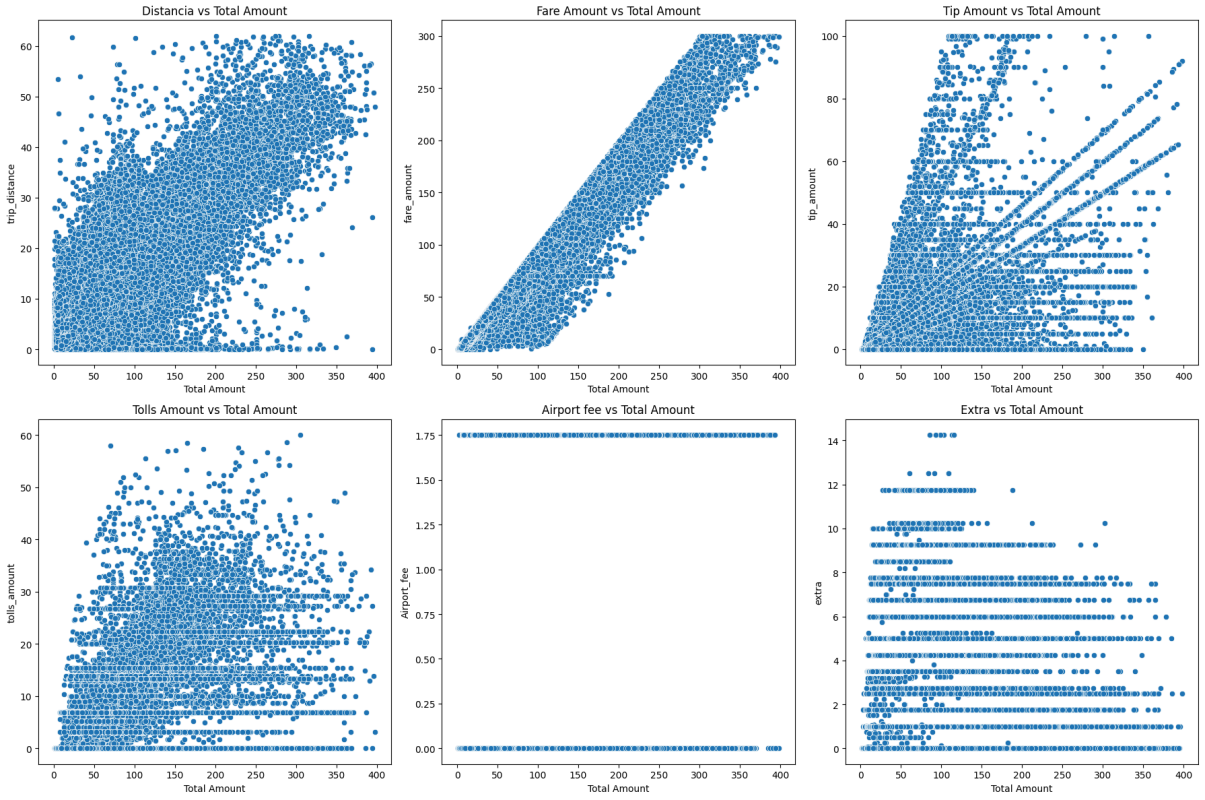
1. **¿Qué variables son más influyentes para incrementar el costo final de un viaje en taxi?**

   Para responder esta pregunta, nos basamos en el análisis de correlación hecho anteriormente, en el cuál, pudimos observar que el costo final (`total_amount`) tiene una alta correlación con las variables:
   - `fare_amount` con un coeficiente de correlación de $0,98$.
   - `tolls_amount` con un coeficiente de correlación de 0.67.
   - `tip_amount` con un coeficiente de correlación de 0.69.
   - `Airport_fee` con un coeficiente de correlación de 0.62.



   - *Heatmap* de la matriz de correlación para visualizar los coeficientes de correlación.

A continuación se realizarán los *Scatter plots* de cada una de estas variables y otras más para ver la correlación de cada una con el costo final:



Viendo los distintos gráficos de dispersión podemos observar que:

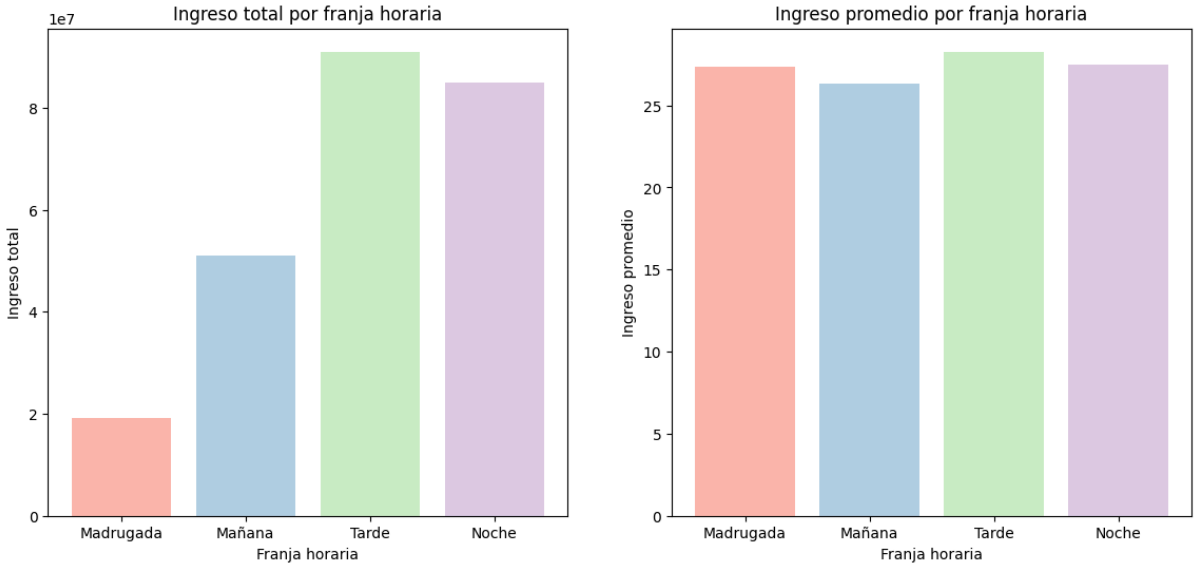
- `trip_distance` y `total_amount` tienen los valores muy dispersos por lo que no se ve una relación clara. Además tiene un coeficiente de correlación cercano a cero $(0,014)$. Por lo que podemos refutar uno de nuestros supuestos: “A mayor distancia recorrida, mayor monto a pagar”.
- `fare_amount` tiene una relación lineal muy fuerte respecto a `total_amount`. Entonces, podemos decir que a mayor tarifa base, mayor costo final se tendrá.
- `tip_amount` tiene una relación lineal con `total_amount` pero más dispersa. Entonces, podemos decir que las propinas elevadas hacen crecer significativamente al costo final, aunque esto depende del pasajero.
- `tolls_amount` y `total_amount` tienen los valores muy dispersos cuando el eje Y es cero, pero, se ve una tendencia de crecimiento cuando ambos valores crecen. Además, estas tienen un coeficiente de correlación considerable.
- `Airport_fee` y `total_amount` no tienen un *scatter plot* muy descriptivo debido a la poca cantidad de números que toma `Airport_fee`. Además de tener una frecuencia baja por su naturaleza, ya que si bien se tienen viajes que salen desde los aeropuertos, los mismos no representan una cantidad significativa en comparación de la totalidad viajes que se hacen. Sin embargo, estas tienen un coeficiente de correlación considerable.

- `extra` y `total_amount` no tienen un *scatter plot* muy descriptivo debido a la poca cantidad de números que toma `extra`, sin embargo, estas tienen un coeficiente de correlación bajo respecto de los demás pero aún considerable $(0,26)$.

En conclusión, podemos decir que el costo total a pagar en un viaje (`total_amount`), depende principalmente de la tarifa base (`fare_amount`) seguido en un segundo orden por el monto total de los peajes (`tolls_amount`) y la propina (`tip_amount`). Ya en un tercer escalón irían `Airport_fee` y `extra`.

2. ¿En qué franja horaria (madrugada, mañana, tarde, noche) los taxistas obtienen mayores ingresos promedio por viaje?

Para responder esta pregunta lo que hicimos fue sumar el ingreso total de cada viaje para cada franja horaria (*madrugada*, *mañana*, *tarde* y *noche*) y a ese total dividirlo por la cantidad de viajes realizados en el horario correspondiente.



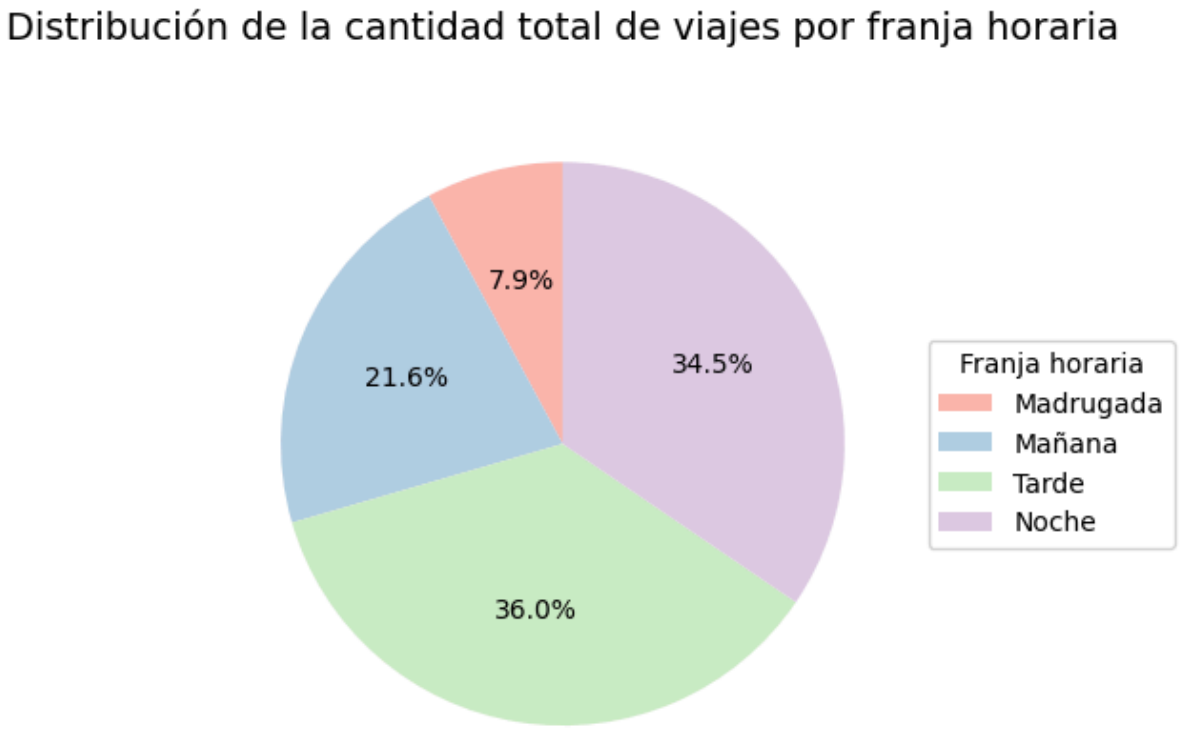
- Se decidió usar *Bar plots* ya que tenemos un soporte discreto en el eje X y una cantidad en el eje Y.

Podemos observar que la franja horaria en la cual los taxistas obtienen mayores ingresos promedio por viaje es la *tarde* (de $12{:}00$ *a* $17{:}59$), que es también en donde mayores ingresos totales se generaron.

Observaciones:

- Los ingresos promedio por viaje en cada rango horario no son muy distantes entre sí.
- La franja horaria de la *madrugada* (de $00{:}00$ *a* $05{:}59$) es la que generó el menor ingreso total. Sin embargo, no es la que menor ingreso promedio tiene. Es decir, son viajes rentables pero poco frecuentes.

3. **¿Qué proporción de los viajes se produce en cada horario del día (madrugada, mañana, tarde y noche)?**

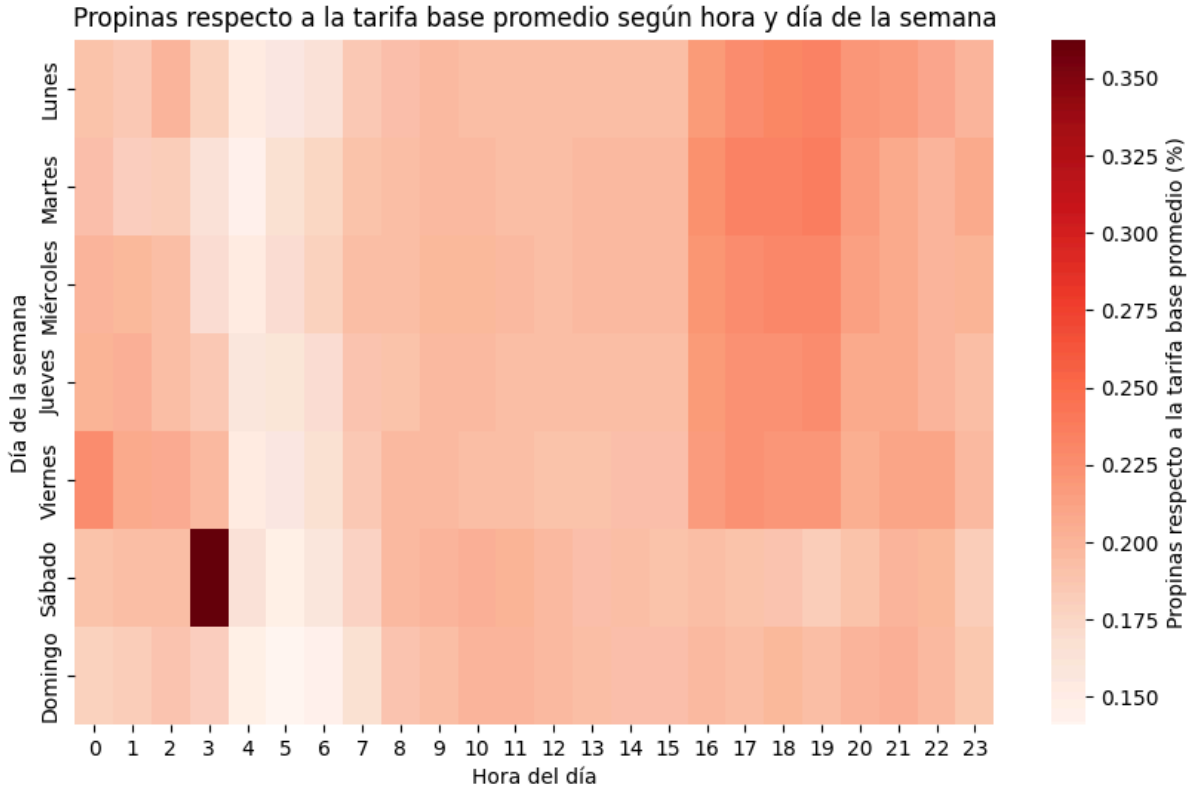


   - Se decidió utilizar un *gráfico de torta, ya que* estos se usan para mostrar proporciones (se ven porcentajes).

Podemos observar que la mayor proporción de los viajes son en la *tarde* $(12{:}00\ a\ 17{:}59)$ seguido muy de cerca por la *noche* $(18{:}00\ a\ 23{:}59)$. Esto se puede deber a que la hora pico se encuentra distribuida entre estas dos franjas horarias. El tercer rango horario con mayor proporción de viajes es la *mañana* $(06{:}00\ a\ 11{:}59)$ pero muy alejada de las dos primeras. La franja con el menor porcentaje de viajes es la *madrugada* $(00{:}00\ a\ 05{:}59)$.

4. **¿En qué horario y día de la semana los pasajeros dejan propinas más generosas en proporción a lo que pagan?**

Para responder esta pregunta lo que se hizo es a cada propina dividirla por la tarifa base para obtener la proporción relativa que ésta representa, para así después sacar un promedio por día y hora. Esto permitió normalizar el efecto de trayectos más largos o más costosos y enfocarse en la generosidad relativa del pasajero.

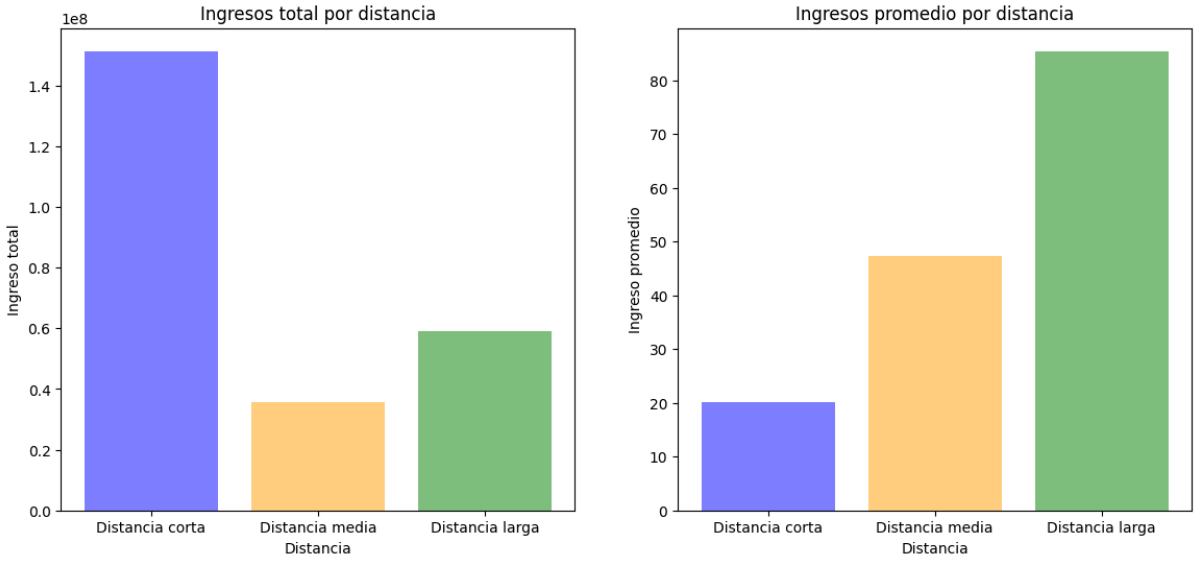


- Dado que el análisis involucra tres dimensiones simultáneamente (día, hora y propina relativa promedio), se decidió utilizar un Heatmap.

En este gráfico podemos observar que los pasajeros dejan las propinas más generosas los días sábados a las tres de la mañana. Esto se podría asociar a viajes nocturnos vinculados a actividades recreativas, como salidas a bares o clubes.

5. **¿Los taxistas obtienen mayores ingresos por viajes cortos (distancia menor a 5 millas), viajes medianos (distancia mayor o igual a 5 millas y menor a 10 millas) o viajes largos (distancia mayor a 10 millas)?**

   Para responder esta pregunta, primero se calculó el ingreso total para cada categoría de distancia (*corta*, *mediana* y *larga*). Después, a cada total se lo dividió por la cantidad de viajes realizados por cada categoría:



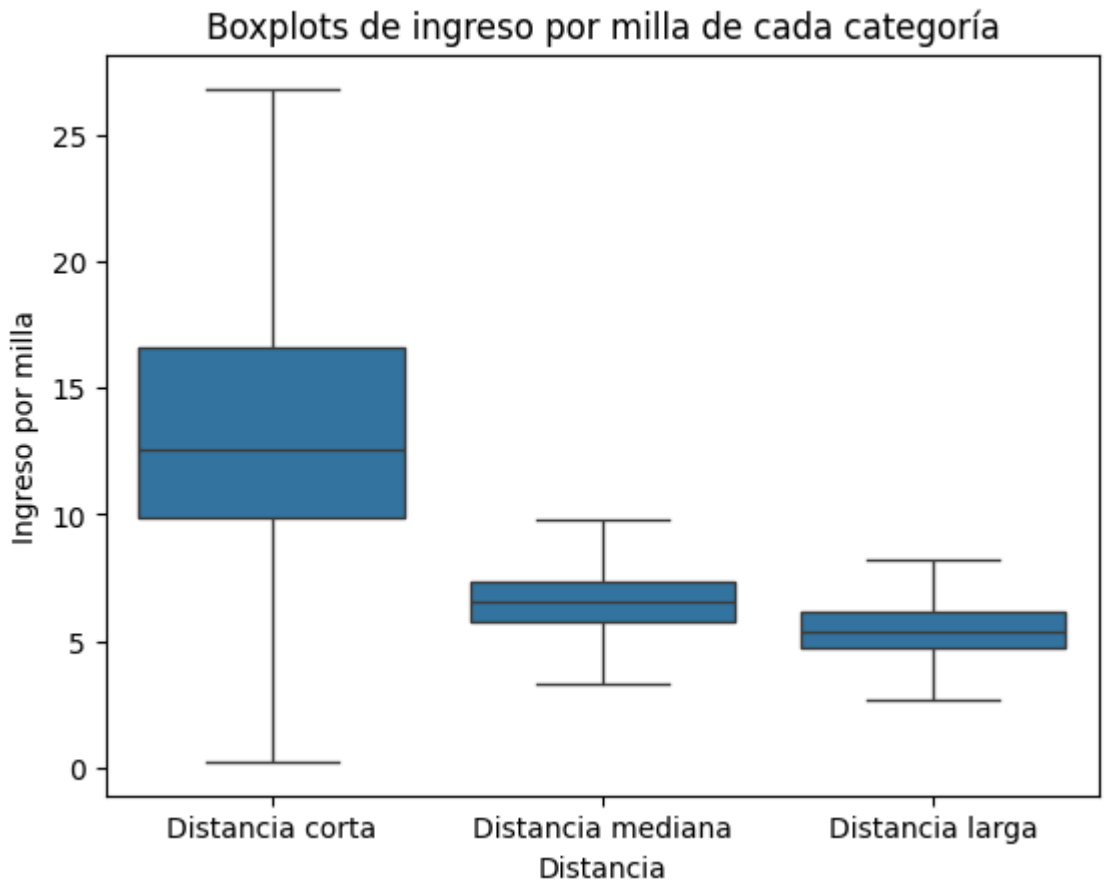
- Se decidió usar *Bar plots*, ya que tenemos un soporte discreto en el eje X y una cantidad en el eje Y.

En el primer gráfico podemos observar que el mayor ingreso total lo generaron los viajes de *distancia corta*.
En el segundo gráfico podemos observar que la categoría de distancia con mayor ingreso promedio es la *distancia larga*.

Observaciones:

- La categoría que menor ingreso promedio tiene es la categoría *distancia corta*, incluso siendo la que mayor ingreso total generó al ser la más frecuente.
- Lo que está pasando es que los viajes de *larga distancia* y *media distancia* son menos frecuentes y estos tienen algunos viajes con mucho ingreso, lo que influye en su promedio.

Como no se ve una respuesta clara, se decidió sacar el ingreso normalizado por milla recorrida de cada viaje para cada categoría de distancia y realizar un Boxplot para cada una:

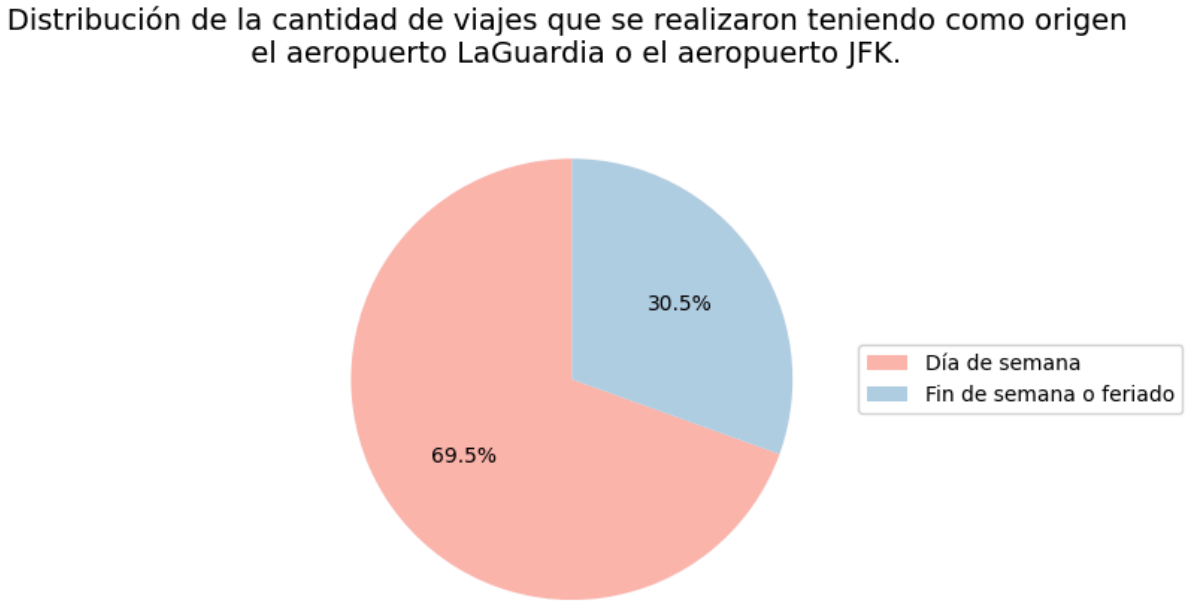


- Se decidió utilizar *Boxplots* para ver la distribución de los valores que toman los ingresos por milla de cada categoría de distancia.

En este gráfico se puede observar que la mejor media de ingreso por milla lo tienen los viajes *de distancia corta*. Por lo que podemos concluir que para los taxistas es más conveniente hacer viajes cortos, ya que tienen un mayor ingreso por milla que los viajes de mediana y larga distancia a pesar de que estos tengan un mejor ingreso promedio.

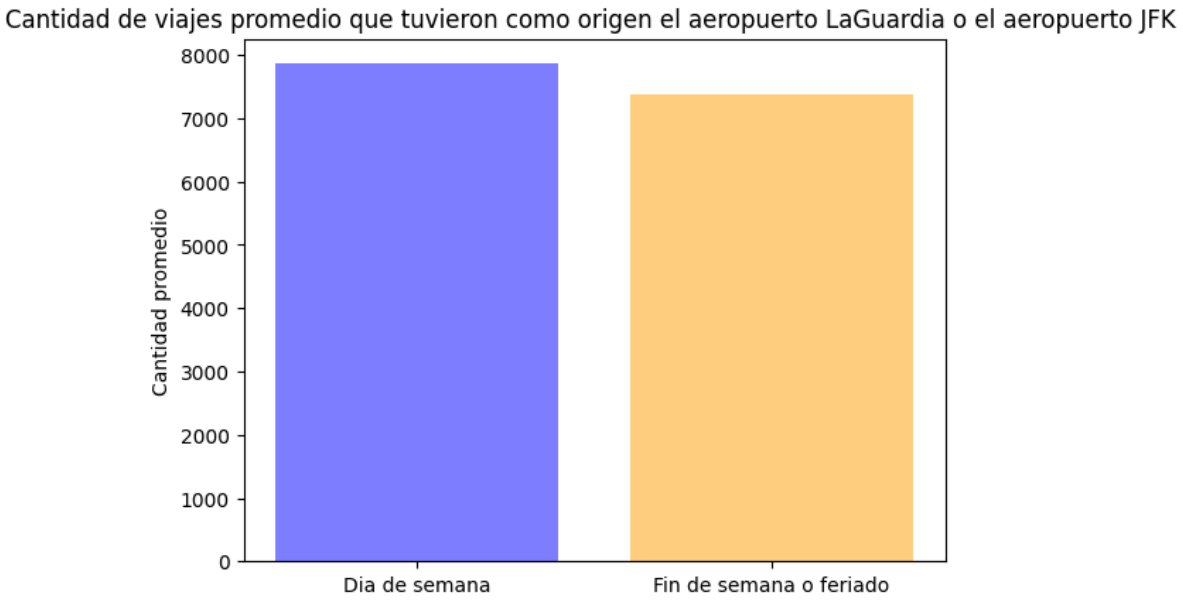
6. **¿Los viajes iniciados desde los aeropuertos de LaGuardia o John F. Kennedy son más frecuentes en fines de semana o feriados?**

Para responder esta pregunta, lo primero que se hizo es ver la proporción de viajes iniciados desde alguno de los aeropuertos mencionados en días de semana y en fines de semana/feriados.



- Se decidió utilizar un *gráfico de torta* ya que estos se usan para mostrar proporciones (se ven porcentajes).

En este gráfico podemos observar que la gran mayoría de los viajes se realizan en días de semana.
Sin embargo, para obtener una visión más precisa, se decidió normalizar la medición, ya que en el período analizado (enero a marzo de 2024) existen más días de semana que fines de semana/feriados. Para ello, se dividió la cantidad de viajes de cada tipo de día por la cantidad de días de ese tipo en el período considerado.



- Se decidió usar *Bar plots* ya que tenemos un soporte discreto en el eje X y una cantidad en el eje Y.

En este gráfico podemos ver que la cantidad de viajes promedio iniciados desde alguno de los aeropuerto mencionados que se realizaron es ligeramente mayor en los días de semana. Por lo que podemos concluir que estos viajes son más frecuentes en los días de semana, aunque la diferencia no es tan amplia.

7. **¿Cuáles son las zonas de destino más frecuentes en los viajes iniciados desde los aeropuertos de LaGuardia o John F. Kennedy?**

Treemap de destinos seleccionados que parten de los aeropuertos LaGuardia y JKF

- Se decidió utilizar un *Treemap* ya que se pueden visualizar las zonas y sub-zonas más frecuentemente seleccionadas como destino de una manera clara.

En este gráfico podemos observar las zonas más frecuentemente seleccionadas como destino desde los Aeropuertos John F. Kennedy o LaGuardia son principalmente las de **Manhattan,** lo cual tiene bastante sentido, ya que es el lugar más turístico de Nueva York.

Ejercicio 2 - Modelos de Clasificación Binaria:

Descripción del dataset:

El dataset utilizado son de datos brindados por estaciones meteorológicas de Australia con el objetivo de poder predecir si lloverá el día de mañana, estos datos son de distintas regiones y localizaciones del país oceánico. Para nuestro análisis y armado de nuestro modelo, sólo se tomaron aquellos que fueron tomados en localidades pertenecientes a las regiones de Nueva Gales del Sur o Victoria.

Una vez filtrados los datos de interés, finalmente el dataset estuvo formado por 74.492 filas y 23 columnas.

A continuación, se definirá el tipo de variable para cada columna:

Primero se definirán las categóricas:

- `Date:` Indica la fecha en la que se tomó la observación.
  - ➢ **Tipo:** *Categórica numérica ordinal*, ya que hay un orden. La diferencia de valores depende de la escala (año/mes/día).
  - ➢ **Tipo de dato:** $object$

- `Location:` Indica el nombre de la localidad donde se encuentra la estación meteorológica en donde fue tomada la observación.
  - ➢ **Tipo:** *Categórica texto nominal*, ya que cada categoría representa una localidad y estas no siguen un orden.
  - ➢ **Tipo de dato:** $object$

- `WindGustDir:` Indica la dirección del viento de la ráfaga más fuerte.
  - **Tipo:** *Categórica texto nominal*, ya que cada categoría representa un punto cardinal y no existe un orden natural entre ellos.
  - **Tipo de dato:** $object$

- `WindDir9am:` Indica la dirección del viento a las $9\ am.$
  - ➢ **Tipo:** *Categórica texto nominal*, ya que cada categoría representa un punto cardinal y no existe un orden natural entre ellos.
  - ➢ **Tipo de dato:** $object$

- `WindDir3pm:` Indica la dirección del viento a las $3\ pm.$
  - ➢ **Tipo:** *Categórica texto nominal*, ya que cada categoría representa una localidad y estas no siguen un orden.
  - ➢ **Tipo de dato:** $object$

- `RainToday:` Indica si llovió el día en el que se registró la observación. Esta variable puede tomar los siguientes valores:
  - $Yes = Llovió.$

- $No = No\ llovió.$
- ➢ **Tipo:** *Categórica texto nominal*, ya que cada letra representa una categoría diferente (Sí o No) pero no sigue un orden ni una jerarquía específica.
- ➢ **Tipo de dato:** $object$

- `RainTomorrow`: Indica si lloverá el día siguiente al que se tomó la observación. Esta variable puede tomar los siguientes valores:
  - $Yes = Lloverá.$
  - $No = No\ lloverá.$
  - ➢ **Tipo:** *Categórica texto nominal*, ya que cada letra representa una categoría diferente (Sí o No) pero no sigue un orden ni una jerarquía específica.
  - ➢ **Tipo de dato:** $object$

Ahora, se definirán las variables cuantitativas:

- `MinTemp`: Indica la mínima temperatura en grados celsius.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `MaxTemp`: Indica la máxima temperatura en grados celsius.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Rainfall`: Indica la cantidad de lluvia registrada durante el día en milímetros.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Evaporation`: Indica la cantidad de agua evaporada en milímetros.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Sunshine`: Indica el número de horas de sol brillante en el día.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `WindGustSpeed`: Indica la velocidad de la ráfaga de viento más fuerte en las últimas 24 horas en kilómetros por hora.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `WindSpeed9am`: Indica la velocidad del viento promediada durante diez minutos antes de las $9\ am$ en kilómetros por hora.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `WindSpeed3pm`: Indica la velocidad del viento promediada durante diez minutos antes de las $3\ pm$ en kilómetros por hora.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Humidity9am`: Indica la humedad (porcentaje)a las $9\ am$.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que puede tomar cualquier valor real entre 0% y 100%.
  - ➢ **Tipo de dato:** $float64$

- `Humidity3pm`: Indica la humedad (porcentaje)a las $3\ pm$.
  - ➢ *Cuantitativa continua*, ya que puede tomar cualquier valor real entre 0% y 100%.
  - ➢ **Tipo de dato:** $float64$

- `Pressure9am`: Indica la presión atmosférica a las $9\ am$.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Pressure3pm`: Indica la presión atmosférica a las $3\ pm$.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Cloud9am`: Indica la fracción del cielo oscurecida por nubes a las $9\ am$.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Cloud3pm`: Indica la fracción del cielo oscurecida por nubes a las $3\ pm$.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

- `Temp9am`: Indica la temperatura a las $9\ am$ en grados celsius.
  - ➢ **Tipo:** *Cuantitativa continua*, ya que es una medición y puede tomar infinitos valores en un rango.

- ➢ **Tipo de dato:** $float64$

- `Temp3pm`: Indica la temperatura a las $3\ pm$ en grados celsius.
  - ➢ **Tipo:** ***Cuantitativa continua***, ya que es una medición y puede tomar infinitos valores en un rango.
  - ➢ **Tipo de dato:** $float64$

Una vez definidos los tipos de variables y realizada la exploración de datos, decidimos no borrar ningúna columna, ya que todas las variables nos parecen importantes para poder entrenar nuestros modelos.

Tratamiento de valores faltantes:
Dado que no tenemos una cantidad inicial de observaciones muy grande, es muy caro tener que eliminar registros por datos faltantes debido a la información que pueden tener en los demás campos por lo que se optó por no eliminar filas exceptuando a las que tengan datos nulos en los campos `RainToday`, `RainTomorrow` y `RainFall` al mismo tiempo, ya que no queremos inventar datos que puedan sesgar los modelos (de `RainTomorrow` se eliminarán todas las observaciones que tengan *nulos*). A continuación se describirán las diferentes decisiones tomadas respecto a la imputación de los *datos faltantes* en las columnas que presentan estos tipos de datos:

- Para las columnas con un porcentaje menor o igual al 10% de *datos faltantes*, se imputaron los valores faltantes por *sustitución por mediana* (en caso de que la variable sea *cuantitativa*) o por sustitución por categoría más frecuente (en caso de que la variable sea categórica). Las únicas excepciones a esto serán las columnas `RainFall` y `RainToday` que quedaron con *valores faltantes* para evitar sobreajuste en los modelos.
- Para las comunas con un porcentaje mayor al 30% de *datos faltantes* (el resto de las columnas), se decidió no imputarlas, ya que estaríamos inventando más de un tercio de los valores de la columna, lo que ocasiona un sesgo muy grande. Además, se aprovechó que los modelos entrenados soportan trabajar con datos nulos.

En la eliminación de las filas con *datos faltantes* en las columnas `RainToday`, `RainTomorrow` y `RainFall` se filtró el 1.78% del dataset original. El resto de observaciones con *valores faltantes* en `RainTomorrow` significó un 0.98% del total.

Tratamiento de valores atípicos:
Para el análisis de los *valores atípicos,* primero se realizó un filtrado de *outliers* con las columnas `WindGustSpeed`, `WindSpeed9am` y `WindSpeed3pm`, ya que las dos últimas no pueden ser mayores que `WindGustSpeed` porque esta representa la mayor velocidad del viento en las últimas 24 horas. En el dataframe se encontraron 134 observaciones (el 0.18% del dataset) que no cumplían esta condición, por lo que se decidió borrarlas al ser un porcentaje bastante bajo.

También se realizó un filtrado de *outliers* con las columnas `MaxTemp`, `MinTemp`, `Temp9am` y `Temp3pm`, ya que las dos últimas no pueden ser mayores que `MaxTemp` porque esta

representa la mayor temperatura y no pueden ser menores que `MinTemp` porque esta representa la menor temperatura. En total, en el dataframe se encontraron 1486 observaciones (el 1.59% del dataset) que no cumplían con alguna de estas condiciones, por lo que se decidió borrarlas al ser un porcentaje bastante bajo.

Después, se realizaron los *Boxplots* de las *variables cuantitativas* para ver cómo se distribuyen y visualizar los *outliers* que están por fuera de los bigotes tanto inferior como superior. En estos gráficos se pudo observar que algunas columnas tienen valores que superan a los bigotes inferiores y superiores, pero en general, estos no representan una cantidad significativa ni tampoco son valores irreales. Por esta razón, se decidió conservar dichos *outliers*.

Valores atípicos multivariados:
Si bien se analizó la existencia de valores atípicos multivariados y se hicieron gráficos, estos son sobre un subconjunto del dataset, por lo que se decidió no eliminar ninguna columna, ya que no es muy representativo del conjunto total de observaciones.

Finalmente, después del tratamiento de valores faltantes y el análisis de valores atípicos se eliminó un $4,53\%$ del conjunto de datos inicial que es un porcentaje aceptable para poder entrenar los modelos.

Transformaciones variables:
Creación y eliminación de variables: Se decidió crear dos nuevas variables llamadas `Year` (que tendrá el año en que se tomó la observación) y `Month` (que tendrá el mes en que se tomó la observación). Con estas variables creadas se tomó la decisión de borrar la columna `Date`, ya que la información importante que esta posee la dividimos en las nuevas columnas.

Encoding: Como vamos a utilizar modelos necesitan trabajar con variables numéricas, se decidió que a cada una de las variables categóricas como `Location`, `WindGustDir`, `WindDir9am`, `WindDir3pm`, `RainToday` y `RainTomorrow` recodificarlas en variables dummies (One Hot Encoding) aprovechando que estas no contienen demasiadas categorías.

Normalización: Se decidió utilizar *normalización z-score* en las variables `WindGustSpeed`, `WindSpeed9am`, `WindSpeed3pm`, `Humidity9am`, `Humidity3pm`, `Pressure9am` y `Pressure3pm`. Esta técnica permite centrar los datos en torno a la media y escalarlos en función de la desviación estándar, esto nos ayudará a tener un mayor control sobre la influencia de los valores atípicos.

Árbol de decisión:

Primero nos fijamos si hay un desbalance en las clases de la variable objetivo `RainTomorrow`:

- Hay 55.179 de filas con NO (aproximadamente el 77.59% del dataset).

- Hay 15.939 de filas con SI (aproximadamente el 22.41% del dataset).

Como podemos observar, hay un claro desbalance en donde la clase NO predomina sobre la clase SI, lo que puede ocasionar un sesgo en el modelo. Para evitar esto se utilizó el método de *Oversampling (*con un `sampling_strategy: 0,5)` sobre el conjunto que se utilizó para entrenar el modelo (80% del dataset final). Una vez aplicado el método nos quedó:

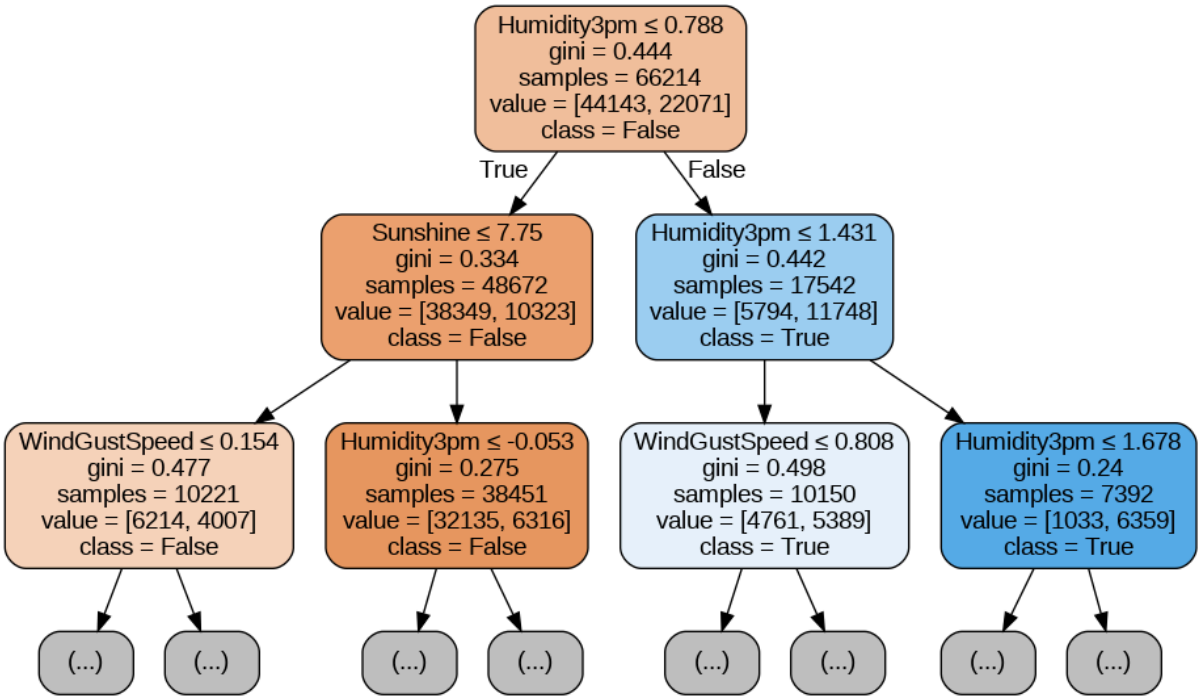
- 44.143 filas con NO (aproximadamente un 66.67% del dataset de entrenamiento)
- 22.071 filas con SI (aproximadamente un 33.33% del dataset de entrenamiento).

Para la construcción de un *árbol de decisión* se optimizó los *hiperparámetros* con la técnica de *5-fold Cross Validation* utilizando la exploración de hiperparámetros por *Random Search* debido a su eficiencia en la búsqueda dentro de un espacio amplio de combinaciones. Los hiperparámetros elegidos fueron:

- `criterion`: Es la función utilizada para medir la calidad de una división (criterio).
- `max_depth`: Es la profundidad máxima en la que puede crecer el árbol.
- `min_samples_split`: Es el número mínimo de muestras necesarias para dividir un nodo interno.
- `min_samples_leaf`: Es el número mínimo de muestras que debe haber en un nodo hoja.

Se priorizó la métrica *f1-score* para tener un mejor equilibrio entre el *recall* y la *precisión*. Una vez obtenidos los hiperparámetros después de probar con distintos rangos de valores, se entrenó el *árbol de decisión*.

Por lo extenso que es el árbol se decidió hacer un análisis hasta el segundo nivel del mismo:



En el gráfico del *árbol de decisión* podemos ver que tiene:

- Un *nodo raíz*, que es aquella variable que nos da la mayor *ganancia de información*.
- *Nodos internos*, que representan *preguntas* con respecto al *valor* de uno de los atributos.

Podemos ver que la variable que mayor ganancia de información nos da es `Humidity3pm`.

- La *regla principal* es `Humidity3pm` $\leq 0,788$ (si la humedad a las $3\ pm$ pm es menor o igual que $0,788$ sigue por el lado izquierdo, sino por el derecho).
- Por la raíz pasaron 66.214 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).

<u>Por el lado izquierdo de la raíz:</u>

Podemos ver que el siguiente *nodo interno* es sobre la variable `Sunshine`.

- La regla es `Sunshine` $\leq 7,75$.
- Por este nodo pasaron 48.672 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).

Después de este nodo, siguen los nodos internos sobre las variables:

- `WindGustSpeed`:
  - La regla es `WindGustSpeed` $\leq 0,154$.
  - Por este nodo pasaron 10.221 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).
- `Humidity3pm`:
  - La regla es `Humidity3pm` $\leq -0,053$.
  - Por este nodo pasaron 38.451 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).

<u>Por el lado derecho de la raíz:</u>

Podemos ver que el siguiente *nodo interno* es otra vez sobre la variable `Humidity3pm`.

- La regla es `Humidity3pm` $\leq 1,431$.
- Por este nodo pasaron 17.542 muestras de las cuales mayoritariamente son $True$ (lloverá al día siguiente).

Después de este nodo, siguen los nodos internos sobre las variables:

- `WindGustSpeed`:
  - La regla es `WindGustSpeed` $\leq 0,808$.
  - Por este nodo pasaron 10.150 muestras de las cuales mayoritariamente son $True$ (lloverá al día siguiente).
- `Humidity3pm`:
  - La regla es `Humidity3pm` $\leq 1,678$.
  - Por este nodo pasaron 7.392 muestras de las cuales mayoritariamente son $True$ (lloverá al día siguiente).

Viendo el *árbol de decisión* podemos concluir que la variable más influyente para predecir si *va a llover al día siguiente* es `Humidity3pm`, que aparece como la raíz. Esto indica que la *humedad a las* $3\ pm,$ es el factor con mayor poder de separación entre los días con y sin

lluvia al día siguiente. Podemos reforzar la importancia de esta variable viendo que aparece en todos los niveles del árbol vistos en el gráfico.

En los niveles inferiores, las divisiones se hacen principalmente por `Sunshine`, `WindGustSpeed` y `Humidity3pm`, lo que nos puede indicar que la cantidad de horas de sol y la velocidad máxima del viento también tienen una relación significativa con la probabilidad de lluvia.

Nota: Los valores de las condiciones (como las que involucran a las variables `Humidity3pm` y `WindGustSpeed`) se encuentran en *escala estandarizada* debido a la *normalización por z-score* realizada en estos atributos, por lo que representan desviaciones respecto al promedio en lugar de valores físicos directos (como $\frac{km}{h}$ o porcentaje de humedad). Esto **no** altera las decisiones del modelo, pero hace que los puntos de corte sean menos interpretables en unidades reales.

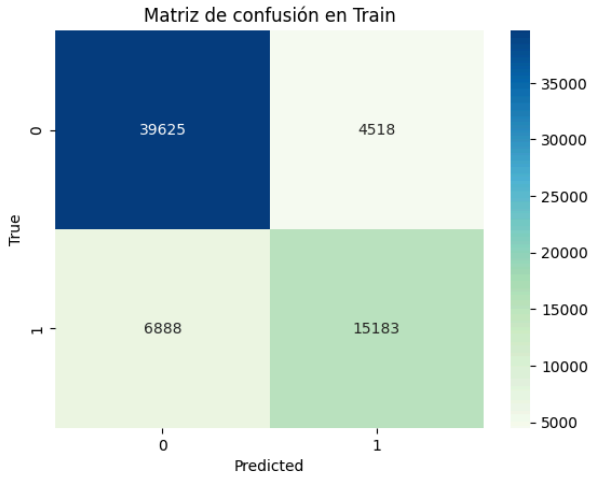
Al evaluar la performance del modelo con los *datos de entrenamiento* (*train*) se obtuvo los siguientes resultados para las métricas:

- $Accuracy$: $0,8277$
- $Recall$: $0,6879$
- $Precision$: $0,7707$
- $f1 - score$: $0,7269$

Después, se evaluó la performance del modelo con los *datos de prueba* (*test*) con los que se obtuvo los siguientes resultados para las métricas:

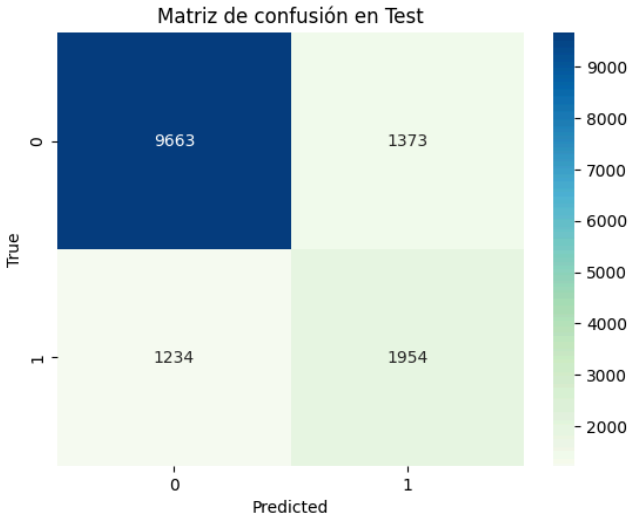
- $Accuracy$: $0,8167$
- $Recall$: $0,6129$
- $Precision$: $0,5873$
- $f1 - score$: $0,5998$

<u>Matriz de Confusión en entrenamiento:</u>



- $True\ Positive\ (TP)$: 39625
- $True\ Negative\ (TN)$: 15183
- $False\ Positive\ (FP)$: 4518
- $False\ Negative\ (FN)$: 6888

Matriz de Confusión en test:



- $True\ Positive\ (TP)$: 1954
- $True\ Negative\ (TN)$: 9663
- $False\ Positive\ (FP)$: 1373
- $False\ Negative\ (FN)$: 1234

Podemos observar que el modelo tiene una buena capacidad para predecir los días que **no** va a llover el día siguiente (clase mayoritaria) pero tiene una mayor dificultad con los días que **si** va a llover el día siguiente (clase minoritaria). Entonces, se puede decir que este modelo no logra capturar de buena forma los patrones de la clase minoritaria.

Random Forest:

Para evitar el desbalance de las clases en la variable objetivo esto se utilizó el método de *Oversampling (*con un `sampling_strategy: 0,65`*)* sobre el conjunto que se utilizó para entrenar el modelo (80% del dataset final). Una vez aplicado el método nos quedó:

- 44.143 filas con NO (aproximadamente un 60.61% del dataset de entrenamiento)
- 28.692 filas con SI (aproximadamente un 39.39% del dataset de entrenamiento).

Antes de utilizar el algoritmo de *Random Forest* se optimizó los hiperparámetros con la técnica *de 5-fold Cross Validation* utilizando la exploración de hiperparámetros por Random Search debido a su eficiencia en la búsqueda dentro de un espacio amplio de combinaciones. Los hiperparámetros elegidos fueron:

- `criterion`: Es la función utilizada para medir la calidad de una división (criterio).
- `max_depth`: Es la profundidad máxima en la que puede crecer el árbol.

- `min_samples_split`: Es el número mínimo de muestras necesarias para dividir un nodo interno.
- `min_samples_leaf`: Es el número mínimo de muestras que debe haber en un nodo hoja.
- `n_estimators`: Es el número de árboles que va a tener el bosque.
- `max_features`: Es la cantidad de características a tener en cuenta al buscar la mejor división.
- `max_samples`: Es la cantidad de muestras a extraer del conjunto de datos de entrenamiento para entrenar cada modelo base.

Para la búsqueda de los óptimos, se priorizó la métrica *f1-score* para tener un mejor equilibrio entre el *recall* y la *precisión*. Una vez obtenidos los hiperparámetros después de probar con distintos rangos de valores, se entrenó el *Random Forest*.

Ya entrenado el modelo, nos pusimos a ver cuáles fueron las *variables* (*features*) más importantes y obtuvimos que las tres más importantes fueron las siguientes:

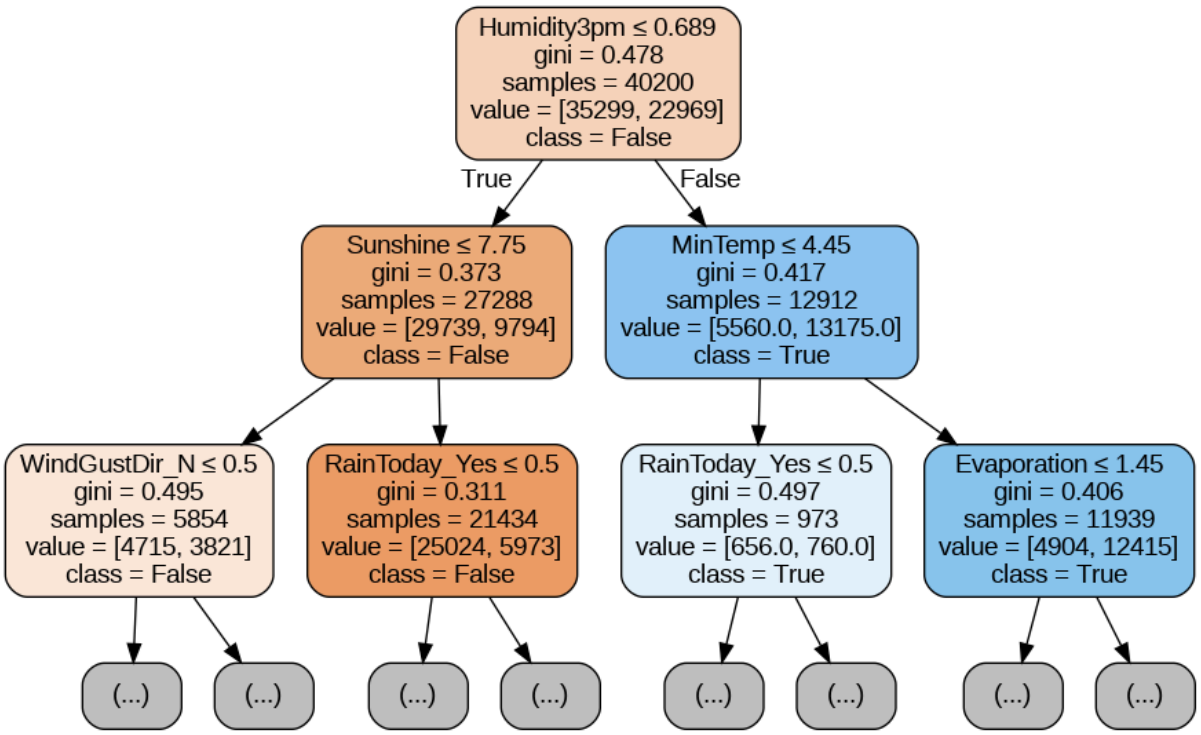
- `Humidity3pm`: $0,199$
- `Sunshine`: $0,078$
- `Cloud3pm`: $0,074$

Como se puede observar, la variable `Humidity3pm` es, por amplio margen, la más influyente en la predicción de si lloverá o no. Esto indica que la *humedad a las 3 pm* es un fuerte indicador del comportamiento del clima, ya que valores altos pueden estar asociados con una mayor probabilidad de lluvia.

En segundo y tercer lugar nos encontramos con las variables `Sunshine` y `Cloud3pm` respectivamente, lo que nos dice que la *cantidad de horas de sol* y la *nubosidad a las 3 p.m.* también aportan información relevante al modelo, aunque con un menor peso que la *humedad a las 3 pm*.

En conjunto, estas variables relacionadas con *humedad*, *luz solar* y *nubosidad* reflejan condiciones atmosféricas coherentes con los factores que típicamente influyen en la ocurrencia de lluvias.

A continuación se mostrará el gráfico de los principales niveles de uno de los árboles del *Random Forest* y no el modelo completo, ya que este se muy extenso:



Podemos ver que la variable que mayor ganancia de información nos da es `Humidity3pm`.

- La *regla principal* es `Humidity3pm` $\leq 0,689$ (si la presión a las 3 pm es menor o igual que $0,689$ sigue por el lado izquierdo, sino por el derecho).
- Por la raíz pasaron 40.200 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).

<u>Por el lado izquierdo de la raíz:</u>

Podemos ver que el siguiente *nodo interno* es sobre la variable `Sunshine`.

- La regla es `Sunshine` $\leq 7,75$.
- Por este nodo pasaron 27.288 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).

Después de este nodo, siguen los nodos internos sobre las variables:

- `WindSpeed3pm`:
  - La regla es `WindGustDir_N` $\leq 0,5$.
  - Por este nodo pasaron 5.854 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).
- `RainToday_Yes`:
  - La regla es `WindSpeed3pm` $\leq 0,5$.
  - Por este nodo pasaron 21.434 muestras de las cuales mayoritariamente son $False$ (no lloverá al día siguiente).

<u>Por el lado derecho de la raíz:</u>

Podemos ver que el siguiente *nodo interno* es otra vez sobre la variable `MinTemp`.

- La regla es `MinTemp` $\leq 0,417$.

- Por este nodo pasaron 12.912 muestras de las cuales mayoritariamente son $True$ (lloverá al día siguiente).

Después de este nodo, siguen los nodos internos sobre las variables:

- `RainToday_Yes`:
  - La regla es `RainToday_Yes` $\leq 0,5$.
  - Por este nodo pasaron 973 muestras de las cuales mayoritariamente son $True$ (lloverá al día siguiente).
- `Evaporation`:
  - La regla es `Evaporation` $\leq 1,45$.
  - Por este nodo pasaron 11.939 muestras de las cuales mayoritariamente son $True$ (lloverá al día siguiente).

Viendo el *árbol de decisión* podemos concluir que la variable más influyente para predecir si *va a llover al día siguiente* en este modelo base es `Humidity3pm` (justamente la variable más importante), que aparece como la raíz. Esto indica que la *humedad a las* 3 *pm* es el factor con mayor poder de separación entre los días con y sin lluvia al día siguiente.

En los niveles inferiores, las divisiones se hacen principalmente por `Sunshine`, `Evaporation`, `MinTemp` y *variables dummies* como `WindGustDir_N` o `RainToday_Yes`, lo que nos puede indicar que para este modelo, la luz del sol, la evaporación, la mínima temperatura, sí el viento más fuerte viene del norte y si llovió ese día son *features* que tienen una relación significativa con la probabilidad de lluvia.

Nota: Los valores de la condición que involucra a la variable `Humidity3pm` se encuentran en *escala estandarizada* debido a la *normalización por z-score*, por lo que representan desviaciones respecto al promedio en lugar de valores físicos directos (como porcentaje de evaporación). Esto **no** altera las decisiones del modelo, pero hace que los puntos de corte sean menos interpretables en unidades reales.

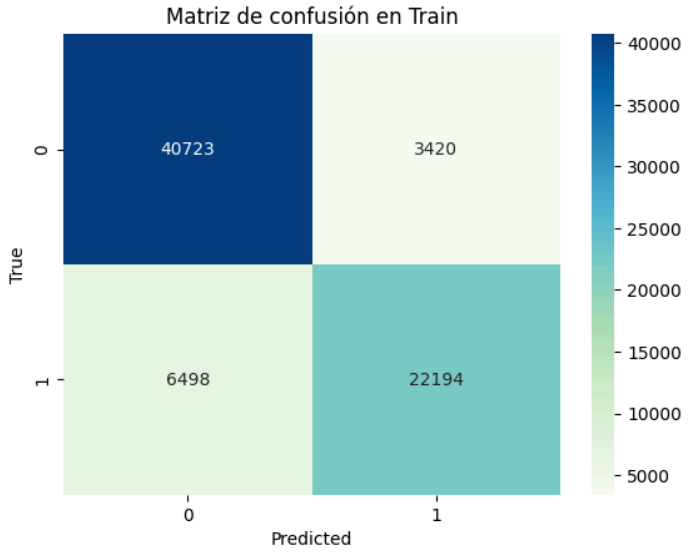
Al evaluar la performance del modelo con los *datos de entrenamiento* (*train*) se obtuvo los siguientes resultados para las métricas:

- $Accuracy$: $0,8638$
- $Recall$: $0,7735$
- $Precision$: $0,8665$
- $f1 - score$: $0,8174$

Después, se evaluó la performance del modelo con los *datos de prueba* (*test*) con los que se obtuvo los siguientes resultados para las métricas:

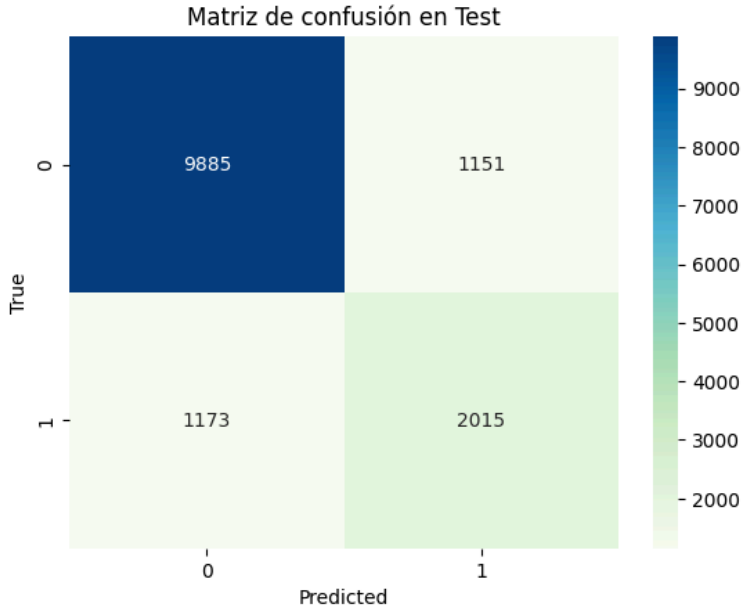
- $Accuracy$: $0,8366$
- $Recall$: $0,6321$
- $Precision$: $0,6364$
- $f1 - score$: $0,6342$

Matriz de Confusión en entrenamiento:



- $True\ Positive\ (TP)$: 40723
- $True\ Negative\ (TN)$: 22194
- $False\ Positive\ (FP)$: 3420
- $False\ Negative\ (FN)$: 6498

Matriz de Confusión en test:



- $True\ Positive\ (TP)$: 2015
- $True\ Negative\ (TN)$: 9885
- $False\ Positive\ (FP)$: 1151
- $False\ Negative\ (FN)$: 1173

Podemos observar que el modelo tiene una buena capacidad para predecir los días que **no** va a llover el día siguiente (clase mayoritaria) y una mejor capacidad para los días que **si** va

a llover el día siguiente (clase minoritaria). Entonces, se puede decir que este modelo logra una mejor generalización.

Modelo a elección:

Elección del modelo: El modelo que elegimos para esta sección es *Extreme Gradient Boosting* (*XGBoost*) principalmente debido a que en nuestro dataset final quedaron columnas con *valores faltantes* (como `Evaporation` y `Sunshine`) a las que se las decidió dejar sin tratamiento, ya que hacer alguna imputación en estas significaba un sesgo muy grande porque la cantidad de datos nulos en estos *features* es muy elevada. Por esto se decidió decantarnos por este modelo que soporta *valores faltantes* al igual que *Árboles de Decisión* y *Random Forest*. Otra de las razones es que este modelo suele tener una buena regularización para prevenir el *overfitting*.

Para evitar el *overfitting* lo que se hizo es usar dos hiper-parámetros llamados `reg_alpha` y `reg_lambda` los cuales se utilizan para evitar que los árboles se adapten al ruido del data set. El `reg_alpha` fuerza al modelo a usar menos splits importantes y el `reg_lambda` hace que los valores de los nodos hoja no crezcan demasiado.

Para evitar el desbalance de clases en este modelo, se utilizó el hiper-parámetro `scale_pos_weight` el cúal se encarga de darle una mayor importancia a la clase minoritaria por sobre la mayoritaria.

Antes de utilizar el algoritmo de *XGBoost* se optimizó los hiperparámetros con la técnica *de 5-fold Cross Validation* utilizando la exploración de hiperparámetros por Random Search debido a su rapidez. Los hiperparámetros elegidos fueron:

- `n_estimators`: Es el número de etapas de refuerzo de gradiente a realizar.
- `max_depth`: Es la profundidad máxima de los estimadores de regresión individuales.
- `learning_rate`: Reduce la contribución de cada árbol.
- `subsample`: Es la fracción de muestras que se utilizará para ajustar los aprendices base individuales.
- `colsample_bytree`: Controla qué porcentaje de las columnas se seleccionan aleatoriamente para construir cada árbol.
- `reg_alpha`: Penaliza los coeficientes grandes forzando algunos a ser exactamente cero.
- `reg_lambda`: Penaliza los coeficientes grandes, pero no los lleva a cero, solo los achica.
- `scale_pos_weight`:Le indica al modelo cuánto más importante es la clase minoritaria respecto de la clase mayoritaria.

Para la búsqueda de los óptimos, se priorizó la métrica *f1-score* para tener un mejor equilibrio entre el *recall* y la *precisión*. Una vez obtenidos los hiperparámetros después de probar con distintos rangos de valores, se entrenó el *XGBoost*.

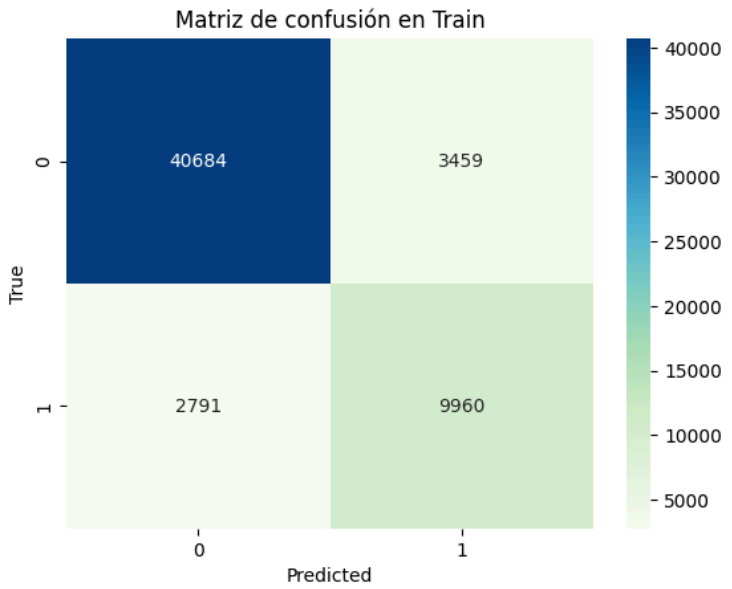
Al evaluar la performance del modelo con los *datos de entrenamiento* (*train*) se obtuvo los siguientes resultados para las métricas:

- $Accuracy: 0,8901$
- $Recall: 0,7811$
- $Precision: 0,7422$
- $f1 - score: 0,76111$

Después, se evaluó la performance del modelo con los *datos de prueba* (*test*) con los que se obtuvo los siguientes resultados para las métricas:

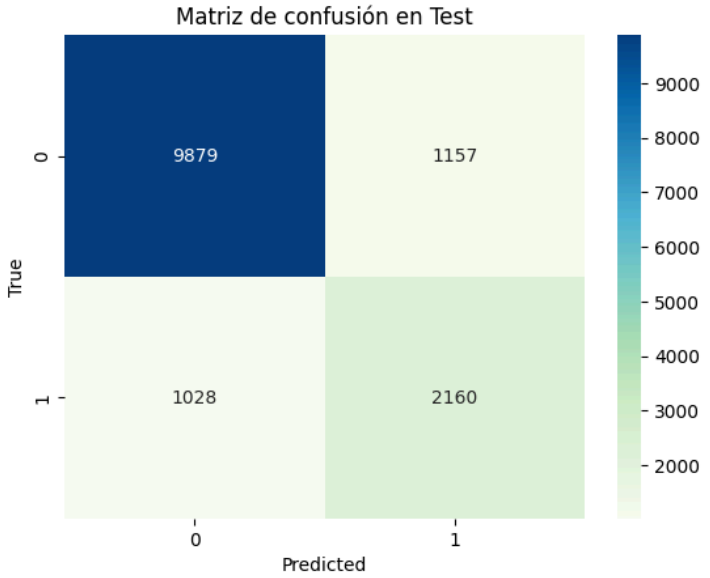
- $Accuracy : 0,8464$
- $Recall: 0,6775$
- $Precision: 0,6512$
- $f1 - score: 0,6641$

Matriz de Confusión en entrenamiento:



- $True\ Positive\ (TP): 40684$
- $True\ Negative\ (TN): 9960$
- $False\ Positive\ (FP): 3459$
- $False\ Negative\ (FN): 2791$

Matriz de Confusión en test:



- $True\ Positive\ (TP)$: 2160
- $True\ Negative\ (TN)$: 9879
- $False\ Positive\ (FP)$: 1157
- $False\ Negative\ (FN)$: 1028

Podemos observar que el modelo tiene una buena capacidad para predecir los días que **no** va a llover el día siguiente (clase mayoritaria) y una mejor capacidad (aunque leve) para los días que **si** va a llover el día siguiente (clase minoritaria). Entonces, se puede decir que este modelo logra una mejor generalización pero no tan grande.

Comparación de resultados:

Modelo	F1-Score	Precisión	Recall	Accuracy
Árbol de Decisión	$0,5998$	$0,5873$	$0,6129$	$0,8167$
Random Forest	$0,6342$	$0,6364$	$0,6321$	$0,8366$
XGBoost	$0,6641$	$0,6512$	$0,6775$	$0,8464$

El *Árbol de Decisión* consiste en: `{'random_state': 10,'min_samples_split': 57, 'min_samples_leaf': 50, 'max_depth': 16, 'criterion': 'gini'}`

El *Random Forest* consiste en: `{'random_state': 10, 'bootstrap': True, 'oob_score': True, 'n_estimators': 100, 'min_samples_split': 29, 'min_samples_leaf': 10, 'max_samples': 0.8, 'max_features': 'log2', 'max_depth': 28, 'criterion': 'gini'}`

El *XGBoost* consiste en: `{'random_state': 10, 'use_label_encoder': False, 'subsample': 0.7, 'scale_pos_weight': 2, 'reg_lambda': 8, 'reg_alpha': 0.5, 'n_estimators': 450, 'max_depth': 5, 'learning_rate': 0.1, 'colsample_bytree': 0.8}`

¿Cómo resultó el performance de cada modelo con respecto al set de entrenamiento?

Performance del *Árbol de Decisión*:

Árbol de Decisión			
F1-Score	**Precisión**	**Recall**	**Accuracy**
$0,7269$	$0,7707$	$0,6879$	$0,8277$

- Tiene el rendimiento más bajo de los tres, lo que es esperable porque es el modelo más simple.
- Aún así, tiene un balance entre precisión y recall razonable por lo que podemos decir que logra captar los patrones generales.
- Tiene un *recall* muy bajo en comparación con los demás.
- Es el que más rendimiento pierde con el conjunto de prueba.

Performance de *Random Forest*:

Random Forest			
F1-Score	**Precisión**	**Recall**	**Accuracy**
$0,8174$	$0,8665$	$0,7735$	$0,8638$

- Hay una mejora en todas las métricas respecto al *Árbol de Decisión*.
- Esto nos dice que hay una mejora en la capacidad de generalización.
- Pierde menos rendimiento con el conjunto de prueba a comparación de Árbol de decisión por lo que vemos que se redujo el overfitting.

Performance de *XGBoost*:

XGBoost			
F1-Score	**Precisión**	**Recall**	**Accuracy**
$0,8296$	$0,8658$	$0,7781$	$0,8651$

- Obtiene los mejores resultados globales, aunque la diferencia con *Random Forest* no es muy grande.
- Tiene el *f1-score* más alto, lo que muestra el mejor equilibrio entre *precisión* y *recall*.
- No mejora casi nada el *recall* respecto a *Random Forest*.
- Es el que tiene el rendimiento más alto con el conjunto de prueba, pero no tan distanciado de *Random Forest*.

Elección del Modelo:

En base a los resultados obtenidos y observando las matrices de confusión de cada modelo, se decidió que el mejor modelo para predecir si lloverá o no al día siguiente es al modelo de XGBoost ya que es el que es el que tiene las mejores métricas de las que se midieron (*f1-score*, *precisión*, *recall* y *accuracy*). Además, un punto muy fuerte es que tenga el *recall* más alto debido a que esto significa que no se le escapan tantos días de lluvia como a los demás modelos que es, para nosotros, lo más importante porque es más preferible llevar un paraguas y no usarlo que no llevarlo y por eso mojarse.

Ejercicio 3 - Modelos de Regresión:

Descripción del Dataset:

En este ejercicio se trabajó sobre un subconjunto de datos de alojamientos de la plataforma AirBnB correspondiente a la ciudad de Los Angeles. El dataset sobre el cual se llevará a cabo el análisis contiene 45421 filas y 79 columnas.

El objetivo es predecir el precio de alquiler en función de la información publicada en los avisos haciendo uso de los modelos de regresión.

Preprocesamiento:

Eliminación de columnas:

En principio, dada la gran cantidad de variables involucradas en el dataset, se realizó un análisis para determinar aquellas que nos servirían para cumplir el objetivo de nuestro ejercicio. Para ello, en base a las definiciones de las variables proporcionadas por la página de AirBnB a través de un data dictionary, descartamos aquellas variables que, a nuestro parecer, no tendrían gran influencia sobre el precio de los alquileres. Esto nos llevó a descartar variables como el ID del alojamiento, el nombre, descripción, información del host, disponibilidad, entre otras. Finalmente, nos quedamos con las siguientes:

- `neighbourhood_cleansed` (*categórica*): Vecindario donde está ubicado el alojamiento. Éste se geocodifica usando la latitud y longitud, comparándolas con los vecindarios definidos por shapefiles digitales abiertos o públicos.
  ➔ **Tipo de dato:** $object$
- `property_type` (*categórica*): Tipo de propiedad autoelegido. Los hoteles y los *bed and breakfast* son descritos como tales por sus anfitriones en este campo.
  ➔ **Tipo de dato:** $object$
- `room_type` (*categórica*): Puede tomar los siguiente valores:
  - $Entire\ home/apt$ (Casa completa)
  - $Private\ room$ (Habitaciones privadas)
  - $Shared\ room$ (Habitaciones compartidas)
  - $Hotel$
  ➔ **Tipo de dato:** $object$
- `accommodates` (*cuantitativa discreta*): La máxima capacidad del hospedaje.
  ➔ **Tipo de dato:** $int64$
- `bathrooms` (*cuantitativa discreta*): La cantidad de baños del hospedaje. En general toma valores como 0.5, 1, 1.5, etc. representando un entero como un baño completo que incluye inodoro, lavabo y ducha o bañera, y medio como un baño de servicios que incluye solo inodoro y lavabo, sin ducha ni bañera.
  ➔ **Tipo de dato:** $float64$
- `bathrooms_text` (*categórica*): La cantidad de baños del hospedaje. En el sitio web de Airbnb, el campo de baños ha evolucionado de un número a una descripción textual.

  En base a esta variable podemos obtener la cantidad de baños y si son de tipo *shared* o *private*.

➔ **Tipo de dato:** $object$

- `bedrooms` (*cuantitativa discreta*): La cantidad de dormitorios del hospedaje.
  ➔ **Tipo de dato:** $float64$
- `beds` (*cuantitativa discreta*): La cantidad de camas del hospedaje.
  ➔ **Tipo de dato:** $float64$
- `amenities` (*categórica*)
  ➔ **Tipo de dato:** $object$
- `price` (*cuantitativa continua*): Precio diario en la moneda local.
  ➔ **Tipo de dato:** $object$
- `minimum_nights` (*cuantitativa discreta*): Número mínimo de noches de estadía para el anuncio (las reglas del calendario pueden ser diferentes).
  ➔ **Tipo de dato:** $int64$
- `maximum_nights` (*cuantitativa discreta*): Número máximo de noches de estadía para el anuncio (las reglas del calendario pueden ser diferentes).
  ➔ **Tipo de dato:** $int64$
- `review_scores_rating` (*cuantitativa continua*): Puntuación general o promedio que los huéspedes le dan al alojamiento.
  ➔ **Tipo de dato:** $float64$
- `review_scores_value` (*cuantitativa continua*): Puntuación que los huéspedes le dan a la relación calidad-precio.
  ➔ **Tipo de dato:** $float64$

Consideramos que las variables seleccionadas son relevantes para explicar el precio del alojamiento, en base a las siguientes **suposiciones**:

- Ubicación: es esperable que existan barrios o zonas de Los Ángeles donde los alojamientos sean más caros o más económicos, debido a factores como el atractivo turístico, la demanda, la seguridad o la accesibilidad.
- Capacidad y tipo de propiedad: alojamientos con mayor capacidad (más huéspedes, dormitorios, camas o baños), con más *amenities*, o que ofrecen propiedades completas en lugar de habitaciones, podrían tener precios más altos debido a los mayores costos.
- Duración mínima y máxima de estadía: alojamientos con estadías mínimas o máximas más largas suelen orientarse a huéspedes de largo plazo y no al turismo, por lo que el precio por noche podría ser menor. En cambio, alojamientos que permiten estadías cortas podrían cobrar precios más altos por noche.
- Reseñas: una mejor valoración por parte de los huéspedes puede reflejar mejores características del alojamiento (comodidad, ubicación, limpieza, atención, etc.), lo que podría asociarse a precios más altos.

Transformación de variables:

Transformamos algunas variables para mejorar el análisis del dataset y la capacidad de generalización de los modelos.

- `neighbourhood_cleansed`: como habían 266 barrios distintos, agrupamos los barrios en regiones que definimos a partir del siguiente mapa:

Mapa de regiones de Los Angeles, tomado de este enlace.

Así, definimos una nueva variable `area` que puede tener 9 valores distintos, de acuerdo a la región donde se encontraba el respectivo barrio. Estos valores son: *Santa Clarita Valley*, *Antelope Valley*, *San Fernando Valley*, *San Gabriel Valley*, *Westside Cities*, *Central LA*, *Gateway Cities*, *South Bay*, y *Santa Catalina Island*.

- `property_type`: como esta variable también tomaba una gran cantidad de valores distintos (94), decidimos volver a agrupar los valores en una nueva variable `property_group` en cinco grupos principales según el grado de privacidad, el tipo de unidad ofrecida y su orientación de uso:
  - *Entire place*: propiedades completas de uso exclusivo del huésped (casas, departamentos, etc.).
  - *Private room*: habitaciones privadas dentro de una vivienda o establecimiento compartido.
  - *Shared room*: habitaciones compartidas con otros huéspedes.
  - *Serviced apartment*: unidades de tipo apartamento con servicios hoteleros o de *aparthotel*.
  - *Specialty*: alojamientos no convencionales o de nicho (barcos, casas en árboles, granjas, etc.).
- `bathrooms_text`: como la cantidad de baños se encuentra en la variable `bathrooms`, entonces definimos que la variable `bathrooms_text` describa solo el tipo de baño. Definimos así la variable `bathrooms_type`.
  Consideraciones:

    - Los baños de servicio fueron descritos con el decimal 0.5 en la variable `bathrooms`, por ello no los asignamos como un tipo en sí.
    - Aquellos baños que no fueron definidos como *shared* ni *private* fueron tomados como *private*.
- `amenities`: teniendo en cuenta la gran cantidad de *amenities* distintos (10546), no nos hubiese convenido formar una variable categórica con cada uno de sus valores. Además, un análisis de los valores para agruparlos (como hicimos anteriormente) llevaría también mucho tiempo y trabajo. Por esto decidimos crear una nueva variable `amenities_num` que represente la cantidad de *amenities* que tiene cada alojamiento.
- `price`: aplicamos el formato debido a esta variable (eliminando el signo de pesos y eliminando comas innecesarias) para transformar su tipo de dato de $object$ a $float$.

Tras la transformación de las variables, nos quedaron las siguientes en el dataset: `room_type`, `accommodates`, `bathrooms`, `bedrooms`, `beds`, `price`, `minimum_nights`, `maximum_nights`, `review_scores_rating`, `review_scores_value`, `area`, `property_group`, `bathrooms_type` y `amenities_num`.

Tratamiento de valores atípicos:

Realizamos análisis univariados y multivariados para el tratamiento de valores atípicos.
Para el **análisis univariado**, graficamos *box plots* por cada variable cuantitativa y observamos la distribución resultante.

- Eliminamos los datos para los cuales había variables con valores sin sentido. Estos fueron aquellos para los que había una cantidad nula de baños o camas.
- Muchas variables tenían una desviación relevante (`accommodates`, `bathrooms`, `bedrooms`, `beds`, `price`, `minimum_nights`, y `maximum_nights`), tomando valores muy altos. A pesar de ser valores posibles, decidimos eliminarlos para no sesgar la distribución. También decidimos reexaminar esas variables en conjunto con otras variables en el análisis multivariado.
- No se detectaron outliers univariados para las variables `review_scores_rating` y `review_scores_value`.

Para el **análisis multivariado** graficamos la matriz de correlaciones y *scatter plots* entre todas las variables cuantitativas. Con esto, identificamos outliers considerando la distribución de las variables con alta correlación. Se definieron como valores atípicos aquellos que se encuentran alejados de la mayoría de las observaciones. Este análisis se realizó mediante el cálculo de las distancias de Mahalanobis.
Para definir el umbral de distancia a partir del cual considerar un valor como atípico, probamos distintos percentiles (90%, 95% y 99%), graficando *scatter plots* con los outliers señalados. Finalmente, seleccionamos el percentil que coincidía con los outliers visualmente identificados.
Tras eliminar los outliers y volver a graficar, observamos que la dispersión de los datos se redujo, mostrando una mayor concentración de los puntos en torno a la nube central de observaciones.

En total se eliminó un 5.45% del dataset, lo que consideramos un valor aceptable para no afectar la representatividad de los datos.

Tratamiento de valores faltantes:

Para el tratamiento de los datos faltantes, lo primero que se hizo es ver cuáles variables tenían datos faltantes y qué porcentaje representan respecto de la cantidad total de observaciones en el dataframe. Estas fueron:

- `bathrooms`: 19.95%
- `bedrooms`: 6.97%
- `beds`: 20.05%
- `price`: 20.36%
- `review_scores_rating`: 26.95%
- `review_scores_value`: 26.98%

A continuación se explicará el tratamiento individual que se aplicó sobre cada variable:

- `bathrooms`: La ausencia de datos en la variable `bathrooms` es de tipo *Missing At Random* (MAR) ya que los datos faltantes pueden ser explicados por valores en las otras columnas (como por ejemplo la cantidad de dormitorios o la cantidad máxima de huéspedes), pero no por valores de esa columna. Es por ello que decidimos imputar los datos faltantes con la estrategia de MICE (*Multivariate Imputation by Chained Equations*).
  A pesar de que la variable `bathrooms` es cuantitativa, como explicamos anteriormente, en la práctica toma valores como 0.5, 1, 1.5, etc. Es por ello que, luego de imputar los datos con MICE, truncamos los valores a su entero más cercano o a su $.5$ más cercano.
- `bedrooms` y `beds`: consideramos la ausencia de datos en las variables `bedrooms` y `beds` también como *Missing At Random*, y nuevamente volvemos a imputar los datos faltantes con la estrategia de MICE.
  Como ambas variables son cuantitativas discretas, truncamos el valor imputado al entero más cercano.
- `price`: para el caso de la variable `price` decidimos no imputar los datos faltantes, sino eliminar aquellas filas que los contengan. Esto ya que nuestro objetivo es entrenar un modelo para predecir el valor de esta variable. Imputar los datos faltantes introduciría ruido y sesgos difíciles de controlar, el modelo aprendería a "perseguir" valores no observados.
- `review_scores_rating` y `review_scores_value`: consideramos las variables `review_scores_rating` y `review_scores_value` como *Missing Completely At Random* (MCAR), pues no podemos predecir sus valores a partir de otros datos del dataframe. Por esto, decidimos imputar los valores faltantes con la media.

Modelos entrenados:

En principio, las variables categóricas se codificaron con One Hot Encoding para convertirlas en variables numéricas y permitir que los modelos las procesen sin asignar relaciones ordinales inexistentes entre las categorías.

1. **Regresión lineal múltiple**

Transformación de los datos:

- Como la variable `price` tenía muchos valores bajos y unos pocos altos (pero que son outliers posibles), entonces presentaba una gran dispersión que provocaba que la regresión lineal dé más peso a las propiedades más caras, dificultando que el modelo capture correctamente la relación entre las variables predictoras y los precios promedio. Esto ocurre porque el modelo ajusta sus coeficientes minimizando el MSE (*Mean Squared Error*), que eleva al cuadrado las diferencias entre las predicciones y los valores reales. Esto hacía que el modelo "trate de acertar más" sobre las propiedades caras porque sus errores dominan la función de pérdida, aunque sean pocas. Para mitigar este problema, se aplicó la transformación logarítmica del precio, que comprime las diferencias relativas y reduce la asimetría de la variable. Así pasamos a tener una variable `log_price`, que tomamos como variable dependiente.
- Las variables cuantitativas se normalizaron con Z-score para hacer comparables los coeficientes.

Resultados del entrenamiento:

$MSE = 23761.91$

$RMSE = 154.15$

$MAE = 78.27$

$R^2 = 0.53$

Como $R^2 = 0.53$, el modelo explica $\sim 53\%$ de la variabilidad del precio: es útil como estimación orientativa, pero no es lo bastante preciso.

$RMSE >> MAE$ probablemente se deba a la gran dispersión de los datos de la variable del precio.

Para identificar las variables más influyentes comparamos los coeficientes y fijamos como umbral de 50% para el efecto en porcentaje sobre el precio (`cambio_%`). Toda variable que excedió ese valor la consideramos influyente.

Bajo este criterio, destacaron las variables: `area_Santa Catalina Island`, `area_Westside Cities`, `area_South Bay`, `area_Central LA`, `area_San Fernando Valley`, `area_Santa Clarita Valley`, junto con `room_type_Hotel room` y `property_group_Private room`. Observamos que la mayoría proviene de la codificación de área, lo que sugiere que el precio está fuertemente asociado a la localización, algo razonable, pues los turistas priorizan seguridad, comodidad, cercanía a atractivos y accesibilidad. Además, variables ligadas a la configuración del alojamiento (tipo de habitación/propiedad) también muestran peso, indicando que las características del espacio, además del lugar, contribuyen de forma relevante a la formación del precio.

2. **XGBoost**

Elección de hiperparámetros:

Primero seleccionamos la cantidad de árboles a entrenar con validación cruzada, utilizando *early stopping* para detener el entrenamiento cuando el error de validación deja de mejorar. El modelo dejó de mejorar a partir de 108 árboles, pero como el error promedio era similar para aproximadamente 100 árboles, se definió ese valor como óptimo para evitar overfitting.

Luego aplicamos validación cruzada probando con distinta cantidad de *folds* (de 2 a 10) para encontrar el óptimo. Como el menor error resultó para 7 folds, elegimos ese valor para entrenar el modelo.

Utilizamos `GridSearchCV` para buscar hiperparámetros con validación cruzada, utilizando la cantidad de árboles y folds hallados previamente.

Los hiperparámtros quedaron definidos de la siguiente manera:

```
{'colsample_bytree': 1.0,
 'learning_rate': 0.2,
 'max_depth': 7,
 'min_child_weight': 3,
 'subsample': 0.7}
```

Estos valores tienen sentido: `max_depth=7` limita la complejidad de cada árbol evitando overfitting, `learning_rate=0.2` balancea velocidad y precisión, y `colsample_bytree=1.0` indica que se utilizan todas las variables en cada árbol.

Resultados del entrenamiento:

$MSE = 19175.48$

$RMSE = 138.48$

$MAE = 75.26$

$R^2 = 0.62$

Estos resultados suponen una mejora frente a la regresión lineal múltiple (mejora del 9%). El MAE indica que, en promedio, el error absoluto en la predicción de precios es de aproximadamente \$75, y el R^2 indica que el modelo explica $\sim 62\%$ de la variabilidad del precio. Esto significa que XGBoost logró captar relaciones no lineales e interacciones entre variables que la regresión lineal no modela explícitamente, pero de todas formas aún hay variabilidad del precio que el modelo no captura, por lo que este modelo no es totalmente preciso: permite hacer estimaciones orientativas.

3. **Random Forest Regressor (cross-validation)**

Para el tercer modelo optamos por el modelo Random Forest Regressor ya que tiene un enfoque distinto a XGBoost. XGBoost usa la estrategia de *boosting*: entrena los árboles secuencialmente, corrigiendo errores de los anteriores. Esto suele dar alta precisión pero requiere ajuste cuidadoso de hiperparámetros, y es sensible al *overfitting*. Random Forest usa la estrategia de *bagging*: entrena muchos árboles en paralelo sobre muestras aleatorias y promedia sus predicciones. Esto lo hace más robusto y estable, reduciendo la varianza y teniendo así menos riesgo de *overfitting*.

Random Forest no solo presenta buena tolerancia a outliers frente a enfoques lineales, sino que además aprende no-linealidades, es decir, puede aproximar relaciones que no se explican como una simple suma ponderada de las variables. Un ejemplo típico es la cantidad de baños: el precio no aumenta de forma continua por cada baño extra, sino que muestra saltos (umbrales); pasar de 1→2 baños suele producir un incremento mucho mayor que de 2→3. También capta interacciones (p. ej., el efecto de un baño adicional puede ser distinto según el barrio o el tipo de propiedad).

Elección de hiperparámetros:

Primero seleccionamos la cantidad de árboles a entrenar con validación cruzada para estimar el desempeño de forma robusta y evitar overfitting. Entrenamos el modelo para 50, 100 y 200 y calculamos el RMSE. Como los errores RMSE para las distintas cantidades de árboles dieron valores muy cercanos, y además como éste decrece a medida que aumentamos la cantidad de árboles (el error entre 50 y 100 es mayor que entre 100 y 200), optamos por usar el valor de 100 árboles para entrenar el modelo, así realizamos un balance entre tiempo de ejecución y precisión.

Luego aplicamos validación cruzada probando con distinta cantidad de *folds* (de 2 a 10) para encontrar el óptimo. En este caso, para acelerar el tiempo de prueba usamos una cantidad de árboles menor a la establecida (20 árboles). Como obtuvimos valores cercanos de error para los folds 5 a 10, elegimos el valor de 5 folds para entrenar el modelo.

Utilizamos `GridSearchCV` para buscar hiperparámetros con validación cruzada, utilizando 20 árboles nuevamente y la cantidad de folds hallada previamente.

Los hiperparámtros quedaron definidos de la siguiente manera:

```
{'max_depth': None,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 10}
```

Estos valores tienen sentido: `max_depth=None` permite a cada árbol expandirse completamente y capturar relaciones complejas, `max_features='sqrt'` reduce la correlación entre árboles y mejora la generalización, `min_samples_leaf=1` permite hojas muy detalladas, y `min_samples_split=10` evita que los nodos se dividan con pocos datos, reduciendo el riesgo de overfitting.

Resultados del entrenamiento:

$MSE = 18517.15$

$RMSE = 136.08$

$MAE = 73.71$

$R^2 = 0.64$

Las métricas volvieron a mejorar para el caso de Random Forest. El MAE indica que, en promedio, el error absoluto en la predicción de precios es de aproximadamente

$74, y el R^2 de 0.64 significa que el modelo explica $\sim 64\%$ de la variabilidad del precio. De todas formas, el modelo aún no alcanza una precisión exacta.

Comparación de resultados:

Modelo	MSE	$RMSE$	MAE	R^2
Regresión lineal	23761.91	154.15	78.27	0.53
XGBoost	19175.48	138.48	75.26	0.62
Random Forest	18517.15	136.08	73.71	0.64

En nuestra comparación, la performance mejora de la regresión lineal a XGBoost y, finalmente, a Random Forest. La regresión lineal presenta alto sesgo y baja varianza, por lo que suele mostrar un brecha entre el rendimiento en entrenamiento y en validación/prueba pequeño entre entrenamiento y validación, pero "se queda corta" cuando la relación entre predictores y precio no es estrictamente aditiva/lineal. Eso se ve en sus métricas: $R^2 = 0.53$, $RMSE = 448.9$ y $MAE = 127.2$, que son las peores del conjunto. En cambio, XGBoost incrementa la capacidad del modelo al ir corrigiendo errores secuencialmente con regularización; típicamente logra menor error en entrenamiento que la lineal y una brecha moderada en validación cuando el ajuste de hiperparamétros está bien. En nuestro caso sube a $R^2 = 0.62$ y reduce $RMSE$ y MAE (138.48 y 75.26), lo que es consistente con su habilidad para capturar umbrales e interacciones. Finalmente, Random Forest (*bagging* de árboles) promedia muchos árboles entrenados en muestras *bootstrap*; esto reduce la varianza y lo vuelve robusto al ruido y a outliers, manteniendo buena capacidad para modelar no-linearidades. Por eso obtiene el mejor equilibrio sesgo–varianza y el mejor desempeño global: $R^2 = 0.64$, $RMSE = 136.08$ y $MAE = 73.71$, con un error de entrenamiento bajo y una brecha controlada en validación gracias al promediado.

Elección del Modelo:

Con los resultados obtenidos, elegimos Random Forest como modelo principal para predecir precio: obtiene el mejor desempeño fuera de muestra frente a XGBoost y la lineal. Además, es robusto ante no-linearidades, outliers moderados y variables con escalas distintas; maneja bien interacciones y umbrales sin requerir ingeniería de features compleja y suele tener un gap train-validación acotado gracias al *bagging* (menos varianza).

Ejercicio 4 - Clustering:

Descripción del dataset:

En este ejercicio se trabajó sobre un subconjunto de datos de tracks en la plataforma Spotify. El dataset sobre el cual se llevará a cabo el análisis contiene 750 filas y 13 columnas.

El objetivo es analizar la tendencia de clustering del dataset, estimar la cantidad apropiada de grupos que se deben formar, evaluar la calidad de los grupos formados realizando un análisis de Silhouette y realizar un análisis de cada grupo intentando entender en función de qué características fueron formados.

Preprocesamiento:

En principio, observamos que no existe una cantidad alevosa de variables para analizar, además que las mismas pueden tener relevancia para el cluster, no así otras variables que figuran en un data dictionary provisto por la cátedra pero que dentro del dataset provisto por la misma no estan, un ejemplo claro es el ID.
Estas son las variables a utilizar para nuestro análisis:

- `acousticness` (cuantitativa continua): medida de confianza de que la pista es acústica (1.0 = alta confianza).
  - ➔ **Tipo de dato:** $float64$
- `danceability` (cuantitativa continua): cuán bailable es la pista (tempo, estabilidad rítmica, fuerza del beat, regularidad).
  - ➔ **Tipo de dato:** $float64$
- `energy` (cuantitativa continua): intensidad/actividad percibida (sonoridad, timbre, onsets, entropía).
  - ➔ **Tipo de dato:** $float64$
- `duration_ms` (cuantitativa discreta): duración de la pista en milisegundos.
  - ➔ **Tipo de dato:** $int64$
- `instrumentalness` (cuantitativa continua): probabilidad de ausencia de voz; >≈0.5 suele indicar instrumental.
  - ➔ **Tipo de dato:** $float64$
- `key` — categórica nominal: tono (Pitch Class); −1 = sin detección.
  - ➔ **Tipo de dato:** $int64$
- `liveness` (cuantitativa continua): probabilidad de performance en vivo; >0.8 sugiere que es en vivo.
  - ➔ **Tipo de dato:** $float64$
- `loudness` (cuantitativa continua): sonoridad promedio en decibelios.
  - ➔ **Tipo de dato:** $float64$
- `mode` (categórica nominal binaria): modalidad (1 = mayor, 0 = menor).
  - ➔ **Tipo de dato:**
- `speechiness` (cuantitativa continua): presencia de habla; >≈0.66 hablada, 0.33–0.66 mezcla, <0.33 mayormente musical.
  - ➔ **Tipo de dato:** $int64$

- `tempo` (cuantitativa continua): tempo estimado en pulsos por minuto.
  ➔ **Tipo de dato:** $float64$
- `time_signature` (categórica ordinal): compás estimado (p. ej., 4 ⇒ 4/4).
  ➔ **Tipo de dato:** $int64$
- `valence` (cuantitativa continua): “positividad” percibida (alto = feliz/eufórico; bajo = triste/sombrío).
  ➔ **Tipo de dato:** $float64$

Para el análisis de segmentación utilizamos el algoritmo K-Means. Este método define k centroides iniciales y calcula la distancia euclidiana de cada observación a dichos centroides para asignarla al más cercano. Luego actualiza cada centroide como la media de las observaciones que contiene y repite el proceso hasta la convergencia. Así, se minimiza la suma de distancias cuadráticas dentro de cada grupo, obteniendo clústeres compactos y bien separados.
Para el tratamiento de valores atípicos realizamos análisis univariado y multivariado. En una primera instancia, aplicamos *box plots* por variable; en todos los casos, los valores observados se ubicaron dentro de lo esperable según el criterio IQR. Complementariamente, efectuamos un análisis multivariado mediante una matriz de correlaciones (heatmap) y gráficos de pares (pairplot). En ambas visualizaciones no se detectaron patrones anómalos ni observaciones potencialmente atípicas.
Por último, no observamos datos faltantes por ende no requerimos un tratamiento para estos.

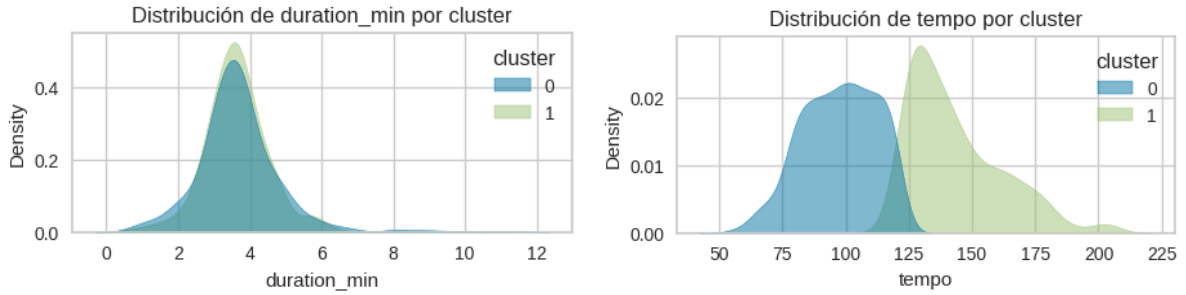
Transformación de los datos:
Para la preparación de datos convertimos `duration` de milisegundos a minutos para facilitar la interpretación humana de descripciones y perfiles por clúster; este cambio no altera las distancias si luego se estandariza, pero mejora la comunicación de resultados. Las variables categóricas, aunque estaban codificadas numéricamente, no tienen un orden intrínseco; por eso aplicamos one-hot encoding y evitamos introducir una jerarquía artificial que sesgue las distancias. Deliberadamente no estandarizamos las variables continuas para preservar su escala original e interpretabilidad, aceptando el *trade-off* de que aquellas con mayor rango/varianza (p. ej., `tempo` o `loudness`) tengan más peso en la distancia; por ello, interpretamos los índices de calidad y los perfiles de clúster considerando este sesgo potencial.

Modelo entrenado:

Como paso previo, evaluamos la tendencia a la agrupación con el estadístico de Hopkins: obtuvimos un valor de ~ 0.74, lo que sugiere una fuerte evidencia de estructura de clústeres (valores cercanos a 1 indican datos “clusterizables”, mientras que ~ 0.5 es aleatorio y ~ 0 se asemeja a una distribución uniforme). Este resultado respalda la viabilidad de aplicar algoritmos de clustering como K-Means y justifica explorar distintos valores de k y validar la estabilidad de las particiones resultantes.
Luego estimamos el número de particiones con el coeficiente Silhouette: el máximo se obtuvo en $k = 3$, pero $k = 2$ presentó un valor muy cercano. Con base en la inspección de perfiles, la consistencia interna y la interpretabilidad de los grupos, optamos por $k = 2$, ya

que ofrece una segmentación más clara y útil para el problema sin sacrificar calidad de partición.

Ya habiendo analizado nuestra cantidad de clústeres, continuamos analizando cada variable y qué relevancia tienen sobre cada cluster. Tras el análisis, tomamos el tempo como eje para perfilar y asignar clústeres, pues la distribución de los datos se veía definida principalmente por esta variable:



Antes de convertir la duración desde milisegundos, esta variable dominaba artificialmente la partición; tras expresarla en minutos, sus curvas por clúster quedan prácticamente superpuestas, lo que indica baja capacidad discriminante univariada (útil para describir, no para separar). En cambio, tempo sí aporta señal: el clúster 0 se concentra en rangos medios–bajos (~75–115 BPM) y el clúster 1 se desplaza a valores altos (~125–200 BPM), con solapamiento acotado en ~120–130 BPM; además, el clúster 1 exhibe cola derecha más larga (temas muy rápidos) y el 0 es más compacto. Por tanto, la separación real proviene de tempo y de combinaciones multivariadas, no de duración por sí sola; si otras variables replican el patrón de duración (unimodales y superpuestas), su rol será principalmente de perfilado. En este sentido, `loudness`, `valence`, `energy`, `danceability` y `acousticness` parecen aportar sobre todo a la caracterización de los grupos, mientras que la asignación de clúster descansa fundamentalmente en `tempo`.

Tiempo dedicado:

Integrante	Tarea	Prom. hs. semanales
Leandro Elias Brizuela	Ejercicios 1 y 2: Colab + reporte.	12hs
José Rafael Patty Morales	Ejercicios 1 y 2: Colab + reporte.	12hs
Jesabel Pugliese	Ejercicios 3 y 4: Colab + reporte.	8hs
Candela Matelica	Ejercicios 3 y 4: Colab + reporte.	8hs