Modelo de detección de notas negativas en Booking.com

Ayudando a revalorizar los establecimientos hoteleros

Trabajo de final de máster

Fecha de entrega

17/09/2020

información de contacto

Albert Hinojosa: albert.hinojosa@gmail.com

Krishna Morales: krishna.morales@hotmail.com

Índice

1. Idea de negocio

1.1 Contexto

Cada día millones de personas acceden a plataformas de reservas online en busca de un establecimiento donde alojarse, los motivos pueden ser muy diversos, ya sea para disfrutar de sus vacaciones, realizar un viaje de trabajo, entre otras razones.

Los usuarios de estas plataformas acceden de forma rápida y sencilla a una gran cantidad y variedad de establecimientos repartidos por todo el mundo generando millones de reservas y a su vez, millones de comentarios sobre las experiencias vividas en dichos establecimientos.

Las puntuaciones y comentarios realizados por los clientes forma parte de una reputación online que se construye sobre cada establecimiento y que es capaz de influir en muchos otros usuarios a la hora de buscar un lugar donde alojarse.

El reto de los establecimientos

Mejorar la reputación online del establecimiento y su nota media establecida por los clientes es un todo un reto, ya que, cada cliente tiene sus propias expectativas y una forma personal de percibir los servicio del establecimiento.

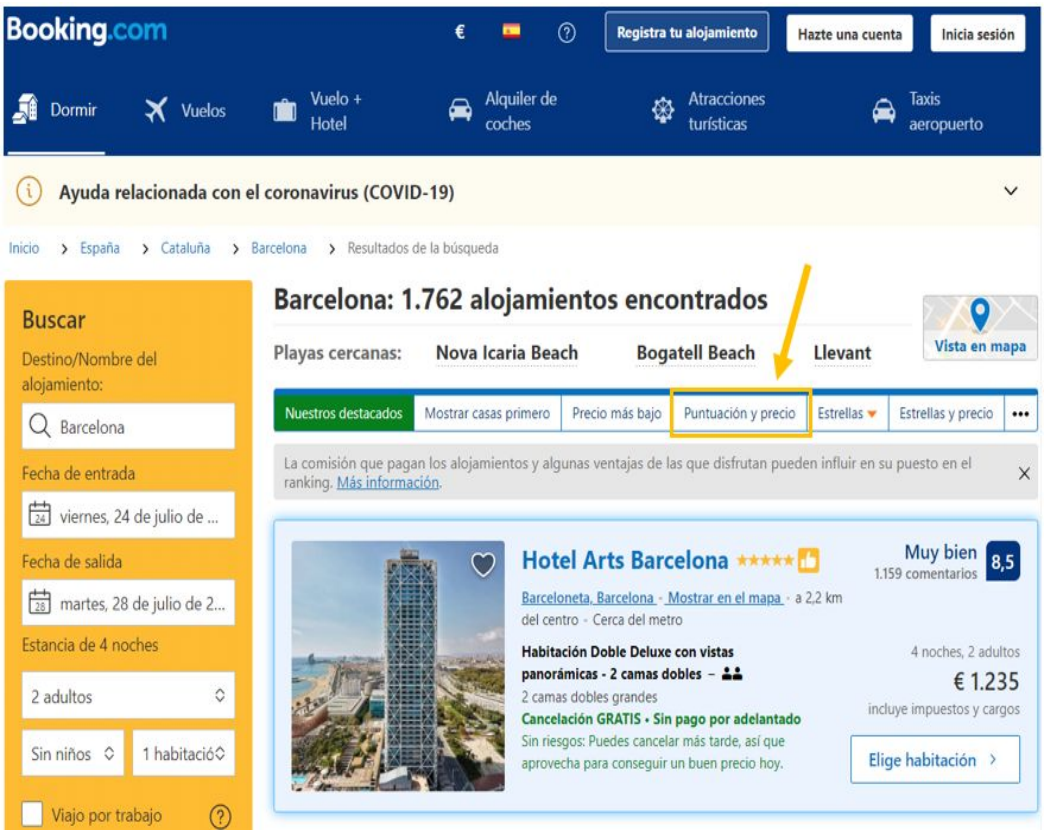
Por otro lado, el proceso de mejora de dicha nota es un proceso complejo y lento para el establecimiento, ya que, debe ser capaz de cubrir muchos aspectos clave a la vez de superar las expectativas del cliente con solvencia. El confort, la limpieza, unas instalaciones de calidad, un trato lo más personalizado posible, una restauración de calidad, una localización apropiada, entre otros muchos aspectos, definirán una opinión que quedará reflejada posteriormente en forma de nota influyendo en la reputación y las ventas.

Contar con información del cliente previo a su llegada puede ser un aspecto determinante para poder ofrecer una experiencia óptima que permita adelantarse a sus necesidades y superar las expectativas creadas. Es por ello que, los establecimientos recaban información del cliente durante y posteriormente a su estancia a través de encuestas de satisfacción. Esta información se almacena en un CRM (Customer Relationship Management), con la que, no solo realizan campañas de marketing personalizadas que incentiven las ventas, sino también implementen acciones que permitan anticiparse a esas necesidades, influyendo en su satisfacción y valoración final.

1.2 Oportunidad de negocio

Mejorar la nota de un establecimiento es todo un reto, pero a su vez, toda una oportunidad de negocio, ya que, los establecimientos capaces de mejorarla tienen una gran oportunidad de hacer crecer su negocio por varias razones:

1. Una mejor nota genera un mayor interés a más clientes potenciando las ventas.
2. Una mejor nota revaloriza el establecimiento permitiéndole establecer precios más altos impactando en las ventas pues cada habitación pasa a ser más rentable.
3. Permite captar nuevos clientes, entre ellos, algunos con mayor poder adquisitivo.
4. Permite mejorar la visibilidad diferenciándose de otros competidores con el mismo cliente objetivo. Si observamos abajo hay un filtro específico destinado a la puntuación, lo que permite al establecimiento tener un mejor posicionamiento si su nota es más alta.



1.3 Propuesta

La pregunta que nos planteamos es la siguiente: ¿qué podemos hacer para ayudar a que los establecimientos mejoren su nota y en consecuencia ayudarles a hacer crecer su negocio?

Objetivo

Nuestro objetivo es crear un modelo que permita detectar aquellas reservas que potencialmente pueden puntuar por debajo de un 7, teniendo en cuenta que la valoración en Booking.com va de 0 a 10, y que más penalizan a la nota media del establecimiento.

Con esta información y a modo de alarma, el establecimiento puede actuar con antelación sobre estas reservas, ya sea, contactando al cliente previamente o durante su estancia, con el fin de conocer sus necesidades y adelantarse a ellas, mejorar su experiencia y en consecuencia su valoración final.

Por otro lado, estamos interesados en saber qué motivos pueden inducir al cliente a poner una valoración por debajo de 7.

2. Exploración del dataset

2.1 Fuentes de información

Kaggle

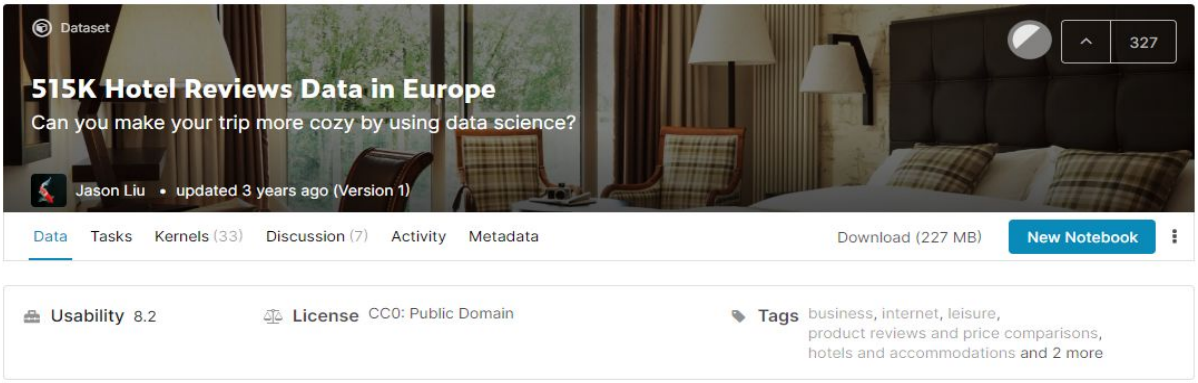
Para realizar este proyecto partiremos de un dataset que encontramos en Kaggle con información de 515.000 reservas realizadas a través del canal de ventas online Booking.com.

Estas reservas se realizaron a lo largo de dos años (2015- 2017) en 1.500 hoteles de 4 y 5 estrellas situados en 6 ciudades europeas: Londres, París, Barcelona, Amsterdam, Milán y Viena.

El dataset se compone de 17 columnas con información sobre las características de la reserva, así como el perfil del cliente que hizo la reserva e información del hotel donde se realizó la reserva.

A lo largo de este segundo apartado iremos viendo cómo el dataset ha ido ampliándose con nuevas variables hasta llegar a 78, con el fin, de aportar más información de valor al estudio.

Link: https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe



Variables originales del dataset

1- **Hotel_Address**: Dirección del hotel (type: String).
2- **Additional_Number_of_Scoring**: Nos dice cuántas reservas puntuaron pero no añadieron un comentario (type: integer)
3- **Review_Date**: Fecha cuando el cliente valoró el establecimiento (type: Date)
4- **Average_Score:** Nota media del establecimiento Average (type: float)
5- **Hotel_Name**: Nombre del hotel (type: String)
6- **Reviewer_Nationality**: Nacionalidad del cliente (type: String)

7- **Negative_Review**: Comentario negativo del cliente. Si el cliente no da un comentario negativo saldrá descrito como: 'No Negative' (type: string)
8- **Total_Number_of_Reviews**: Número de veces que el establecimiento ha sido valorado por un cliente (type: integer)
9- **Review_Total_Negative_Word_Counts**: Número total de palabras negativas que tiene el comentario (type: integer)
10- **Positive_Review**: Comentario positivo que dió el cliente. Si el cliente no da un comentario positivo saldrá descrito como: 'No Positive'. (type: string)
11- **Review_Total_Positive_Word_Counts**: Número total de palabras positivas que tiene el comentario (type: integer)
12- **Total_Number_of_Reviews_Reviewer_Has_Given**: Número de comentarios que el cliente ha realizado en el pasado (type: integer)
13- **Reviewer_Score**: Nota que da el cliente al establecimiento basándose en su experiencia (type: float)
14- **days_since_review**: Duración entre la fecha que el cliente realizó la puntuación y el día que se hizo el “scraping” (type: integer)
15- **lat**: Latitud del hotel (type: float)
16- **lng**: longitud del hotel (type: float)
17- **Tag**: Incluye varias variables dentro de sí de gran relevancia como la estancia media, propósito del viaje, tipo de viaje y perfil del viajero.

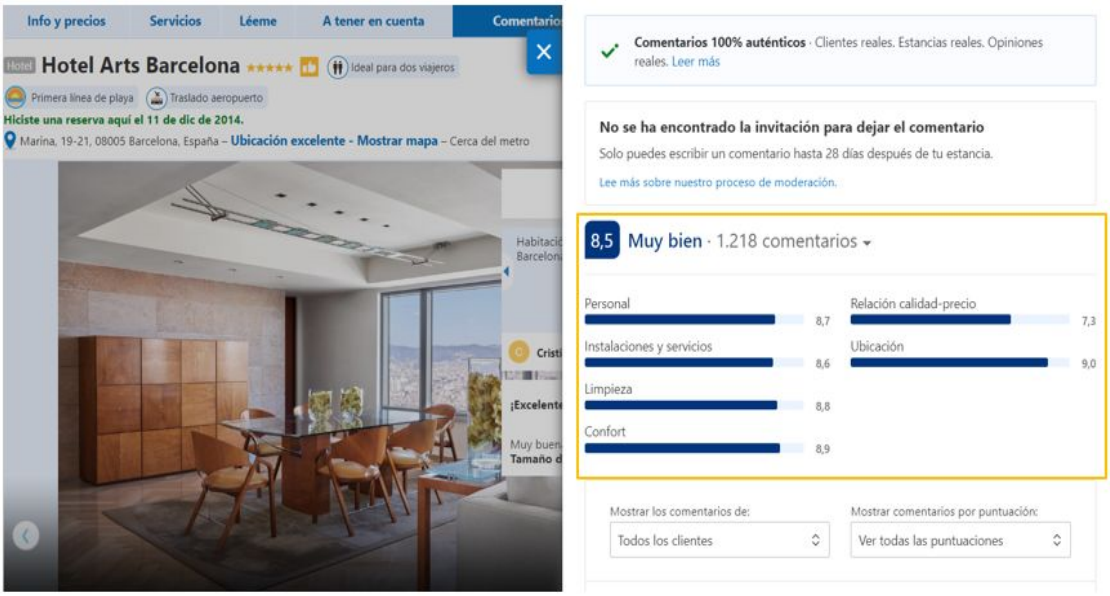
2.2 Criterios de negocio

Criterios de valoración del cliente

Si observamos todas las variables que dispone el dataset y las comparamos con los criterios que el cliente valora a la hora de establecer la nota del establecimiento, nos damos cuenta de que va a ser necesario realizar un trabajo complementario para poder recabar parte de la información más relevante.

En la foto de abajo vemos que la nota media del establecimiento (8.5) resulta de aspectos como la relación calidad-precio, el confort, la limpieza, el personal, las instalaciones y los servicios del establecimiento. Estos son los criterios que Booking.com solicita al cliente que valore, así pues, añadiremos variables que estén relacionadas con estos aspectos clave.

En nuestro análisis cruzaremos esta información junto con la aportada por el dataset para poder entender qué reservas son las que tienen una mayor probabilidad de calificar peor al establecimiento. Para ello utilizaremos varios procesos:



1- A través de páginas de información turística, obtendremos información referente a la **localización** del establecimiento para entender como es de relevante para el cliente la distancia al aeropuerto/ tren, al centro de la ciudad y a puntos de interés turístico.
2- Realizaremos **web scraping** en la página de Booking.com para obtener información clave de los establecimientos, precio medio y categoría del establecimiento.
3- A través de un proceso de **NLP** analizaremos los comentarios de los clientes que hay incluidos en el dataset para entender qué aspectos el cliente valora mejor y peor.
4- Algunos campos del dataset **deberán desglosarse**, ya que, contienen diversas **variables con valor comercial** para nuestro análisis: la estancia media, el perfil del cliente, el motivo del viaje y el dispositivo de reserva.
5- A partir de campos ya existentes crearemos nuevos algún **KPI de negocio** como por ejemplo el precio de la reserva.

A continuación veremos cómo se llevan a cabo estos procesos con el fin de obtener dicha información y que utilizaremos para realizar posteriormente un análisis de las variables de nuestro dataset. Las conclusiones que se extraigan servirán para entender qué tipo de relaciones encontramos entre ellas y así ver la mejor manera de preparar el dataset para su posterior implementación en los modelos..

2.3 Creación de nuevas variables

2.3.1 Variables relacionadas con la localización

Consideramos que la localización es un aspecto de gran valor para los clientes. Es por ello que hemos creado variables de localización tales como distancia al centro de la ciudad y a puntos de comunicación clave como el aeropuerto o la estación de tren principal de cada ciudad.

Dist_Center: Distancia del hotel al centro de la ciudad (type: float) link a Github

Dist_Airport: Distancia del hotel al aeropuerto principal de la ciudad donde está situado el hotel (type: float) link a Github

Dist_Train: Distancia del hotel al aeropuerto principal de la ciudad donde está situado el hotel (type: float) link a Github

Debido a que la gran mayoría de nuestras reservas viajan por motivos de ocio hemos querido añadir un tercer grupo basándonos en la distancia del hotel a los 10 puntos de mayor interés turístico de la ciudad donde está situado el establecimiento.

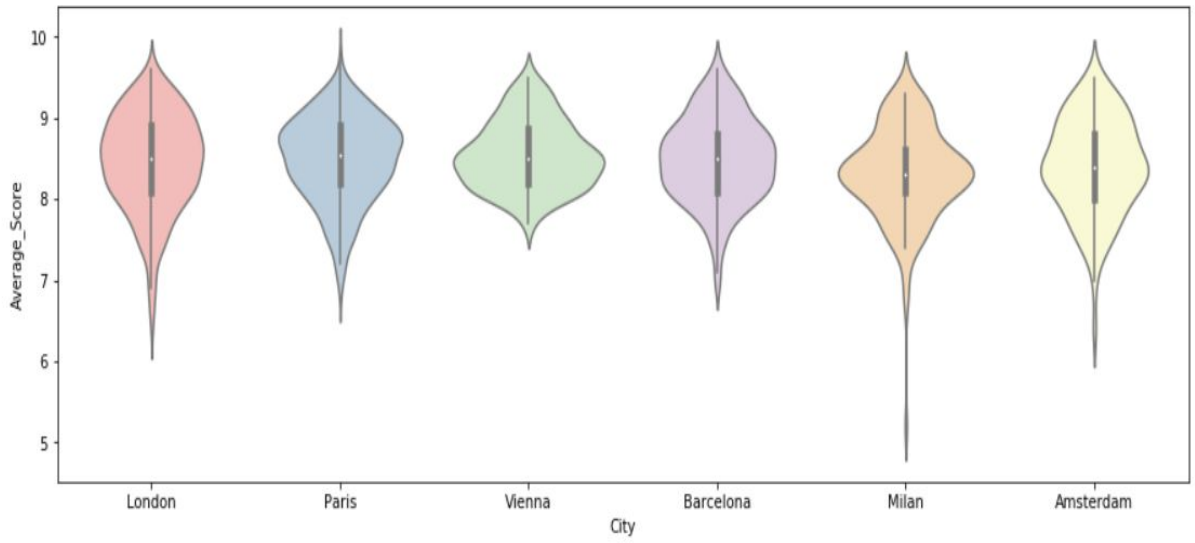
Close_Landmarks: nos dice cuántos de los top 10 landmarks de la ciudad donde está situado el hotel están cerca de éste. Para ello utilizamos la distancia que hay entre el hotel y el punto que geográficamente está más cerca de los 10 landmarks (type: float).
Link a Github1, link a Github2

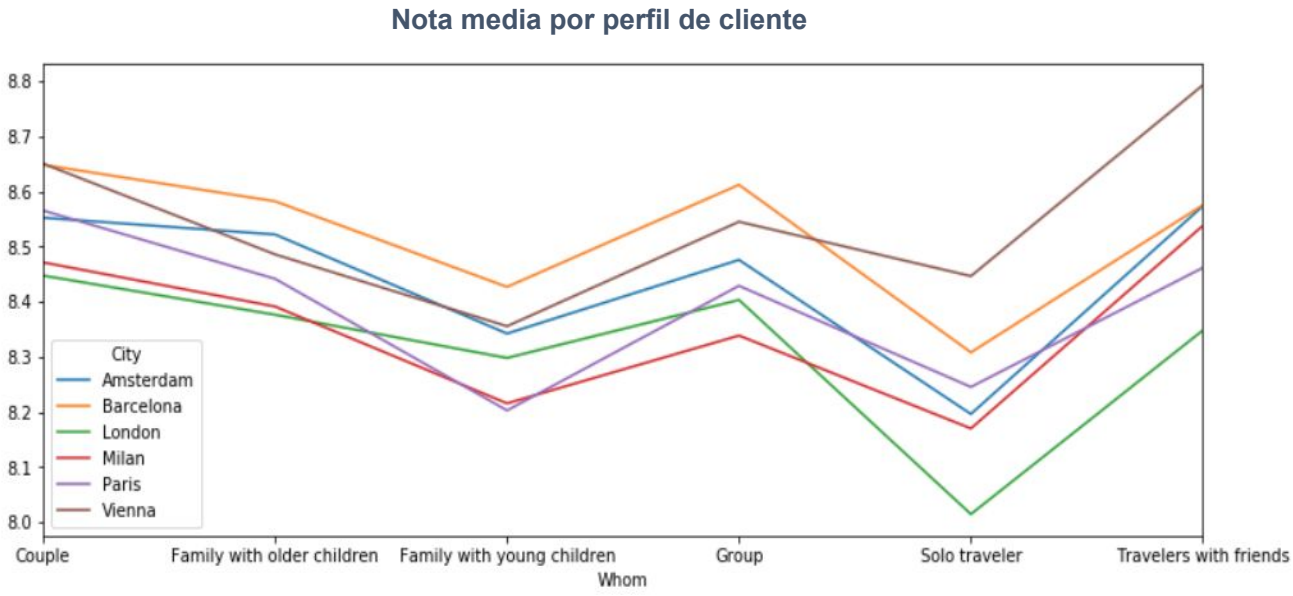
También hemos aprovechado la variable "Hotel Address" y la hemos desglosado en otras variables como la ciudad y país donde está situado el establecimiento, (link a Github)

Country: país donde el hotel está localizado (type: string)
City: ciudad donde el hotel está localizado(type: string)

A través de un "**violin plot**" vemos que la propia ciudad dónde está localizado cada hotel importa dado que Barcelona y Viena son las ciudades con los hoteles con notas medias más altas y las preferidas por la mayoría de clientes independientemente de su perfil.

Nota media por ciudad





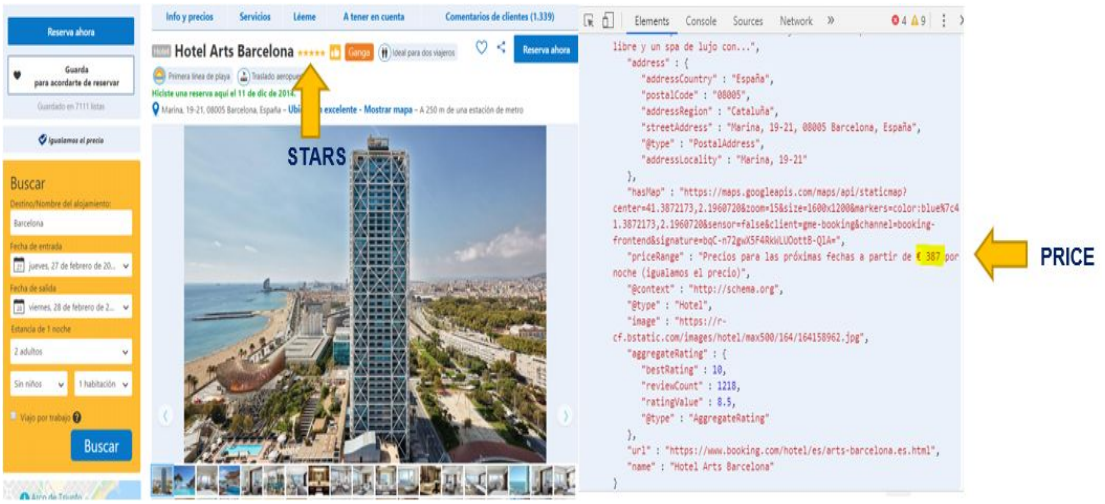
2.3.2 Variables relacionadas con la relación calidad - precio

Aplicación de web scraping para la obtención del precio y la categoría

El dataset no aporta una métrica tan importante como el precio de la reserva, aspecto de gran influencia a la hora de realizar la reserva y aspecto que el cliente deberá valorar posteriormente en función del servicio recibido. ¿Cómo lo obtenemos?

Price: A través de un proceso de "web scraping" extraemos el precio de referencia que se establece en la página de Booking.com para cada hotel (type: float) link a Github

Stars: el mismo proceso de web scraping nos permitirá conocer también las estrellas del establecimiento (type: Strings)



Verificando los resultados obtenidos nos encontramos que hay un conjunto de establecimientos de los que que no hemos obtenido precio a lo largo del proceso. Decidimos probar varios modelos de regresión para para calcular dichos precios siendo un **Random Forest Regressor** el que mejor resultado nos da. Los datos del modelo los incluimos dentro del dataset, (link a Github).

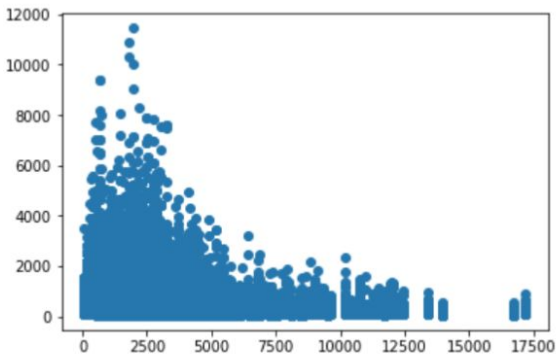
Creación de un KPI de negocio

> **Reservation_ADR**: una vez obtengamos el "Price" lo multiplicaremos por el número de noches de la reserva (Length_N) obteniendo así el coste total de la reserva (type: float). Link a Githhub

A continuación mostramos cómo a pesar de no tener el precio original de la reserva, el precio orientativo establecido a través de web scraping toma todo el sentido de negocio cuando lo cruzamos con otras variables. Por ejemplo, vemos que las reservas son más caras en establecimientos más cercanos al centro de la ciudad.

```
plt.scatter(df['Dist_Center'], df['Reservation_ADR'])
```

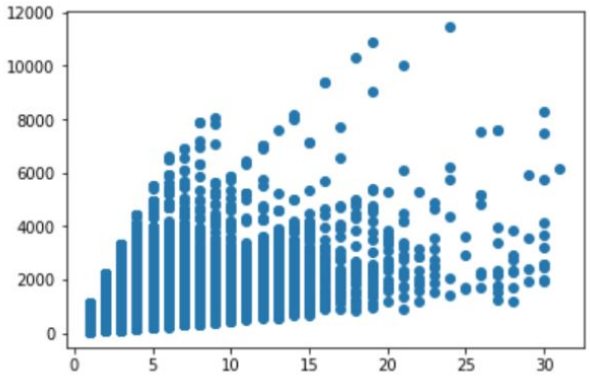
```
<matplotlib.collections.PathCollection at 0x7fa746d2ff10>
```



También observamos que las reservas que incluyen más noches suelen ser más caras, como era de esperar, aunque esto puede variar en función de la ciudad y su nivel económico.

```
plt.scatter(df['Length_N'], df['Reservation_ADR'])
```

```
<matplotlib.collections.PathCollection at 0x7fa746cb6450>
```

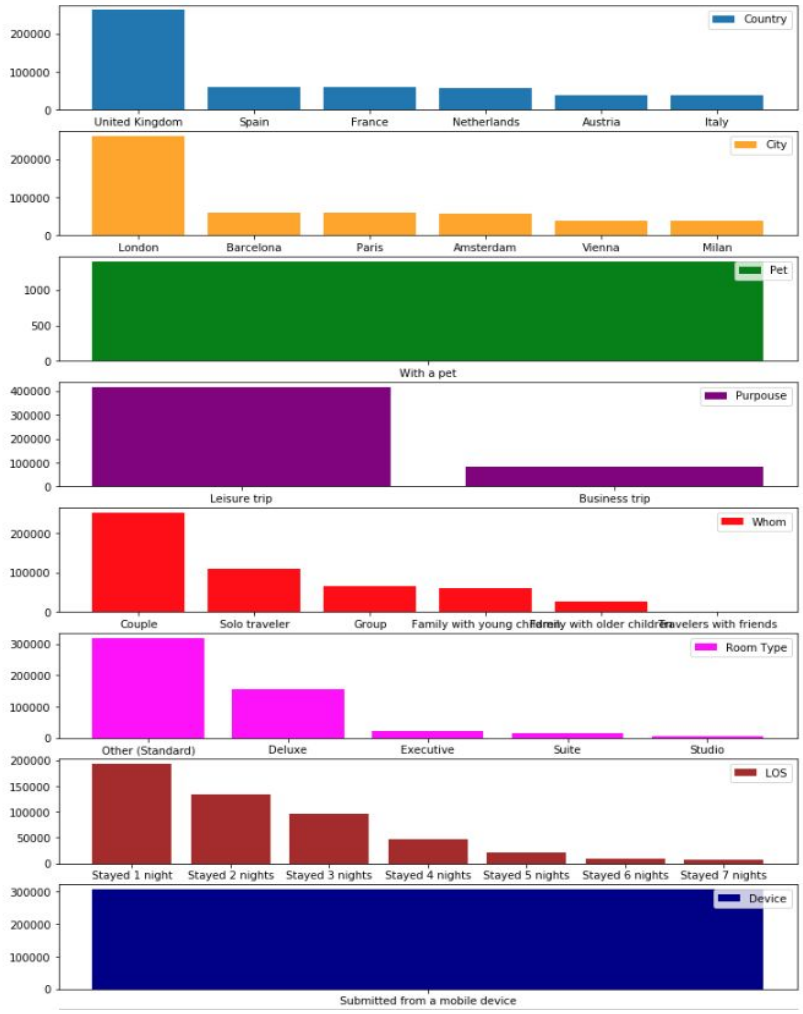


2.3.3 Variables relacionadas con las características de la reserva y el perfil del cliente

A partir del campo "**Tag**" somos capaces de obtener nuevas variables referentes al perfil del cliente y de la reserva. Esta información es muy relevante, ya que, algunas de sus variables influyen claramente en la valoración final del cliente. Las variable extraídas son las siguientes:link a Github

Purpose: motivo del viaje (type: string)
Whom: Perfil del cliente (type: string)
Room: Tipo de habitación. (type: String)
Length: Número de noches alojadas (type: string)
Device: Dispositivo por el cual se hizo la reserva, móvil o ordenador (type: string)
Pet: la reserva incluye un animal doméstico (type: string)

Si visualizamos todas estas variables observamos que casi la mitad de las reservas fueron realizadas por clientes de U.K que viajaron por ocio y en pareja, entre uno y dos días y se alojaron en una habitación estándar o deluxe.

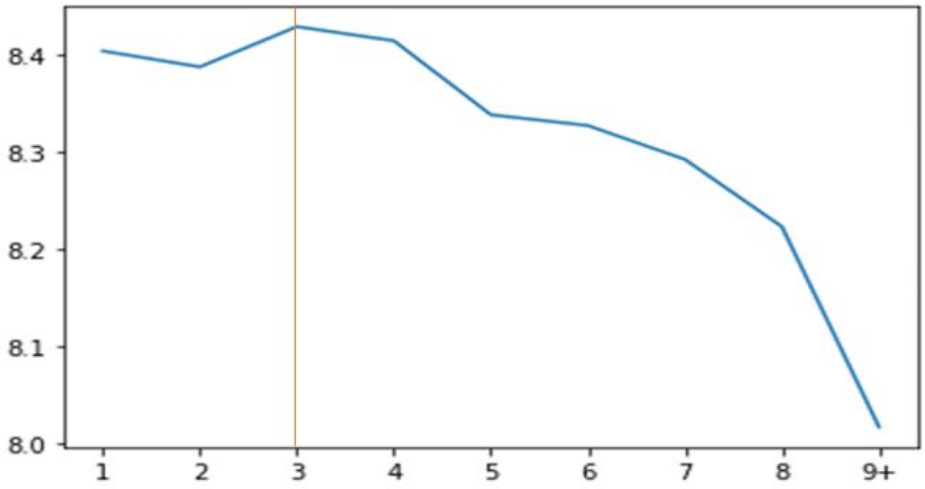


Posteriormente, estas variables pasan a tomar otro formato con la idea de facilitar cálculos posteriores y reducir la gran variedad de categorías que tienen algunos de estos campos.

Room -> Room_Recode: es el Room type agrupado en habitaciones que son de la misma categoría y que muchos establecimientos denominan de manera distinta siendo en realidad la misma tipología. Por ejemplo habitación doble o estándar (type: String) link a Github

Length -> Length_N: Representa el número de noches que el cliente se alojó transformando los valores de string a integers. Se establecen grupos de 1 a 9 o más noches, ya que, la mayoría de reserva en zonas urbanas son suelen tener una estancia media menor a una semana y todos aquello que supere las 9 noches lo agrupamos bajo una misma categoría (type:integer)
link a Github

Con estas modificaciones observamos que los clientes que duermen más noches tienden a dar una peor valoración siendo los que pernoctan 3 noches los que dan una mejor opinión.



La influencia del origen del cliente

A parte del campo Tag encontramos otro de gran interés como es el origen del cliente. Consideramos que el origen cultural puede influir en la manera de ver las cosas y la valoración final que pueden hacer.

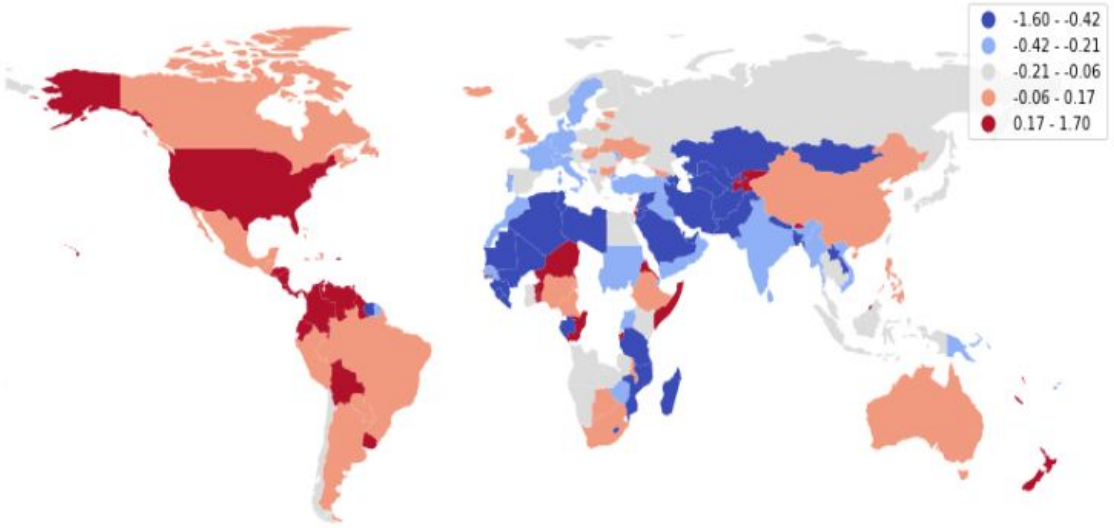
Queremos pues analizar qué nacionalidades tienden a valorar mejor o peor al establecimiento, es decir, que nacionalidades influyen más positiva o negativamente en la nota media. Para ello creamos una nueva variable:

Diff: representa la diferencia que hay entre la nota del cliente y la nota media del hotel.

A partir de esta variable somos capaces de transformar el campo **Reviewer_Nationality** en una nueva variable denominada:

Nationality_Recode: representa las regiones del mundo cuya nota final tiende a favorecer más o menos al establecimiento dado que tienden a tener más o menos diferencia (**Diff**) respecto a la nota media del hotel (type: String). Link a Github

Con geopandas visualizamos dicha variable observando qué regiones del mundo tienden a favorecer más o menos la nota media del hotel. Vemos por colores y tonalidades este aspecto, cuanto más rojo es, más diferencia hay en positivo respecto a la nota media del hotel. Cuanto más azul es, más diferencia hay en negativo respecto a la nota media del hotel.



La influencia de la estacionalidad en la nota del cliente

Otro campo del que desglosar información es **Review_Date**, de donde podremos obtener el mes y el año y los pasaremos de string a integer para su posterior análisis.

Review_Month: mes que efectuó la opinión el cliente (type: factor)
Review_Year: año que efectuó la opinión el cliente (type: integer)

Observamos que tenemos comentarios de dos años seguidos de agosto 2015 a septiembre 2017 aspecto positivo porque toma dos temporadas altas que es cuando suelen concentrarse un mayor número de reservas.

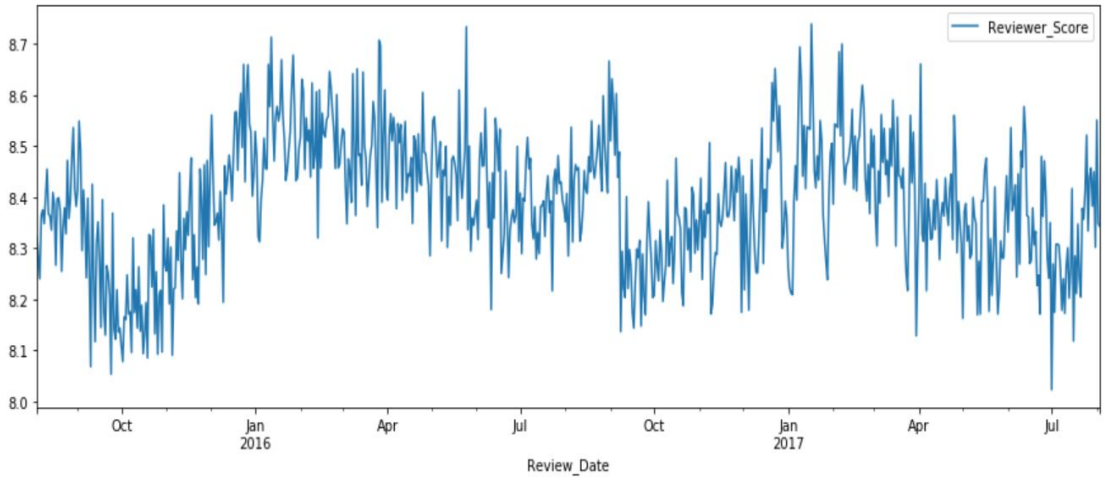
```
# Convert 'Review_Date' field to Date column type
df['Review_Date'] = df['Review_Date'].apply(lambda x: dt.datetime.strptime(x,'%m/%d/%Y'))

df['Review_Month'] = df.Review_Date.apply(lambda x: x.month)
df['Review_Year'] = df.Review_Date.apply(lambda x: x.year)

df['Review_Year'].unique(), df['Review_Month'].unique()
```

```
(array([2017, 2016, 2015]),
 array([ 8,  7,  6,  5,  4,  3,  2,  1, 12, 11, 10,  9]))
```

Si cruzamos esta información con la nota media de los establecimientos vemos que existe una estacionalidad que se corresponde con las temporadas turísticas. La nota media sube o baja en la temporada baja o alta respectivamente.

```
#1 There's a seasonality with lower scores around october and higher values at the beginning of the year
df[['Review_Date','Reviewer_Score']].groupby('Review_Date').mean().plot(figsize=(15,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa74c0f3a90>
```



2.4 Dataset final

Como consecuencia de los procesos realizados previamente obtenemos un nuevo dataset de 37 variables que incluye cuatro aspectos relevantes:

1. Características de la reserva.
2. Información sobre el perfil del hotel
3. Información sobre el perfil del cliente
4. Opinión del cliente sobre su experiencia

```
df.columns
```

```
Index(['Hotel_Address', 'Additional_Number_of_Scoring', 'Review_Date',
       'Average_Score', 'Hotel_Name', 'Reviewer_Nationality',
       'Negative_Review', 'Review_Total_Negative_Word_Counts',
       'Total_Number_of_Reviews', 'Positive_Review',
       'Review_Total_Positive_Word_Counts',
       'Total_Number_of_Reviews_Reviewer_Has_Given', 'Reviewer_Score',
       'days_since_review', 'lat', 'lng', 'Diff', 'Review_Month',
       'Review_Year', 'Country', 'City', 'Pet', 'Purpose', 'Whom', 'Room',
       'Length', 'Device', 'Room_Recode', 'Nationality_Recode',
       'Length_Recode', 'Close_Landmarks', 'Dist_Center', 'Dist_Airport',
       'Dist_Train', 'Price', 'Stars', 'Length_N', 'Reservation_ADR'],
      dtype='object')
```

2.4.1 Análisis de las variables

A partir de este dataset realizamos un análisis de las variables desde diferentes perspectivas.

Por un lado, verificamos si las variables del dataset corroboran parte de nuestras suposiciones basadas en nuestro conocimiento de negocio. Algunas de las conclusiones ya se han ido viendo previamente como por ejemplo:

- Existe una estacionalidad en la nota.
- Hay nacionalidades que influyen más en positivo que otras en la nota.
- La influencia del número de noches alojadas incide en la valoración final.

Por otro lado, realizamos un análisis univariable y multivariable de las variables cualitativas y cuantitativas para observar aspectos como:

- La diferencia entre medias y medianas observando el impacto de los outliers.
- Vemos qué variables cualitativas tienen una mayor frecuencia definiendo la tipología dominante de cliente y reserva.
- También vemos qué variables cuantitativas tienen una mayor correlación entre ellas.

Estas conclusiones se pueden encontrar en el siguiente notebook, (link a Github).

2.4.2 Proceso de NLP aplicado a los comentarios

Realizado este análisis preliminar decidimos ampliar una vez más nuestro dataset aprovechado la información que nos aportan los comentarios de los clientes. Utilizaremos estas opiniones para conocer qué palabras son las más frecuentes. Esto nos permitirá establecer posteriormente unos grupos de palabras que crearemos en base a los criterios de valoración que establece Booking.com: ubicación, relación calidad-precio, confort, instalaciones y servicios, personal, limpieza, entre otros. Finalmente, estos grupos serán añadidos al dataset.

Para ello procedemos con un proceso de NLP que implica varios pasos y que se muestra de forma detallada en los siguientes notebooks (paso 1: link a Github, paso 2: link a Github):

Tokenización: donde definimos que tipo de palabra es cada una: artículo, adjetivo nombre, verbo, adverbio...
Lematización: nos quedamos exclusivamente con los nombres, los verbos (que pasamos a infinitivo) y los adjetivos. Descartamos el gran grueso de palabras como artículos, determinantes, signos de puntuación que no tienen ningún significado en nuestro análisis.
Ranking de palabras: finalmente vemos aquellas que más se repiten de cada una de las tres clases definidas

Estos pasos los realizamos tanto para el conjunto de comentarios positivos como negativos y el resultado en ambos casos de las palabras más frecuentes fue el siguiente:

Conjunto resultante de los comentarios positivos

```
pd.DataFrame.from_dict(count_dict, orient='index').sort_values(by=0, ascending=False).head(50).index
```

```
Index(['staff', 'location', 'room', 'hotel', 'good', 'great', 'breakfast',
       'friendly', 'helpful', 'nice', 'clean', 'bed', 'excellent',
       'comfortable', 'stay', 'positive', 'lovely', 'station', 'close', 'walk',
       'service', 'everything', 'restaurant', 'perfect', 'view', 'bar',
       'quiet', 'amazing', 'comfy', 'bathroom', 'area', 'would', 'facility',
       'get', 'modern', 'love', 'food', 'london', 'metro', 'fantastic',
       'reception', 'free', 'make', 'spacious', 'city', 'place', 'easy',
       'minute', 'beautiful', 'central'],
      dtype='object')
```

Conjunto resultante de los comentarios negativos

```
pd.DataFrame.from_dict(count_dict, orient='index').sort_values(by=0, ascending=False).head(50).index
```

```
Index(['room', 'negative', 'hotel', 'breakfast', 'small', 'staff', 'nothing',
       'bed', 'get', 'would', 'could', 'good', 'night', 'bathroom', 'stay',
       'bit', 'time', 'work', 'shower', 'little', 'check', 'service', 'day',
       'bar', 'need', 'price', 'make', 'pay', 'go', 'book', 'ask', 'reception',
       'floor', 'poor', 'clean', 'expensive', 'door', 'window', 'take', 'use',
       'water', 'noisy', 'air', 'coffee', 'give', 'wifi', 'bad', 'noise',
       'restaurant', 'area'],
      dtype='object')
```

Llegados a este punto agrupamos las palabras en función de los aspectos que Booking solicita al cliente que valore: ubicación, relación calidad-precio, confort, instalaciones y servicios, personal y limpieza y lo añadimos al dataset alcanzando las **69 variables**:

```
df_features_hotels.shape
```

```
(515738, 69)
```

```
df_features_hotels.columns
```

```
Index(['Hotel_Address', 'Additional_Number_of_Scoring', 'Review_Date',
       'Average_Score', 'Hotel_Name', 'Reviewer_Nationality',
       'Review_Total_Negative_Word_Counts', 'Total_Number_of_Reviews',
       'Review_Total_Positive_Word_Counts',
       'Total_Number_of_Reviews_Reviewer_Has_Given', 'Reviewer_Score',
       'days_since_review', 'lat', 'lng', 'Diff', 'Diff_Percentage',
       'Review_Month', 'Review_Year', 'Country', 'City', 'Pet', 'Purpose',
       'Whom', 'Room', 'Length', 'Device', 'Room_Recode', 'Nationality_Recode',
       'Length_Recode', 'Close_Landmarks', 'Dist_Center', 'Dist_Airport',
       'Dist_Train', 'Price', 'Stars', 'Length_N', 'Reservation_ADR',
       'food_Neg', 'staff_Neg', 'location_Neg', 'value_Neg', 'comfort_Neg',
       'room_Neg', 'facilities_Neg', 'cleanliness_Neg', 'food_Pos',
       'staff_Pos', 'location_Pos', 'value_Pos', 'comfort_Pos', 'room_Pos',
       'facilities_Pos', 'cleanliness_Pos', 'food_Neg_Hotel',
       'staff_Neg_Hotel', 'location_Neg_Hotel', 'value_Neg_Hotel',
       'comfort_Neg_Hotel', 'room_Neg_Hotel', 'facilities_Neg_Hotel',
       'cleanliness_Neg_Hotel', 'food_Pos_Hotel', 'staff_Pos_Hotel',
       'location_Pos_Hotel', 'value_Pos_Hotel', 'comfort_Pos_Hotel',
       'room_Pos_Hotel', 'facilities_Pos_Hotel', 'cleanliness_Pos_Hotel'],
      dtype='object')
```

2.4.3 Estadísticos de cada hotel

Vemos que hay establecimientos cuya nota media es producto de una variante de notas bastante amplia entre 0 y 10, así como hoteles que tienden a concentrar la mayoría de sus nota alrededor de la media. Esto es importante tenerlo en cuenta dado que estamos buscando predecir aquellas reservas que van a poner una nota negativa, por debajo de 7, con lo que, hay establecimientos con una mayor tendencia a una mayor rango de nota y en consecuencia con más posibilidades de obtener notas por debajo de 7 que otros.

Así pues, decidimos añadir como últimas variables todas aquellas que hacen referencia a la distribución de la nota dentro de cada establecimiento y de esta manera saber cómo de dispersa o concentrada tiene la nota cada hotel.

count: número de notas por hotel (type: integer).
mean: media (type: float).
Std: desviación estándar (type: float).
min: nota mínima (type: float).
max: nota máxima (type: float).
25%, 50% y 75%: cuartiles (type: float).

Con estas variables el dataset alcanza las 77 variables.

3. Preparación de los datos

3.1 Selección de la mejor estrategia de negocio

Una vez se ha realizado todo el trabajo de creación, modificación y análisis de nuestras variables, decidimos realizar unas primeras pruebas lanzando algunos modelos de clasificación. Pretendemos ver cómo se comporta de inicio y valorar si hay que adaptar nuestras variables más allá de las adaptaciones realizadas hasta el momento.

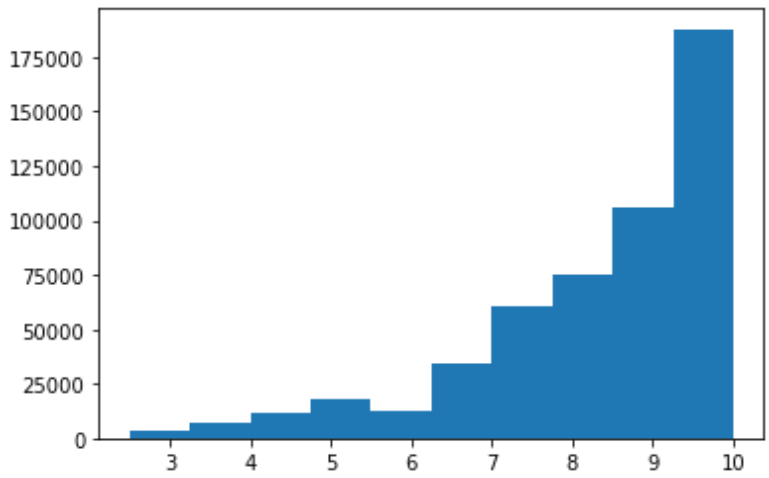
Para asegurarnos de que seguimos las mejor estrategia de negocio (predecir aquellas notas que queden por debajo de 7) decidimos añadir 2 puntos de vista más y compararlos:

1. Predecir aquellas notas que estén dos puntos por debajo de la media del hotel.
2. Predecir aquellas notas que sean un 70% más bajas respecto a la media del hotel.

Los resultados de las pruebas preliminares nos muestran una clara diferencia a favor de nuestra premisa basada en **predecir aquellas notas que quedan por debajo de 7**. Así pues, decidimos continuar nuestro análisis desde esta óptica de negocio.

3.1.1 Creación de categorías por encima y debajo de 7

El siguiente paso consistirá en crear una última variable (nº 78) a nuestro dataset denominada **Category** (type: String) y que será la **variable a modelizar**. Con ella, se diferenciarán aquellas notas que estén por debajo de un 7, considerándose negativas (“Bad”), respecto de las que estén por encima de un 7, considerándose positivas (“Good”).



A partir del histograma de notas podemos ver que aunque las notas suelen estar muy por encima, vamos a tener suficientes casos con la categoría que queremos predecir gracias al gran tamaño de nuestro dataset.

A continuación se prueban dos modelos:

1. Un **Random Forest** por ser un modelo de clasificación que suele tener buenos resultados, incluso sin optimizar hiperparametros. Además tiende a controlar la varianza y evita modelos con overfitting.

2. Una **Regresión Logística**, ya que, es un modelo de clasificación que no está basado en árboles de decisión.

3.1.2 Primeros resultados

Los resultados de ambos modelos alcanza un alto **Accuracy (0.825)**. Este dato no debe llevarnos a engaño, ya que, como la mayoría de las notas del dataset están por encima de 7 la tendencia del modelo es a predecir que la nota estará por encima de 7, obteniendo como no, un Accuracy muy alto de inicio.

Sin embargo, si calculamos otras métricas que complementan nuestro análisis del modelo de clasificación (score <7), tales como el **Kappa (0.05)** o el **F1-score (0.088)** comprobaremos que los resultados son muy bajos, tanto en el random forest com en la regresión logística. Así que, deberemos adaptar mejor nuestro dataset para poder sacar el máximo partido a nuestros modelos de clasificación, (link a Github). Veamos los resultados de ambos modelos:

Random Forest

Predict \ Real	Bad	Good
Bad	17	22
Good	328	1633

```
Accuracy: 0.8250
   Kappa: 0.0554
F1-Score: 0.0885
```

Regresión Logística

Predict \ Real	Bad	Good
Bad	45	79
Good	300	1576

```
Accuary: 0.8105
  Kappa: 0.1108
F1-Score: 0.1919
```

3.2 Balanceo de las categorías

Después de lanzar los dos primeros modelos y ver los primeros resultados decidimos seguir adaptando nuestro dataset. Dado que hay una clase mayoritaria, notas por encima de un 7, equilibramos las categorías positivas y negativas al 50% cada una.

Al disponer de una base de datos amplia, con más de medio millón de casos, se selecciona aleatoriamente el mismo número de notas positivas y negativas igualando al 50% las dos categorías. Esto deja una dataset resultante de 173.702 reservas.

Después de ejecutar el Random Forest vemos que el Accuracy baja considerablemente pero se consigue que el resto de métricas suban especialmente el F1- Score y el Recall, dando así un primer paso en nuestro proceso de equilibrar nuestro modelo de clasificación, (link a Github).

Original Dataset

		Real: Bad	Real: Good
Predict	Bad	45	79
	Good	300	1576

```
Accuracy: 0.8105
   Kappa: 0.1108
F1-Score: 0.1919
```

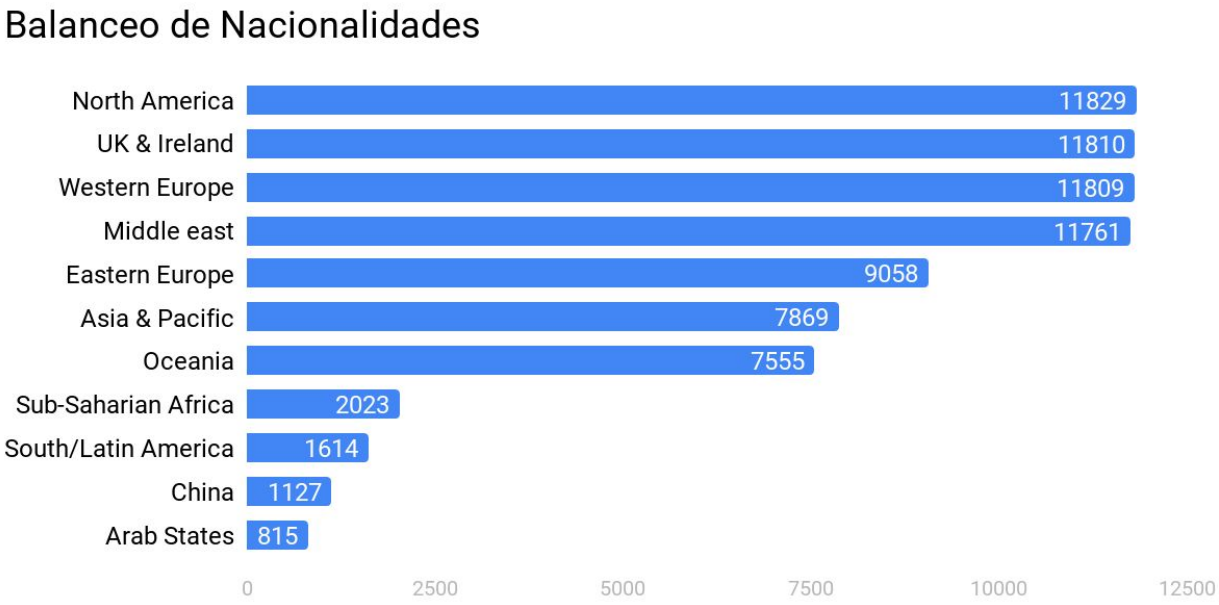
Balanced Categories

		Real: Bad	Real: Good
Predict	Bad	611	420
	Good	384	585

```
Accuracy: 0.5980
   Kappa: 0.1961
F1-Score: 0.6031
```

3.3 Balanceo de nacionalidades

Se decide equilibrar también las nacionalidades, ya que, hay una representación de clientes procedentes de UK que roza el 50% de todas las reservas del dataset y buscamos que haya una representación más equilibrada entre todas las nacionalidades. En esta segunda fase el número de reservas queda en 77.270, ya que, inevitablemente se reduce la categoría mayoritaria (U.K).



Original Dataset

Predict \ Real	Bad	Good
Bad	45	79
Good	300	1576

```
Accuary: 0.8105
  Kappa: 0.1108
F1-Score: 0.1919
```

Balanced Categories

Predict \ Real	Bad	Good
Bad	611	420
Good	384	585

```
Accuracy: 0.5980
   Kappa: 0.1961
F1-Score: 0.6031
```

Balanced Nationality

Predict \ Real	Bad	Good
Bad	615	381
Good	385	619

✔
```
Accuracy: 0.6170
   Kappa: 0.2334
F1-Score: 0.6162
```

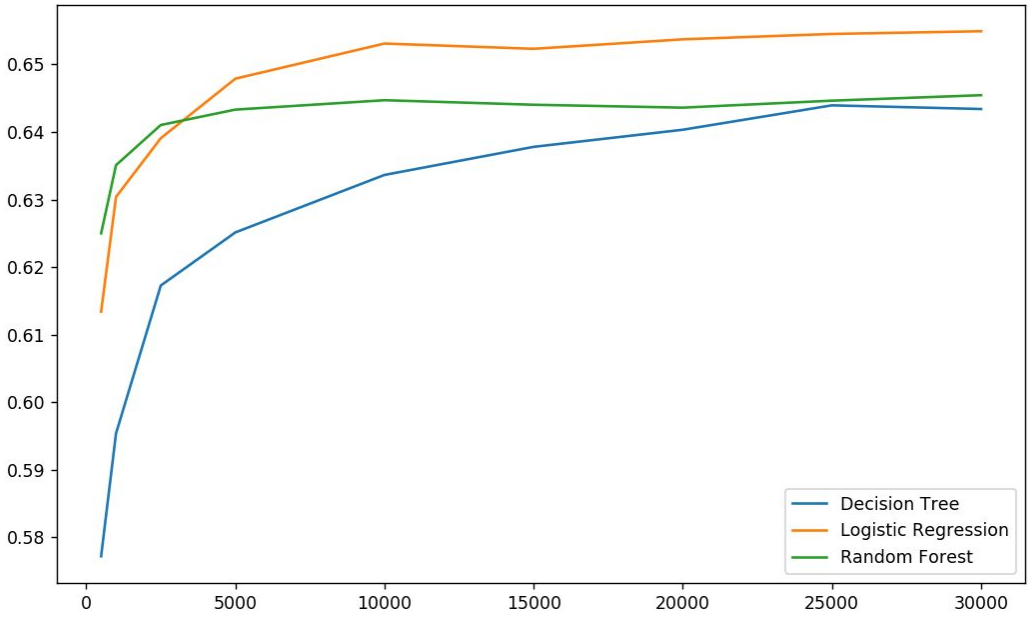
Tras balancear las nacionalidades comprobamos que los resultados son mejores en nuestro último modelo usando Random Forest, que en los modelos previos ya que mejoramos todos los Scores analizados (link a Github).

4. Optimización del modelo de clasificación

4.1 Tamaño óptimo de la muestra

Llegados a este punto nos preguntamos cuál sería el tamaño óptimo del dataset para modelizar. Para ello se comparan 3 modelos con distintos tamaños crecientes: **Random Forest** y un **Árbol de Decisión** a los que se les marca una restricción mínima de profundidad de 5 para poner un límite con sentido que agilice el proceso. Utilizamos un tercer modelo que será una **Regresión Logística** tal y como veníamos haciendo anteriormente.

Vemos que independientemente de que la regresión logística obtiene el mejor Accuracy, el tamaño óptimo puede estar aproximadamente sobre las 10.000 reservas, (link a Github).



	500	1000	2500	5000	10000	15000	20000	25000	30000
DT	0.5772	0.5954	0.6172	0.6251	**0.6336**	0.6378	0.6403	0.6439	0.6434
LR	0.6134	0.6304	0.6390	0.6479	**0.6531**	0.6523	0.6537	0.6545	0.6549
RF	0.6250	0.6351	0.6410	0.6433	**0.6447**	0.6440	0.6436	0.6446	0.6454

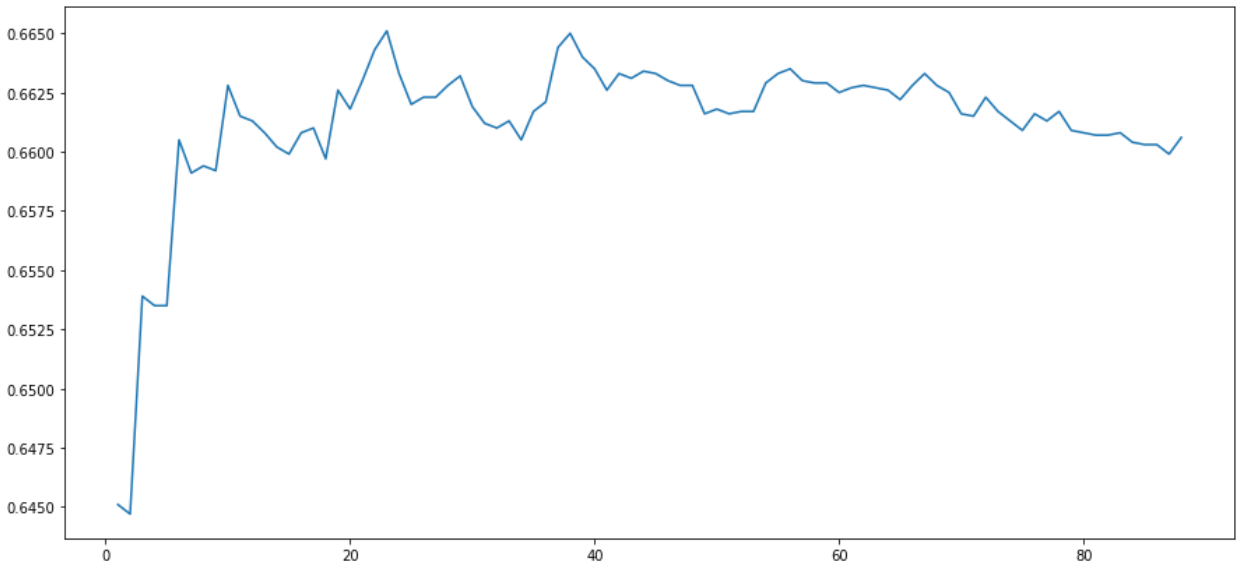
4.2 Selección de variables (RFE)

Tras conocer el tamaño óptimo de reservas y haber realizado las adaptaciones previas, vemos que el número de variables con el que trabajamos es muy grande y nos planteamos tres aspectos:

1. Reducir el número de variables con el que trabajamos, ya que, es posible que no todas las variables estén aportando realmente información relevante y den valor a los modelos ejecutados hasta ahora.
2. Si reducimos el número de variables podremos reproducir los modelos de manera más rápida.
3. Si escogemos las variables adecuadas, podremos aumentar el score de nuestros modelos.

Calculamos en ambos modelos cuál sería la accuracy en función del número de variables óptimo aplicando un **RFE** (Recursive Feature Selection). Veamos los resultados de forma gráfica, (link a Github).

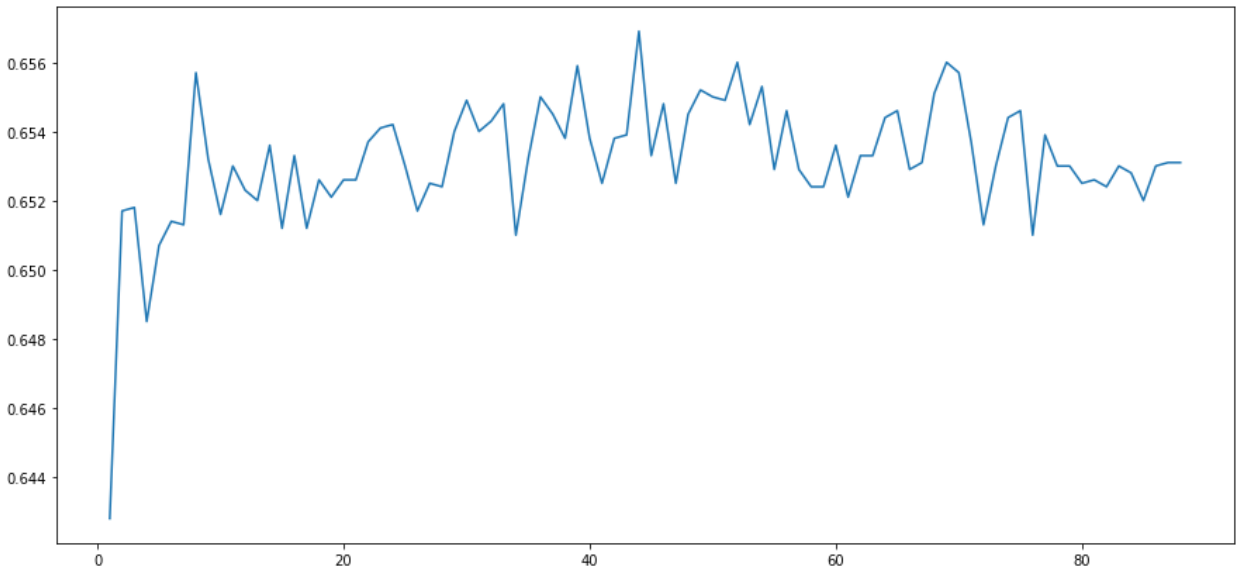
Regresión logística: Observamos que el número de variables óptimo es de 23 o 38.



Observamos 2 picos con Accuracy máximo con 22 y 38 features. A parte de estos 2 picos, la forma de la curva es bastante estable y tiene una tendencia creciente hasta el punto máximo, momento en el que añadir más variables hace ir decreciendo el accuracy de manera continua y constante.

De entre los 2 picos, consideramos mejor coger 38 features ya que no sabemos si coger pocas variables puede ser perjudicial en otros modelos.

Random Forest: Observamos que el número de variables óptimo es 44.



Con Random Forest, quizás por su componente aleatoria vemos saltos muy grandes y poca continuidad en el Accuracy planteado por cada N seleccionada como selección de variables. Entre las opciones planteadas con Regresión Logística o Random Forest elegimos la solución proporcionada por la Regresión Logística por ser mucho más consistente.

Logistic Regression

Features	Accuracy
23	0.6651
38 ✔	0.6650
37	0.6644
22	0.6643
39	0.6635

Random Forest

Features	Accuracy
44	0.6569
69	0.6560
52	0.6560
39	0.6559
8	0.6557

Finalmente seleccionamos las 38 de la Regresión Logística como mejor solución. Todo este trabajo nos permite mejorar nuestro modelo de clasificación dado que sabemos la proporción adecuada del dataset, con 10.000 reservas, y el número de variables más relevantes que son las siguientes:

```
Index(['Average_Score', 'Total_Number_of_Reviews_Reviewer_Has_Given', 'mean',
       'std', 'max', 'Review_Month_1', 'Review_Month_10', 'Review_Month_2',
       'Review_Month_3', 'Review_Month_4', 'Review_Month_7', 'Review_Month_8',
       'Review_Month_9', 'City_Amsterdam', 'City_London', 'Pet_With a pet',
       'Purpose_Leisure trip', 'Whom_Family with older children',
       'Whom_Family with young children', 'Whom_Travelers with friends',
       'Room_Recode_Deluxe', 'Room_Recode_Other (Standard)',
       'Room_Recode_Studio', 'Nationality_Recode_Arab States',
       'Nationality_Recode_Asia & Pacific',
       'Nationality_Recode_Eastern Europe', 'Nationality_Recode_Middle east',
       'Nationality_Recode_North America', 'Nationality_Recode_Oceania',
       'Nationality_Recode_UK & Ireland', 'Nationality_Recode_Western Europe',
       'Length_Recode_Stayed 2 nights', 'Length_Recode_Stayed 5 nights',
       'Length_Recode_Stayed 6 nights', 'Length_Recode_Stayed 7 nights',
       'Length_Recode_Stayed 8 nights', 'Stars_Pension',
       'Stars_hotel de 3 estrellas'],
      dtype='object')
```

Observamos que sobre todo, las variables predominantes son las más básicas relacionadas con la reserva, mientras que las variables añadidas a posteriori tienen menor importancia.

5 Evaluación del modelo

5.1 Valoración de otros modelos de clasificación

Llegados a este punto, y partiendo de la adaptación realizada al dataset, pasamos a una fase en la que se establecen 3 objetivos:

1. **Definir la metrica a maximizar:**
    - En el transcurso de nuestro trabajo hasta ahora se ha ido trabajando básicamente con el **Accuracy** de los modelos. Está métrica es la usada de manera más común para evaluar los modelos de clasificación. Sin embargo en nuestro estudio, en el que nos interesa sobre todo maximizar los aciertos en reservas por debajo de 7 puntos, puede que esta no sea la mejor de las opciones, ya que, trata por igual todas las clases analizadas.

    - Después de descartar el accuracy como métrica principal a maximizar se ha usado como métrica el **F1-Score** para la clase de interés. Sin embargo, se ha observado que algunos modelos que maximizan nuestro F1 bajan mucho el Accuracy debido a un fuerte desbalance entre Recall y Precision. Por lo general está disminuyendo la Precision con lo que aumentan los Falsos Positivos.

    - Para intentar reducir estos Falsos Positivos se ha creado un score ad-hoc que hemos llamado **H-Score** (Harmonic Score) que no es más que la media armónica entre el Accuracy y el F1-Score y que probaremos a maximizar.

2. **Explorar diversos modelos de clasificación** que puedan afinar mejor nuestro score. Evaluaremos modelos que ya veníamos utilizando como los Árboles de Decisión, el Random Forest y la Regresión Logística y añadiremos un KNN(KD Tree), Gradient Boosted Trees, XG Boosting, SVM y en una última instancia un Catboost.

3. **Seleccionar los mejores parámetros**: para asegurarnos de que los resultados de evaluación del modelo son independientes de la muestra elegida realizaremos una validación cruzada en la que los grupos de datos de entrenamiento se escogen aleatoriamente. Para la selección de los mejores parámetros se han probado distintas técnicas de búsqueda como Grid Search, Random Search y Optimización Bayesiana que ha sido la optimización final elegida, ya que, de manera general ha resultado ser la mejor en términos de rapidez y maximización del score.

Así pues, se han creado funciones para la evaluación y maximización de nuestros distintos Scores mediante validación cruzada y optimización bayesiana de hiperparametros en distintos modelos que se pueden encontrar dentro de los ficheros de optimización para cada una de las métricas (ver apartado 6.2). El proceso es el mismo en cada optimización con la diferencia de que los resultados difieren según el score elegido a maximizar.

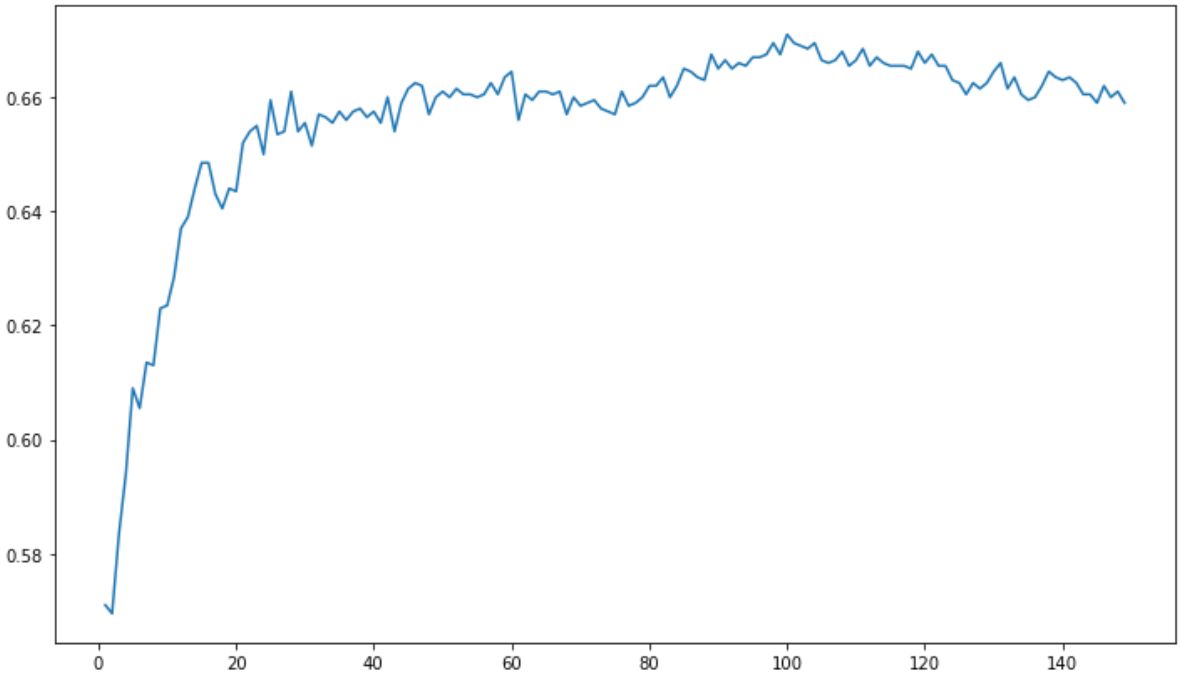
5.2 Resultados de los Modelos de Clasificación

Tras encontrar los hiperparametros que maximizan cada métrica para cada modelo llegamos a las siguientes conclusiones:

Maximización del Accuracy: (link a Github)

Los modelos que maximizan el Accuracy no consiguen valores excesivamente altos, quedando de manera general alrededor del 66%. La maximización de esta métrica estima valores similares para el resto de métricas de interés (F1-Score, Recall). Parecen modelos muy generales.

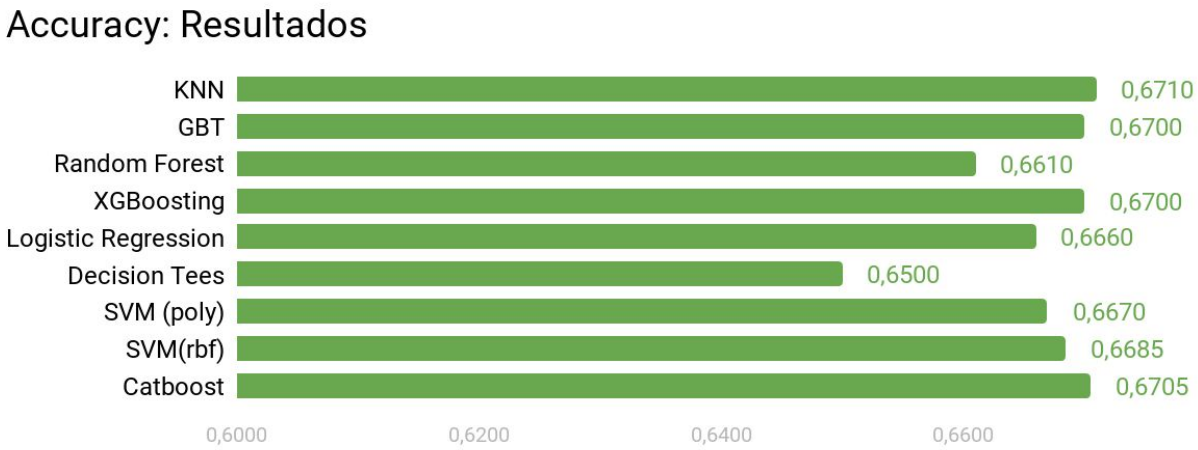
El modelo con mejores resultados es el más simple de todos, el **KNN** con un accuracy de 67.1% contra nuestro test set. Sin embargo, el K óptimo es de 100 vecinos, lo que también nos puede estar indicando lo complicado de acertar en nuestro modelo y lo generalista que es, ya que, precisa de una gran cantidad de vecinos para encontrar su valor óptimo.



Accuracy vs selección de K vecinos

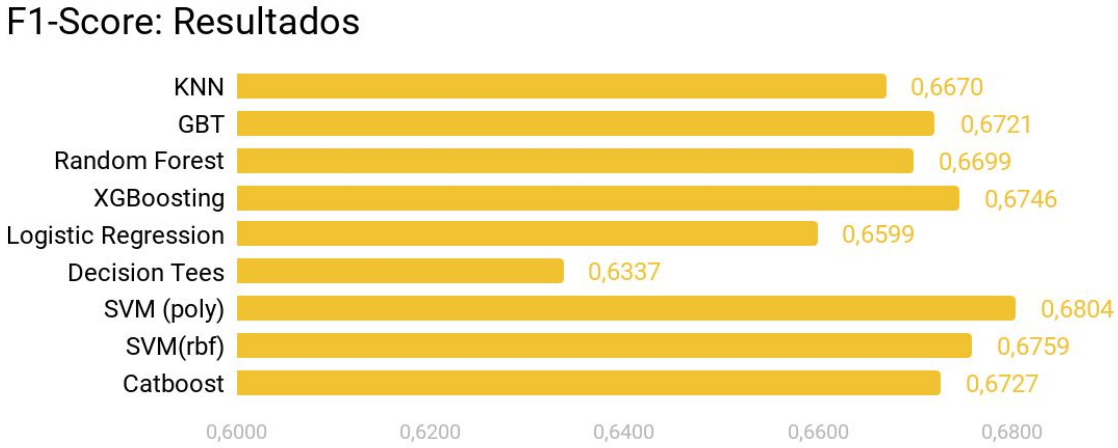
De entre el resto de modelos el que ha dado mejores resultados destacan Catboost, GBT y XGB.. (link a Github).El modelo con peores resultados ha sido el Decision Trees.

Uno de los problemas que ya se observa en el análisis del Accuracy y que descarta la métrica como óptima es que por lo general acierta en menor porcentaje la clase de interés (notas por debajo de 7) y los resultados se suponen mejorables.



F1-Score: (link a Github)

Intentando conseguir acertar en mayor medida la clase de interés se han calculado modelos que maximizaran el F1-Score. En este caso los resultados han variado más entre modelos situándose entre el 63% del modelo con menor score y un 68% del modelo que tiene mejores resultados un **SVM con kernel polinomial**. Su Accuracy en este caso baja hasta el 65%.

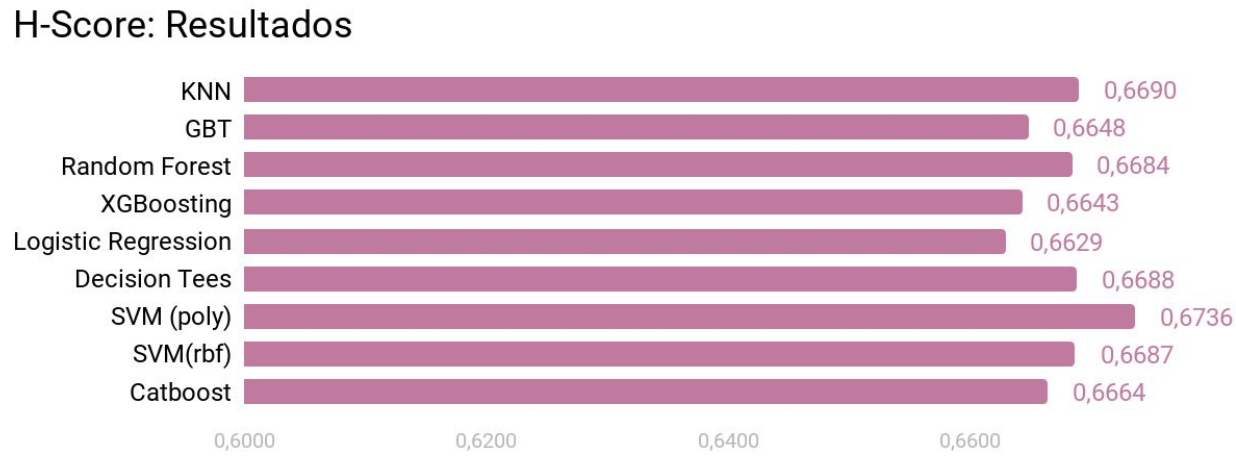


No obstante nos hemos dado cuenta que nuestro F1 de SVM empieza a tener datos desbalanceados entre Recall y Precision lo que potencialmente puede conllevar gran cantidad de Falsos Positivos y bajo Accuracy. Intentando buscar un compromiso entre las 2 métricas vistas y esperando que Falsos Positivos o negativos se incorporen a su clase correcta, se ha creado un score llamado H-Score que no es más que la media armónica del Accuracy y el F1-Score y que se va a intentar maximizar

H-Score: (link a Github)

Se han vuelto a lanzar los modelos intentando encontrar el mejor H-Score, esperando un mejor balance entre Recall y Precision y que esto no perjudique los Scores de F1.

Estas optimizaciones han devuelto resultados no muy diferentes a los que se estaban consiguiendo hasta ahora aunque sí es verdad que algunos modelos que habían devuelto predicciones desbalanceadas entre Recall y Precision consiguen balancearlos.



El modelo con mejores resultados vs el test set vuelve a ser el SVM con kernel polinómico con unos scores idénticos a los conseguidos para F1-Score. El mejor modelo en validación cruzada también ha sido un SVM pero con kernel RBF con buenos resultados en todos los indicadores. También se observa que las diferencias entre modelos son mínimas.

Viendo que los resultados no mejoran los resultados que ya se tenían se decide continuar con las predicciones del modelo F1-Score

Comparativo Accuracy, F1-Score y H-Score:

Accuracy(KNN)

		Real	
		Bad	Good
Predict	Bad	659	317
	Good	341	683

F1 (SVM-Poly) ✔

		Real	
		Bad	Good
Predict	Bad	709	375
	Good	291	625

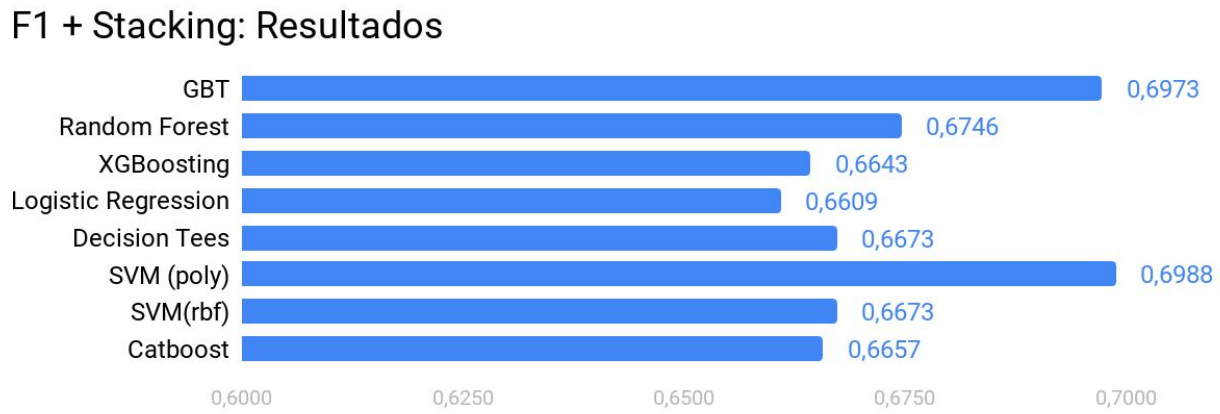
H-Score(SVM-RBF)

		Real	
		Bad	Good
Predict	Bad	670	333
	Good	330	667

5.3 Modelos con Stacking

Una vez optimizados los modelos y descartados los resultados de los modelos de Accuracy y H-Score, se intentan mejorar los resultados mediante Stacking del modelo F1-Score, añadiendo las predicciones de los modelos a una nueva selección aleatoria de nuestro data frame original.

Los modelos F1 con Stacking han mejorado de manera importante en algunos algoritmos al añadir las predicciones de los modelos, aproximándose en algunos casos al 70%. Sin embargo nos encontramos en la situación no deseada y comentada anteriormente, ya que el aumento en F1 se basa en aumentar de manera considerable el Recall llevándola a más del 90% pero bajando la Precision a menos de 60% lo que genera un problema de Falsos Positivos.



Destacan 2 modelos modelos muy por encima del resto:

- GBT: Es el modelo con mejor score en validación cruzada y tiene un F1-Score vs Test Set de 69.7% muy cimentado en una alta Recall (91.8%) aunque Accuracy y Precision caen por debajo del 60%.

- SVM: Tiene resultados muy parecidos y un F1-Score algo más alto vs el test set. La diferencia entre Recall y Precision es más extrema y el Accuracy es incluso más bajo que en el modelo GBT.

El modelo con GBT parece el menos extremo aunque de todas formas no parece que ninguno de los 2 sea mejor que el modelo original F1 con SVM.

Comparativo F1-Score (SVM) vs F1 + Stacking (GBT):

F1-Score (SVM) ✔️

		Real	
		Bad	Good
Predict	Bad	**709**	375
	Good	291	**625**

F1 + Stacking (GBT)

		Real	
		Bad	Good
Predict	Bad	**934**	**727**
	Good	84	255

5.4 Modelo Elegido

Después de múltiples pruebas, parece que el mejor modelo es el **SVM original optimizando F1-Score**. No es el modelo que haya dado mayor F1 de todos los probados pero sí que es el más equilibrado entre todos los testeados.

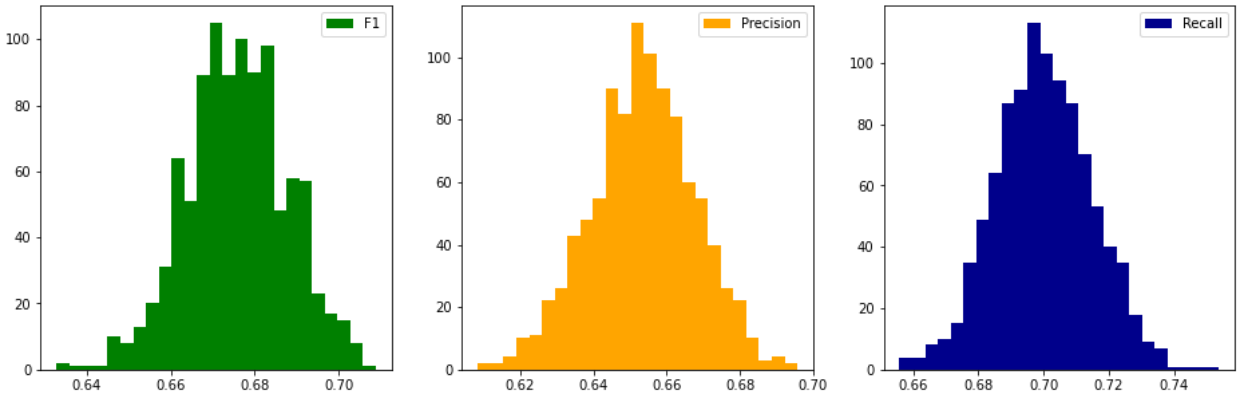
5.5 Intervalo de confianza de las predicciones

Finalmente, complementamos todas estas operaciones estableciendo un intervalo de confianza a las predicciones de las principales métricas del modelo elegido (Accuracy, Recall y Precision).

Para encontrar estos intervalos de confianza se han cogido muestras del test set haciendo Bootstrapping y se han lanzado 1000 simulaciones. A continuación se ha cogido el percentil 0.5 y el percentil 99.5 para encontrar los límites superior e inferior de cada métrica y asegurarnos que nuestra predicción está entre este intervalo con un **99% de probabilidad**.

A parte de los datos de los intervalos también graficamos los histogramas de estas simulaciones para comprobar que siguen una distribución normal y validar los datos, (link a Github).

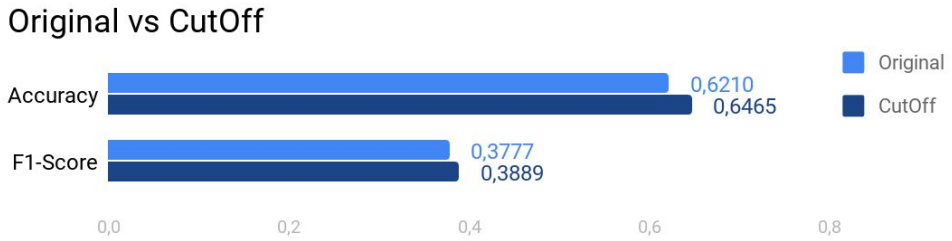
	Intervalo Inferior	Mediana	Intervalo Superior
Accuracy	0.6320	0.6590	0.6835
F1-Score	**0.6454**	**0.6758**	**0.7038**
Recall	0.6607	0.6994	0.7370
Precision	0.6267	0.6536	0.6888


F1
Precision
Recall

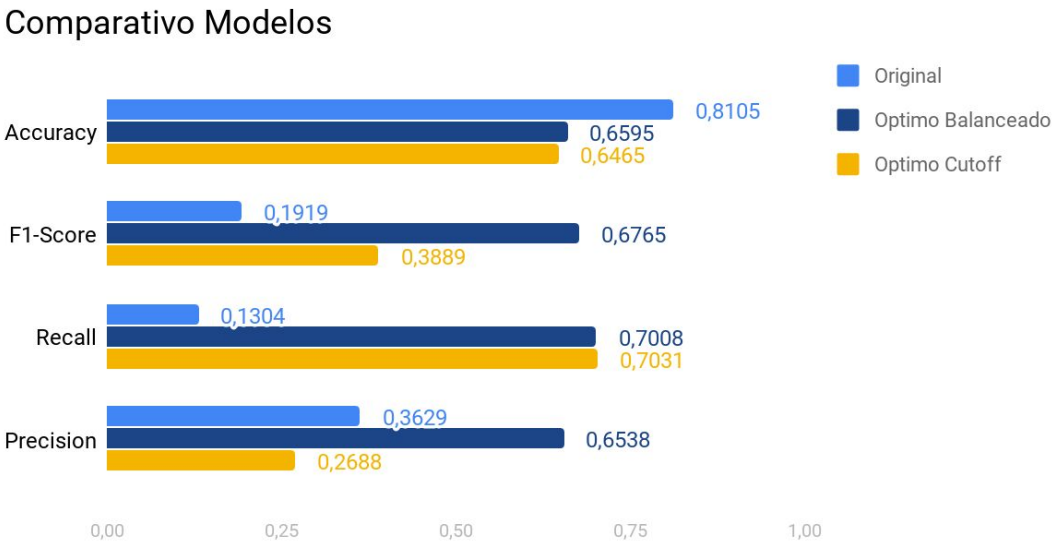
6. Resultados en Dataset Original

Los resultados finales, sin embargo, no son los analizados en el último punto, si no que debemos comprobar los resultados del modelo elegido en el dataset original ya que nuestra realidad está formada por un dataset original desbalanceado.

Al estar importando un modelo realizado con un balanceo muy distinto al que se está usando como dataset final se ha analizado el efecto de cambiar ligeramente el cutoff para la clasificación de clases con las probabilidades del modelo. En nuestro caso, dejar el **cutoff en 0.505** nos permite optimizar el F1-Score y aumentarlo 1 punto respecto el modelo original. A su vez también conseguimos mejorar el Accuracy en más de 2 puntos.



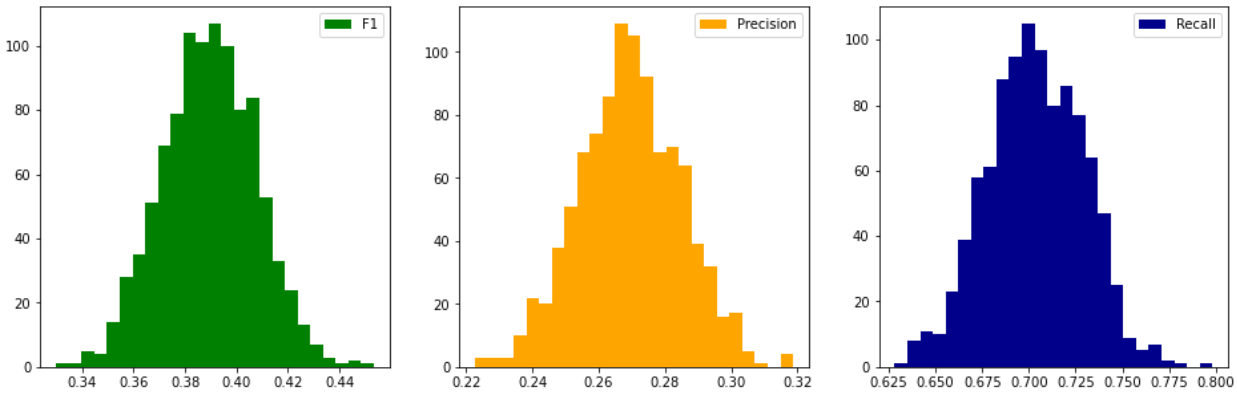
Al aplicar el SVM que maximiza F1 al dataset original los resultados empeoran respecto los últimos modelos con el dataset balanceado, pero mejoran y hasta **doblan el F1 de los primeros modelos**. Mientras que la Precision es bajo tanto en los modelos iniciales como en el final, si que se consigue subir el Recall del modelo desde un 13% hasta un 70% lo que es destacable.



Al igual que hemos hicimos con las predicciones en el dataset balanceado volvemos a calcular unos intervalos de confianza para nuestras métricas más relevantes en el dataset sin balancear.

En este caso el F1-Score que es nuestra métrica de interés queda entre un 34% y un 43%. Destaca el alto valor de el Recall que se estima entre un 64% y 77% (link a Github).

	Intervalo Inferior	Mediana	Intervalo Superior
Accuracy	0.6215	0.6465	0.6760
F1-Score	**0.3432**	**0.3895**	**0.4354**
Recall	0.6399	0.7030	0.7667
Precision	0.2287	0.2696	0.3069



Como output final, se espera que la clasificación de futuras reservas siga la siguiente distribución. Observamos que las reservas malas prácticamente no escaparían de ser supervisadas aunque con el coste de supervigilar reservas buenas en exceso, lo que indica mucho potencial de mejora en los modelos

		Real	
		Bad	Good
Predicted	Bad	11,5%	33,4%
	Good	4,5%	50.6%

7. Siguientes Pasos

La idea de predecir una potencial nota negativa, antes incluso de que el huésped llegue al hotel, abre una puerta a poder evitar que esta nota se produzca, siempre que se sepa qué es lo que puede provocarla.

Para este cometido se ha intentado modelizar si una reserva iba a dar una nota negativa en cada una de las categorías definidas en el proceso de NLP, en el que se analizaron los comentarios positivos y negativos.

Sin embargo los resultados preliminares usando métricas y modelos similares a los anteriores han arrojado resultados muy alejados a unos mínimos exigibles y esto nos reafirma en la falta de información (falta de features y no de casos).

8. Conclusiones generales

Después de una gran cantidad de pasos y después de optimizar gran variedad de modelos para distintas métricas nos hemos encontrado diversos patrones que se han ido repitiendo.

- Hemos estado trabajando con unos modelos cuyos resultados son muy débiles. A pesar de tener grandes cantidades de información, los patrones son escasos y en los momentos que una variable ha conseguido discriminar ha sido por hacerlo de manera muy general.

- Trabajar con unos modelos tan débiles nos ha conducido a situaciones inesperadas como que el mejor modelo para la Accuracy fuera un KNN con una K muy alta, o que los modelos para predecir el motivo de queja no se diferencian casi de lo que hubiera sido una decisión aleatoria.

- A pesar de haber elegido el modelo que maximiza F1-Score con SVM los resultados de los modelos se han ido repitiendo de manera muy similar independientemente de cuál fuera la métrica a maximizar o el algoritmo usado.

- Sí que se han ido consiguiendo pequeñas mejoras respecto al punto inicial, pero en ningún momento se ha conseguido un salto destacable. Al final se han ido seleccionando modelos óptimos que solo conseguían milésimas en los Scores respecto a los anteriores.

- Parece que las variables que consiguen aportar más a los modelos son todas las relacionadas con la reserva y el cliente, pero también son los aspectos con un menor número de variables.

- Las variables introducidas sobre el Hotel no consiguen explicar mucho y se repiten en cada reserva para el mismo hotel con lo que no son excesivamente discriminantes.

Finalmente, consideramos que los modelos se podrían mejorar si tuviéramos información más relevante y detallada referente al cliente. Variables como por ejemplo: el histórico de valoraciones del cliente cuando opina sobre un hotel, solicitudes del cliente en el momento de realizar la reserva, tipo de establecimientos que suele reservar, datos provientes de redes sociales con los que conocer gustos y tendencias de la persona, sexo, edad, entre otros aspectos podrían ayudar a mejorar los resultados obtenidos hasta ahora.

9. Apéndice

Links de acceso a Booking_Analysis

Creación de variables

- Create features 1 (recodes)
- Create features 2 (nr close landmarks)
- Create features 3 (distance to city center)
- Create features 4 (Distance to airport)
- Create features 5 (Distance to train)
- Create features 6 (Scrapping info + Length_N)
- Create features 7 (NLP Scores by Hotel Category)
- Create features 8 (estimate Price NA - RF)

Acciones previas para la creación de variables

- Features work – find top 10 landmarks by city
- Features work - NLP Prepare & Lemmatize
- Features work - Web Scraping

Primera exploración del dataset y sus variables

- Exploratory -EDA
- Exploratory - General

Preparación del dataset y elaboración de los modelos de clasificación

- Model 01_First Models