Implementación código

El objetivo del presente documento es el de realizar una explicación del código implementado en la realización del proyecto. Dicho código, se encuentra implementado en Python y está disponible en su totalidad en el repositorio de github bajo el nombre de procesar fichero.jpynb. El código implementado para el proyecto se encuentra focalizado en la lectura de los dos ficheros .xlsx (bookings y hotels) y el tratamiento de su contenido para la aplicación de una serie de cambios requeridos para la posterior visualización de la información.

A continuación, se explicará tanto el código, así como las decisiones tomadas en cada paso. En primer lugar, detallamos la carga de las librerías necesarias para la implementación:

```
import pandas as pd
import random
import datetime
```

- pandas: se requiere para la lectura de los ficheros y la carga de los datos en data sets que nos facilitarán trabajar con los datos durante la implementación, así como en la posterior generación de los ficheros modificados.
- random: se requerirá dicha librería para la asignación de valores de manera aleatoria.
- datetime: se requerirá para la generación de campos de tipo fecha.

Posteriormente, se realiza la lectura de los ficheros originales con las reservas y el listado de hoteles con la ayuda de la librería pandas. Adicionalmente, se confirma que no existen valores null en los data sets obtenidos:

```
#Importamos los ficheros de reservas y hoteles:
bookings = pd.read_excel('bookings.xlsx')
hotels = pd.read_excel('hotels.xlsx')
#Se verifica la no existencia de valores null en los ficheros:
print('Fichero bookings.xlsx:')
print(bookings.isnull().sum(), '\n')
print('Fichero hotels.xlsx:')
print(hotels.isnull().sum(), '\n')
```

Seguidamente, generamos los listados de segmentos, clientes y regímenes que asignaremos posteriormente de manera aleatoria. Además, se crean e inicializan vacías las nuevas columnas que se añadirán a los data sets de reservas y hoteles, así como la variable formar la cual contendrá el formato de la fecha para los campos de tipo fecha:

```
#Generamos los listados de segmentos, clientes y regímenes que
asignaremos:
listSegments = ['Hoteles sostenibles', 'Solo adultos', 'Hoteles que
aceptan mascotas', 'LGTBI friendly', 'Hoteles de negocios', 'Hoteles
```

```
rurales', 'Hoteles para familias', 'Hoteles con actividades
deportivas', 'Sin segmento']
listClients = ['Travel fun', 'Viajes Fernández', 'A todo viaje',
'Travel with us', 'Viajes paraiso', 'Viaja sin límites', 'Party
travel', 'Unique experiences travel', 'World travel']
listBoards = ['Solo alojamiento', 'Alojamiento y desayuno', 'Todo
incluido', 'Media pensión', 'Pensión completa']
bookings['Hotel ID'] = None
bookings['Roomnights'] = None
bookings['Total Price'] = None
bookings['Client'] = None
bookings['Lead time bucket'] = None
bookings['Arrival date'] = None
bookings['Booking date'] = None
format = '%d/%m/%Y'
hotels['Segment'] = None
```

En cuanto a la generación de las listas, estas incluyen 9 segmentos de hotel, 9 clientes y 5 regímenes distintos. En referencia a los segmentos de hotel, se han incluido aquellos que considero más relevantes actualmente, así como un volumen reducido de clientes sobre el cual repartir las diferentes reservas. Por último, en referencia al listado de regímenes se han generados los que considero más habituales y por tanto aquellos que suelen ofrecer casi todos los hoteles, si bien es cierto que podrían existir otros regímenes adicionales menos frecuentes como el “desayuno inglés”, “desayuno continental” … que no ofrecen todos los hoteles. Además, dado el volumen de datos de los ficheros, así como las posibles combinaciones con los distintos valores de los listados, se obtienen unos resultados variados y cercanos a lo que podríamos encontrar en una empresa del sector.

Posteriormente, se asignan los segmentos a los hoteles de manera aleatoria, si bien hay segmentos que tienen un mayor peso y por tanto habrá más hoteles de aquellos segmentos cuyo peso sea mayor. Por ejemplo, he considerado que el segmento “Solo adultos” o “Sin segmento” tengan mayor peso que el resto de los segmentos dado que es más frecuente encontrar estos tipos de hoteles a otros pertenecientes al resto de segmentos. Seguidamente se eliminan caracteres especiales generados en el nombre del segmento derivados de la asignación aleatoria.

```
#Asignamos el segmento del hotel:
for h in range(len(hotels)):
  hotels['Segment'][h] = random.choices(listSegments, weights =
[3,4,2,3,2,2,2,1,5])
  hotels['Segment'][h] = str(hotels['Segment'][h]).replace("'", "")
  hotels['Segment'][h] = str(hotels['Segment'][h]).replace("]", "")
  hotels['Segment'][h] = str(hotels['Segment'][h]).replace("[", "")
```

Una vez asignados los segmentos, recorremos cada una de las reservas. Lo primero que hacemos es eliminar las reservas en las que no haya adultos (no pueden existir reservas sin un adulto) o bien haya más de 3 niños por reserva, ya que tampoco es habitual encontrar reservas donde la ocupación sea de 1/2 adultos y más de 3 niños. También se eliminarán aquellas reservas en las cuales el número total de noches sean 0 ya que para crear una reserva válida debe contener al menos 1 noche de estancia.

Una vez hecho esto, asignamos un hotel a la reserva, teniendo en cuenta que si la reserva contiene niños se descartan los hoteles de los segmentos “Solo adultos” y ”Hoteles de negocios” y si la reserva no contiene niños, se descartan los hoteles del segmento “Hoteles para familias” ya que en general son hoteles dirigidos a familias con niños. Posteriormente se asigna el hotel a la reserva de uno de los segmentos resultantes de manera aleatoria.

```
#Recorremos cada una de las reservas:
for b in range(len(bookings)):
  #Eliminamos las líneas de reservas en las que no hay adultos o más
de 3 niños o el número total de noches de la reserva es 0:
  if bookings['no_of_adults'][b] == 0 or
bookings['no_of_children'][b] > 3 or
(bookings['no_of_weekend_nights'][b] +
bookings['no_of_week_nights'][b] == 0):
    bookings = bookings.drop(b)
    continue
  #Asignamos un hotel, si hay niños se descartan los segmentos de
solo adultos y hoteles de negocios y si no los hay se descarta el
segmento de hoteles para familias:
  if (bookings['no_of_children'][b]) > 1:
    hotelsAux = hotels[(hotels.Segment.isin(["Hoteles sostenibles",
"Hoteles para familias", "Hoteles que aceptan mascotas", "LGTBI
friendly", "Hoteles rurales", "Hoteles con actividades deportivas",
"Sin segmento"]))]
  else:
    hotelsAux = hotels[(hotels.Segment.isin(["Hoteles sostenibles",
"Solo adultos", "Hoteles que aceptan mascotas", "LGTBI friendly",
"Hoteles de negocios", "Hoteles rurales", "Hoteles con actividades
deportivas", "Sin segmento"]))]
  hotels_id = list(hotelsAux['Hotel ID'])
  bookings['Hotel ID'][b] = random.choice(hotels_id)
```

A continuación, se asignan a la reserva el cliente y el régimen. En cuanto a los clientes (se entiende cliente no como el titular de la reserva sino como la agencia, tour operador… a través de la cual se ha realizado la reserva), se le asigna un peso a cada uno de ellos dado que por norma general habrá clientes que reserven más que otros y no todos reservaran por igual. Por otro lado, con respecto al régimen, también se asignan de manera aleatoria con distintos pesos siendo los regímenes de “Solo alojamiento” y “Alojamiento y desayuno” aquellos con mayor peso dado que suelen ser los más habitualmente reservados.

Seguidamente se eliminan caracteres especiales generados en el nombre del cliente y el régimen derivados de la asignación aleatoria.

```
  #Asignamos un cliente:
  bookings['Client'][b] = random.choices(listClients, weights = [7,
5, 1, 2, 1, 3, 1, 2, 1])
  bookings['Client'][b] = str(bookings['Client'][b]).replace("'", "")
  bookings['Client'][b] = str(bookings['Client'][b]).replace("[", "")
  bookings['Client'][b] = str(bookings['Client'][b]).replace("]", "")
  #Asignamos un régimen:
  bookings['type_of_meal_plan'][b] = random.choices(listBoards,
weights = [10, 5, 1, 2, 1])
  bookings['type_of_meal_plan'][b] =
str(bookings['type_of_meal_plan'][b]).replace("'", "")
  bookings['type_of_meal_plan'][b] =
str(bookings['type_of_meal_plan'][b]).replace("]", "")
  bookings['type_of_meal_plan'][b] =
str(bookings['type_of_meal_plan'][b]).replace("[", "")
```

Después se continúa generando el resto de nuevos campos para el data set de las reservas. Son los siguientes:

- Roomnights: Es el número total de noches de la reserva, el cual se obtiene de la suma de los campos no_of_weekend_nights + no_of_week_nights.
- Total Price: Es el precio total de la reserva, el cual se obtiene multiplicando el avg_price_per_room por el número total de noches (roomnights). Hay casos en que el total price es 0 o un valor muy bajo (debido a que el avg_price_per_room incluido en el fichero es 0 o muy bajo), lo cual no es realista para el precio de una reserva, si bien no se ha realizado ninguna acción dado que este campo no se ha incluido en las visualizaciones. En caso de querer incluirlo, habría que tomar una decisión de que hacer en estos casos.
- Arrival date: Es la fecha de llegada de la reserva (YYYY/MM/DD) generada en base a los campos arrival_year, arrival_month y arrival_date.
- Booking date: Es la fecha de creación de la reserva (YYYY/MM/DD) generada en base a la fecha de llegada (arrival date) y los días de antelación de la reserva.
- Lead time bucket: Rango de antelación de la reserva en base a los días de antelación con la cual se ha realizado la reserva. Se han establecido 7 rangos diferentes, por ejemplo: “De 0 a 1 día”, “De 2 a 7 días”, “De 8 a 14 días”…

```
  #Obtenemos el número total de noches:
  bookings['Roomnights'][b] = bookings['no_of_weekend_nights'][b] +
bookings['no_of_week_nights'][b]
  #Obtenemos el precio total de la reserva:
  bookings['Total Price'][b] = bookings['Roomnights'][b] *
bookings['avg_price_per_room'][b]
  #Obtenemos la fecha de llegada:
```

```
  fecha = str(bookings['arrival_date'][b]) + '/' +
str(bookings['arrival_month'][b]) + '/' +
str(bookings['arrival_year'][b])
  bookings['Arrival date'][b] = datetime.datetime.strptime(fecha,
format)
  bookings['Arrival date'][b] = bookings['Arrival date'][b].date()
  #Obtenemos la fecha de reserva:
  bookings['Booking date'][b] = bookings['Arrival date'][b] -
datetime.timedelta(days=int(bookings['lead_time'][b]))
  #Asignamos los rangos de antelación de la reserva:
  if bookings['lead_time'][b] == 0 or bookings['lead_time'][b] == 1:
    bookings['Lead time bucket'][b] = 'De 0 a 1 día'
  elif bookings['lead_time'][b] > 1 and bookings['lead_time'][b] <=
7:
    bookings['Lead time bucket'][b] = 'De 2 a 7 días'
  elif bookings['lead_time'][b] > 7 and bookings['lead_time'][b] <=
14:
    bookings['Lead time bucket'][b] = 'De 8 a 14 días'
  elif bookings['lead_time'][b] > 15 and bookings['lead_time'][b] <=
30:
    bookings['Lead time bucket'][b] = 'De 15 a 30 días'
  elif bookings['lead_time'][b] > 31 and bookings['lead_time'][b] <=
90:
    bookings['Lead time bucket'][b] = 'De 1 a 3 meses'
  elif bookings['lead_time'][b] > 91 and bookings['lead_time'][b] <=
180:
    bookings['Lead time bucket'][b] = 'De 3 a 6 meses'
  else:
    bookings['Lead time bucket'][b] = 'Más de 6 meses'
```

El último paso es asignar el tipo de habitación a la reserva. Para ello, se ha tenido en cuenta la ocupación de la habitación. Es decir, si el número total de paxes es 1, el tipo de habitación asignada a la reserva será uno de la gama “Individual”, si el número de paxes es 2, será una habitación del tipo “Doble”, si son 3 será del tipo “Triple” y si son más de 3 paxes en la reserva serán habitaciones cuádruples o apartamentos con múltiples habitaciones. Dentro de cada tipo de habitación (individual, doble, triple…) hay diferentes características de habitación, la estándar, con balcón, superior…por lo que una vez identificada la ocupación de la habitación se le asignará de manera aleatoria el tipo de habitación, donde la habitación cuya característica es “estándar” suele ser la más habitual y por tanto tendrá un peso mayor al resto en las asignaciones. Seguidamente se eliminan caracteres especiales generados en el nombre de la habitación derivados de la asignación aleatoria.

```
  #Asignamos el tipo de habitación en función de la ocupación:
  if (bookings['no_of_adults'][b] + bookings['no_of_children'][b]) ==
1:
```

```
    bookings['room_type_reserved'][b] = random.choices(['Individual
estándar', 'Individual superior', 'Individual con balcón',
'Individual cama matrimonio'], weights = [3, 2, 1, 1])
  elif (bookings['no_of_adults'][b] + bookings['no_of_children'][b])
== 2:
    bookings['room_type_reserved'][b] = random.choices(['Doble
estándar', 'Doble superior', 'Doble con balcón', 'Doble cama
matrimonio', 'Doble 2 camas'], weights = [3, 2, 1, 2, 1])
  elif (bookings['no_of_adults'][b] + bookings['no_of_children'][b])
== 3:
    bookings['room_type_reserved'][b] = random.choices(['Triple
estándar', 'Triple superior', 'Triple con balcón', 'Triple 2
dormitorios', 'Triple 3 camas'], weights = [3, 2, 1, 2, 1])
  else:
    bookings['room_type_reserved'][b] = random.choices(['Cuádruple
estándar', 'Apartamento 3 habitaciones', 'Cuádruple 3 camas'],
weights = [3, 2, 1])
  bookings['room_type_reserved'][b] =
str(bookings['room_type_reserved'][b]).replace("'", "")
  bookings['room_type_reserved'][b] =
str(bookings['room_type_reserved'][b]).replace("]", "")
  bookings['room_type_reserved'][b] =
str(bookings['room_type_reserved'][b]).replace("[", "")
```

El último paso del código una vez recorridas las reservas es la generación de los dos ficheros con las transformaciones realizadas.

```
#Generamos los ficheros con las transformaciones:
bookings.to_excel('bookings_processed.xlsx', index=False)
hotels.to_excel('hotels_processed.xlsx', index=False)
```

Repositorio github

https://github.com/dpriegob/TFM-Analisis_de_reservas_hoteleras_por_segmento_de_hotel_mediante_tecnicas_de_visualizacion_de_datos.git