Universidad Internacional de La Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Análisis y Visualización de Datos Masivos

Sistema de recomendación para nuevos usuarios de Airbnb

Trabajo Fin de Máster

presentado por: Pardo Cuesta, Raquel

Director: García García, Óscar

Ciudad: Bilbao

Fecha: 23 de julio de 2019

”Every expert started out as a beginner"

”Todos los expertos empezaron como principiantes"

- Helen Hayes

Abstract

Product recommendations for new users of a company, problem known as cold start, suppose new artificial intelligence methods to be able to carry out the recommendation of a product with a certain proportion of correct answers. The aim of this work is the treatment of an Airbnb user's database with machine learning techniques to achieve a recommendation system. This recommendation system will suggest, among a number of possible destinations, the most likely choice taking into account the user's activity into the website, his/her age and gender and other factors. To do so, several automatic learning techniques are applied, both in Big Data and conventional environments, and the best predictive results are going to be used to elaborate the recommendation system.

Keywords: Airbnb, machine learning, recommendation systems, artificial intelligence, applied statistic.

Resumen

Las recomendaciones de productos para nuevos usuarios de una compañía, problema conocido como *cold start*, supone nuevos métodos de inteligencia artificial para poder llevar a cabo la recomendación de un producto con cierta proporción de aciertos. El objetivo de este trabajo es el tratamiento de una base de datos de usuarios de Airbnb con técnicas de *machine learning* para lograr un sistema de recomendación. Este sistema de recomendación sugerirá, entre un número determinado de destinos, la elección más probable en función de la actividad del usuario con la página web, su edad y género y otros factores. Para ello, se aplicarán varias técnicas de aprendizaje automático, tanto en entornos Big Data como en los convencionales, y se utilizarán los de mejores resultados para elaborar el sistema de recomendación.

Palabras Clave: Airbnb, *machine learning*, sistemas de recomendación, inteligencia artificial, estadística aplicada.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Lista de acrónimos

AUC	Area Under the Curve
BD	Big Data
BI	Business Intelligence
DCG	Discounted Cumulative Gain
ETL	Extract Transform Load
FN	False Negative
FP	False Positive
GB	Gradient Boosting
GBDT	Gradient Boosted Decision Trees
GLM	Generalized Linear Models
HDFS	Hadoop Distributed File System
NDCG	Normalized Discounted Cumulative Gain
NN	Neural Network
NLP	Natural Language Procesing
OvA	One Versus All
RF	Random Forest
ROC	Receiver Operating Characteristic
SQL	Structured Query Language
SVM	Support Vector Machine
TF –IDF	Term Frequency –Inverse Term Frequency
TFM	Trabajo de Fin de Máster
TN	True Negative
TP	True Positive

1. Introducción

Históricamente, ha habido una tendencia generalizada de recopilar datos (Burkov, 1956), ya sea para hacer un inventario de los productos restantes de un almacén (Retail, 2019), hacer censos de la población (Srinivasan, 2017), comprobar el buen funcionamiento de un proceso industrial (Srinivasan, 2017) o hacer un histórico del proceso de devolución de un préstamo (AdelaideX, 2018). A pesar de que pueda parecer que, ante esta situación, la única dificultad es la del análisis de los datos, si al conjunto de datos se le añaden atributos como un gran **v**olumen de datos, **v**ariedad del contenido, **v**elocidad de almacenamiento y de actualización y la falta de **v**eracidad, se da lugar a un entorno *Big Data* (Rezaul y Alla, 2017). Cada una de estas palabras que empiezan por **v** aporta por su cuenta nuevas carácterísticas a los conjuntos de datos pero, cuando aparecen a la vez, es el momento en el que dan lugar al *Big Data* (AdelaideX, 2018). Por su cuenta, cada término aporta la siguiente información:

1. **V**olumen: Hace referencia a la gran cantidad de datos con los que se tiene que lidiar para extraer el conocimiento que reside en los datos (AdelaideX, 2018). Sensores, cámaras de videovigilancia, transacciones a nivel global e interacciones con una red social son ejemplos en los que en poco tiempo se llegan a generar mucho datos, llegando 2,3 trillones de gigabytes de información al día (IBM, 2018).
2. **V**elocidad: Representa la rapidez en la que se cargan, eliminan o actualizan los datos, siendo a veces necesario el análisis en tiempo real para mayor eficacia (Firican, 2019). Como ejemplos que corroboren la velocidad de los datos, hay un terabyte de información bursatil por cada sesión (generalemente compuesta de muchas interacciones) (IBM, 2018), en un coche hay más de 100 sensores que recogen información sobre el estado del combustible y otro tanto para la presión de los neumáticos (IBM, 2018) y, por último, tratar con los datos en tiempo real en el proceso de construcción de una pieza crítica para aviones y/o cohétes puede servir para mejorar los recursos de fabricación, haciendo un seguimiento (Cheng, Chen, Sun, Zhang y Tao, 2018).
3. **V**ariedad: Los datos ya no tienen por qué ser estructurados, ni provenir de una misma fuente y ni estar en un mismo formato (AdelaideX, 2018). Es por ello que, entre otros, el estudio del fraude de una empresa se puede hacer con datos estructurados, semi-estructurados o no estructurados al ser lo primordial la calidad del dato frente a la estructura (Cheng y col., 2018), las estaciones meteorlógicas emplean sensores, videos, cámaras de infrarrojos e históricos de datos para hacer más precisas sus predicciones

(Chouksey y Chauhan, 2017) y las redes sociales guardan al mes más de 30 billones de datos de contenido en el caso de Facebook (IBM, 2018) y 400 millones de tweet en cuanto a Twitter (IBM, 2018) donde cada interacción presenta un formato y un comportamiento diferente.

4. **V**eracidad: Se refiere a la incertidumbre que causan los datos debido a que los datos pueden contener ruido o estar incompletos, entre otras opciones (AdelaideX, 2018). Debido a la falta de franqueza, por ejemplo uno de cada tres empresarios duda de la veracidad de los datos con los que está tratando (IBM, 2018), el 80 % del tiempo necesario para un análisis de datos se destina a la preparación de los datos (limpieza, filtrado, transformaciones, etc.) (AdelaideX, 2018), y a los Estados Unidos la pobreza de lo datos recabados les cuesta, al año, 3,1 billones de dólares (IBM, 2018).

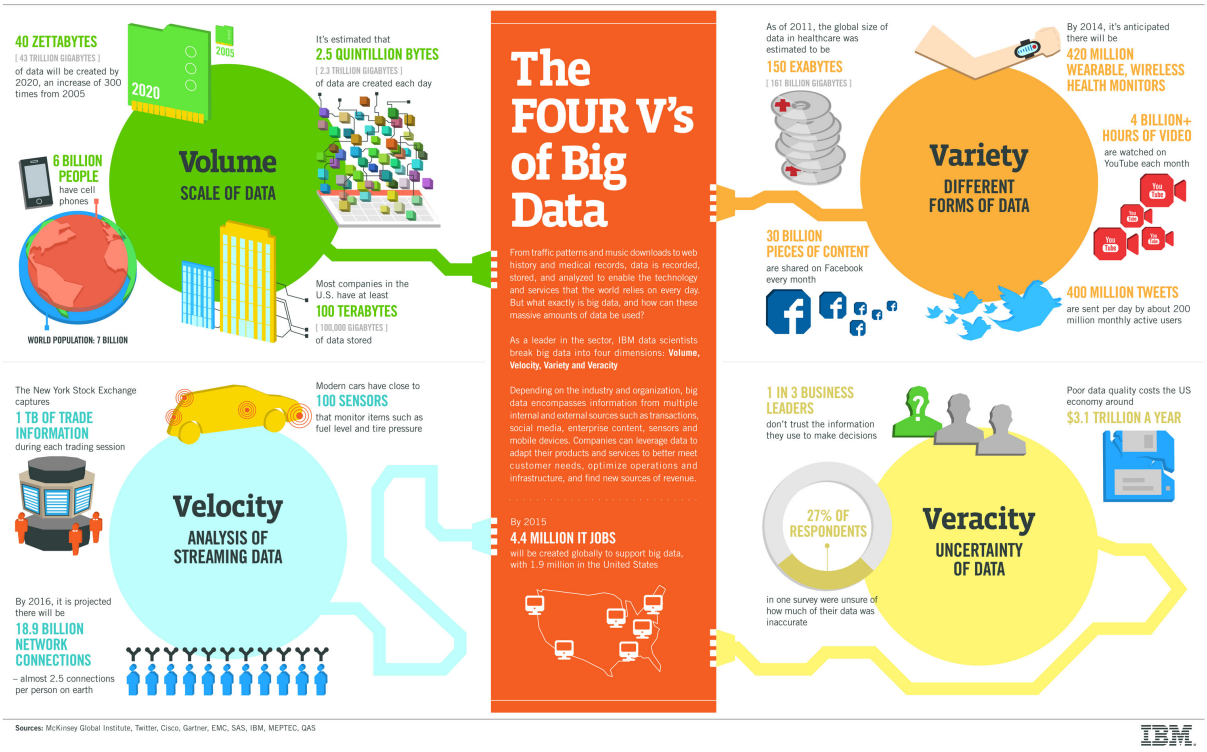

Figura 1: Las 4**V**s del *Big Data* junto a hechos de cada una de ellas (IBM, 2018).

De este modo quedan definidas las 4**V**s del *Big Data*, existiendo fuentes que amplían esta definición añadiendo hasta un total de 10 **V**s en algunos casos (AdelaideX, 2018), (Firican, 2019).

A pesar de las nuevas propiedades de este tipo de datos, en un entorno *Big Data* se suelen seguir empleando las mismas técnicas de análisis de datos que en el entorno convencional: regresiones logísticas, árboles de decisión, procesamiento del lenguaje natural, redes

neuronales, etc (Burkov, 1956). La diferencia se encuentra en que estas técnicas se usan con herramientas que optimizan su rendimiento (Rezaul y Alla, 2017). Debido al potencial que presentan para las empresa los conocimientos que se pueden extraer de los datos y las herramientas de las que se dispone para ello, muchas compañías están interesadas en actualizar sus sistemas y añadir esta forma de procesar grandes cantidades de datos (Adomavicius y Tuzhilin, 2005). Por otro lado, otras empresas como Airbnb ya llevan unos años incorporando estos métodos de procesamiento de información (Grbovic y Cheng, 2018), (Grbovic, Cheng y col., 2018), (Haldar y col., 2018).

Airbnb es una empresa de alquiler de apartamentos vacacionales que fue fundada en 2008 en San Francisco, EE.UU (Airbnb, 2019b). Su finalidad es la de poner en contacto, a cambio de una pequeña comisión, al propietario o anfitrión de la propiedad con el viajero o huésped interesado en alojarse en ella. Al igual que otras empresas, Airbnb también está interesada en extraer toda la información posible de los datos para hacer recomendaciones a sus usuarios, personalizarles las búsquedas, predecir los intereses de los huéspedes, mejorar el posicionamiento de los apartamentos de los anfitriones en el buscador de Airbnb y hacer recomendaciones de experiencias y restaurantes a los huéspedes en función de su próximo destino vacacional, gustos e interesas, entre otras acciones (Grbovic, Cheng y col., 2018).

Las recomendaciones de las que se ha hablado se pueden hacer, por ejemplo, utilizando técnicas de inteligencia artificial, donde los métodos que se usan se engloban dentro del concepto *sistemas de recomendación* (Gorakala y Usuelli, 2015). Hay distintos tipos de sistemas de recomendación: de filtrado colaborativo (Adomavicius y Tuzhilin, 2005), demográficos (Montaner, López y de la Rosa, 2003), basados en contenido (Adomavicius y Tuzhilin, 2005), basados en conocimiento (Gorakala y Usuelli, 2015), híbridos (Montaner y col., 2003), etc. quedando la utilidad y empleabilidad de cada uno de ellos ligada al conjunto de datos con los que se trabaja (Rezaul y Alla, 2017). Muchas compañías han apostado por incorporar las recomendaciones en su forma de gestionar el contenido, como es el caso de Amazon en 2003 con su sistema de filtrado colaborativo *ítem-a-ítem* (Linden, Smith y York, 2003), Netflix en 2012 con la incorporación de un sistema de filtrado colabotarivo (Amatriain y Basilico, 2012a), (Amatriain y Basilico, 2012b), Youtube en 2016 con un sistema de filtrado híbrido entre los sistemas de recomendación colaborativos y los basados en contenido (Covington, Adams y Sargin, 2016) y Airbnb en 2018 con un sistema de filtrado colaborativo basado en *deep-learning* (Grbovic y Cheng, 2018), entre otros.

En resumidas cuentas, los sistemas de recomendación satisfacen muchas de las nece-

sidades que hoy en día tienen las empresas en cuanto a la personalización del contenido al usuario (Gorakala y Usuelli, 2015). A pesar de ello, cuando un nuevo producto y/o usuario entran en la base de datos, conocido como *cold start*, es el momento en el que gran parte de los sistemas de recomendación presentan sus carencias (Adomavicius y Tuzhilin, 2005). Por ese motivo, este Trabajo de Fin de Máster tiene como objetivo la creación de un sistema de recomendación para nuevos usuarios de Airbnb. Con este fin se va a estudiar el comportamiento que tuvieron 135.484 primeros usuarios junto a las características de estos usuarios para así poder clasificar, con este conocimiento, a un posible nuevo usuario de la siguiente forma: primero se va a predecir si, en función de sus interacciones con la web de Airbnb, el usuario posiblemente acabe reservando o no. Si es clasificado como un usuario que no va reservar, ahí concluye el sistema de recomendación. Si, por el contrario, el algoritmo predice que el usuario va a reservar, es entonces cuando se le recomienda, en base al resultado de un modelo evaluado, un destino de entre 10 posibles países.

Este problema tiene varios retos a resolver: diferenciar entre los usuarios que son huéspedes y los anfitriones, enriquecer la base de datos en la medida de lo posible para tener tanto datos de calidad como buenos resultados predictivos y diferenciar la solución que se propone en esta memoria de los estudios ya realizados en una página de concursos llamada Kaggle. Kaggle (Kaggle, 2019) es una página de competiciones y una comunidad para científicos de datos y *machine learners* donde, entre otras cosas, se les ofrece formación y concursos a toda persona interesada. Hace 3 años, Kaggle (Kaggle, 2019) lanzó una competición para predecir el primer destino de los nuevos usuarios de Airbnb con la misma base de datos de esta memoria. Para darle novedad a este trabajo y mejorarlo respecto a las propuestas del concurso, se modificará la base de datos original eliminando varias variables predictivas, complicando el trabajo. Asimismo, la estructura, las herramientas y la forma de hacerle frente al problema también van a ser diferentes, optimizando su uso y el funcionamiento del sistema en conjunto, a parte de ofrecer otra forma diferente de resolver al problema.

Completando la parte innovadora del trabajo, siendo la memoria autocontenida, no sólo se va complicar el problema reduciendo la dimensionalidad de una de las fuentes de datos, sino que también se va a crear un sistema de recomendación original y propio para resolver la dificultad de hacer una recomendación a un nuevo usuario. Además, se van a llevar a cabo otras técnicas y procesos de preparación de los datos que los desarrollados en la competición y se le va a dar un doble enfoque al problema: una visión de trabajo en un entorno de *Business Intelligence* y otro enfoque en un entorno de *Big Data*. Muchas veces el límite de tamaño que

separa una base de datos donde es mejor usar herramientas de *Big Data* de las que no es muy incierto (AdelaideX, 2018). Este motivo ha acarreado que, en esta memoria, por un lado se vayan a usar herramientas convencionales, es decir, herramientas que no son de *Big Data* como Python y R para lograr el objetivo (entorno de *Business Intelligence*) y, por otro lado, que se vayan a usar herramientas de *Big Data* como Spark con el mismo fin (entorno de *Big Data*). De este modo se va decidir qué herramienta presenta mejores resultados (tiempos de ejecución, facilidad de uso, resultados en los algoritmos, etc.) y qué entorno aconsejar con estos datos.

En cuanto a la estructura del Trabajo de Fin de Máster, a continuación se va a contextualizar al lector con el estado del arte, Sección 2. Después se van a recordar los objetivos y la metodología de esta memoria en la Sección 3 y luego se va a entrar de lleno en la Sección 4 con el procesamiento de los datos. En esta última sección, tras un análisis exploratorio de los datos y limpieza de los mismos, se va a seleccionar la base de datos sobre la que aplicar los algoritmos de *machine learning* y se aplicarán varios algoritmos en la Sección 4. Finalmente, se estudiarán los resultados obtenidos en la Sección 6, se alcanzará unas conclusiones finales en la Sección 7 y se comentarán las líneas de futuro por la que se puede seguir este trabajo, Sección 8. Además, en el anexo se puede encontrar el código empleado, Sección 9.

2. Estado del arte

Para comprender la solución que se encuentra al final del TFM, es necesario presentar previamente el proceso que se ha seguido hasta este objetivo. Al tratarse de una memoria autocontenida dirigida a un público con conocimientos básicos de *machine learning*, a continuación se van a tratar, primero, los principales sistemas de recomendación de productos que existen junto a sus carencias. Después, se van a analizar los avances que, a día de esta memoria, se han hecho por y para Airbnb en el tema de la personalización del contenido y la recomendación. Finalmente, se planteará la estrategia a seguir para hacer la recomendación de un primer destino a los nuevos usuarios de Airbnb y que se desarrollará en las siguientes secciones de esta memoria.

2.1. Sistemas de recomendación

Todo comenzó en 1969, cuando nació una nueva herramienta que facilitaba la comunicación entre ordenadores: Internet (Abbate, 1999). A raíz de ese momento, no sólo apareció en el mercado un nuevo instrumento para compartir contenido, sino que fue modificando la forma de almacenar, acceder y buscar información paulatinamente. En la actualidad, Internet es capaz de facilitarle al usuario información a cerca de casi cualquier cosa en cuestión de milisegundos (Abbate, 1999). Es más, debido a la gran cantidad de contenido que puede proporcionar sobre casi todo, es necesario tener alguna forma de organizar el contenido y facilitárselo al usuario, estando en auge las técnicas que permiten clasificar, ordenar, asociar, filtrar o recomendar la información (Ricci, Rokach y Shapira, 2011).

Los sistemas de recomendación proporcionan un medio para asociar a los usuarios con el contenido, personalizándoles los resultados y sugiriéndoles información que el sistema cree que les puede interesar (Montaner y col., 2003). Los datos de los que se vale para ello pueden ser los proporcionados por los usuarios, ya sea de manera directa como completando un formulario o de manera indirecta con la huella digital, o pueden ser datos sobre los productos, por ejemplo el número de ventas de cada uno, su descripción y el perfil de cliente al que está dirigido (Rezaul y Alla, 2017). Además, algunos sistemas deciden hacer la recomendación basándose en la similitud de los usuarios (Gorakala y Usuelli, 2015) mientras que otros sugieren productos similares a los consultados y/o adquiridos (Gorakala y Usuelli, 2015).

Antes de entrar más en detalle sobre los principales sistemas de recomendación, es

necesario saber que, en todo sistema de recomendación, el primer paso que se tiene que dar para poder hacer las recomendaciones lo tiene que dar el usuario (Montaner y col., 2003). Para ello, el usuario debe registrarse en la compañía completando un formulario para tener un perfil de usuario. Esto es imprescindible porque muchas de los sistemas se apoyan en él para hacer las sugerencias (Montaner y col., 2003). Además, es necesario que el perfil del usuario aporte la mayor cantidad de información posible sobre el cliente para poder saber más sobre sus hábitos, gustos y patrones de consumo y llevar a cabo recomendaciones más precisas (Ricci y col., 2011).

Una vez se tienen usuarios registrados conun mínimo de actividad, el sistema de recomendación puede empezar a funcionar y hacer sugerencias. A continuación, se describen los principales sistemas de recomendación de productos.

2.1.1. Sistema de recomendación basado en filtrado demográfico

Este tipo de sistema de recomendación se vale de las descripciones proporcionadas por los usuarios en su perfil para establecer relaciones entre el producto y el tipo de persona al que le puede interesar (Montaner y col., 2003). Para ello, los usuarios son agrupados usando la información que han aportado en su perfil en base a descripciones estereotipadas (edad, sexo, ubicación, rango social, etcétera). Asimismo, en muchas ocasiones también se les ofrece cuestionarios extras en el registro para recabar más información personal y poder obtener resultados más precisos en la clasificación.

Este sistema ha sido implantado por SurveyMonkey (SurveyMonkey, 2017) y Ed Tech (Tech, 2013). A pesar de que su funcionamiento parezca útil, de forma práctica mayormente se usa en combinación de otros sistemas de recomendación que se estudiarán después, como es el caso de Amazon (Amazon, 2001).

Las carencias de los sistemas basados en filtrado demográfico residen en que ofrecen siempre el mismo tipo de contenido a las personas con semejantes características, es decir, si el interés de un usuario ha cambiado el sistema no es capaz de adaptarse, ofreciéndole todavía el contenido habitual. Además, no es capaz de ofrecer recomendaciones individualizadas y puede que no todos los individuos que cumplan unas mismas características demográficas encajen en un estereotipo en concreto (Montaner y col., 2003).

2.1.2. Sistema de recomendación basado en contenido

Los sistemas de recomendación basados en contenido, conocidos también como de filtrado cognitivo, se valen de la comparación entre la metainformación de los productos y la información del perfil de los usuarios para hacer recomendaciones, pudiendo completar la información sobre el usuario añadiendo sus preferencias históricas al cruce (Rezaul y Alla, 2017). Esto es, hacen sugerencias en función de los productos que a este mismo usuario le han gustado en un pasado o que encajan con los gustos indicados en su perfil. Para que el sistema de recomendación funcione, cada producto se representa con una descripción o con una serie de palabras sueltas que expresan correctamente en qué consiste el producto (Gorakala y Usuelli, 2015). El perfil de usuario consta, en un principio, de la información aportada por el usuario y se va completando a la vez que consulta o consume productos con las palabras de la descripción de los mismos. De este modo se consigue tratar de forma individualizada a cada usuario (Ricci y col., 2011).

Este sistema de recomendación es uno de los más empleados en la actualidad, siendo ejemplo de ello empresas como Webmate (Webmate, 2019), ACR News (News, 2014) e Infofinder (Transfinder, 2019). Además, para obtener buenos resultados de precisión no es necesario un gran conjunto de datos. Aparte, como medidas de similitud entre el producto y el usuario se suele utilizar la ponderación TF-IDF (Term Frequency-Inverse Term Frequency) y otras técnicas de procesamiento del lenguaje natural (Gorakala y Usuelli, 2015).

Aunque sea capaz de ofrecer buenas recomendaciones tanto para nuevos productos como para nuevos usuarios (si éstos completan su perfil con toda la información necesaria), sólo es capaz de hacer recomendaciones objetivas, dejando de lado otros factores subjetivos que pueden influir en la opinión del usuario (la veracidad de los especificado en la descripción, si hay algún fallo técnico al acceder al producto, la opinión de cada usuario, etc.) (Gorakala y Usuelli, 2015). Asimismo, las recomendaciones son tan individualizadas que al usuario únicamente se le recomienda lo que le ha gustado en un pasado o lo que ha puesto en su perfil que le puede gustar, imposibilitando que se le hagan sugerencias sobre otro contenido que ofrezca la empresa y haciendo que la recomendación pierda precisión en este aspecto (Ricci y col., 2011).

2.1.3. Sistema de recomendación basado en filtrado colaborativo

Los sistemas de recomendación de filtrado colaborativo, también conocidos como de filtrado social, asocian personas con intereses pasados similares y toman esta asociación como punto de partida para hacer recomendaciones (Gorakala y Usuelli, 2015). Dicho de otra forma, la base de este tipo de recomendación reside en que, si dos usuarios tenían el mismo interés en el pasado, es muy probable que también tengan gustos similares en el futuro (Rezaul y Alla, 2017). Esto se debe a que, cuando una persona quiere ver una película y está indecisa, suele acudir en busca de sugerencias a la gente de su entorno que sabe que tiene intereses o gustos parecidos a los suyos. Busca consejo en estos individuos porque la persona indecisa sabe que la probabilidad de que acierten en su recomendación es mayor si consulta a gente que se le parezca. Pues bien, estos sistemas emulan la respuesta que le podría dar al usuario titubeante una persona física. Para hacer las sugerencias se tienen que abordar dificultades como las de cómo establecer una medida de similitud entre los productos, cómo establecer la medida de similitud entre los usuarios y cómo lidiar con los nuevos productos y usuarios (Gorakala y Usuelli, 2015). Es por ello que utiliza datos estadísticos sobre las evaluaciones de los productos y los patrones de comportamiento de los usuarios para establecer la similitud, usando medidas de similitud como las medidas de correlación o la similitud del coseno (Rezaul y Alla, 2017).

Este sistema de recomendación es el más empleado en la actualidad y se caracteriza por el hecho de que cuanto mayor es la base de datos más precisos son los resultados que se logran (Ricci y col., 2011). Es por ello que empresas como Amazon (Linden y col., 2003), Netflix (Amatriain y Basilico, 2012a), (Amatriain y Basilico, 2012b), Youtube (Davidson y col., 2010), LinkedIn (Wu, Shah, Choi, Tiwari y Posse, 2014) y Airbnb (Grbovic y Cheng, 2018) han apostado por este sistema en algún momento de su historia o sigue apostando todavía para ciertas necesidades.

A pesar de ello, presenta, entre otros, problemas para hacer sugerencias cuando se incorpora tanto un nuevo producto como un nuevo usuario al no tener forma de compararlos con los registros que posee la compañía (Ricci y col., 2011). Por otro lado, como ya se ha comentado, los mejores resultados se obtienen cuanto mayor es el número de datos disponibles. Pero reunir, por ejemplo, 100 valoraciones de distintos usuarios por cada producto de la web es una tarea costosa y a menudo imposible. Además, si la web consta de pocos usuarios activos en comparación con el resto de información que posee, se corre el riesgo de que las

recomendaciones lleguen a ser poco variadas (Montaner y col., 2003).

2.1.4. Sistema de recomendación híbrido

Como se ha visto hasta ahora, es difícil encontrar un sistema de recomendación que responda bien ante todas las situaciones. Una técnica muy común es la de combinar varias de las técnicas existentes en los sistemas de recomendación, dando lugar a los sistemas híbridos (Ricci y col., 2011) como se puede ver en la Figura 2.

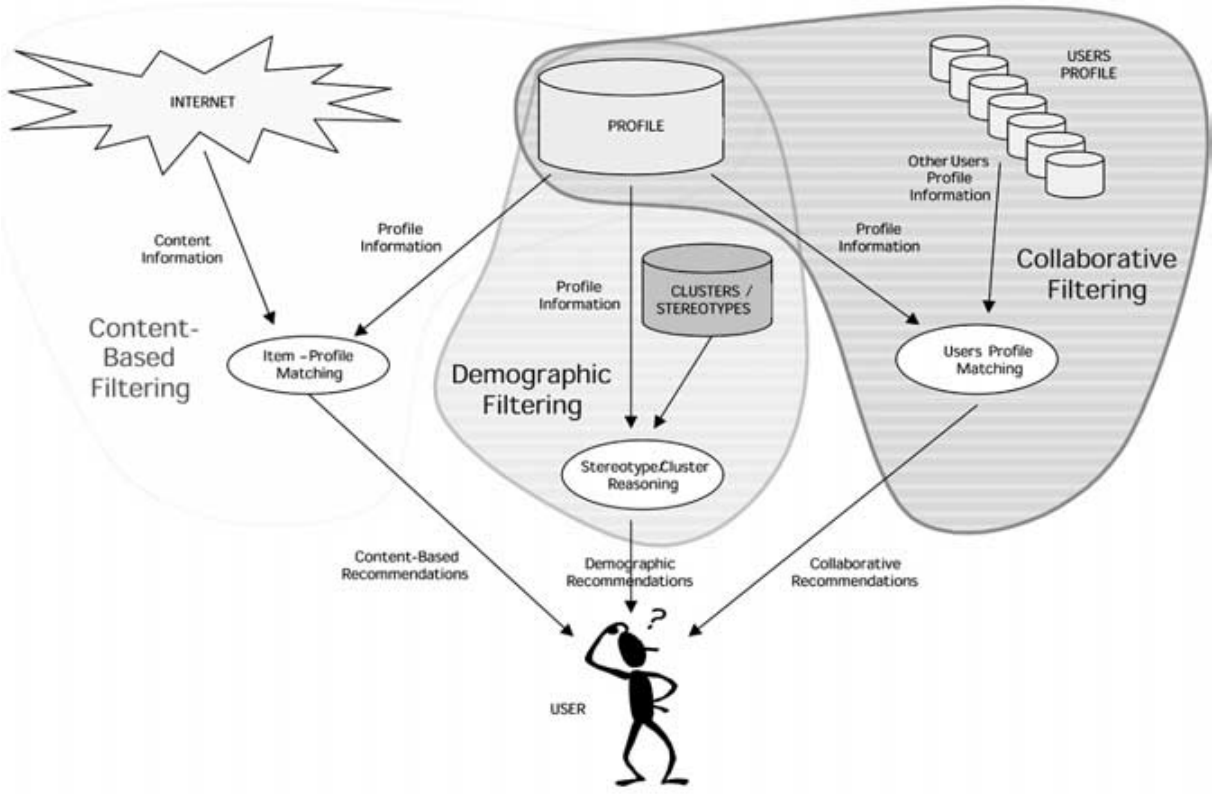

Figura 2: Ejemplo de la estructura de un sistema de recomendación híbrido (Montaner, López y de la Rosa, 2003).

Uno de los principales problemas que los sistemas híbridos pretenden abordar es la de hacer una recomendación a los nuevos usuarios (Montaner y col., 2003). La combinación entre los sistemas basados en contenido y los basados en filtrado colaborativo darían lugar a un tipo de sistema híbrido que, ante un nuevo producto, podría medir la similitud entre productos, con evaluaciones y descripciones parecidas, para poderle asignar una probabilidad de ser sugerido (Ricci y col., 2011). Por otro lado, solicitando a los nuevos usuarios que rellenen un formulario complementario al inicial, las sugerencias se llevarán a cabo usando el sistema basado en contenido. Además, a largo plazo se pueden incorporar las opiniones del resto de

usuarios para dotarlo de subjetividad y mejorar en las recomendaciones (Rezaul y Alla, 2017). En la figura superior se puede ver cómo se complementan en un sistema de recomendación híbrido los tres tipos de sistemas vistos hasta ahora: los sistemas de filtrado demográfico y colaborativos se alimentan de los perfiles de los usuarios para hacer las sugerencias, subsanando las carencias de ambos, y el sistema basado en contenido completa la recomendación añadiendo información sobre productos similares (Montaner y col., 2003). De este modo, si el usuario rellena el formula complementario, el motor de recomendación tirará del sistema basado en contenido, mientras que si se niega, el sistema de filtrado demográfico es el que entrará en acción.

Los sistemas de recomendación híbridos presentan menos carencias que el resto de sistemas de recomendación ya que nacen de la combinación de varios métodos (Montaner y col., 2003). Como ejemplo de empresas que implementan este tipo de recomendaciones están Facebook (Kabiljo e Ilic, 2015), LinkedIn (Wu y col., 2014), y Youtube (Covington y col., 2016) entre muchas otras.

A pesar de ello, si un nuevo usuario se niega a completar el formulario extra, como la recomendación se haría con el filtrado demográfico, heredaría sus carencias (Ricci y col., 2011). Es decir, al emplear estereotipos que no tratan a los usuarios de forma individualizada, de tener un usuario atípico la probabilidad de errar es elevada. Como se acaba de comprobar, encontrar la combinación de sistemas de recomendación que traigan consigo un sistema híbrido y que presente la menor cantidad de problemas posibles con los nuevos usuarios puede ser una tarea tediosa. Es por ello que se va a introducir otro tipo de sistema de recomendación menos conocido pero que obtiene buenos resultados con pocos datos y un perfil de usuario concreto.

2.1.5. Sistema de recomendación basado en conocimiento

Los sistemas de recomendación basados en conocimiento permiten llevar a cabo recomendaciones valiéndose de datos que le proporcionan conocimiento tanto de los productos como de los usuarios (Rezaul y Alla, 2017). Estos datos pueden ser las características del producto, los patrones de compra de ciertos usuarios o los criterios de recomendación, por ejemplo (Rezaul y Alla, 2017).

Suelen emplearse cuando el historial de actividad de los usuarios es pequeño y son una gran alternativa para las situaciones cuando tanto los sistemas de recomendación basados en

contenido como los basados en filtrado colaborativo presentan carencias (Gorakala y Usuelli, 2015). Empresas como Bestbuy (Rajendra, Dewan y Can Colakoglu, 2019) son un ejemplo de compañías que apuestan por este tipo sistema.

A pesar de ello, son un tipo de sistemas poco estudiados y empleados, pero son altamente valiosos y pueden emplearse en combinación con otros sistemas de recomendación para subsanar sus carencias (Pelánek, 2018), dando lugar a sistemas de recomendación híbridos. Además, los sistemas basados en conocimiento no individualizan al usuario al ser reducido el grupo de usuarios al que le pueden hacer las sugerencias (Rezaul y Alla, 2017) y, como este sistema de recomendación se construye en función de las necesidades de los datos, no es posible definir una técnica general de aplicación (Gorakala y Usuelli, 2015). Por otro lado, es necesario que el usuario empiece a interactuar con el contenido de la compañía por iniciativa propia para que el sistema de recomendación empiece a funcionar. Esto se puede incentivar mediante publicidad para captar usuarios y con una página de inicio clara e intuitiva para que comience a generar registros (Pelánek, 2018).

2.2. Antecedentes del problema

Conocer y estudiar los principales sistemas de recomendación es necesario para determinar las diferentes opciones que hay para abordar el objetivo de recomendar a un nuevo usuario de Airbnb su primer destino. Asimismo, se van a invertigar los antecedentes de este problema y otros parecidos estudiados por la compañía y otras empresas con el mismo fin. De este modo, se reparará en las técnicas de *machine learning* que han aplicado en cada caso y los motivos que han llevado a esa decisión principalmente.

2.2.1. Antecedentes proporcionados por Airbnb

La base de datos con la que se trabaja a lo largo de esta memoria contiene datos de la compañía Airbnb, dedicada al alquiler de apartamentos vacacionales. Debido a que se va a construir un sistema de recomendación a partir de ellos, se va a analizar cómo Airbnb resuelve el problema de hacer recomendaciones a sus usuarios y les personaliza el contenido.

Hay que recordar que el problema de recomendación de usuarios de Airbnb es único, al tener en cuenta que tanto el anfitrión como el huésped son considerados usuarios (Grbovic, Wu y col., 2018). Por ello,la compañía debe centrar sus esfuerzos en el estudio de ambos per-

files. Para los anfitriones, existe un estudio de la universidad del estado de Oklahoma sobre la construcción de un modelo para ayudar a los anfitriones de Airbnb a establecer el precio de su propiedad en alquiler (Guggilla, Gutha y Chakraborty, 2011). En cuanto a los huéspedes, Airbnb ha desarrollado varios algoritmos para hacer recomendaciones de apartamentos similares y para posicionar los resultados obtenidos tras una búsqueda a cada usuarios, todo ello de forma personalizada (Grbovic y Cheng, 2018).

Es importante resaltar que la empresa Airbnb ha ido evolucionando con el tiempo, pasando de ofrecer en 2008 únicamente alquileres de apartamentos a incluir en 2016 el poder reservar experiencias y desde 2017 también permite hacer reservas en restaurantes (Airbnb, 2019b). Por consiguiente, cuenta con un posicionador de búsquedas para los apartamentos y otro para las experiencias. Para posicionar apartamentos emplea técnicas como Lambda Rank, Natural Language Procesing (NLP), Gradient Boosted Decision Trees (GBDT) (Grbovic y Cheng, 2018) y redes neuronales de *deep learning* (Haldar y col., 2018), entre otras técnicas. Además, para mejorar los resultados, los analistas comentan que se pueden añadir al algoritmo de búsqueda tanto el sistema de recomendación de contenido similar que se estudiará a continuación como la personalización a tiempo real (Grbovic, Cheng y col., 2018), métodos que usan NLP, redes neuronales y técnicas de *negative sampling*. Por otro lado, para posicionar experiencias, se basa en el algoritmo Gradient Boosted Decision Trees (Grbovic, Wu y col., 2018). Cabe mencionar que en ambos casos también se aborda el *cold start* tanto para posicionar los nuevos apartamentos (Grbovic y Cheng, 2018) como para las nuevas experiencias (Grbovic, Wu y col., 2018).

En cuanto a los sistemas de recomendación que Airbnb ha desarrollado, el mencionado en el párrafo anterior, (Grbovic, Cheng y col., 2018), se basa en recomendar a los huéspedes contenido similar a lo que hayan buscado anteriormente y sirve de complemento para mejorar el posicionamiento de las búsquedas, llegando con ambas técnicas a un porcentaje de conversión del 99 % (Grbovic y Cheng, 2018). Concretando más el procedimiento que emplean, por cada usuario y apartamento crean un vector donde cada una de las posiciones corresponde a los atributos que influyen tanto en la elección por parte del huésped de un apartamento en concreto como en la aceptación del anfitrión de un huésped en su propiedad. En base a similitudes incrustadas (*embedding similarities*) entre estos dos vectores se aumenta o disminuye la probabilidad de que ambos usuarios estén satisfechos y se efectúe una reserva. Además, incorporan un procedimiento de *negative sampling* en el que se les da un peso extra a los atributos con malas evaluaciones (tanto para el huésped como para la propiedad) y así reducir la

probabilidad de rechazo de forma significativa (Grbovic y Cheng, 2018). Por lo tanto, este sistema de recomendación se podría englobar dentro de uno híbrido compuesto por un sistema de recomendación de filtrado colaborativo (al basarse en las opiniones de otros usuarios para construir los vectores) y por un sistema de recomendación basado en contenido (al emplearse también la información proporcionada por ambos usuarios para construir los vectores).

2.2.2. Antecedentes proporcionados por Kaggle

La base de datos de este TFM fue proporcionada por Kaggle en 2015 como parte de un concurso cuyo objetivo era el de predecir el próximo destino de un nuevo usuario (Kaggle, 2015b). La página de la competición ha sido consultada para extraer los datos, comprender las variables de las que se compone y para garantizar la originalidad del trabajo. Esto último se va a lograr reduciendo el número de variables de la base de datos y presentando una forma propia y diferente de predecir el primer destino a un nuevo usuario con un sistema de recomendación.

Es importante recordar que Kaggle es una página que lanza competiciones para analistas de datos donde muchas empresas o/e instituciones aportan una base de datos propia junto a un problema a resolver por los concursantes a cambio de, a veces, premios económicos. Tras concluir con una competición, se evalúa el resultado de cada participante y se entregan premios por la mejor contribución en la competición (resultado global de la evaluación), por el mejor kernel (la calidad del código empleado) y por la mejor discusión (la aportación en un foro). En cada una de estas categorías se entregan 3 premios (oro, plata y bronce, donde puede llegar a haber más de un bronce). De entre estas categorías, son los de mejor contribución los que reciben el premio económico y a los que también se les hace una entrevista. Es por ello que, de entre los ganadores de la competición de la que proviene la base de datos, se van a comentar los resultados de los ganadores a los que les hicieron la entrevista al ser los que, de forma global, mejor han resuelto el problema. En esta competición, el concursante que quedó en primera posición, es decir, el de mejor contribución en la competición, ni concedió entrevista ni compartió el proceso que siguió. Es por ello que sólo se comentarán los resultados de los concursantes que quedaron en segunda y tercera posición y que sí concedieron entrevista.

La base de datos proporcionada por Kaggle está compuesta de cinco conjuntos de datos:

- `countries.csv`: Resumen estadístico de los países de destino y su ubicación.
- `age_gender.csv`: Resumen estadístico del intervalo de edad, género y país de destino de los usuarios.

- `train.csv`: Conjunto de datos de entrenamiento.
- `test.csv`: Conjunto de datos de prueba.
- `sessions.csv`: Registro de las sesiones web de algunos usuarios.

Se recomienda la lectura de las dos primeras páginas del planteamiento del problema en la Sección 4 para más información sobre el contenido de cada una de las tablas, pues es ahí donde se introducen con más detalle. Así el lector logrará entender mejor los párrafos siguientes donde se estudiarán la segunda y tercera mejor solución.

La segunda mejor solución (Team, 2016a) toma la decisión de cruzar las tablas `countries`, `age_gender` y `train` por un lado y `countries`, `age_gender` y `test` por otro. Además, también modifica atributos, pues convierte la edad en intervalos y elimina los valores que se alejan del límite real, categoriza las variables categóricas usando el codificador One Hot, calcula la diferencia de tiempo transcurrido entre la primera actividad del usuario, la creación de la cuenta y la primera reserva (añadiendo 7 nuevos atributos a los que les designa el nombre de `lag_features` para referirse a ellos de forma colectiva) y agrupa el tiempo empleado por cada usuario en cada sesión.

En cuanto a las técnicas que ha aplicado para realizar la predicción, construye un modelo de 3 capas donde utiliza el modelo XGBoost (Brownlee, 2016), GLM (Generalized Linear Models) (Burkov, 1956) y la validación cruzada (Villalonga, 2019). A continuación se recupera la figura proporcionada por la campeona que resume el orden de aplicación de estos métodos, donde todo el código lo ha llevado a cabo usando R.

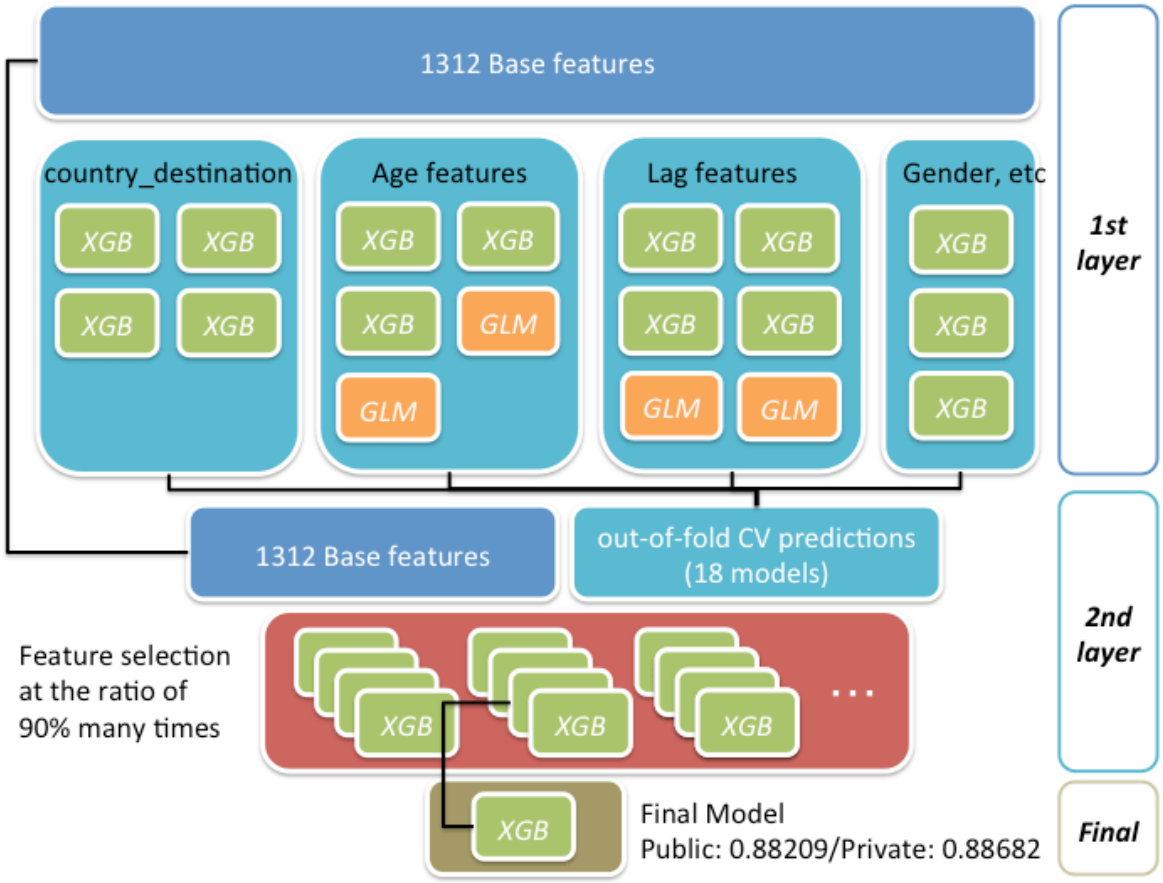

Figura 3: Resumen de las tres capas empleadas en la segunda mejor solución del concurso de Kaggle (Team, 2016a).

Por otro lado, la tercera mejor solución (Team, 2016b) se obtiene cruzando las tablas de `train` y `sessions`. Además, en `train` cuenta la cantidad de valores nulos, rellena los valores fuera de rango de la edad, desglosa las variables de fecha en año, temporada, hora, día de la semana y otras más y codifica las variables categóricas usando One Hot Encoding. En `sessions` cuenta el total y la frecuencia de cada una de las acciones, calcula varias medidas estadísticas del tiempo empleado en cada sesión y hace el *bincounts* de este tiempo.

Para realizar la predicción, este concursante también ha utilizado un modelo de tres capas compuesta de una combinación de algoritmos como Random Forest (Ho, 1995), XGBoost Gradient Boosting (Burkov, 1956), redes neuronales (Rezaul y Alla, 2017) y el clasificador Extra-trees (Bhandari, 2018). El código lo ha llevado a cabo utilizando Python y en la Figura 4, proporcionada por el campeón, se resume su propuesta.

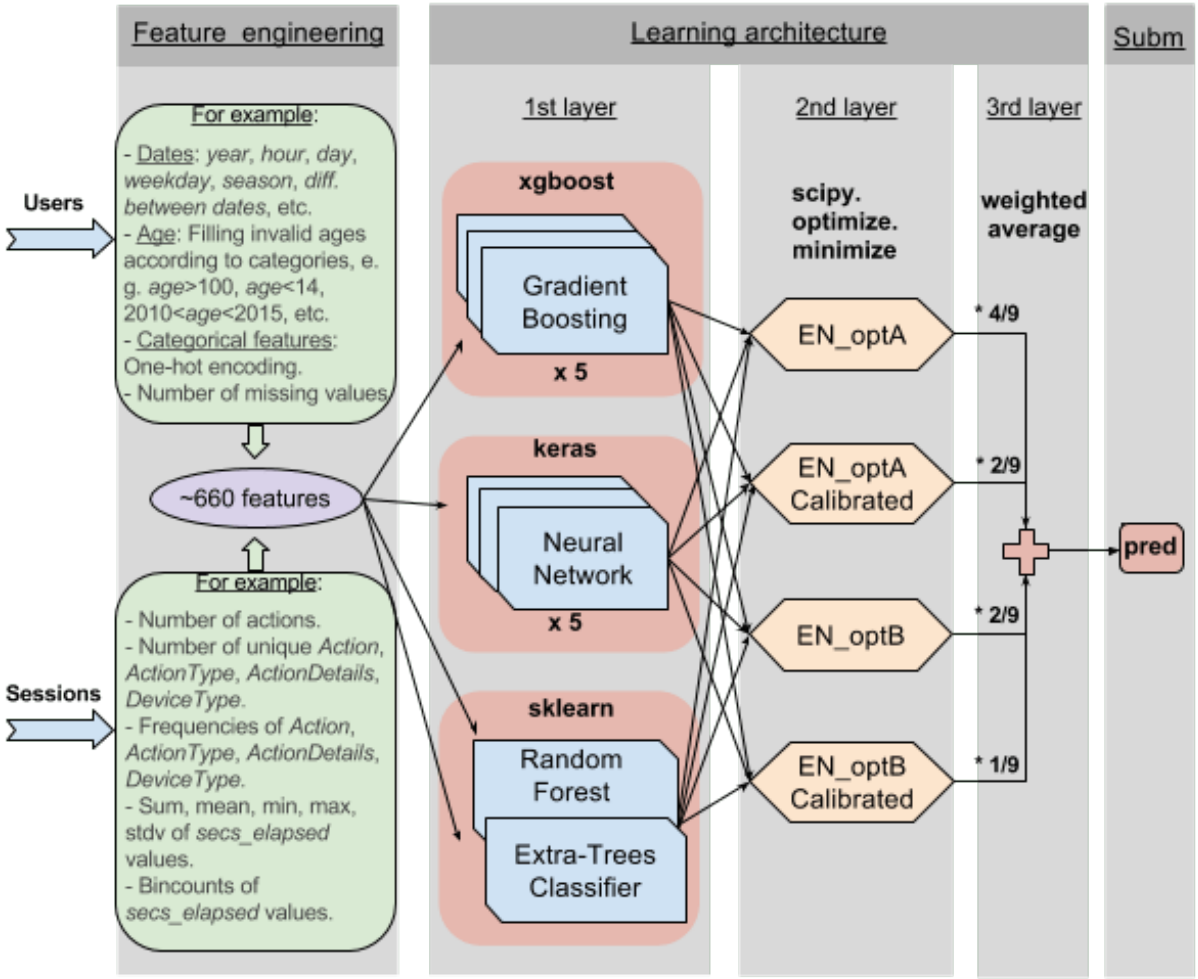

Figura 4: Resumen de las tres capas empleadas en la tercera mejor solución del concurso de Kaggle (Team, 2016b).

A la hora de determinar los ganadores (primer, segundo y tercer mejor resultado) y evaluar las soluciones de cada participante, Kaggle solicita a cada participante que genere un csv con dos variables: los identificadores de los usuarios y el destino al que el modelo elaborado predice que acudirá. Para ello, una vez ha entrenado el modelo, se ejecuta el modelo con los datos de `test.csv`, tabla que está compuesta de las mismas variables que `train.csv` salvo la variable a predecir (el destino de los usuarios), siendo todos los identificadores de `train.csv` y `test.csv` distintos. Los resultados de esta ejecución se guardan y se les da el formato demandado por Kaggle para enviar los resultado y así poder ser evaluados.

Una vez se reciben los resultado, Kaggle calcula la ganancia acumulada descontada (NDCG de sus siglas en inglés Normalized Discounted Cumulative Gain) de cada uno para medir la proximidad de la predicción respecto al resultado real (Kaggle, 2015a). NDCG es una medida de clasificación que evalua la utilidad de una respuesta en función de su posición en

una lista de resultados (Kaggle, 2015a), siendo su fórmula la siguiente:

$$NDCG_k = \frac{DCG_k}{IDCG_k},\ donde\ DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)},$$

$IDCG_k$ es el DCG (Discounted Cumulative Gain) ideal, es decir, el máximo posible, rel_i es la relevancia de la solución en la posición i y k es la cantidad de soluciones que puede haber en la lista de resultados.

Esta medida, al ser una valor normalizado, proporciona un valor entre el cero y el uno, ambos inclusive. Además, Kaggle permite a los participantes incluir hasta 5 posibles destinos por usuario de Airbnb, siendo $k = 5$ en el cómputo de la media. Por ello, cobra sentido que en la propuesta de la segunda y tercera mejor solución, resumidas en las Figuras 2 y 3, combinen varios algoritmos para hacer la predicción.

Como Kaggle no ha facilitado tras concluir este concurso los destinos de los usuarios de la tabla `test.csv`, no hay forma de poder otorgar una puntuación al modelo que se logra con esta memoria y compararlo con los resultados de los campeones estudiados. Por ello, con este TFM se pretende proponer otra forma de resolver el problema ajena a los requisitos de la competición y recomendando un único destino, el que el modelo crea más probable. Asimismo, la precisión del modelo se medirá con la base de datos de entrenamiento.

2.2.3. Antecedentes proporcionados por otras fuentes

Airbnb no sólo pone a disposición del público la base de datos de la página de concursos Kaggle, sino que también ofrece a través de su web bases de datos *open source* para quien quiera trabajar con ellas. A continuación, se va a hacer referencia a un problema resuelto con una de estas bases de datos libre y cuyo objetivo es el de hacer un sistema de recomendación basado en el perfil del usuario (Mahajan, 2017).

Para ello se vale de una base de datos de 1,3 millones de registros con información a cerca de las reservas anteriores de ciertos usuarios de Airbnb y de los atributos o características de algunos apartamentos (número de habitaciones, comodidades, evaluaciones, etc.) (Mahajan, 2017). Tras un reacondicionamiento de los datos (normalizando los datos con la normalización min-max), se vale de redes neuronales (perceptrones multicapa con retropropagación) para calcular la posible puntuación que le daría cada huésped a la propiedad en función de su rango de edad, profesión, intereses y temporada. Para llevar a cabo la recomen-

dación y predecir el futuro apartamento, emplean el algoritmo recomendador Matchbox para entrenar las recomendaciones y la puntuación de este recomendador para cumplir con el objetivo, mostrando los 7 mejores resultado que serán los que se recomienden (Mahajan, 2017). Además, culminan todo este proceso creando un servicio web del modelo (Mahajan, 2017).

Este estudio sirve como ejemplo del potencial que tienen los datos de Airbnb para generar sistemas de recomendación y presenta un problema resuelto: el de recomendar apartamentos en función del perfil del usuario. Con la base de datos de esta memoria, dadas las variables de las que se compone, el objetivo va a ser recomendar el primer destino a un nuevo usuario de Airbnb en función de su actividad con la web y de su perfil.

2.3. Estrategia propia para abordar el problema

Tras la recapitulación de los sistemas de recomendación más empleados y el posterior análisis de los antecedentes del problema a tratar, ya se tiene abundante información a cerca del contexto que rodea el objetivo de este Trabajo de Fin de Máster. Debido a la finalidad y las situaciones de uso de cada uno de los sistemas de recomendación, los candidatos más atractivos son el sistema de recomendación basado en filtrado colaborativo, el sistema de recomendación basado en contenido y el sistema de recomendación híbrido.

El sistema de recomendación que más carencias presenta para el objetivo de hacer una recomendación a un nuevo usuario, también conocido como *cold start*, es el sistema de recomendación basado en filtrado colaborativo pues, pese a que está diseñado para recomendar contenido a los usuarios en base a las decisiones tomadas por usuarios parecidos, una de las carencias de este sistema de recomendación se encuentra en los nuevos usuarios. Cuando un nuevo usuario accede a la web que está basada en este sistema de recomendación, al no tener forma de comparar su actividad con la de otros usuarios y encontrar similitudes (ya que un nuevo usuario se caracteriza por no haber accedido antes al contenido de la web), el sistema no puede hacerle una recomendación.

Una forma de subsanar las carencias del sistema anterior es combinándolo con un sistema basado en contenido y creando un sistema de recomendación híbrido. De esta forma, la parte de filtrado colaborativo es útil cuando un usuario habitual accede a la web, mientras que la de filtrado basado en contenido se aplica con los nuevos usuarios tras pedirles que rellenen una encuesta para poder determinar sus intereses y personalizar el contenido. Aún así, si el usuario no ha querido facilitar esos datos, no es posible hacer una recomendación acertada.

Otra opción es la de construir un sistema de recomendación basado en conocimientos previos, donde se conocen las decisiones que han tomado otros usuarios cuando estaban en la misma situación del usuario actual, es decir, cuando fueron nuevos usuarios y, en base a ese conocimiento y al tipo de interacciones que se registran con la web, se puede realizar la recomendación. Este sistema encaja muy bien con el objetivo de la memoria pero pude ser reforzado y mejorado combinándolo con otro sistema de recomendación y dando lugar a un sistema híbrido de nuevo.

Para tomar una decisión entre estas dos posibilidades de sistemas de recomendación, conviene consultar los atributos de la base de datos y la información que aporta cada una de las tablas, al ser esta información con la que se va a trabajar. Los datos tienen las siguientes características:

- Las tablas no son muy grandes (11, 421, 213.452, 62.097 y 10.567.738 registros en `countries`, `age_gender`, `train`, `test` y `sessions` respectivamente).
- Ofrecen información sobre el tipo de interacción que ha tenido el usuario con Airbnb, considerando como usuario tanto al huésped como al anfitrión.
- Aportan conocimiento sobre datos estadísticos de los usuarios de Airbnb y los países de destino.
- Dan información sobre características demográficas y otro tipo de interacciones (esta vez no sólo con Airbnb, sino que con otros entornos) que han culminado en reserva.

Por un lado, las tablas de datos originales no tienen excesivamente demasiados registros, además, con los cruces entre tablas para mejorar la calidad de los datos, esta candidad se reducirá. Por el otro, las recomendaciones para nuevos usuarios están dirigidas a un colectivo con características muy concretas, al tener pocos o ningún registro sobre los individuos y su actividad. Debido a que estas características coinciden con más puntos de los comentados en los sistemas de recomendación basados en conocimientos que en los híbridos, es este tipo de sistema de recomendación, el basado en conocimientos, en que se va a llevar a cabo.

A pesar de esta elección, otra línea futura de investigación puede ser la de construir un sistema de recomendación híbrido para esta misma base de datos y así tener los dos enfoques. Resaltar que se ha optado por las contrucción de un único sistema de recomendación al estar condicionada la elaboración del Trabajo de Fin de Máster por el plazo de entrega, habiendo elegido el tipo de sistema de recomendación que, en base a los conocimientos adquiridos a raíz de la investigación sobre los sistemas de recomendación, es el que mejores resultados

puede aportar.

En cuanto a cómo va a ser abordada la investigación en la memoria, en la Sección 4 se hace toda la preparación de los datos para elegir la tabla o tablas en este caso a las que se les aplicarán los algoritmos de *machine learning*. Este proceso se divide en el análisis de las tablas originales, la limpieza de los datos, el filtrado (estudio del potencial de cada variable) y la transformación (crear nuevas variables a partir de las originales) de los atributos, la integración o cruce de las tablas más ricas en contenido y la reducción (disminución de variables) de los atributos en los cruces. Una vez se disponen de las tablas finales, se les aplica algoritmos de aprendizaje automático como C5.0, Random Forest y One Vs All en la Sección 5. Finalmente, en la Sección 6 se estudian los resultados que se han obtenido con los algoritmos anteriores y se selecciona el que mejor se adapta a los datos. Para resumir estas fases se facilita la Figura 5, resaltando la proporción de tiempo empleada en cada una de las fases.

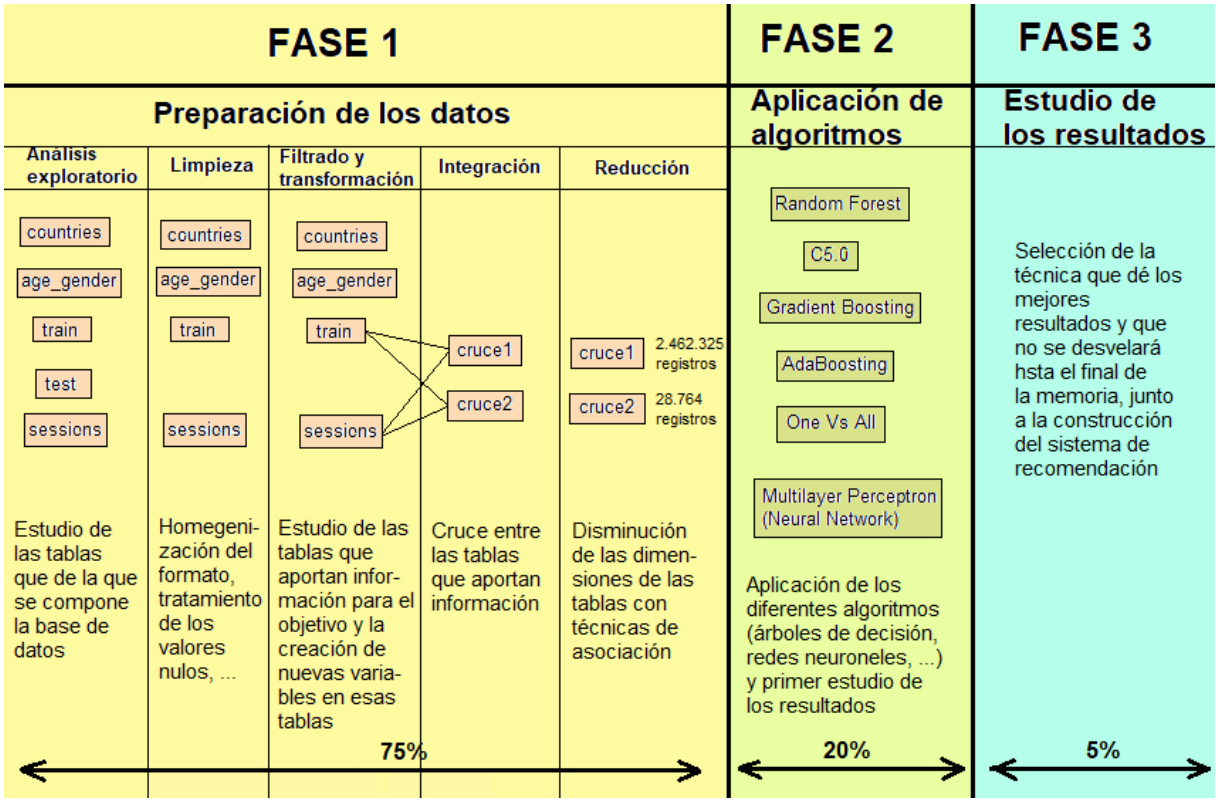

Figura 5: Resumen de las diferentes fases del estudio llevado a cabo en la memoria.

Por último, resaltar que, al tratarse de un sistema basado en conocimientos el que se va a construir, va a ser necesario el estudio y la aplicación de diversos algoritmos de *machine learning*, buscando el que dé mejores resultados. Para consultar el código desarrollado a lo largo de la memoria, éste se encuentra en el anexo, Sección 9, con el objetivo de mantener

al lector centrado en el contenido y en los resultados en vez de en la parte técnica de la programación.

3. Objetivos y metodologías

El objetivo principal de este Trabajo de Fin de Máster es el de crear un sistema de recomendación basado en conocimiento a partir de los datos sobre las interacciones de otros nuevos usuarios anteriores y las características de los mismos. De entre las variables que forman parte de la base de datos se encuentran el tipo de dispositivo empleado, la edad y sexo de los usuarios, el tiempo hasta la primera reserva, su actividad y el destino.

Para ello, se va a seguir el siguiente proceso donde se comentan los objetivos más específicos que se van a abordar:

1. Acondicionamiento de los datos mediante el análisis exploratorio y la limpieza de los datos.
2. Filtrado y transformación de las variables de las tablas más interesantes.
3. Cruce entre tablas para enriquecer las bases de datos y posterior reducción de las variables.
4. Estudio y aplicación de las técnicas de aprendizaje supervisado para construir el sistema de recomendación basado en conocimiento, donde se van a aplicar técnicas como árboles de decisión y redes neuronales.
5. Evaluación de los resultados obtenidos.
6. Conclusiones que se obtienen tras concluir el procedimiento y líneas futuras de investigación.

Con este TFM también se pretende profundizar más en los sistemas de recomendación basados en conocimiento y hacer de este trabajo un modelo que sirva para ayudar tanto a Airbnb como a otras empresas a mejorar sus sistemas de recomendación y ser fuente de inspiración y nuevas ideas. A pesar de ello, el principal interesado ante la forma de enfrentarse al problema seguramente sea Airbnb porque, como ya se ha visto en el estado del arte, su problema es único debido a que nuevos usuarios pueden ser tanto los huéspedes como los anfitriones. Además, Airbnb ya cuenta con estudios estadísticos para los nuevos huéspedes sobre cómo actuar con su apartamento, faltándole sólo a la compañía mejorar sus conocimientos acerca de los nuevos huéspedes para abordar completamente el problema.

Por otro lado, el esquema conceptual que va a seguir el sistema de recomendación se resume con la siguiente figura.

Figura 6: Esquema del planteamiento del sistema de recomendación que será abordado en la memoria.

Ante un nuevo registro de un nuevo usuario, primero se va a predecir si, en función de su interacción con la web de Airbnb, va a acabar reservando o no. Si el resultado de esta predicción es que no va a reservar, se detiene la recomendación al no tener datos en la base de datos como para responder a esta situación. Si, por el contrario, resulta que acabará reservando, se le hace una recomendación entre unos posibles destinos que aparecen en verde en la Figura 6: Alemania, Australia, Canadá, EE.UU., España, Francia, Holanda, Italia, Portugal, Reino Unido, No se sabe y Otro posible destino. Para predecir si el usuario va a reservar y después predecir el destino en caso de reserva, se van a emplear dos tablas distintas como se comentará a lo largo de las siguientes páginas.

En cuanto a la metodología empleada, se puede dividir en dos grupos: la metodología para la toma de decisiones en cuanto a los datos y la metodología que hace referencia a las herramientas empleadas.

A la hora de tomar decisiones sobre la base de datos, se han llevado a cabo empleando el conocimiento que se han adquirido tanto de la compañía, en el estado del arte, como de los datos, en el planteamiento del problema. Junto a ésto, añadir la fuerte base analítica de la autora de la memoria que ha llevado a cabo el modelado de los datos y el conocimiento como usuaria de Airbnb desde hace más de cinco años que le ha dado el uso de la web. Gracias a todo lo anterior, el entendimiento de los datos ha sido muy bueno y ha acarreado buenas decisiones donde, en caso de duda, se estudiaban todas las opciones posibles.

En cuanto a la elección de herramientas, recordar que se le ha dado un doble enfoque. Es de conocimiento general que en el mundo del *Big Data* es difícil diferenciar cuándo un conjunto de datos es lo suficientemente grande como para aplicar herramientas de *Big Data* de cuándo es suficiente hacer lo mismo con otras herramientas no tan potentes en cuanto a tiempos de ejecución. En esta memoria también se aborda este problema al reproducir los pasos con herramientas de *Big Data* y con herramientas convencionales. En cuanto a las herramientas de *Big Data*, se han empleado SparkSQL y SparkMLib (trabajando en un entorno de *Big Data*), mientras que, como herramientas convencionales se ha trabajado con SQL y R (desarrollado en un entorno de *Business Intelligence*). Es por ello que en las conclusiones no sólo se tratarán temas sobre el algoritmo de *machine learning* que mejores resultados aporta, sino que también se dará respuesta a, para este conjunto de datos, cuál de los dos caminos es el más eficiente con un ordenador de 4GB de RAM. Para ello, se comenta a continuación cada entorno, comenzando por el de *Business Intelligence* con la Figura 7.

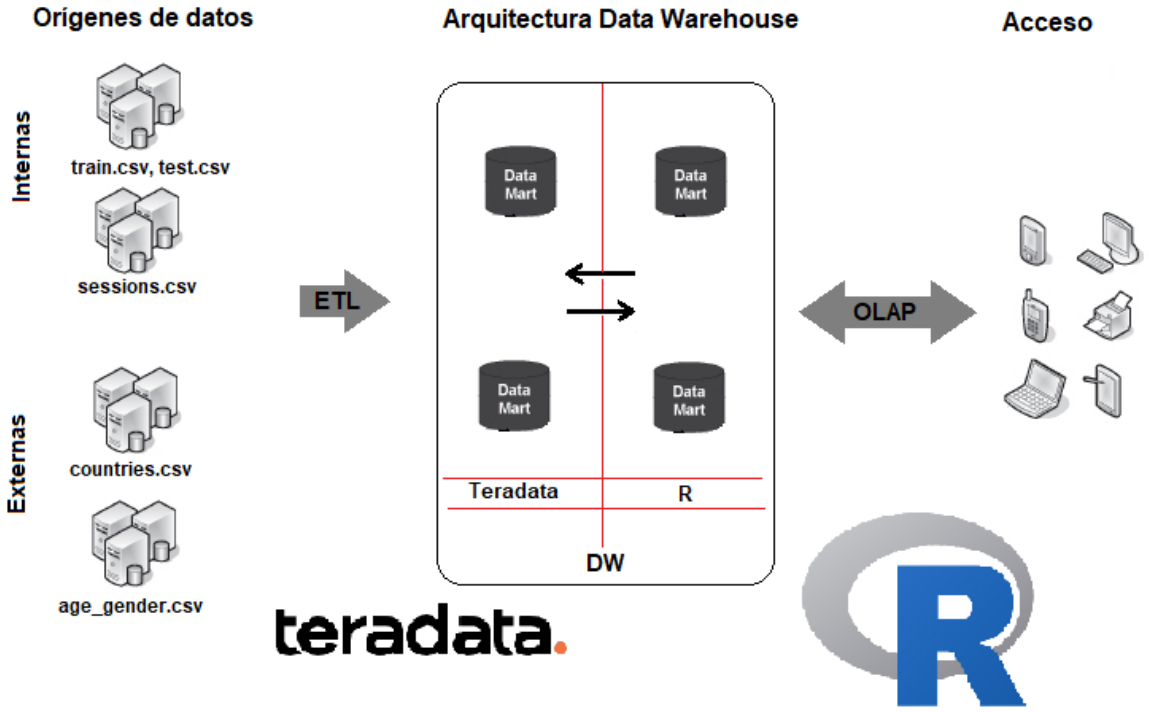

Figura 7: Entorno de *Business Intelligence* donde se trabajará en la memoria.

La figura superior se basa en el libro (Conesa y Curto, 2010). En ella se aprecia la procedencia de las 5 tablas que constituyen la base de datos, junto un proceso ETL (*extract*, *transform*, *load*) que no se concreta y con el que se llevan los datos al almacén de datos o *data warehouse*. Para elaborar el análisis de BI se presupone la existencia de este entorno y se parte del data warehouse para dar comienzo al Trabajo de Fin de Máster. En este caso,

los datos (ya acondicionados por el ETL) se cargan en Teradata (Teradata, 2019), herramienta de SQL especializada en *data warehouse*s, con la que se terminan de preparar los datos y, posteriormente, se analizan estadísticamente y gráficamente con R, herramienta *open source* que destaca por su utilidad para estudios estadísticos y el modelado de los datos (R, 2019b). Resaltar que las dos flechas entre Teradata y R indican el traspaso de información entre el almacén de datos (Teradata) y la herramienta estadística. Finalmente, para la elaboración de algunos gráficos se emplea Excel.

Para el entorno de *Big Data*, se resume con la siguiente figura.

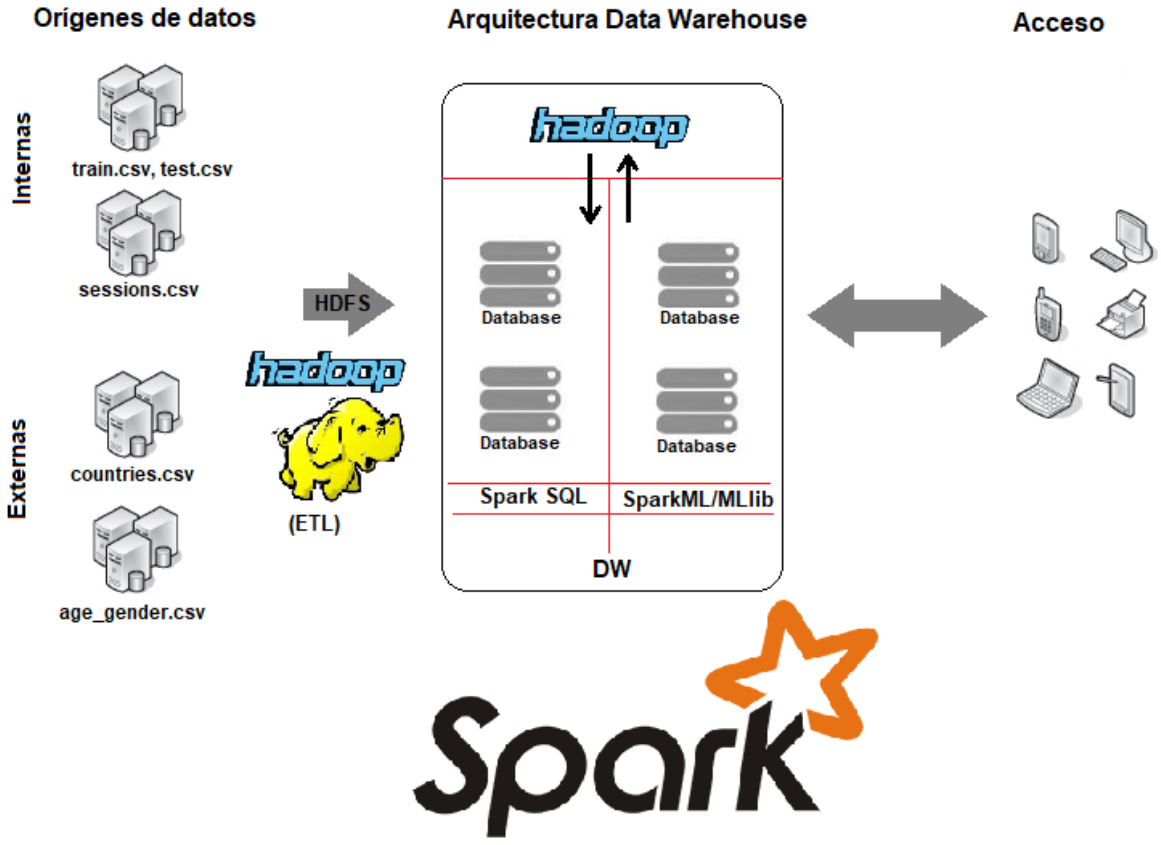

Figura 8: Entorno de *Big Data* donde se trabajará en la memoria.

En este caso, los datos se importan primero a un entorno distribuido como Hadoop (Hadoop, 2019a), usando Hadoop para el almacenamiento de datos, y se empieza a trabajar con HDFS (*Hadoop Distributed File System*) (Hadoop, 2019b). Después, se recogen los datos de este entorno (de ahí las dos flechas de distinto sentido en el Figura 8, representando el traspaso de información) y se llevan a Spark (Spark, 2019f), siendo Spark el motor de procesamiento de datos, punto en donde se da comienzo al código de la memoria. Con SparkSQL (SparkSQL, 2019) se hace el mismo proceso que el correspondiente a la herramienta de bases de datos

relacionales de BI, mientras que tanto SparkML como SparkMLlib (SparkMLib, 2019) se emplean para la parte analítica. Se ha elegido trabajar en la API de Spark de Scala debido a que, como Spark está escrito en Scala, entre las 4 APIs que tiene la de Scala es la primera en crear y la más completa (Rezaul y Alla, 2017). Además, gran parte de los gráficos se han hecho con Excel (al trabajar con datos resumidos o agrupados) y el cambio de .csv al formato con el que por defecto se trabaja en SparkML/MLlib se ha hecho con Python. Finalmente, el código empleado para cada entorno está disponible en el anexo, Sección 9.

4. Planteamiento del problema

El objetivo o problema a resolver de este Trabajo de Fin de Máster es el de hacer una recomendación de destino a un nuevo usuario de Airbnb. Las dificultades que presenta esta cuestión son la poca cantidad de información que se tiene para hacer sugerencias a los nuevos usuarios y que sobre los sistemas de recomendación basados en conocimiento no se tiene ni tanta información ni un procedimiento establecido como en los otros sistemas (Pelánek, 2018). Debido a ello, se le prestará especial dedicación a la preparación de los datos que se compone de las siguentes fases:

- Análisis exploratorio de los datos: primer contacto con las tablas de la base de datos donde se describirá cada variable además de establecer un esquema global de la relación existente entre las tablas.
- Limpieza de los datos: se modificará el formato original de las variables y se hará el análisis y la gestión de los valores nulos y fuera de rango, entre otras cosas.
- Filtrado de la tabla `countries`: se analizará cada variable y la aportación que hace cada una al objetivo del TFM.
- Filtrado de la tabla `age_gender`: se estudiarán las variables y la aportación que hacen al objetivo del TFM.
- Filtrado y transformación de la tabla `train`: se estudiarán todas las variables y la contribución que hace cada una de ellas al objetivo. Además, se modificará la tabla original, enriqueciéndola con nuevas variables y eliminando otras para diferenciar esta la tabla de la proporcionada por Kaggle (para garantizar la originalidad del trabajo). Asimismo, se estudiarán las características de las nuevas variables.
- Filtrado y transformación de la tabla `sessions`: se analizarán todas las variables y las contribución que hace cada una al objetivo del trabajo. Asimismo, se estudiará el enriquecimiento de la tabla mediante la incorporación de nuevas variables.
- Integración de los datos: se llevará a cabo el cruce entre las tablas con los datos más ricos en contenido para llegar al objetivo.
- Reducción de los datos: se estudiará si existe alguna relación de influencia entre las variables de la tabla integrada. De ser así, se procederá a la reducción de la dimensión de la misma.

4.1. Análisis exploratorio de los datos

En el momento en el que se quiere explotar una base de datos para extraer el conocimiento que reside en los datos, es necesario familiarizarse con los elementos de los que está compuesta primero. Este primer contacto entre el *data scientist* o analista de datos y la base de datos se hace a través del análisis exploratorio de los datos, es decir, mediante un examen de los datos previo a la aplicación de técnicas estadísticas y/o de *machine learning*. De este modo, se conoce el contenido de las tablas, la relación entre las mismas y el estado en el que se encuentra cada atributo.

En este caso, la base de datos consta de 5 tablas: `countries`, `age_gender`, `train`, `test` y `sessions` que se encuentran en los documentos `countries.csv`, `age_gender.csv`, `train.csv`, `test.csv` y `sessions.csv`, respectivamente. A continuación, se describen brevemente los atributos de los que se componen cada una de las tablas, su formato y descripción, y la cantidad de registros de cada una.

countries: Resumen estadístico de los países de destino y su ubicación.

Atributos	Formato (longitud)	Descripción
country_destination	String(2)	País de destino
lat_destination	String/Float	Latitud del destino
lng_destination	String/Float	Longitud del destino
distance_km	String/Float	Distancia desde el país de origen hasta el destino
destination_km2	String/Float	Área del destino
destination_language	String(3)	Idioma del destino
language_levenshtein_distance	String/Float	Diferencia fonética levenshtein entre el idioma del país de origen y el destino

Tabla 1: Contenido de la tabla `countries`.

age_gender: Resumen estadístico del intervalo de edad, género y país de destino de los usuarios.

Atributos	Formato (longitud)	Descripción
age_bucket	String	Intervalo de edad
country_destination	String(2)	País de destino
gender	String	Género
population_in_thousands	String/Float	Población del destino en miles
year	String/Float	Año del registro de los datos

Tabla 2: Contenido de la tabla `age_gender`.

train y test: Conjunto de datos de entrenamiento y de prueba respectivamente.		
Atributos	**Formato (longitud)**	**Descripción**
id	String(10)	Identificador del usuario
date_account_created	String/Date	Fecha de creación de la cuenta (necesario para recomendar)
timestamp_first_active	String/Timestamp	Fecha de la primera actividad del usuario (puede ser anterior a date_account_created o date_first_booking porque los usuarios pueden buscar antes de registrarse)
date_first_booking**	String/Date	Fecha de la primera reserva
gender	String	Género
age**	String/Float	Edad
signup_method	String	Método de registro
signup_flow	String/Int	La página de la que vino el usuario para registrarse
language	String(2)	Idioma
affiliate_channel	String	Canal de afiliación (el tipo de marketing de pago)
affiliate_provider	String	Proveedor de afiliación (dónde se desarrolló el marketing: google, . . .)
first_affiliate_tracked**	String	Determina la primera comercialización con la que interactuó el usuario antes de registrarse
signup_app	String	Aplicación del registro
first_device_type	String	Tipo del primer dispositivo
first_browser	String	Primer navegador
country_destination*	String(2)	País de destino

* sólo se encuentra en `train`. ** valores nulos tanto en `train` como en `test`.

Tabla 3: Contenido de las tablas `train` y `test`.

sessions: Registro de las sesiones web de algunos usuarios.		
Atributos	**Formato (longitud)**	**Descripción**
user_id**	String(10)	Identificador del usuario
action**	String	Acción realizada
action_type**	String	Tipo de acción
action_details**	String	Descripción de la acción
device_type	String	Tipo de dispositivo
secs_elapsed**	String/Float	Segundos transcurridos para realizar la acción

** valores nulos.

Tabla 4: Contenido de la tabla `sessions`.

Cantidad de registros en la base de datos	
Tablas	**Cantidad de registros**
countries	11
age_gender	421
train	213.452
test	62.097
sessions	10.567.738

Tabla 5: Cantidad de registros de cada tabla de la base de datos.

Todas las tablas se proporcionan en formato .csv destacando que, cuando en la tabla en la columna de ”formato” aparecen dos formatos separados por una barra, el primero corresponde a cómo se reciben los los datos en la herramienta de almacenamiento de *Big Data*, mientras que el segundo formato hace referencia al entorno *Business Intelligence*. En cuanto a la longitud, no todos los atributos siguen siempre, dentro de cada tabla, una misma dimensión por lo que sólo se ha indicado la longitud en el caso que sí se cumple. La descripción se ha elaborado completando la información de la página de la que proviene la base de datos (Kaggle, 2015b) con el buscador Google (Google, 2019) para los casos donde no existe información sobre los atributos. Además, las tablas tienen variables que presentan valores vacíos, también conocidos como nulos. Para facilitar la comprensión acerca de la relación que guardan las cinco tablas se proporciona la Figura 9.

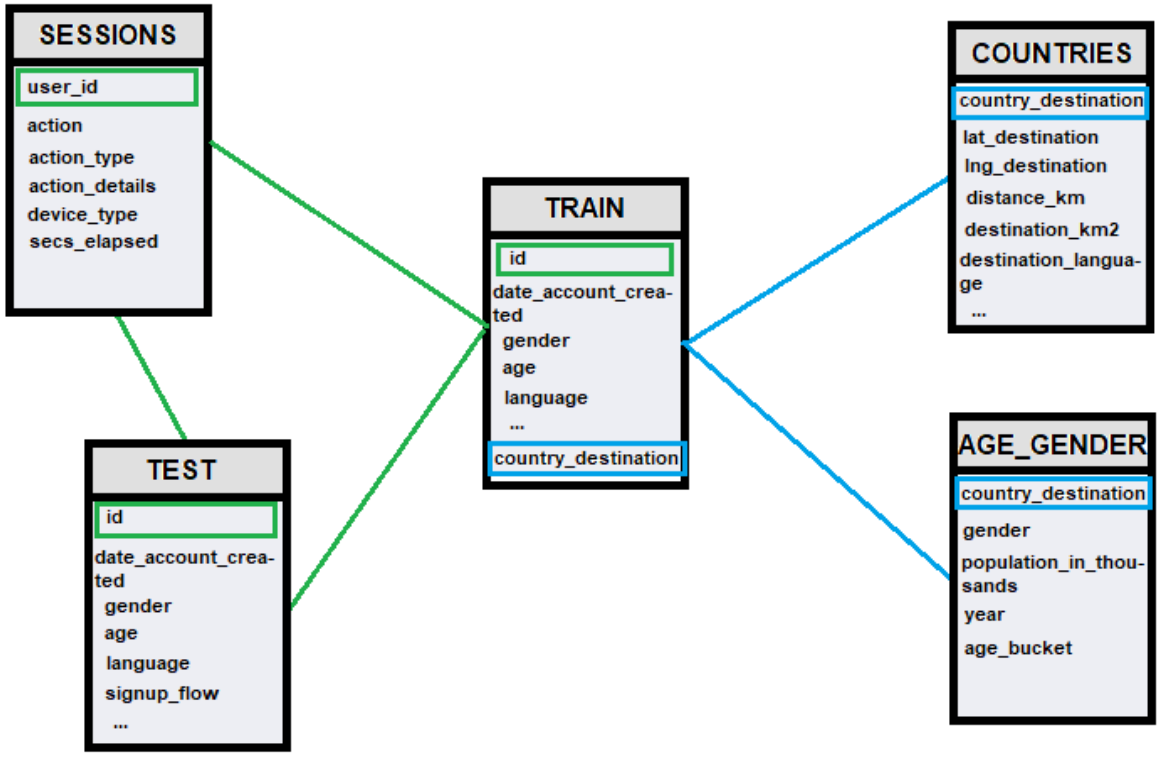

Figura 9: Relación entre las tablas de la base de datos.

Se aprecia que los atributos `id` y `user_id` aportan la misma información y que sirven para interconectar los datos de `train`, `test` y `sessions`. Por otro lado, `country_destination` conecta las tablas `train`, `age_gender` y `countries`. Esta información puede ser muy útil para estudiar el tipo de cruces que se pueden hacer para enriquecer los conjuntos de datos.

Cabe mencionar que, llegados a este punto, toda información acerca del contexto de la base de datos y de conocimiento que puede aportar Kaggle (Kaggle, 2019) llega a su fin, separándose en este punto las directrices del concurso y el objetivo del Trabajo de Fin de Máster. De este modo, la tabla `test` pasará a un segundo plano porque es la tabla de la que Kaggle (Kaggle, 2019) se vale para puntuar las soluciones y medir la cantidad de aciertos de cada modelo participante, no pudiendo hacer lo mismo con el modelo que se propone en este trabajo al desconocer la elección de destino final de los usuarios de esa tabla. Recordar que en esta memoria la calidad de la solución se va a medir dividiendo la base de datos entre el subconjunto de entrenamiento y el de validación. A pesar de ello, la tabla no se descarta porque, aunque no sea de ayuda para predecir el primer destino de un nuevo usuario, sí que sirve para determinar qué tipo de interacciones indican predisposición a reservar, como se verá más adelante.

4.2. Limpieza de los datos: tratamiento del formado, de los valores nulos y de los valores fuera de rango

En las Tablas 1, 2, 3 y 4 se ha podido comprobar la necesidad de dotar de formato distinto a algunas de las variables. En las tablas siguientes se resumen las modificaciones llevadas a cabo, habiendo modificado en ciertos casos también la longitud del atributo y estableciendo el mismo formato para cada variable tanto en el entorno *Big Data* como en el de *Business Intelligence*.

countries		
Atributos	**Formato original**	**Nuevo formato**
country_destination	String(2)	String(2)
lat_destination	String/Float	Float
lng_destination	String/Float	Float
distance_km	String/Float	Float
destination_km2	String/Float	Float
destination_language	String(3)	String(2)
language_levenshtein_distance	String/Float	Float

Tabla 6: Modificaciones efectuadas en la tabla `countries`.

age_gender		
Atributos	**Formato original**	**Nuevo formato**
age_bucket	String	String
country_destination	String(2)	String(2)
gender	String	String
population_in_thousands	String/Float	Float
year	String/Float	Int

Tabla 7: Modificaciones efectuadas en la tabla `age_gender`.

train		
Atributos	**Formato original**	**Nuevo formato**
id	String(10)	String(10)
date_account_created	String/Date	Date
timestamp_first_active	String/Timestamp	Date
date_first_booking**	String/Date	Date
gender	String	String
age**	String/Float	String
signup_method	String	String
signup_flow	String/Int	Int
language	String(2)	String(2)
affiliate_channel	String	String
affiliate_provider	String	String
first_affiliate_tracked**	String	String
signup_app	String	String
first_device_type	String	String
first_browser	String	String
country_destination	String(2)	String(2)

** valores nulos.

Tabla 8: Modificaciones efectuadas en la tabla `train`.

sessions		
Atributos	**Formato original**	**Nuevo formato**
user_id**	String(10)	String(10)
action**	String	String
action_type**	String	String
action_details**	String	String
device_type	String	String
secs_elapsed**	String/Float	Float

** valores nulos.

Tabla 9: Modificaciones efectuadas en la tabla `sessions`.

En la tabla `countries`, Tabla 6, a las cinco variables que eran de tipo caracter en el entorno *Big Data* y numérico en el *Business Intelligence*, se les ha dado el formato float para ambos. Además, se ha modificado la extensión del atributo `destination_language` para que

sea el mismo que el de la variable `language` de la tabla `train` al tener ambas idiomas en común, uniformando sus formatos. En la tabla `age_gender`, Tabla 7, en ambos entornos también se les ha dado formato numérico a las variables que lo demandaban y el atributo `gender` se ha cambiado a mayúsculas para dotarle del mismo formato que la variable `gender` de `train`. En cuanto a la tabla `train`, Tabla 8, se ha dado el mismo formato de fecha (DD-MM-YYYY) a `date_account_created`, `timestam_first_active` y `date_first_booking` en ambos entornos para que, posteriormente, sean comparables entre sí. La variable `age` se ha agrupado por los mismos intervalos que la variable `age_bucket` de la tabla `age_gender` y a `signup_flow` se le ha dado el formato de número entero. En la tabla `sessions`, Tabla 9, al atributo `secs_elapsed` se le ha dado el formato float. Mencionar que se ha unificado la cantidad de decimales en toda la base de datos en los dos entornos, estableciendo en dos el número de decimales en las variables de tipo float.

En cuanto a los valores nulos, se encuentran en un total de 8 variables repartidas en dos tablas. Seguidamente, se resumen la proporción de valores nulos por cada variable en cada tabla, donde las tablas y atributos no mencionados carecen de valores nulos.

train	
Atributos	**Proporción de nulos**
date_first_booking	58,34 %
age	41,22 %
first_affiliate_tracked	2,84 %

Tabla 10: Proporción de los valores nulos en las variables de la tabla `train`.

sessions	
Atributos	**Proporción de nulos**
user_id	0,32 %
action	0,75 %
action_type	10,66 %
action_details	10,66 %
secs_elapsed	1,29 %

Tabla 11: Proporción de los valores nulos en las variables de la tabla `sessions`.

Se aprecia que la única variable donde la proporción de valores nulos es mayor al 50 %, es decir, la única variable donde la cantidad de información válida es menor que el número de información sin completar, es `date_first_booking` que se encuentra en la Tabla 10. Ante esta situación, se pueden tomar dos decisiones respecto a este atributo: eliminarlo de la tabla porque hay más información sin completar que la registrada o eliminar de `train` todos los

registros donde el atributo esté vacío. Ante la importancia contextual que tiene la variable para el objetivo (la variable a predecir es el primer destino para un nuevo usuario, donde un usuario en potencia no se convierte en nuevo usuario hasta que no reserva), se ha optado por la segunda opción, eliminando los registros de `train` que en `date_first_booking` no tengan datos. De este modo, la tabla `train` pasa de tener 213.452 instancias a 88.908 registros.

Debido a esta reducción en la cantidad de registros de `train`, que se mantendrá a lo largo de la memoria hasta nuevo aviso, la proporción de valores nulos del resto de variables de la misma tabla se han visto alterados. En la siguiente tabla se recogen las nuevas proporciones.

train	
Atributos	**Proporción de nulos**
date_first_booking	0 %
age	23,36 %
first_affiliate_tracked	1,95 %

Tabla 12: Proporción de los valores nulos en las variables de la tabla `train` reducida, compuesta de 88.908 registros en vez de 213.451.

En cuanto al resto de variables de las Tablas 11 y 12, como la proporción de nulos no supera el 50 % tanto en `train` como en `sessions` no suponen ningún problema y no se aplicará ninguna modificación.

Una vez se han localizado los valores nulos, se procede a su tratamiento. La forma de tratar los valores nulos se va a diferenciar en función de si la variable es cualitativa o cuantitativa. En cuanto a las variables cualitativas, como es el caso de `user_id`, `action`, `action_type`, `action_detail` y `age` (se recuerda que la edad está en intervalos ahora), los valores nulos se sustituyen por ”null”. Además, en el caso de la variable `age` también se le ha asignado el valor ”-unknown-” a los valores que son superiores a 123 años (redondeando hacia arriba, la edad de la persona más longeva del mundo (ABC, 2019)). Para las variables cuantitativas, `secs_elapsed` y `first_affiliate_tracked`, se sustituyen por ”0” o ”0.0” en función de si la variable es entera (integer) o racional (float), respectivamente.

En estos momentos ya se conocen las tablas, las variables de las que se componen cada una de ellas y se han reacondicionado los datos, modificando el formato y tratando los valores nulos y los fuera de rango de cada atributo. Antes de proceder con el enriquecimiento de la tabla a la que se le van a aplicar las técnicas de *machine learning*, se van a analizar cada una

de las tablas para elegir qué variables pueden aportar más información a la hora de predecir si el usuario tiene intención en reservar y, de haberla, predecir el primer destino. A priori, las tablas a enriquecer son `sessions` para predecir la intencionalidad del usuario de reservar en base a su interacción con la web y `train` para vaticinar el país de destino del nuevo usuario.

4.3. Filtrado de la tabla countries

En esta tabla se encuentran datos estadísticos como la latitud, longitud, idioma y otros más sobre los posibles países de destino. Gracias a la variable `distance_km` se ha podido deducir que todos los usuarios de la base de datos son estadounidenses, ya que para este país de destino la distancia respecto al origen es cero. A pesar de ello, como una de las variables a predecir es el país de destino y, dado que los atributos de `countries` sirven para completar esta variable y no la tabla `train`, es descartada. Dicho de otra forma, de cruzar las tablas `train` y `countries`, daría lugar a una tabla donde la variable a predecir se habría enriquecido, habiendo una correlación muy grande entre estas variables, y donde el resto de las variables predictivas se habrían mantenido sin modificación. Debido a esta elección, no se procede al análisis estadístico de los datos de esta tabla.

Pese a haber descartado la tabla `countries` para el cruce, podría haber sido de utilidad si en vez de querer trabajar con un atributo de salida cualitativo nominal se hubiera querido trabajar con uno cuantitativo contínuo. Esto es posible porque las variables `lat_destination`, `lng_destination`, `distance_km` y `destination_km2` son únicas por cada país de destino, pudiendo sustituir como atributo de salida la variable `country_destination` por cualquiera de estas 4 opciones, pasando de un atributo de salida cualitativo a uno cuantitativo. En esta memoria se ha mantenido como variable a predecir `country_destination`.

4.4. Filtrado de la tabla age_gender

La tabla `age_gender` aporta información sobre datos estadísticos de la edad, el sexo y el país de destino de los usuarios de Airbnb (Airbnb, 2019b) en 2015 junto a la población de cada país elegido. Este dato podría ser útil para enriquecer el perfil demográfico que accede a cada uno de los países de destino, dicho de otra forma, podría servir para mejorar la calidad de las variables de `train`, enriqueciéndolas.

Para garantizar que cruzando `train` con `age_gender` se está incrementando la calidad

del dato, se va a analizar si efectivamente `age_gender` aporta con sus resúmenes estadísticos información extra. Para resolver esta incógnita se presenta las figuras 10 y 11.

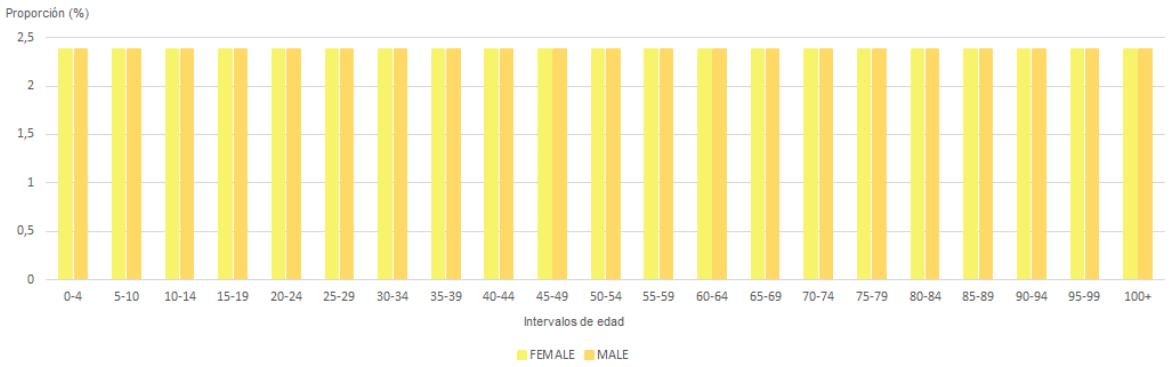

Figura 10: Gráfica de la distribución de la variable `age_gender` agregada (de todos los países), donde se aprecia que no hay variación.

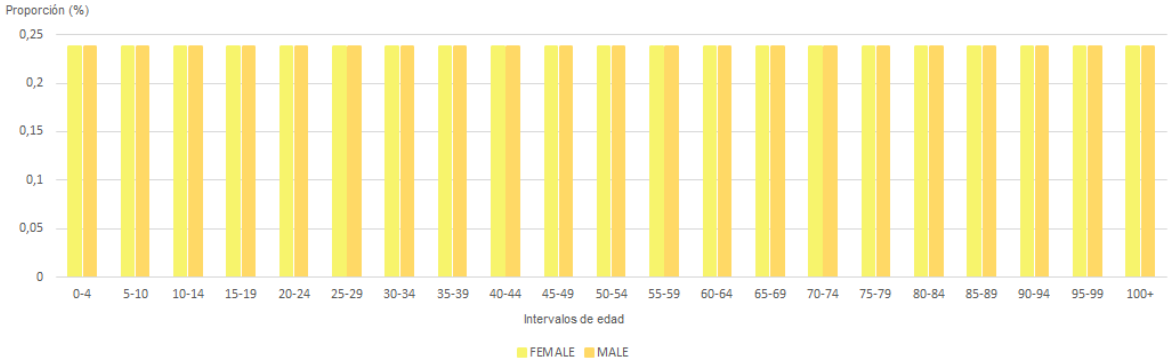

Figura 11: Gráfica de la distribución de la variable `age_gender` en cualquier país, donde se aprecia que no hay variación.

La primera de las figuras, la Figura 10, corresponde a la distribución de la proporción de los datos agrupados por edad y género teniendo en cuenta todos los posibles destinos vacacionales, es decir, trabajando con datos agregados, ofreciendo una visión general. Mientras tanto, la Figura 11 muestra la misma información pero esta vez para uno de los 10 posibles destinos (como para cada uno de los 10 destinos la gráfica no varía, se ha incluido sólo un gráfico en vez de 10 iguales). Si se presta atención a la información que se extrae de estas dos tablas, de ambas se puede deducir que la tabla `age_gender` no aporta nueva información. Esto se debe a que para ambos géneros y para todos los intervalos de edad, independientemente del país de destino, la proporción de registros que comparten dichas propiedades es siempre la misma, es decir, no hay variaciones.

Como la información que aporta esta tabla por edad, sexo y país de destino es la misma

para todas las categorías, se ha tomado la decisión de no cruzar la tabla `train` con `age_gender`. Explicando más esta decisión, se plantea un ejemplo suponiendo que se ha llevado a cabo el cruce: un usuario empieza a interactuar con la web de Airbnb y el primer modelo predice que tiene intenciones de reservar. De esta persona se conoce la edad y el sexo entonces, ¿cómo se le puede recomendar un primer destino si para estas dos variables la probabilidad de que vaya a cada uno de los posibles destinos es la misma? De este modo queda en evidencia que, añadir la misma información a todos registros, no añadirla o desconocer estos datos aporta el mismo conocimiento para responder a la pregunta de ¿qué primer destino es el más probable?.

Al no cruzar `train` con `age_gender`, esta tabla carece de interés para el ojetivo de la memoria y es descartada. Por consiguiente, no se va a llevar a cabo el análisis estadístico de los datos de `age_gender`.

4.5. Filtrado y transformación de la tabla train

La tabla `train` es el conjunto de datos que se va a emplear para predecir el primer destino de un nuevo usuario y poder hacerles, de este modo, la recomendación. La tabla está compuesta de 15 atributos de entrada y uno de salida, donde la mayoría de las variables de entrada son cualitativas nominales y la variable a predecir es también cualitativa nominal. Como `country_destination`, el atributo de salida, es cualitativo nominal y tiene más de 2 categorías, para saber cuál será el primer destino de un nuevo usuario se va a tener que resolver un problema de clasificación multinominal.

Antes de profundizar más en el estudio de las variables, se va a comentar en detalle el efecto que han tenido las decisiones que se han tomado hasta este momento. La reducción de registros no sólo ha supuesto pasar de 213.451 instancias a 88.908, sino que también la variable `date_first_booking` ha dejado de tener campos vacíos y se ha reducido la proporción de nulos en `age` y `first_affiliate_tracked`. Además, también ha acarreado la reducción de las categorías del atributo de salida porque, antes de la reducción, había 12 posibles destinos contando ”other” (otros) y ”NDF” (*not defined*) y, tras la reducción, quedan 11 al haber desechado todos los registros que tenían como país de destino ”NDF”. Por ello, a partir de ahora el objetivo va a ser predecir el próximo destino vacacional entre 11 y no 12 opciones.

Descartar la tabla `age_gender` también ha tenido efecto en la tabla `train` porque, si antes se habían agrupado las edades para poder comparar los registros de `train` con los de `age_gender`, ahora se deshace este cambio y se mantiene `age` como número entero. Además,

debido a la política de privacidad de Airbnb de 2019 (recordar que aunque los datos fueron recabados entre 2010-2014 la política de privacidad que se debe aplicar es la vigente en el momento del análisis de los datos (Martínez, 2019)), los menores de 16 años no pueden reservar apartamentos dentro de la web (Airbnb, 2019a). Es por ello que a la variable `age` de `train` se le asignará el valor de ”0” tanto a los valores nulos y fuera de rango (en vez de ”null” como sucedía tras categorizar la variable) como a las edades de entre 0 y 15 años. A continuación se presenta el fragmento de la política de privacidad de Airbnb donde se trata lo comentado (Airbnb, 2019a):

*”**2.2 Datos de menores**.*
Nuestros sitios web y aplicaciones no están dirigidos a menores de 16 años y no recopilamos ninguna información personal directamente de los menores de 16 años. Si usted cree que estamos tratando la información personal de un menor de forma inapropiada, nos lo tomaremos muy en serio y le instamos a que se ponga en contacto con nosotros utilizando la información proporcionada en la sección ”Contacto” que aparece a continuación.”

La siguiente tabla resume los cambios comentados hasta ahora para facilitar al lector la comprensión de los mismo.

train		
Atributos	**Formato**	**Variaciones**
id	String(10)	
date_account_created	Date	
timestamp_first_active	Date	
date_first_booking	Date	Eliminación de los registros nulos
gender	String	
age	Int	Modificación del format, reducción de nulos y aplicación de la politica de privacidad
signup_method	String	
signup_flow	Int	
language	String(2)	
affiliate_channel	String	
affiliate_provider	String	
first_affiliate_tracked	String	Reducción de datos nulos
signup_app	String	
first_device_type	String	
first_browser	String	
country_destination	String(2)	Reducción de categorías, de 12 a 11 (”NDF” desaparece)

Tabla 13: Modificaciones efectuadas en la tabla `train` reducida, compuesta de 88.908 registros en vez de 213.451.

Por otro lado, en el estado del arte ya se comentó que se iban a eliminar algunas variables de la base de datos para garantizar la originalidad de la memoria y diferenciar este trabajo de las soluciones que se propusieron en la competición de Kaggle (Kaggle, 2019). Las variables que se van a eliminar pertenecen a esta tabla y son `signup_method` y `signup_flow`. Esta elección se ha tomado teniendo en cuenta el significado de cada atributo y no mediante algún algoritmo de selección de variables porque muchas de las técnicas de *machine learning* que se van a aplicar a continuación ya tienen incorporado un proceso de selección de atributos, descartando el mismo algoritmo las variables menos relevantes para el objetivo.

Una vez que se tiene la tabla `train` completamente filtrada, tanto por el número de registros (para tener datos de calidad) como por las variables (por cuestión de resultados únicos), se procede a la transformación de las variables. En esta parte se va a estudiar cada variable y las formas que hay para enriquecer los datos. Dando comienzo por las variables de tipo fecha, se observa la siguiente tendencia a lo largo del tiempo en las Figuras 12, 13 y 14.

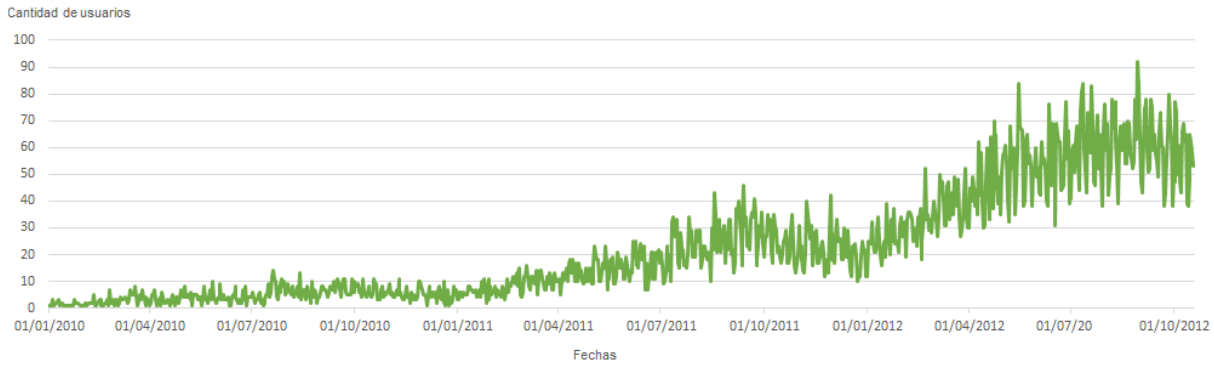

Figura 12: Serie temporal de la creación de cuenta de los usuarios en `train`, donde la tendencia indica que cada vez se crean más cuentas.

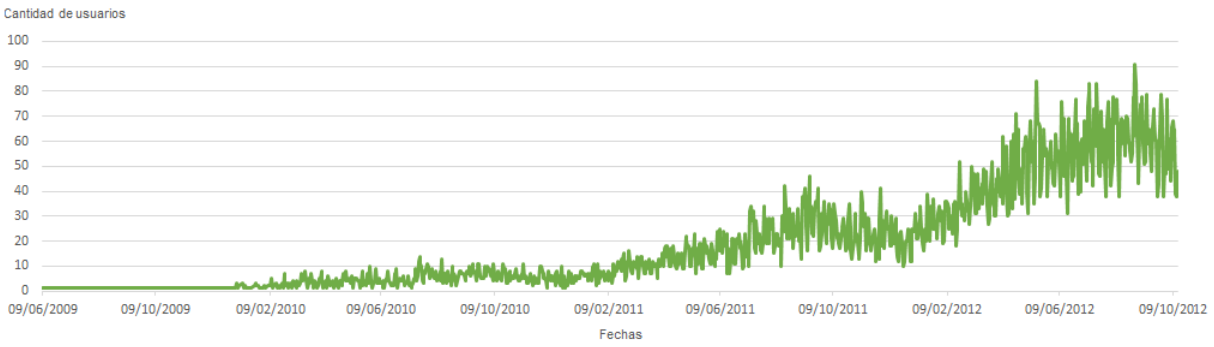

Figura 13: Serie temporal de la primera actividad de los usuarios en `train`.

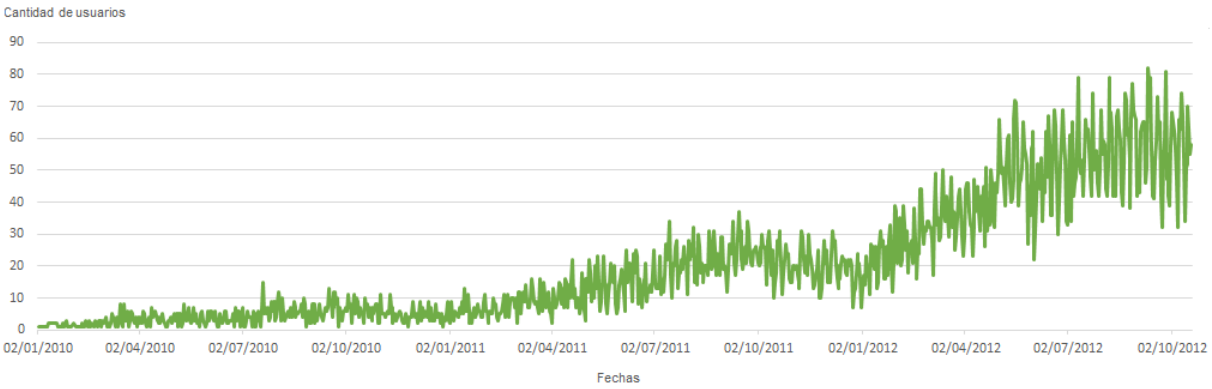

Figura 14: Serie temporal de la primera reserva de los usuarios en `train`.

Se puede apreciar que para las tres variables de tipo fecha, `date_account_created`, `timestamp_first_active` y `date_first_booking`, hay una tendencia de que aumente el número de primeros usuarios que crean una cuenta y reservan a medida que se avanza en el tiempo. Asimismo, en enero de 2012 se percibe un descenso de estas actividades que rápidamente se recupera en los siguientes meses. Por otro lado, como las tres gráficas son muy parecidas, puede que haya una correlación alta en estas tres variables, correlación que será estudiada más adelante.

Vista la información de esta forma, no es posible conocer, por ejemplo, el mes del año en el que más cuentas se crean, el mes donde más primera actividad hay y el mes en el que se llevan a cabo más reservas. Para dar respuesta a estas preguntas, se han llevado a cabo las tres siguientes figuras.

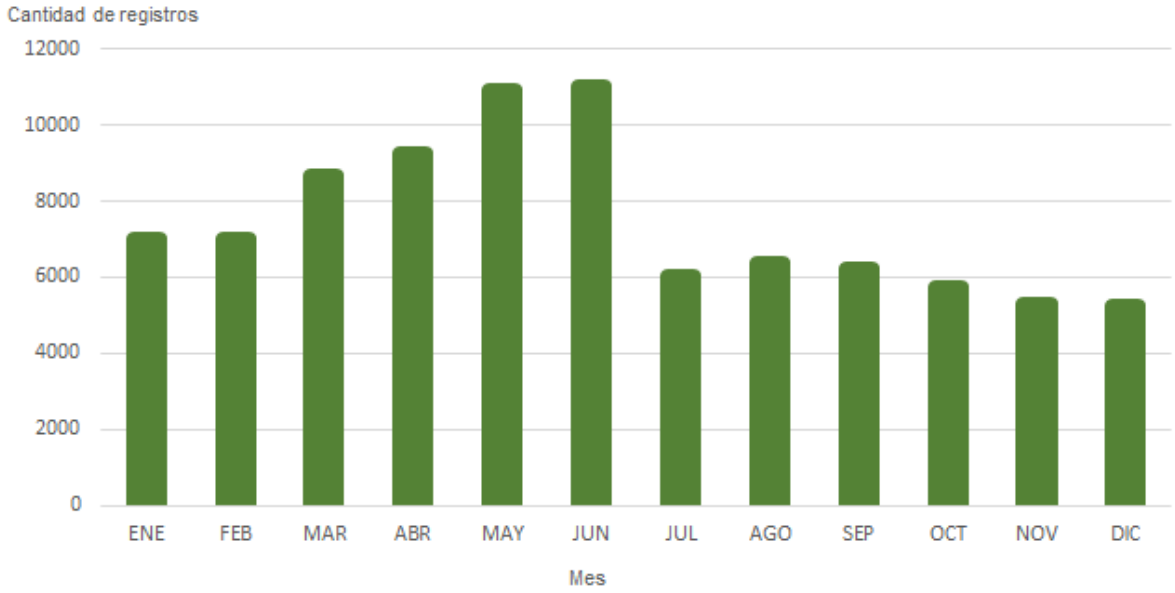

Figura 15: Gráfica de barras de la distribución mensual de la creación de una cuenta para los usuarios en `train`, incrementando en el primer semestre del año y disminuyendo ligeramente en el segundo.

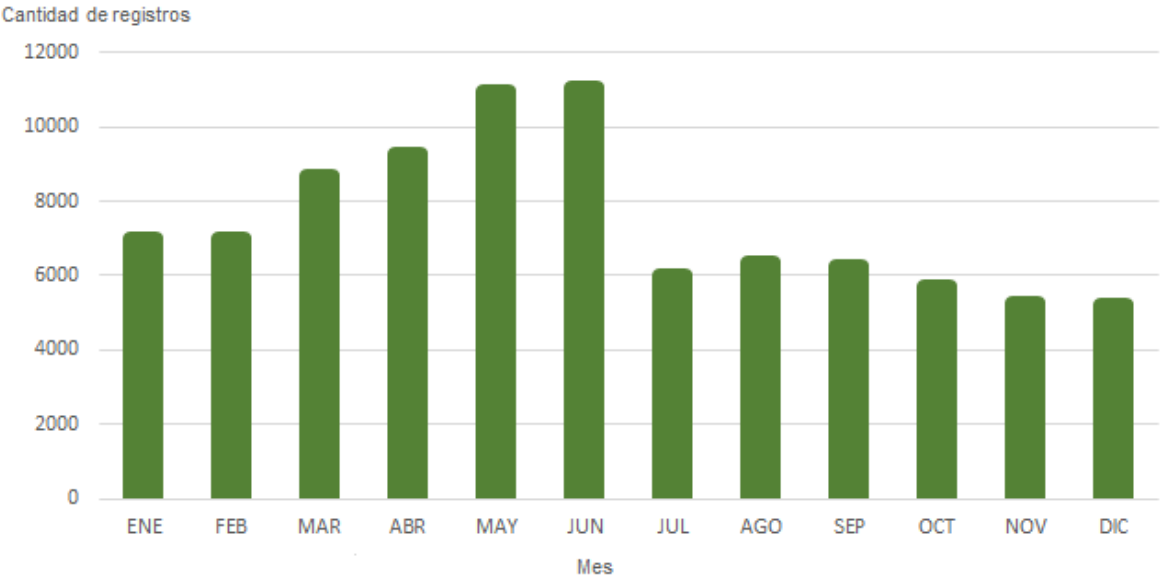

Figura 16: Gráfica de barras de la distribución mensual de la primera actividad de los usuarios en `train`, incrementando en el primer semestre del año y disminuyendo ligeramente en el segundo.

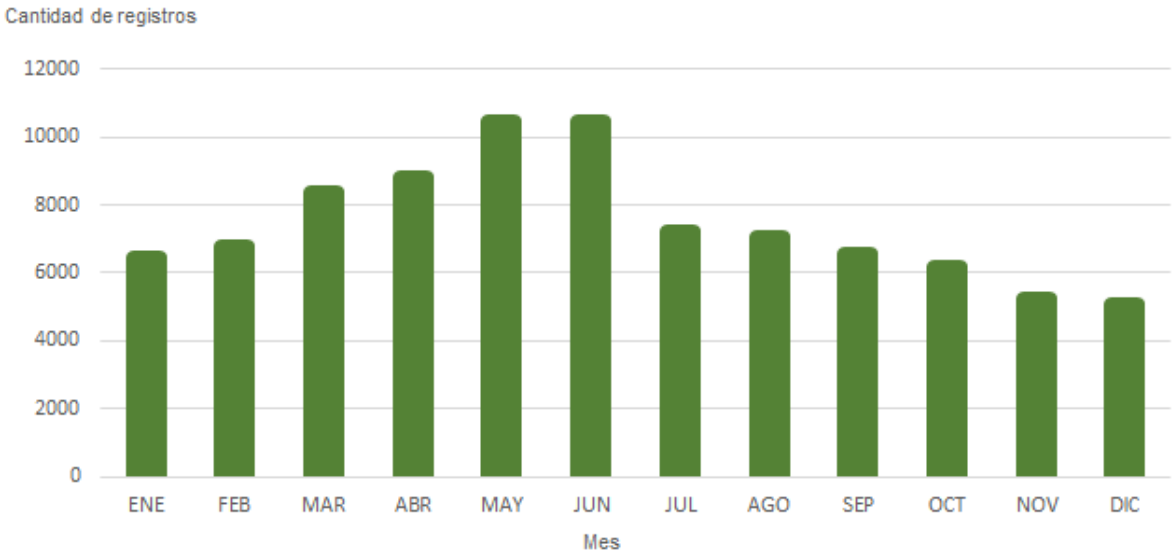

Figura 17: Gráfica de barras de la distribución mensual de la primera reserva de los usuarios en `train`, incrementando en el primer semestre del año y disminuyendo ligeramente en el segundo.

En las Figuras 15, 16 y 17 se puede apreciar que en las tres situaciones los meses de mayor cantidad de nuevas cuentas, de primera actividad y de primera reserva son junio, mayo, abril y marzo, es decir, los meses más próximos al verano. Por contrario, los meses de menor actividad son noviembre y diciembre. Además, el número de primeras reservas no decae tanto en julio y agosto como lo hace para la creación de una cuenta y la primera actividad.

Las variables de tipo fecha permiten extraer mucha información acerca del comportamiento de los usuarios como se ha podido comprobar hasta el momento. Para enriquecer los datos de `train`, a parte del tratamiento de los valores nulos que se le hizo, se van a definir nuevas variables para extraer toda la información posible de los atributos de tipo fecha. Las nuevas variables, junto a las que ya estaban, se resumen en la Tabla 14, resaltando las nuevas con naranja.

train		
Atributos	**Formato (longitud)**	**Descripción**
id	String(10)	Identificador del usuario
date_account_created_2009	Int	Diferencia en días entre la fecha de creación de la cuenta y el 1 de enero de 2009
timestamp_first_active_2009	Int	Diferencia en días entre la fecha de la primera actividad del usuario y el 1 de enero de 2009
date_first_booking_2009	Int	Diferencia en días entre la fecha de la primera reserva y el 1 de enero de 2009
dif_first_active_created	Int	Diferencia en días entre la creación de la cuenta de Airbnb y su primera actividad
dif_create_first_booking	Int	Diferencia en días entre la creación de la cuenta de Airbnb y su primera reserva
dif_first_active_booking	Int	Diferencia en días entre la primera actividad de la cuenta de Airbnb y su primera reserva
month_account_created	Int	Mes de la fecha en la que se creó la cuenta
month_first_active	Int	Mes de la fecha de la primera actividad
month_first_booking	Int	Mes de la fecha de la primera reserva
year_month_account_created_2009	Int	Diferencia entre el primer día del mes en el que se creó la cuenta y el 1 de enero de 2009
year_month_first_active_2009	Int	Diferencia entre el primer día del mes en el que la cuenta tuvo actividad por primera vez y el 1 de enero de 2009
year_month_first_booking_2009	Int	Diferencia entre el primer día del mes en el que se reservó por primera vez y el 1 de enero de 2009

gender	String	Género
age	Int	Edad
language	String(2)	Idioma
affiliate_channel	String	Canal de afiliación (el tipo de marketing de pago)
affiliate_provider	String	Proveedor de afiliación (dónde se desarrolló el marketing: google, . . .)
first_affiliate_tracked	String	Determina la primera comercialización con la que interactuó el usuario antes de registrarse
signup_app	String	Aplicación del registro
first_device_type	String	Tipo del primer dispositivo
first_browser	String	Primer navegador
country_destination	String(2)	País de destino

Tabla 14: Contenido de la tabla `train` enriquecida, compuesta de 88.908 registros.

En la información teórica relativa al tipo de variable con la que se puede trabajar en los algoritmos de *machine learning* que se van a aplicar en la Sección 5, se especificada que permiten tanto variables cualitativas como cuantitativas de entrada. Como no comentan nada sobre las variables de tipo fecha, se ha tomado la decisión de convertir estas variables en números, calculando en cada caso la diferencia en días entre la fecha y el 1 de enero de 2009, al ser éste el comienzo de año más próximo a la fecha mínima entre `date_account_created`, `timestamp_first_active` y `date_first_booking`. Así mismo, se ha calculado la diferencia entre estas tres fechas para calcular el tiempo empleado por el usuario entre cada una de las acciones y el mes donde se llevó acabo. Finalmente, se ha calculado la diferencia de inicio de mes de cada una de las acciones y el 1 de enero de 2009 para tener en días el mes y el año en el que la acción tuvo lugar. De este modo, quedan desglosadas las variables de tipo fecha para extraer la máxima información posible y enriquecer la tabla.

En cuanto a los atributos correspondientes a la edad y al sexo, la edad toma valores entre los 16 y los 123 años, habiendo valores nulos, y la media se encuentra en los 29 años con una desviación típica de 20 años. La variable `gender` consta de cuatro categorías, resumidas en la siguiente tabla.

Variable género de train	
Género	**Número de apariciones**
female	31 993
-unknown-	29 018
male	27 721
other	176

Tabla 15: Contenido de la variable `gender` de la tabla `train` enriquecida, compuesta de 88.908 registros.

En la Tabla 15 se ve que en `train` hay más mujeres que hombres, estando como la segunda categoría más frecuente la de género desconocido y hay pocos registros para el sexo ”other”. Para estudiar los atributos de edad y sexo en conjunto, se ha analizando la distribución de la edad por cada género mediante los gráficos que vienen a continuación. Cabe mencionar que la edad se ha agrupado por intervalos sólo para resumir la variable de forma visual y no porque el formato de `age` sea ese (recordar que la edad primero se agrupó por intervalos para luego deshacer este cambio y dejarlo con el formato de los números enteros).

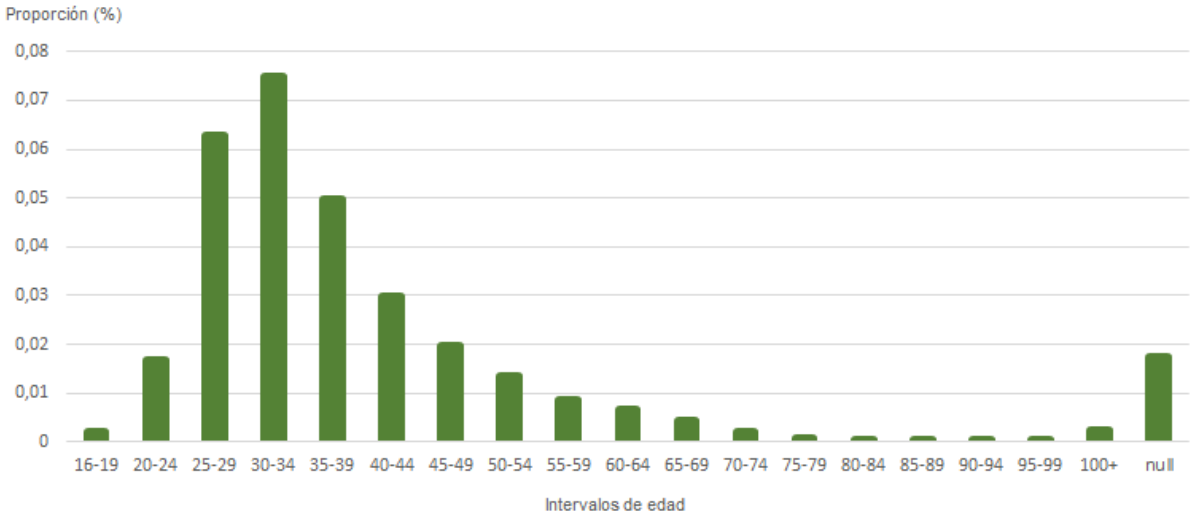

Figura 18: Gráfica de barras de la distribución del género ”male” en la tabla `train`.

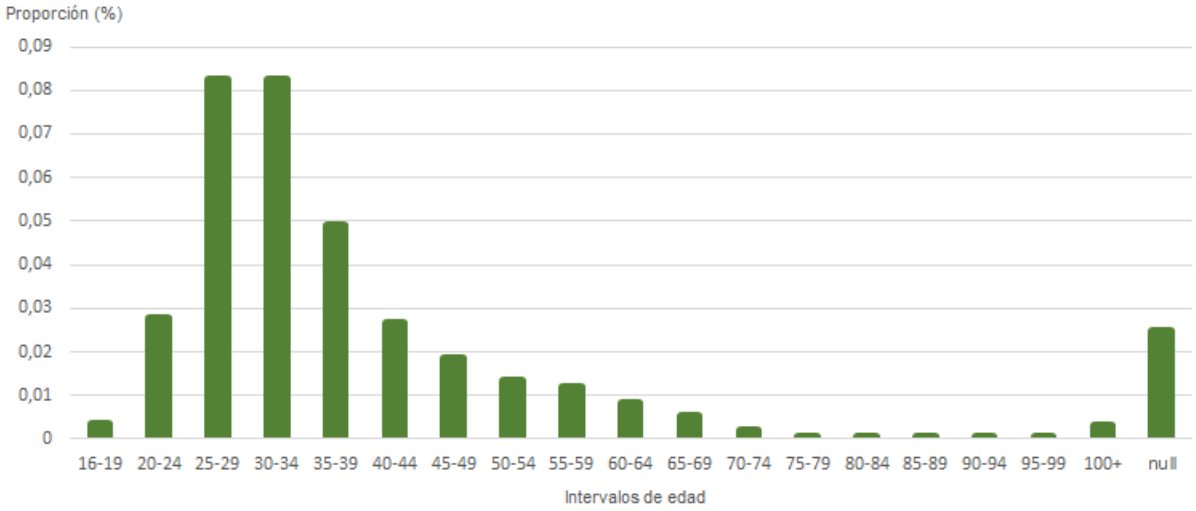

Figura 19: Gráfica de barras de la distribución del género ”female” en la tabla `train`.

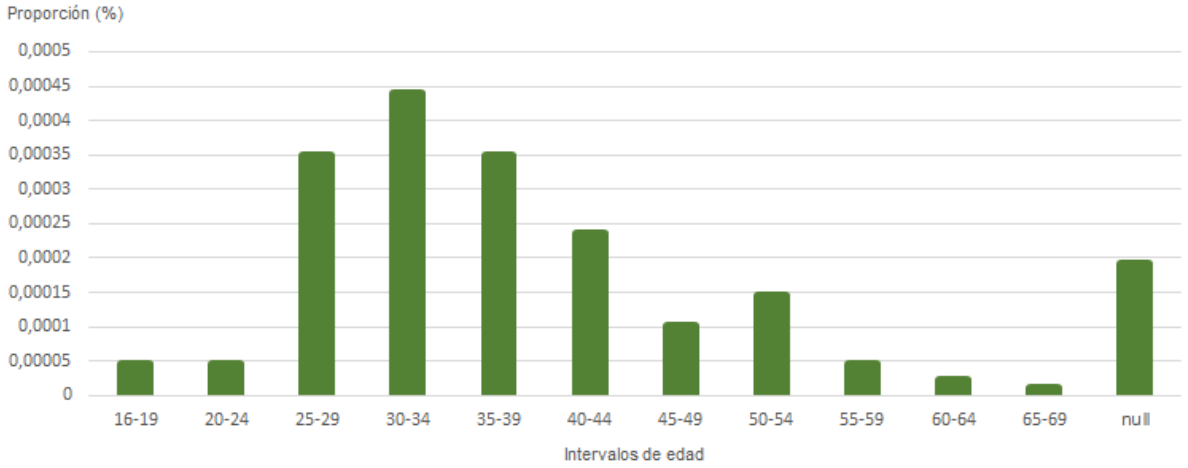

Figura 20: Gráfica de barras de la distribución del género ”other” en la tabla `train`.

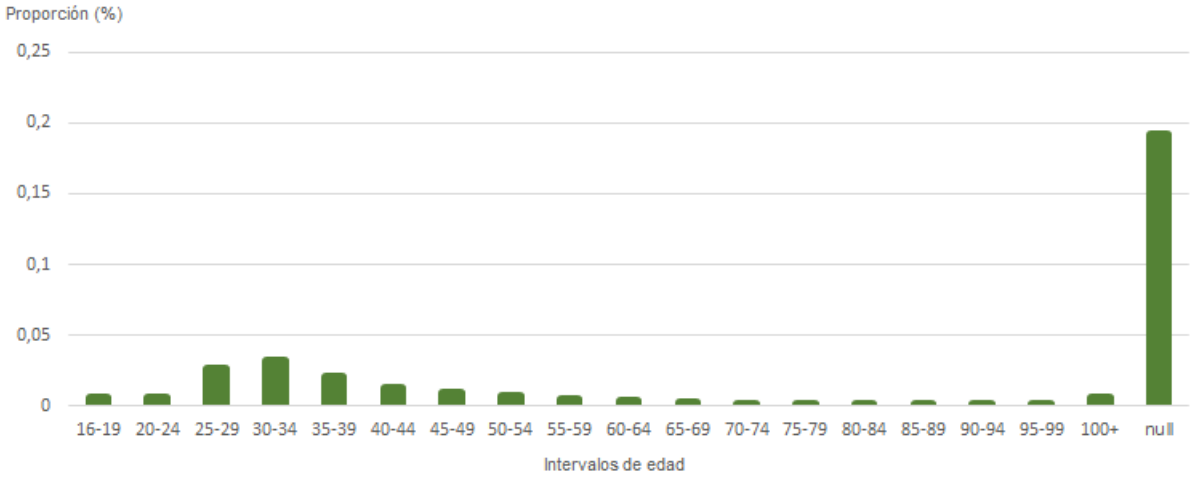

Figura 21: Gráfica de barras de la distribución del género ”unknown” en la tabla `train`.

A partir de las Figuras 18, 19, 20 y 21 se da a conocer que la mayoría de los usuarios masculinos y femeninos se encuentran en el intervalo de edad entre los 25 y los 39 años. Además, la cantidad de valores nulos es especialmente elevada para el sexo ”unknown” al estar cerca del 20 %, mientras que en el resto de categorías se mantiene por debajo del 2 %. Finalmente, en las cuatro categorías de `gender` se sigue el mismo patrón: la cantidad de usuarios disminuye a partir de que la edad supera los 50-55 años.

Acerca del resto de variables, salvo el atributo de salida, se resumen tras este párrafo lo más destacable de cada una, completando esta información en el anexo, Sección 9. Se ha tomado esta decisión porque, como son todas cualitativas con alrededor de 10 categorías o más, la carga descriptiva de este apartado es muy grande, derivando este contenido al apéndice. Seguidamente, se exponen los atributos comentados en la Tabla 16 de forma resumida.

Variables restantes de train		
Atributo	**Número de categorías**	**Categoría más frecuente**
language	23	en
affiliate_channel	8	direct
affiliate_provider	17	direct
first_affiliate_tracked	8	untracked
signup_app	4	Web
first_device_type	9	Mac Desktop
first_browser	29	-unknown-

Tabla 16: Contenido de las variables restantes de la tabla `train` enriquecida, compuesta de 88.908 registros.

Finalmente, a la hora de utilizar un algoritmo de aprendizaje supervisado, es importante que en la fase de entrenamiento del modelo todas las categorías de la variable que se quiere aprender a predecir aparezcan de forma equitativa. De no ser así, el algoritmo que se logra no tendrá una capacidad de predicción equiprobable entre los posibles resultados, teniendo una habilidad de predicción mejor o peor en función de la cantidad de instancias recibidas para cada categoría en la fase de entrenamiento (a mayor cantidad de instancias de una categoría mayor capacidad de predicción).

En cuanto al conjunto de datos `train` enriquecido, los posibles países de destino son AU (Australia), CA (Canadá), DE (Alemania), ES (España), FR (Francia), GB (Gran Bretaña), IT (Italia), NL (Holanda), PT (Portugal), US (Estados Unidos) y ”other” (otro posible destino). Para que el algoritmo de aprendizaje supervisado que mejor se adapte a los datos pueda predecir igual de bien cada uno de los posibles destinos, en `train` deberían aparecer con una proporción del 8,33 % cada uno de los posibles destinos. En la Figura 22 se ve si esto ocurre.

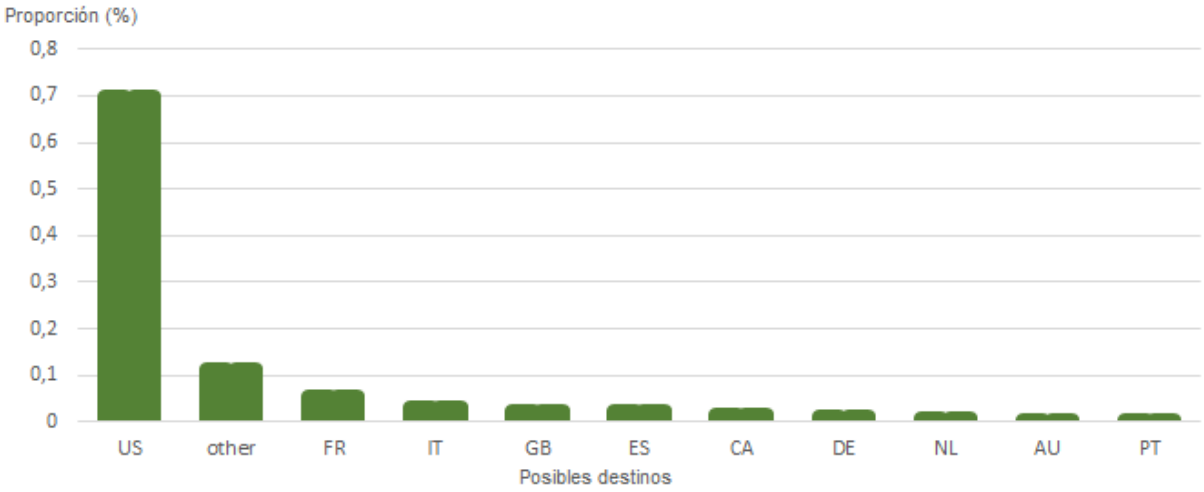

Figura 22: Gráfica de barras con la proporción de la elección de cada destino vacacional de los usuarios en `train`.

En la figura superior se observa que algo más del 70 % de los usuarios viaja dentro de Estados Unidos, seguido en segundo lugar por el destino ”other” con poco más del 10 % y en tercer lugar Francia al recibir el 6 % de los usuarios. El resto de destinos reciben un 3 % de los usuarios en el caso de Italia, Reino Unido y España, un 2 % Canadá, un 1 % Alemania, Holanda y Australia y menos de un 1 % Portugal.

Es evidente que no todas las categorías del atributo de salida presentan la misma proporción, es decir, el conjunto de datos de entrenamiento no está equilibrado. Por consiguiente, el modelo predictivo que se va a llevar a cabo con el conjunto de datos `train` va a tener una mayor capacidad predictiva para los destinos de Estados Unidos u ”other” que para Australia o Portugal, por ejemplo. Es por ello que, si se quiere igualar la capacidad predictiva del modelo se debería reducir el tamaño de `train` a 217 instancias por cada variable de salida o destino, siendo 217 la cantidad mínima de instancias que hay entre los posibles destinos. Como la cantidad de instancias correspondientes a cada variable de salida es resultado de la recopilación directa de datos en un intervalo de tiempo, no se va a modificar porque, si es verdad que los estadounidenses tienen más tendencia a viajar dentro de su país que fuera, el modelo al que se llegará por este planteamiento lo va a representar.

4.6. Filtrado y transformación de la tabla sessions

Por último, la tabla `sessions` contiene los registros de las interacciones que ha tenido cada usuario con la web de Airbnb (Airbnb, 2019a), teniendo en cuenta que un mismo usuario

puede tener más de una interacción en una sesión. Está compuesta de 6 variables, todas cualitativas nominales menos `sec_elapsed` que es cuantitativa continua. Esta tabla contiene información suficiente para predecir, en función del tipo de interacción, si un usuario tiene intención de reservar o no. Para ello, es necesario una varible a predecir que, aunque por el objetivo del TFM está claro cuál debe ser, no está incluida en la tabal todavía.

Debido al objetivo para el que se va a emplear esta tabla, la variable a predecir se va a llamar `book` y va a ser un atributo de salida cualitativo nominal con dos categorías posibles: ”Y” (*yes*) y ”N” (*No*). Es decir, el atributo `book` va a ser una variable booleana. De este modo, para anteponerse a la intencionalidad del usuario se va a tener que solucionar un problema de clasificación binomial. En la Tabla 17 se resume lo comentado, resaltando con naranja la nueva variable.

sessions		
Atributos	**Formato (longitud)**	**Descripción**
user_id	String(10)	Identificador del usuario
action	String	Acción realizada
action_type	String	Tipo de acción
action_details	String	Descripción de la acción
device_type	String	Tipo de dispositivo
secs_elapsed	Float	Segundos transcurridos para realizar la acción
book	String	Intención del usuario en reserva (Y/N)

Tabla 17: Contenido de la tabla `sessions` enriquecida.

Para computar los valores de ”Y” y ”N” del atributo de salida a cada registro, primero se van a comparar los identificadores de los usuario de las tablas `sessions`, `train` (la no reducida de 213.451 registros) y `test`. De este modo, a los usuarios que tenga `sessions` en común con `train` y `test` se les asignaría ”Y” porque en estas últimas dos tablas aparecen los usuarios que han reservado, y ”N” a los que solamente estén en `sessions`. La cantidad de usuarios que guardan en común estas tres tablas se resume en la Tabla 18.

Cruces			
Tablas	**sessions**	**train**	**test**
sessions	135.484	73.815	61.668
train	73.815	213.452	0
test	61.668	0	62.097

Tabla 18: Cantidad de registros de las tablas `train`, `test` y `sessions`.

Debido a que `sessions` se compone de 135.484 identificadores de usuarios diferentes, de

entre los cuales 73.815 comparte con `train`, 61.668 con `test` y en total con `sessions` comparte con `train` y `test` a la vez 135.483 usuarios, es decir sólo hay un usuario de `sessions` que no lo tienen en común. Dicho de otra forma, la tablas `train` y `test` aportan información sobre usuarios que han terminado reservando, mientras que `sessions` contiene datos de usuarios en general. Si `train` y `test` comparten con `sessions` 135.483 de los 135.484 usuarios que tiene la última, esta situación imposibilita crear la fase previa del sistema de recomendación que permitiría predecir cuándo un usuario tiene intención en reservar y cuándo no en función de su interacción. A pesar de que con los datos de los que se dispone no se puede llegar a este objetivo, se añadirá en las líneas futuras de trabajo, Sección 8, como idea pendiente de desarrollar.

Como no es posible construir un modelo para predecir el interés por reservar de un usuario con 135.483 usuarios en los que sí tienen intención frente a uno que no, la variable `book` es descartada, desarrollando los siguientes apartados de la memoria manteniendo esta decisión. Aún así, la tabla `sessions` puede ser de utilidad, ya que sirve para enriquecer el objetivo de predecir el primer destino de un nuevo usuario y construir un sistema de recomendación.

`Sessions` se compone de los atributos cualitativos `user_id` (necesario para el cruce entre `train` y `sessions`), `action` (con 359 categorías diferentes), `action_type` (compuesta de 11 categorías dispares), `action_detail` (con 155 categorías distintas) y `device_type` (con 14 categorías desemejantes). Para tener mayor detalle sobre las categorías de cada variable se recomienda al lector acudir al anexo, Sección 9, donde se resumen todas las variables cualitativas en diferentes tablas. La única variable numérica de la tabla es `secs_elapsed`, cuyo valor mínimo y máximo son 0 y 179.997 segundos, respectivamente. Tiene como media 19.156 segundos y como mediana 1.108 segundos, es decir, se aprecia que la mayoría de las interacciones han durado pocos segundos mientras que hay unas pocas que han durado mucho más, de ahí la diferencia tan grande entre la media y la mediana.

Hasta ahora se han llevado a cabo el análisis exploratorio de los datos, su limpieza y el filtrado y las transformaciones de las variables en las tablas oportunas. En el siguiente apartado se va a construir la tabla (se adelanta que en esta memoria van a ser dos tablas) a la que se le aplicará las técnicas de *machine learning*, profundizando más en el estudio estadístico de las variables de las que se compondrá.

4.7. Preparación de las tablas para modelizar: integración y reducción

El objetivo de esta memoria es el de hacer un sistema de recomendación que sugiera a un nuevo usuario el destino más probable al que puede acudir, es decir, su primer destino. En un principio, la idea era añadir una fase predictiva para predecir la intención de reservar del usuario y, de salir afirmativa, hacerle la recomendación del destino. Como se ha podido comprobar, no hay registros suficientes para modelizar las intenciones de reservar de los usuarios pero sí que hay instancias suficientes para predecir el primer destino. Por ello, en este apartado se van a combinar las tablas que se han considerado que aportan información para afrontar el problema de la recomendación.

La tabla que contiene tanto información útil como la variable objetivo es `train`, que se va a cruzar con `sessions` para ser enriquecida. Como la tabla a mejorar es `train`, el cruce se puede hacer de dos formas: Añadiendo a `train` la información de la tabla `sessions` (*left join*) o cruzando ambas tablas por los identificadores que tienen en común (*join*). Para tomar esta decisión, se va a estudiar la calidad del dato que se obtendría en ambos cruces. La Figura 23 resume gráficamente cada una de las combinaciones posibles de cruce de tablas cruzando por el identificador del usuario.

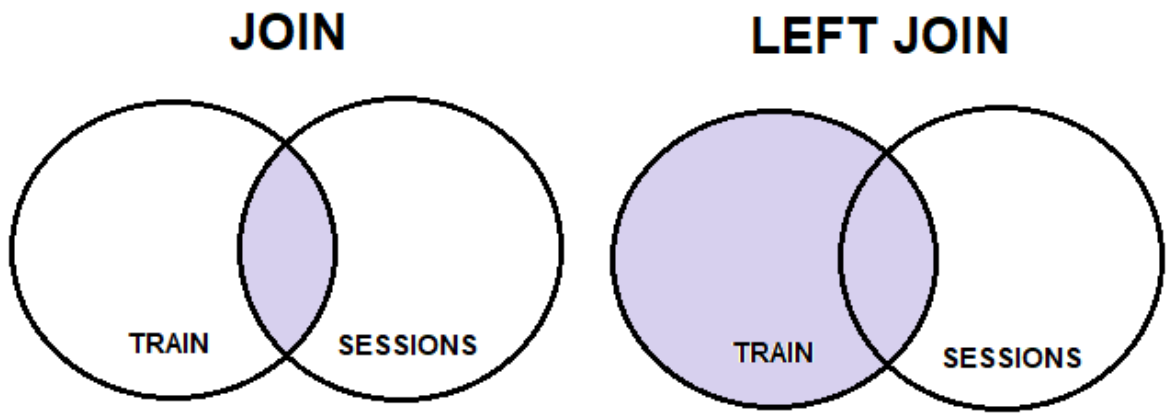

Figura 23: Diagrama de la diferencia entre los cruces con *join* y con *leftjoin*.

Por otro lado, la tabla `sessions` aporta información muy minuciosa. Para analizar si es necesario un nivel de detalle tan profundo para lograr buenas predicciones, se va a cruzar `train` con dos versiones de `sessions`, aplicando en ambos cruces las técnicas de *machine learning*:

1. `sessions1`: La tabla `sessions` sin variaciones.
2. `sessions2`: La tabla resumen de `sessions`, compuesta sólo de tres variables cuantitativas:
   - `num_actions`: atributo cuantitativo discreto que representa la cantidad de acciones que ha tenido el usuario previas a la reserva.
   - `sum_secs_elapsed`: variable cuantitativa contínua que informa sobre el tiempo total, en segundos, que ha transcurrido en todas las acciones previas a la reserva.
   - `secs_per_action`: atributo cuantitativo que mide la media de segundos empleados en cada acción.

De este modo, la tabla `train` se crucará primero con `sessions1` mediante un *join* y luego con un *left join* y, tras decidir cuál de las opciones tiene mejores datos, se hará lo mismo para `sessions2`. Los resultados de la calidad del dato (medida que se anañizará a través de la proporción de valores nulos) que se tiene entre los dos cruces de `train` y las dos posibilidades de la tabla `sessions` se resumen en las Tablas 19 y 20, apareciendo sólo los atributos cuya proporción de nulos es distinta de cero (el identificador del usuario se ha eliminado tras el cruce al considerar que para la predicción no aporta información).

train JOIN sessions1	
Atributos	**Proporción de nulos**
age	17,43 %
first_affiliate_tracked	0,001 %
action	1,18 %
action_type	11,34 %
action_detail	11,34 %
secs_elapsed	2,88 %

Tabla 19: Proporción de los valores nulos en el cruce `train` JOIN `sessions1`, compuesto de 2.480.247 registros en vez de 2.540.381.

train LEFT JOIN sessions1	
Atributos	**Proporción de nulos**
age	17,59 %
first_affiliate_tracked	0,07 %
action	3,53 %
action_type	13,53 %
action_detail	13,53 %
device_type	2,36 %
secs_elapsed	5,18 %

Tabla 20: Proporción de los valores nulos en el cruce `train` LEFT JOIN `sessions1`, compuesto de 2.540.381 registros.

Ambos cruces presentan en todas las variables una proporción de nulos menor al 50 % y no hay una abrupta reducción de registros. Mientras que la Tabla 19 consta de una proporción de nulos más baja que la Tabla 20, también tiene 100.000 registros menos. Ante esta situación de tablas resultantes son tan parecidas que se elige la Tabla 20, la correspondiente al *left join*, al considerarla la más completa por tener más registros y una baja proporción de nulos.

En cuanto al cruce entre `train` y `sessions2`, las tablas de calidad del dato resultantes son las siguiente.

train JOIN sessions2	
Atributos	**Proporción de nulos**
age	20,65 %
first_affiliate_tracked	0,02 %
sum_secs_elapsed	1,40 %
secs_per_action	1,40 %

Tabla 21: Proporción de los valores nulos en el cruce `train` JOIN `sessions2`, compuesto de 25.119 registros en vez de 88.908.

train LEFT JOIN sessions2	
Atributos	**Proporción de nulos**
age	23,36 %
first_affiliate_tracked	1,95 %
numb_actions	67,63 %
sum_secs_elapsed	68,08 %
secs_per_action	68,08 %

Tabla 22: Proporción de los valores nulos en el cruce `train` LEFT JOIN `sessions2`, compuesto de 88.908 registros.

En este caso la Tabla 22 tiene variables cuya proporción de nulos es superior al 50 %, es decir, que para estas variables el cruce aporta más valores vacíos que valores con información. Debido a esto y aunque en el Tabla 21 haya menos registros, se elige como tabla de mayor calidad la Tabla 21, creada a partir del *join* de las tablas.

Una vez se tienen las tablas cruzadas, se procede al análisis de las asociaciones que puede haber entre las variables. Para medir la relación entre las variables cuantitativas se va a utilizar el coeficiente de correlación de Pearson (Pearson, 1895), mientras que para dos variables cualitativas se empleará el coeficiente V de Cramer (Crámer, 1946). Para la asociación entre variables mixtas, es decir, una variable cualitativa y otra cuantitativa, se aplicará el test de Kruskall-Wallis (Kruskal y Wallis, 1952).

A partir de las siguientes subsecciones se va a explicar, primero, la técnica empleada

para medir la asociación entre cada tipo de variable y después, los resultados conseguidos tras aplicarlas en las dos tablas cruzadas (`train` *left join* `sessions1` y `train` *join* `sessions2`).

4.7.1. Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es una técnica estadística para variables cuantitativas cuya finalidad es la de medir el nivel de relación lineal que guardan dos variables (Ríus, 1998). La fórmula con la que se mide esta fuerza de relación lineal es la siguiente, siendo x e y las variables cuya linealidad se quiere medir (Pearson, 1895).

$$\mathsf{r}_{xy} = \frac{1}{n} \sum \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right),$$

donde x_i e y_i son cada uno de los valores de x e y respectivamente, $\bar{x}$ e $\bar{y}$ las medias de las variables y s_x y s_y las desviaciones estándar.

El coefiente de correlación puede tomar cualquier valor entre -1 y 1, ambos inclusive (Ríus, 1998). En función del valor que corresponda a r_{xy}, se extrae la siguiente información (Solutions, 2019) :

- Los valores positivos de r_{xy} indican que la relación es positiva, es decir, cuando x aumenta y también lo hace y viceversa.
- Los valores negativos de r_{xy} expresan que la relación es negativa, dicho de otra forma, cuando x aumenta/decrece y decrece/aumenta y viceversa.
- Si $|r_{xy}| = 1$: indica que existe una correlación perfecta, indicando una dependencia lineal completa.
- Si $0,5 \leq |r_{xy}| < 1$: existe un fuerte grado de correlación entre las variables.
- Si $0,3 \leq |r_{xy}| \leq 0,49$: hay un grado de correlación moderado.
- Si $0 < |r_{xy}| \leq 0,29$: el grado de correlación es bajo.
- Si $r_{xy} = 0$: no existe correlación lineal y las variables son del todo independientes.

A continuación se van a presentar dos tablas, Tabla 23 y **??** una para cada tipo de cruce. En ellas se muestra la correlación entre las variables cuantitativas de cada cruce, marcando con una X los casos en el que la correlación entre variables es igual o superior a 0,3, es decir, cuando hay una correlación de grado medio, alto o perfecta (se ha optado por esta representación debido al número de variables y por mejor visualización).

train LEFT JOIN sessions1														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	X	X	X							X	X	X		
2	X	X	X							X	X	X		
3	X	X	X		X	X				X	X	X		
4				X										
5			X		X	X						X		
6			X		X	X						X		
7							X	X	X					
8							X	X	X					
9							X	X	X					
10	X	X	X							X	X	X		
11	X	X	X							X	X	X		
12	X	X	X		X	X				X	X	X		
13													X	
14														X

Tabla 23: Coeficiente de Pearson de las variables del cruce `train` LEFT JOIN `sessions1`.

En esta tabla, los números que aparecen en la primera fila y columna que van del 1 hasta el 14 corresponden a las variables cuantitativas de `train` y `sessions1` siendo, respectivamente, las siguientes: `date_account_created_2009`, `timestamp_first_active_2009`, `date_first_booking_2009`, `dif_first_active_created`, `dif_create_first_booking`, `dif_first_active_booking`, `month_account_created`, `month_first_active`, `month_first_booking`, `year_month_account_created_2009`, `year_month_first_active_2009`, `year_month_first_booking_2009`, `age` y `secs_elapsed`. Además, analizando los datos, se descubre que las variables `month_account_created` y `month_first_active` son iguales.

train JOIN sessions2															
	1	2	3	5	6	7	8	9	10	11	12	13	14	15	16
1	X	X	X			X	X	X	X	X	X				
2	X	X	X			X	X	X	X	X	X				
3	X	X	X	X	X	X	X	X	X	X	X				
5			X	X	X			X			X				
6			X	X	X			X			X				
7	X	X	X			X	X	X	X	X	X				
8	X	X	X			X	X	X	X	X	X				
9	X	X	X	X	X	X	X	X	X	X	X				
10	X	X	X			X	X	X	X	X	X				
11	X	X	X			X	X	X	X	X	X				
12	X	X	X	X	X	X	X	X	X	X	X				
13												X			
14													X	X	
15													X	X	
16															X

Tabla 24: Coeficiente de Pearson de las variables del cruce `train` JOIN `sessions2`.

Una vez más, los valores que aparecen en la primera fila y columna de la Tabla 24 son los valores cuantitativos de las variables `train` y `sessions2`, donde los valores del 1 al 16 corresponden a `date_account_created_2009`, `timestamp_first_active_2009`, `date_first_booking_2009`, `dif_first_active_created`, `dif_create_first_booking`, `dif_first_active_` `month_account_created`, `month_first_active`, `month_first_booking`, `year_month_account_created_2009`, `year_month_first_active_2009`, `year_month_first_booking_2009`, `age`, `numb_actions`, `sum_secs_elapsed` y `secs_per_action`, respectivamente. En esta tabla no aparece la variable `dif_first_active_created` (la número 4) porque todos sus valores son nulos, implicando que `date_account_created_2009` y `timestamp_first_active_2009` sean iguales, junto a `dif_create_first_booking` y `dif_first_active_booking`, `month_account_created` y `month_first_active`, y `year_month_account_created_2009` y `year_month_first_active_2009`.

A parte de las relaciones entre variables que se han comentado hasta ahora, para seleccionar las variables menos correlacionadas que se van a mantener en cada cruce se han seguido los siguientes criterios:

1. Si un atributo no guarda correlación con ningún otro salvo consigo mismo, se mantiene.
2. Si un atributo guarda relación con otros atributos:
   2.1. Para cada variable con las que guarda una correlación igual o superior a 0,3, calcular la suma.
   2.2. Entre las sumas, mantener el atributo que mayor resultado haya obtenido y rechazar el resto de atributos con los que esté correlacionado.
   2.3. Ante un empate, calcular la suma del coeficiente de correlación de Pearson de cada variable con la que se asocia menos del 0,3 y mantener la variable de menor resultado.
   2.4 Si quedan elementos restantes que han participado en la suma pero no han sido rechazados, volver al paso 2.1 para estos elementos y repetir hasta que no quede ninguno.

Este criterio se ha aplicado para reducir el número de variables correlacionadas, ya que aportan la misma información de forma distinta, y mantener en las tablas las variables que mejor resumen la información de las variables descartadas. Tras este procedimiento, las variables cuantitativas que quedan son:

Variables cuantitativas conservadas	
train LEFT JOIN sessions1	**train JOIN sessions2**
date_first_booking_2009	dif_create_first_booking
dif_first_active_created	month_first_active
month_account_created	age
age	sum_secs_elapsed
secs_elapsed	secs_per_action

Tabla 25: Variables cuantitativas preservadas para ambos cruces.

4.7.2. Coeficiente V de Cramer

El coeficiente V de Cramer sirve para medir el nivel de asociación entre dos variables cualitativas, siendo aplicable cuando los atributos tienen dos o más categorías (Aroztegi, 2017). Dicho de otra forma, es aplicable cuando se tiene una tabla de contingencia de 2x2, 3x3 o nxn. La fórmula con la que calcula la fuerza de la asociación es la siguiente (Crámer, 1946).

$$\text{V}=\sqrt{\frac{\chi^2}{n\cdot(min(f,c)-1)}},$$

donde $\chi^2 = \sum \frac{(f_e - f_t)^2}{f_t}$ (f_t es la frecuencia teórica y f_e la frecuencia empírica (Aroztegi, 2017)), n es el número de observaciones de la tabla, f el número de filas y c el número de columnas.

El coeficiente puede tomar valores entre 0 y 1, ambos inclusive (Crámer, 1946). En cuanto al significado que aporta cada valor de V, varía en función de la fuente consultada. Tomando como referencia una de las más conservadoras (Field, 2013), dependiendo del valor de V se tiene la siguiente información:

- Si $V = 1$: indica que hay una relación perfecta entre las dos variables.
- Si $0,6 < V < 1$: existe un fuerte grado de asociación entre las variables.
- Si $0,2 < V \leq 0,6$: hay una grado de asociación moderada.
- Si $0 < V \leq 0,2$: el grado de asociación es significativo pero bajo.
- Si $V = 0$: no existe relación entre ambas variables.

Debido al gran número de grupos que presentan los atributos, la técnica elegida para el análisis de la asociación entre variables cualitativas ha sido la V de Cramer, donde para las variables de dos categorías el resultado es el mismo que con el test de independencia chi cuadrado (Pearson, 1900) (Sánchez, 2000). A pesar de ello y siguiendo lo que se trató en el anexo, Sección 9, dada la cantidad de categorías de las variables y la baja frecuencia de aparición que presentaban algunas de ellas, se ha decidido agrupar en la categoría ”others”

todos los grupos cuya proporción de aparición en los registros de las tablas cruzadas sea menor al 1 %. Además, como es probable que la agrupación produzca registros repetidos, es decir, información duplicada, se han descartado las réplicas, conllevando a la reducción de instancias de cada tabla. De este modo, las variables cualitativas nominales presentan el siguiente número de categorías, resumido en las Tablas 26 y 27.

train LEFT JOIN sessions1		
Atributos	**Número de categorías original**	**Nuevo número de categorías**
gender	4	4
language	23	2
affiliate_channel	8	7
affiliate_provider	17	5
first_affiliate_tracked	8	5
signup_app	4	4
first_device_type	9	7
first_browser	40	7
action	359	21
action_type	10	8
action_detail	127	19
device_type	15	9

Tabla 26: Reducción de las categorías de las variables del cruce `train` LEFT JOIN `sessions1`, compuesto de 2.469.679 registros.

train JOIN sessions2		
Atributos	**Número de categorías original**	**Nuevo número de categorías**
gender	4	4
language	22	2
affiliate_channel	8	7
affiliate_provider	16	6
first_affiliate_tracked	8	5
signup_app	4	4
first_device_type	9	7
first_browser	29	7

Tabla 27: Reducción de las categorías de las variables del cruce `train` JOIN `sessions2`, compuesto de 28.774 registros.

La reagrupación de las categorías de las variables cualitativas de ambas tablas permite trabajar con categorías que cumplan un mínimo de frecuencia de aparición, manteniendo inalteradas las más relevantes, a parte de aportar simplicidad a los atributos. A continuación, se procede al cálculo del coeficiente V de Cramer para las variables cualitativas de `train` *left join* `sessions1` y para `train` *join* `sessions2`, resumiendo los resultados en las Tablas 28 y 29. Ade-

más, para mayor claridad, los valores cuya asociación es alta (ya que se trabaja con variables con categorías agrupadas) se representan con una X.

train LEFT JOIN sessions1												
	1	2	3	4	5	6	7	8	9	10	11	12
1	X											
2		X										
3			X	X								
4			X	X								
5					X							
6						X	X					X
7						X	X					X
8								X				
9									X	X	X	
10									X	X	X	
11									X	X	X	
12						X	X					X

Tabla 28: Coeficiente V de Cramer de las variables del cruce `train` LEFT JOIN `sessions1`.

Una vez más, se han codificado las variables con números para lograr una tabla más compacta. Los valores del 1 al 12 corresponde a las variables caulitativas `gender`, `language`, `affiliate_channel`, `affiliate_provider`, `first_affiliate_tracked`, `signup_app`, `first_device_type`, `first_browser`, `action`, `action_type`, `action_detail` y `device_type`, respectivamente.

train JOIN sessions2								
	1	2	3	4	5	6	7	8
1	X							
2		X						
3			X	X				
4			X	X				
5					X			
6						X	X	
7						X	X	
8								X

Tabla 29: Coeficiente V de Cramer de las variables del cruce `train` JOIN `sessions2`.

En cuanto a la Tabla 31, los valores que van del 1 al 8 corresponden a los atributos `gender`, `language`, `affiliate_channel`, `affiliate_provider`, `first_affiliate_tracked`, `signup_app`, `first_device_type` y `first_browser`, respectivamente.

Para decidir qué atributos son descartados de los cruces se han aplicado los siguiente criterios, análogos a los empleados en el estudio del coeficiente de correlación de Pearson

para las variables cuantitativas. Los criterios para este caso son los siguientes;

1. Si un atributo no guarda correlación con ningún otro salvo consigo mismo, se mantiene.
2. Si un atributo guarda relación con otros atributos:
   2.1. Para cada variable con las que guarda una asociación igual o superior a 0,6, calcular la suma.
   2.2. Entre las sumas, mantener el atributo que mayor resultado haya obtenido y rechazar el resto de atributos con los que esté correlacionado.
   2.3. Ante un empate, calcular la suma del coeficiente V de Cramer de cada variable con la que se asocia menos del 0,6 y mantener la variable de menor resultado.
   2.4 Si quedan elementos restantes que han participado en la suma pero no han sido rechazados, volver al paso 2.1 para estos elementos y repetir hasta que no quede ninguno.

De este modo, se muestran en la Tabla 30 las variables que cumplen las condiciones de permanencia.

Variables cualitativas conservadas	
train LEFT JOIN sessions1	**train JOIN sessions2**
gender	gender
language	language
affiliate_provider	affiliate_channel
first_affiliate_tracked	first_affiliate_tracked
first_browser	first_device_type
action_detail	first_browser
device_type	

Tabla 30: Variables cualitativas preservadas para ambos cruces.

Finalmente, mencionar que a partir de este punto se trabajará con las variables reagrupadas en el resto de la memoria.

4.7.3. Contraste de Kruskall-Wallis

La prueba, test o contraste de Kruskall-Wallis es una técnica no paramétrica, es decir, la prueba no asume que las variables provienen de una distribución en concreto, y permite determinar si existe correlación entre una variable cuantitativa y otra cualitativa (Kruskal y Wallis, 1952). Además, se trata de una extensión del test U de Mann-Whitney (Mann y Whitney, 1947) para más de dos categorías (Birnbaum, 1956). Generalmente se usa cuando no se puede

aplicar la técnica ANOVA de una vía (Fisher, 1922) debido a la falta de la normalidad del atributo cuantitativo, ya que el contraste de Kruskall-Wallis no necesita esa hipótesis (McDonald, 2014). A pesar de ello, hay estadísticos que creen que este test no es del todo preciso y, si se cumple la condición de normalidad, recomiendan aplicar la técnica ANOVA de una vía al ser más exacta (McDonald, 2014).

Las hipótesis para poder aplicar el contraste de Kruskall-Wallis son la independencia de las observaciones y que todos los grupos tengan la misma forma en la distribución (McDonald, 2014). Una vez se han verificado, se confrontan las siguientes hipótesis en el contraste:

$$\begin{cases} H_0: & Todas\ los\ grupos\ tienen\ la\ misma\ media. \\ H_1: & Todas\ los\ grupos\ no\ tienen\ la\ misma\ media. \end{cases}$$

donde el estadístico empleado es $H = \frac{12}{n(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(n+1)$, siendo n la cantidad de registros, k la cantidad de categorías, n_i con $i \in \{1, ..., k\}$ el número de observaciones por cada categoría y R_i con $i \in \{1, ..., k\}$ la suma de rangos por cada grupo (Kruskal y Wallis, 1952).

Si el p-valor (la probabilidad de que el estadístico tome el valor que ha tomado tras haber recogido la muestra (Aroztegi, 2017)) es menor o igual que α, siendo α el nivel de significación, es decir, la cantidad de error que estamos dispuestos a asumir (Canela, 2019), entonces se refuta la hipótesis nula, H_0. Por el contrario, si el p-valor es menor que α, no hay evidencias estadísticas suficientes para refutar H_0, asumiendo que todas las categorías tienen la misma media y estando las dos variables asociadas (Aroztegi, 2017).

En cuanto a las dos tablas cruzadas, se les ha aplicado el test de Kolmogorov-Smirnov (Smirnov, 1948) para la normalidad a todas las variables cuantitativas, donde el contraste de hipótesis que se plantea para ello es el siguiente:

$$\begin{cases} H_0: & La\ variable\ sigue\ una\ distribucion\ normal \\ H_1: & La\ variable\ no\ sigue\ una\ distribucion\ normal \end{cases}$$

Como para todas las variables el p-valor es menor que $\alpha = 0,05$, se refuta H_0 y se deduce que las variables no siguen una distribución normal, impidiendo aplicar la técnica ANOVA de una vía.

Para poder aplicar el contraste de Kruskall-Wallis es necesario verificar previamente que se cumplen las hipótesis. Para la verificación de que todos los grupos del atributo cualitativo

tienen la misma forma en la distribución (Amat, 2016), se analiza la asimetría, mientras que la independencia de las observaciones se asume debido a la naturaleza de los datos. Dada la gran cantidad de grupos que tienen las variables, no siempre se van a satisfacer las hipótesis, resumiendo los casos en los que sí mediante las Tablas 31 y 32.

Hipótesis misma forma de distribución train LEFT JOIN sessions1					
Cualitativa/ Cuantitativa	**date_first_ booking_2009**	**dif_first_ active_create**	**month_account_ created**	**age**	**secs_ elapsed**
gender		✓	✓		✓
language		✓	✓		
affiliate_provider		✓			
first_affiliate_tracked	✓	✓	✓		
first_browser		✓			
action_detail		✓	✓		
device_type		✓	✓		✓

Tabla 31: Resultado del estudio de la forma de distribución de `train` LEFT JOIN `sessions1`.

Hipótesis misma forma de distribución train JOIN sessions2					
Cualitativa/ Cuantitativa	**dif_create_ first_booking**	**month_first_ active**	**age**	**sum_secs_ elapsed**	**secs_per_ action**
gender	✓	✓		✓	
language	✓	✓			
affiliate_channel	✓	✓	✓	✓	✓
first_affiliate_tracked		✓	✓		
first_device_type	✓	✓		✓	
first_browser					

Tabla 32: Resultado del estudio de la forma de distribución de `train` JOIN `sessions2`.

Tanto en la Tabla 31 como en la Tabla 32 el signo ✓implica que sí se cumple que la forma en la distribución en cada categoría de la variable es la misma, comprobado usando el As de Pearson (Canela, 2019). Por ello, las variables con el signo son las únicas que cumplen las condiciones para aplicar la prueba de Kruskall-Wallis. Tras aplicar esta prueba con un nivel de significación de $0,1$ sólo se refuta H_0 para el contraste entre `language` y `dif_first_active_create` de la Tabla 31 y para el contraste entre `language` y `month_first_active` de la Tabla 32. En consecuencia, se han decidido retirar las variables `dif_first_active_create` y `month_first_active` de las tablas `train` *left join* `sessions1` y `train` *join* `sessions2`, respectivamente, al contrastar realmente las variables cuantitativas en el contraste de hipótesis.

Llegados a este punto, se han estudiado las cinco tablas de la que estaba compuesta la base de datos, llegando a la conclusión que sólo dos servían para enriquecer los datos para obtener mejores resultados predictivos. También se ha hecho dos cruces distintos con las tablas seleccionadas, uno con las tablas originales y otro con tres variables resumen de la tabla que contiene datos sobre los registros de las sesiones de los usuarios. Además, se ha estudiado la asociación entre las variables en los casos en los que se ha podido para evitar la redundancia de la información. Tras este proceso, las dos tablas finales que recibirán el nombre de `cruce1` y `cruce2` y a las que se les aplicarán los algoritmos de *machine learning* son las siguientes (manteniendo sólo los registros no replicados).

cruce1: train LEFT JOIN sessions1		
Atributos	**Formato (longitud)**	**Descripción**
date_first_booking_2009	Int	Diferencia en días entre la fecha de la primera reserva y el 1 de enero de 2009
month_account_created	Int	Mes de la fecha en la que se creó la cuenta
gender	String	Género
age	Int	Edad
language	String(2)	Idioma
affiliate_provider	String	Proveedor de afiliación (dónde se desarrolló el marketing: google, . . .)
first_affiliate_tracked	String	Determina la primera comercialización con la que interactuó el usuario antes de registrarse
first_browser	String	Primer navegador
action_details	String	Descripción de la acción
device_type	String	Tipo de dispositivo
secs_elapsed	Float	Segundos transcurridos para realizar la acción
country_destination	String(2)	País de destino

Tabla 33: Tabla final `cruce1`, compuesta de 2.462.325 registros.

cruce2: train JOIN sessions2		
Atributos	**Formato (longitud)**	**Descripción**
dif_first_active_booking	Int	Diferencia en días entre la primera actividad de la cuenta de Airbnb y su primera reserva
gender	String	Género
age	Int	Edad
language	String(2)	Idioma
affiliate_channel	String	Canal de afiliación (el tipo de marketing de pago)
first_affiliate_tracked	String	Determina la primera comercialización con la que interactuó el usuario antes de registrarse
first_device_type	String	Tipo del primer dispositivo
first_browser	String	Primer navegador
sum_secs_elapsed	Float	Total tiempo transcurrido para realizar las acciones
secs_per_action	Float	Proporción de segundos por acción
country_destination	String(2)	País de destino

Tabla 34: Tabla final `cruce2`, compuesta de 28.764 registros.

5. Desarrollo de la propuesta

El sistema de recomendación que se va a construir para responder al problema de hacer una sugerencia a un nuevo usuario de Airbnb (Airbnb, 2019b) consiste en un sistema de recomendación basado en conocimiento. Estos conocimientos son la información que se tiene tanto de la actividad del usuario (interactuando con la web) como de datos propios del perfil (edad, sexo y fecha en la que se unió a la comunidad de Airbnb (Airbnb, 2019b) creando su cuenta, por ejemplo) junto a otra información como la de su primera actividad, el tipo de dispositivo que usó para reservar y si fue captado a través de alguna campaña de marketing. Además, para hacer la recomendación se va a tomar como base el resultado proporcionado por la predicción que haga la técnica de aprendizaje automático que mejor se adapte a los datos.

Para predecir el primer destino de un usuario, se va a trabajar con una serie de instancias donde se conoce el país de destino, es decir, la clase que toma el atributo de salida para cada individuo es conocida. Por ello, como se pretende caracterizar el destino a partir de registros del mismo, se van a emplear algoritmos de aprendizaje supervisado. Existen gran cantidad de técnicas de inteligencia artificial de clasificación: ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), SVM (Cortes y N., 1999), Prism (Jarník, 1930), Random Forest (Ho, 1995), etc. Para decidir el algoritmo que mejor se adapta a los datos, a parte de un primer filtro para saber si las hipótesis del algoritmo se cumplen y que es aplicable tanto para las variables cualitativas como las cuantitativas de las que se componen las tablas, la estrategia que se va a seguir es la de aplicar distintas técnicas que generalmente dan buenos resultados. Se ha determinado proseguir así ya que no se tiene el tiempo necesario para poder aplicar todos los algoritmos de los que se satisfacen las condiciones iniciales.

Por otro lado, en cuanto al entrenamiento y a la validación de los algoritmos de aprendizaje supervisado, como no se tiene un conjunto de datos de prueba como tal, para corroborar si la clasificación ha sido correcta o no se pueden llevar a cabo la validación cruzada o la k-validación cruzada, principalmente. La validación cruzada permite estimar el ajuste del modelo a un conjunto provisional de datos de prueba cuando no se dispone de ellos de forma explícita (Burkov, 1956). Mientras tanto, la k-validación cruzada se usa en la misma situación que la anterior pero cuando el conjunto de datos no tiene registros suficientes (Villalonga, 2019). Para ello, se divide la base de datos en k subconjuntos del mismo tamaño, de entre los cuales k-1 se emplea para el entrenamiento y el resto como datos de prueba o validación, rotando de for-

ma iterativa hasta la k-ésima iteración (Villalonga, 2019). Ya que tanto para `cruce1` como para `cruce2` se tienen suficientes instancias, se dividirán una sóla vez los datos en el grupo de entrenamiento y de validación, aplicando la validación cruzada y estudiando con qué porcentaje de entrenamiento y de prueba funciona mejor el algoritmo.

Para comprobar la calidad de los resultados obtenidos para cada método, se van a estudiar las siguientes métricas, evaluando la clasificación de los datos de validación:

- Matriz de confusión: Consiste en una matriz de nxn donde "n" es el número de clases del atributo de salida (Kuhn y col., 2008). Las columnas de la matriz aportan información sobre las clases del atributo de salida a las que corresponde cada instancia, mientras que las filas muestran las clases que la técnica de inteligencia artificial ha predicho, siendo los valores de la diagonal principal los aciertos y el resto los fallos (Villalonga, 2019). De esta matriz se puede extraer mucha información para evaluar la clasificación, como son la precisión, la exactitud, el TP Rate, etcétera (Villalonga, 2019). De entre los cuales se prestará atención a los siguientes:
  - Exactitud: Es el porcentaje de registros de los datos de validación que son clasificados correctamente (Kuhn y col., 2008). Se calcula dividiendo la suma del número de instancias correctamente clasificadas sobre el número total de instancias (Villalonga, 2019).
  - Error de clasificación: Es el porcentaje de instancias de los datos de validación que son clasificados de forma errónea (Villalonga, 2019). Se calcula restándole la exactitud a la unidad.
- AUC: Significa *Area Under the Curve*, refiriéndose con la palabra curva a la curva de ROC (Receiver Operating Characteristic) (Aroztegi, 2017). La curva de ROC es una medida de rendimiento que muestra el resultado global de la clasificación (Kuhn y col., 2008). El AUC proporciona una forma de calcular la capacidad del modelo de clasificación, ya que es invariable respecto a la escala y al umbral de clasificación (Aroztegi, 2017). Toma valores entre 0 y 1, siendo el objetivo a lograr el valor más próximo a uno (Aroztegi, 2017).

En la actualidad, existen más formas de medir la calidad de la clasificación: TP Rate, FP Rate, TN Rate, FN Rate, la precisión, la prevalencia, etc. En este trabajo se han elegido las medidas comentadas porque ofrecen una visión general acerca de las predicciones, permitiendo ampliar el detalle de la evaluación de cada clase del atributo de salida en caso de que fuera necesario.

A continuación, se van a describir primero las técnicas de *machine learning* que se han

decidido aplicar sobre los datos y, posteriormente, se van a aplicar a las tablas `cruce1` y `cruce2`. Recordar que los algoritmos se van a aplicar a ambas tablas para ver con qué nivel de detalle se obtienen mejores resultados. Además, debido al doble objetivo de Trabajo de Fin de Máster, se aplicarán los algoritmos con herramientas convencionales, es decir, de *Business Intelligence*, R, y con herramientas de *Big Data*, SparkMLlib y SparkML. Igualmente, si el código empleado por cada herramienta tiene algo característico también se tratará, pudiendo consultar el código empleado en toda la memoria en el anexo, Sección 9. Finalmente, como la de los algoritmos serán ejecuciones pesadas, tendrán un tiempo máximo de 36 horas de ejecución para proporcionar algún resultado, de lo contrario se los tomará como ineficientes.

5.1. Regresión logística multinomial

La regresión logística multinomial es la técnica más simple para atributos de salida de más de dos categorías, es decir, multinomiales, y sirve para predecir la probabilidad de las diferentes clases posibles de la variable de salida en función a los atributos de entrada, sean tanto cualitativos como cuantitativos (Burkov, 1956). Surge a partir de una generalización de la regresión logística, técnica empleada para predecir el resultado de una variable cualitativa, y es habitual que sea la primera técnica en ser aplicada debido a su simpleza (Aroztegi y Barrio, 2018). Es decir, si es posible modelizar los datos con una técnica sencilla que arroje buenos resultados, no hay necesidad de acudir a algoritmos más complejos como árboles de decisión y redes neuronales.

A pesar de que la regresión logística multinomial sirve para hacer una clasificación para más de dos clases en el atributo de salida y permite de entrada tanto atributos cualitativos como cuantitativos, deben cumplirse las siguientes hipótesis para que los resultados que se obtengan sean verídicos (Aroztegi y Barrio, 2018):

- La variable de salida debe ser cualitativa nominal.
- No debe haber variables cualitativas ordinales.
- No debe haber outliers.
- Debe existir una relación lineal entre los atributos cuantitativos de entrada y la variable dependiente o de salida.
- Los atributos de entrada deben de ser independientes.
- No debe haber colinealidad.

Empezando con la verificación del cumplimiento de las hipótesis, la variable a predecir es un atributo cualitativo nominal llamado `country_destination`. Además, todas las variables cualitativas son nominales al no haber un orden entre las categorías. En cuanto a los outliers, mediante los siguientes diagramas de cajas se estudia la distribución de los datos para las variables cuantitativas.

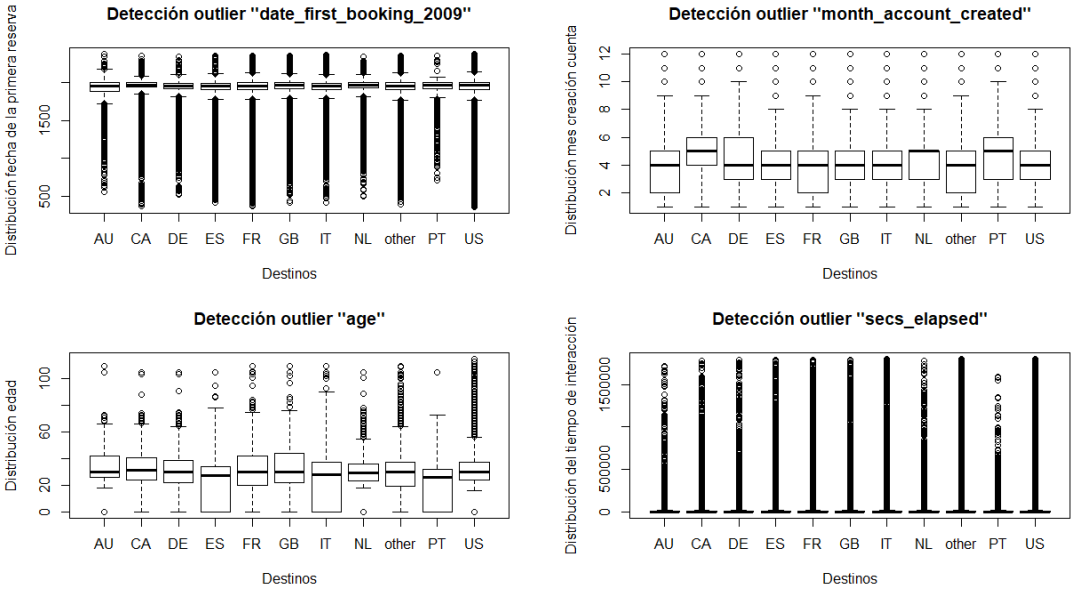

Figura 24: Diagramas de caja de las variables cuantitativas de `cruce1`.

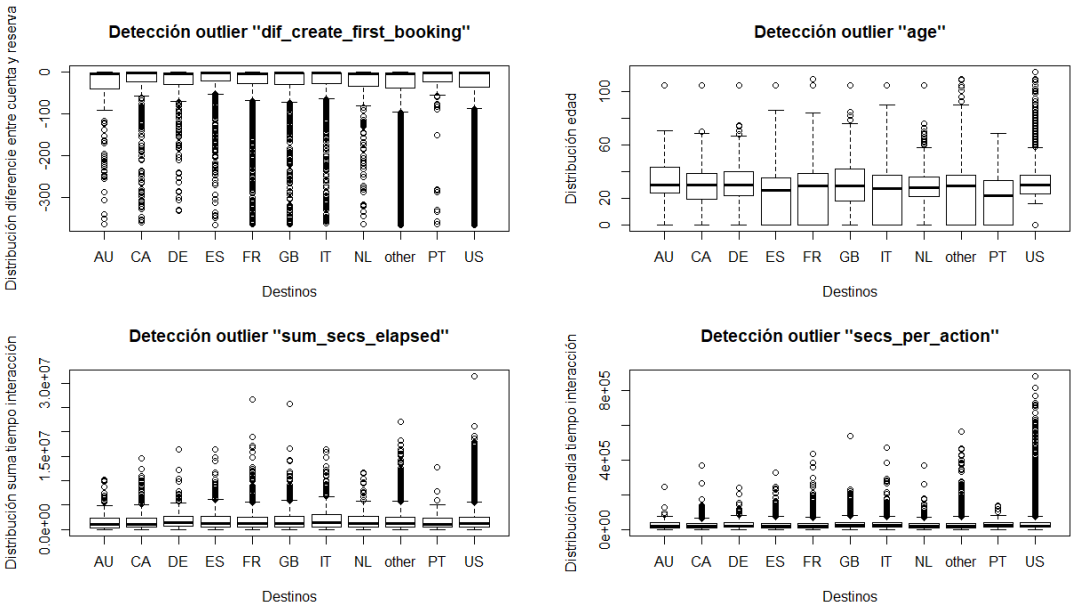

Figura 25: Diagramas de caja de las variables cuantitativas de `cruce2`.

En las Figuras 24 y 25 se puede ver que en todas las variables hay una gran cantidad

de outliers, no cumpliendo con la hipótesis. Como no se cumplen todas las hipótesis para las variables cuantitativas de ambos conjuntos de datos, se van a buscar otras técnicas para dar respuesta al problema. De este modo, se deduce que el objetivo de este TFM no puede ser abordado por la técnica más simple y básica, acudiendo a técnicas más sofisticadas como las siguientes.

5.2. C5.0

C5.0 es una mejora del algoritmo C4.5 que a su vez es una extensión del algoritmo ID3 (Research, 2019). Además, C5.0, al igual que su antecesor, puede generar tanto árboles de decisión como conjuntos de reglas (IBM, 2019). A diferencia de C4.5, C5.0 ofrece resultados más precisos, de una forma más rápida y gasta menos memoria, creando árboles más cortos, entre otras mejoras (Research, 2019). C5.0 divide las diferentes muestras en base al campo que aporta la máxima ganancia de información, subdividiéndolas de nuevo en base a este misma medida de precisión, generalmente con otro campo, hasta que resulte imposible crear más submuestras (IBM, 2019). Finalmente, se examinan las divisiones del nivel inferior y se podan las que no aportan información, realizando un proceso de postpoda (IBM, 2019).

Profundizando más en su capacidad de generar un árbol de decisión, no tiene hipótesis previas para poder aplicarlo y funciona igual que C4.5 (Research, 2019). A pesar de su funcionalidad tanto para generar reglas como para hacer clasificaciones, en esta memoria se va a explotar su capacidad de clasificación. Por otro lado, el algoritmo está implementado en R (con el paquete *C50*) pero no está implementado en Spark. Por ello, la modelización sólo se ha llevado a cabo con R.

A la hora de encontrar los parámetros que mejores resultados dan, se ha consultado la documentación del paquete en la referencia (R, 2018). De este modo, las soluciones de calidad para el algoritmo C5.0 corresponden a las ejecuciones con el parámetro de entrada de la utilización de pesos para el atributo de salida en el caso de `cruce2` y sin pesos para la tabla `cruce1`. Estos resultados se resumen en las dos siguientes tablas, desglosadas en función del porcentaje de registros empleados para el entrenamiento (%Train) y la validación (%Val).

cruce1: Algoritmo C5.0 con R							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	98,03	98,04	98,08	98,14	98,14	98,14	98,16
Tasa Error (%)	1,97	1,96	1,92	1,86	1,86	1,86	1,84
AUC	0,996	0,996	0,997	0,997	0,998	0,998	0,998

Tabla 35: Resultados del algoritmo C5.0 al aplicarlo a los datos de la tabla `cruce1` usando R.

cruce2: Algoritmo C5.0 con R							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	68,59	69,63	70,30	68,18	68,45	69,1	68,34
Tasa Error (%)	31,41	30,37	29,7	31,82	31,55	30,9	31,66
AUC	0,506	0,504	0,500	0,510	0,516	0,506	0,524

Tabla 36: Resultados del algoritmo C5.0 al aplicarlo a los datos de la tabla `cruce2` usando R.

En la Tabla 35 se muestran los resultado para `cruce1` donde se aprecia que el conjunto de datos responde muy bien al algoritmo, logrando unos valores en la validación muy buenos y sin temer al sobreajuste. Además, el AUC está muy cerca de uno. De entre las distintas proporciones para entrenamiento-validación para `cruce1`, se selecciona la de mejores resultados, siendo la de 85-15 la que se va a comentar en la sección correspondiente a los resultados obtenido, Sección 6. Para la Tabla 36, correspondiente a `cruce2`, la mejor proporción ataña a la de 80-20, ya que para este caso el área bajo la curva y la exactitud guardan una buena proporción dentro de los demás resultados.

A continuación, se van a comentar tres algoritmos que no sólo generan árboles de decisión, sino que aplican, previamente, técnicas de *boosting* o *bagging*. Seguidamente se definen estos términos:

- *Boosting*: Consiste en emplear la base de datos de entrenamiento y crear, de forma iterativa, múltiples modelos usando técnicas de modelización débiles, es decir, técnicas que por si solas no aportan buenos resultados. De esta clase de algoritmos son AdaBoosting y Gradient Boosting, por ejemplo.
- *Bagging*: Se basa en la división de la base de entrenamiento en múltiples copias o subconjuntos y aplicar a cada uno de estos un modelo débil. A este tipo pertenece, por ejemplo, el algoritmo Random Forest.

5.3. Random Forest

Random Forest es un algoritmo de aprendizaje supervisado que genera varios árboles de decisión de forma aleatoria, de ahí su nombre (Burkov, 1956). Además, sirve tanto para problemas de clasificación como de regresión y es un tipo de técnica de *bagging* (Ho, 1995). En cuanto a su funcionamiento, comienza con un proceso iterativo donde, de la base de datos completa, se toma cada vez un subconjunto de registros de entrenamiento con reemplazo (Orellana Alvear, 2018). De este subconjunto se selecciona, entre todos los atributos de entrada, un subgrupo que se mantendrá a lo largo de toda la construcción del árbol (Breiman, 1999). Con este último conjunto de elementos reducido es con el que se lleva a cabo un árbol de decisión, creciendo el árbol hasta su máxima extensión sin procesos de poda generalmente (Orellana Alvear, 2018). Antes de la siguiente iteración, se vuelve a la base de datos original y se guarda el árbol generado, volviendo a construir otro árbol (Orellana Alvear, 2018).

El algoritmo Random Forest clasifica una nueva instancia mediante la agregación de los resultados predictivos de todos los árboles, habiendo tantos árboles como iteraciones se hayan llevado a cabo en su construcción (Orellana Alvear, 2018). Esta agregación se puede hacer teniendo en cuenta la mayoría de los votos en la clasificación o el promedio de los resultados en la regresión (Breiman, 1999). Además, el algoritmo produce muy buenos resultados predictivos y con alta precisión (Breiman, 1999). Seguidamente, se van a utilizar tanto las herramientas de R con el paquete *randomforest* y Spark con su componente SparkML para implementarlo con las dos bases de datos de esta memoria.

Los mejores resultados para el algoritmo de Random Forest se obtienen tras el estudio de las posibles combinaciones para ajustar sus parámetros y la fluctuación en la calidad de la exactitud en cada prueba. Los parámetros resultantes en R son 100 árboles de los que se parte para crecer, 4 candidatos aleatorios para cada división, podando el árbol y tomando como relevante la importancia de cada clase de salida para la modelización con la tabla `cruce2`. En Spark y para `cruce2`, el parámetro que se puede regular es el número de árboles, correspondiendo a 100. Para más información sobre los argumentos de los algoritmos se recomiendan las referencias (Liaw, 2019) para R y (Spark, 2019a) para Spark. Como los algoritmos implementados en R y en Spark utilizan distintos argumentos, se esperan distintos resultados, mostrándolos en las Tablas 37 y 38 para cada proporción de datos de entrenamiento y prueba.

cruce2: Algoritmo Random Forest con R							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	69,62	69,26	68,85	69,07	68,89	68,92	70,18
Tasa Error (%)	30,38	30,74	31,15	30,93	31,11	31,08	29,82
AUC	0,519	0,517	0,519	0,528	0,512	0,512	0,518

Tabla 37: Resultados del algoritmo Random Forest al aplicarlo a los datos de la tabla `cruce2` usando R.

cruce2: Algoritmo Random Forest con Spark							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	69,53	69,96	68,86	68,86	69,80	69,63	69,69
Tasa Error (%)	30,47	30,04	31,14	31,14	30,2	30,37	30,31

Tabla 38: Resultados del algoritmo Random Forest al aplicarlo a los datos de la tabla `cruce2` usando Spark.

En las tablas superiores se observa que con las dos herramientas se logran resultados similares, no siendo posible medir el AUC para el entorno de *Big Data* ya que no se ha encontrado una implementación en Spark para esta medida estadística. Entre las diferentes proporciones de entrenamiento-validación y los resultados de las diferentes herramientas, se considera la mejor elección la de 90-10 con R, debido a su alta exactitud, baja tasa de error y elevado AUC en comparación con el resto de ejecuciones en ambos entornos.

La ausencia de explicaciones y parámetros para la tabla `cruce1` tanto en R como en Spark se debe a que no ha sido posible llevar a cabo su ejecución por completo ya que no se dispone de memoria suficiente. Ante esta situación, R proporciona un mensaje expresando que el cómputo que debe realizar excede el tamaño máximo mientras que Spark, tras indicar que no puede realizar el procesamiento in-memory, tira del disco y, como tampoco puede, ejecuta el código con, como máximo, 72,4MB de los 136,54MB que ocupa. Trabajando con esta reducción del tamaño del conjunto de datos, el algoritmo Random Forest en el entorno *Big Data* no compila bien con más de 60 árboles. Esta limitación es importante porque, por casos experimentales que se han visto para esta tabla, cuanto mayor es el número de árboles mejores resultados se obtienen. Los resultados para 60 árboles y con 72,4MB de los 136,54MB se presentan en la Tabla 39.

cruce1 parcial: Algoritmo Random Forest con Spark							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	67,60	67,54	67,55	67,65	67,64	67,62	67,41
Tasa Error (%)	32,40	32,46	32,45	32,35	32,36	32,38	32,59

Tabla 39: Resultados del algoritmo Random Forest al aplicarlo a los datos de la tabla parcial `cruce1` usando Spark.

Trabajando con la mitad de los datos de `cruce1`, se obtienen unos resultados de validación del modelo similiar a los logrados con el algoritmo Random Forest para `cruce2`. Debido a la reducción del conjunto de datos que acarrea el no tener memoria suficiente en un ordenador recién formateado de 4GB de RAM, no se tomará este algoritmo en consideración a la hora de decidir qué algoritmo se adapta mejor a los datos de `cruce1`.

5.4. AdaBoosting

El algoritmo de AdaBoosting pertenece al aprendizaje supervisado y genera árboles de decisión (Burkov, 1956). Se basa en la técnica *boosting*, es decir, combina los resultados de varios algoritmos débiles que darían lugar por separado a predicciones deficientes, para lograr pronósticos más efectivos (Burkov, 1956). Para ello, AdaBoosting comienza entrenando un árbol de decisión (Árbol1), asignándole a cada clase del atributo de salida un mismo peso (Singh, 2018). Tras evaluar el primer modelo, se aumenta el peso de las observaciones con mayor error de clasificación y se reducen los pesos de las de menor error (Singh, 2018). Entonces, se vuelve a construir otro árbol de decisión (Árbol2) que se alimenta de los datos ponderados del anterior, siendo el modelo en esta segunda iteración Árbol1+Árbol2. Luego, se calcula el error del modelo del conjunto de dos árboles y se vuelven a reponderar las clases de salida en base a su dificultad de ser predichas correctamene (Singh, 2018). Este proceso se repite iterativamente tantas veces como se crea necesario, agregando al modelo un nuevo árbol en cada iteración (Singh, 2018).

En AdaBoosting, los árboles anteriores al más reciente en el modelo sirven para mejorar la clasificación de las clases que no están bien asignadas por los árboles anteriores, logrando un modelo efectivo (Burkov, 1956). Cuando entra una nueva instancia en el modelo, el resultado de la variable de salida se calcula con la suma ponderada de los modelos anteriores (Singh, 2018). En cuanto a la implementación del algoritmo para `cruce1` y `cruce2`, no está disponible en Spark, estudiando sólo los resultados aportados por R con el paquete *adabag*.

En cuanto a los argumentos del algoritmo disponible en R, detallados en (Olson, 2019), destacan la profundidad de los árboles y el número de iteraciones para el *boosting*. Se ha intentado llevar a cabo distintas ejecuciones en R tanto para `cruce1` como `cruce2`, utilizando distintos valores para estos parámetros de entrada del algoritmo y quedándose sin espacio en todas ellas. A pesar de que el algoritmo parecía prometedor, como no está implementado ni en SparkML ni en SparkMLlib y al no ser capaz de ejecutarlo en R por problemas de espacio,

este algoritmo no se ha podido implementar para la evaluación de la mejor técnica de *machine learning* que mejor se adapta a los datos.

5.5. Gradient Boosting

Otro algoritmo de aprendizaje supervisado que genera árboles de decisión y que es del tipo *boosting* es Gradient Boosting (Burkov, 1956). A diferencia de AdaBoosting, no utiliza pesos para mejorar la predicción sino que se vale de la función pérdida (Burkov, 1956) . Esta función mide lo bueno que son los coeficientes del modelo para predecir correctamente el atributo de salida, dependiendo la forma de la función pérdida del resultado que se intente optimizar: para la regresión la función pérdida se basa en el error de la predicción mientras que para la clasificación se mide cuán bueno es el modelo a la hora de presuponer la clase correcta (Singh, 2018).

Aparte de lo comentado, Gradient Boosting aplica de forma iterativa varios modelos débiles para obtener un algoritmo efectivo (Singh, 2018). De este modo, si el proceso de repite n veces, el modelo resultante sería a_1·Árbol1 + a_2·Árbol2+ ...+a_n·ÁrbolN, siendo a_i con i desde 1 hasta N los coeficientes a precisar (Singh, 2018). En cuanto a su implementación, ésta ha sido realizada gracias al paquete *gbm* con R (Greenwell, Boehmke, Cunningham y Developers, 2019). SparkML proporciona a sus usuarios una implementación del algoritmo Gradient Boosting en la referencia (Spark, 2019b), pero sólo está disponible para clasificaciones binomiales y no multinomiales, como es la de esta memoria.

Llevando a la práctica el algoritmo proporcionado por R, la herramienta no es capaz de ejecutar la técnica de aprendizaje automático con los datos de `cruce1` por problemas de espacio. En cambio, `cruce2` sí que logra compilarlo, siendo para 100 árboles, 10 observaciones mínimas en cada nodo del árbol y 0,01 para el parámetro shrinkage, que condiciona la expansión del árbol, los valores de los parámetros con los que mejores resultados se tienen con el algoritmo. A continuación, se presenta la Tabla 40 a modo de resumen de los resultados para cada proporción de entrenamiento-validación.

cruce2: Algoritmo Gradient Boosting con R							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,08	69,92	69,56	69,75	70,12	70,10	69,38
Tasa Error (%)	29,92	30,08	30,44	30,25	29,88	29,9	30,62
AUC	0,524	0,517	0,520	0,516	0,521	0,524	0,519

Tabla 40: Resultados del algoritmo Gradient Boosting al aplicarlo a los datos de la tabla `cruce2` usando R.

En este caso, tras observar los resultados y siguiendo el mismo planteamiento que hasta ahora, la proporción de mejores resultados es la de 85 % entrenamiento y 15 % validación. Esto se debe a que tiene, en comparación con el resto, los mejores valores de AUC y exactitud, guardando este resultado para el análisis de los resultados obtenidos de la siguiente sección, Sección 6.

5.6. One Versus All

En esta subsección se van a explicar tanto el algoritmo Support Vector Machine (SVM) como One Versus All, ya que el segundo es una generalización del primero (Burkov, 1956). SVM es un algoritmo de aprendizaje supervisado cuyo objetivo es el de encontrar un hiperplano n-dimensional, siendo n el número de atributos de entrada, que sea capaz de separar las clases de salida (Cortes y N., 1999). Forma parte del aprendizaje supervisado y sirve para atributos de salida binomiales (dos clases) (Cortes y N., 1999). En cuanto al hiperplano, es un concepto geométrico cuya representación varía en función del espacio en el que se trabaja: en espacios unidimensionales el hiperplano corresponde a un punto, en bidimensionales es una recta y en tridimensionales es un plano, por ejemplo (Mardones, 2015). Además, no tiene porqué existir un único hiperplano que separe distinguiblemente ambas clases, siendo el hiperplano elegido por el algoritmo SVM el que tenga margen máximo con el registro más próximo al hiperplano en ambas clases, proporcionando mayor confianza para clasificar futuras instancias (Cortes y N., 1999). A la hora de clasificar una nueva instancia, se le asigna la clase correspondiente al lado del hiperplano en el que cae (Burkov, 1956).

En cuanto al método One Versus All, es la generalización de SVM para modelos multinomiales, es decir, con más de dos clases en el atributo de salida (Burkov, 1956). El algoritmo convierte el problema multinomial en tantos binomiales como combinaciones de clases haya, aplicando a cada subdivisión binomial el algoritmo SVM (Burkov, 1956). De este modo, si un modelo tiene n clases de salida, One Versus All calcula $\frac{n!}{2!(n-2)!}$ SVM para cada combinación

de las categorías binomiales. Al ser un problema de predicción multinomial el de esta memoria, en las siguientes tablas aparecen los resultados tras la aplicación de One Versus All. En la implementación, la herramiente R dispone del paquete *e1071* (Meyer y col., 2019) que es válido tanto para SVM como para One Versus All debido a que el algoritmo de ese paquete consta de un parámetro de entrada que sirve para especificar si el atributo de salida es binomial o multinomial. Spark, por su parte, tiene una implementación correspondiente al algoritmo SVM (Spark, 2019c) y otra para One Versus All (Spark, 2019e).

En las diferentes ejecuciones que se han llevado a cabo, en R se ha comprobado que los mejores resultados para `cruce2` se logran para los parámetros de un kernel radial y de 0,1 en la variable gamma necesaria para el kernel, 5 como coste y la clasificación de tipo C, trabajando con el atributo de salida multinomial. La herramienta Spark proporciona menos parámetros a definir por el usuario para el algoritmo One Versus All, logrando los mejores resultado cuando se fija en 30 el número máximo de iteraciones para `cruce2`. Las Tablas 41 y 42 resumen las medidas de validación para estos argumentos.

cruce2: Algoritmo One Versus All con R							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,07	69,96	69,27	70,60	70,94	69,04	69,14
Tasa Error (%)	29,93	30,04	30,73	29,4	29,06	30,96	30,86
AUC	0,519	0,516	0,514	0,523	0,525	0,525	0,520

Tabla 41: Resultados del algoritmo One Versus All al aplicarlo a los datos de la tabla `cruce2` usando R.

cruce2: Algoritmo One Versus All con Spark							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	69,68	69,69	69,61	70,11	70,12	69,20	70,30
Tasa Error (%)	30,32	30,31	30,39	29,89	29,88	30,8	29,7

Tabla 42: Resultados del algoritmo One Versus All al aplicarlo a los datos de la tabla `cruce2` usando Spark.

Prestando atención a las tablas superiores, se ve que los resultados obtenidos para ambas herramientas son muy parecidos. La proporción de entrenamiento-validación que obtiene los mejores valores, deduciendo que el algoritmo es más preciso para ese caso y para esa herramienta, corresponde a la proporción 80-20 con la herramienta R. Por este buen rendimiento, es decir, por contener los parámetros con el que el algoritmo mejor se adapta a los datos, este es el resultado que se comentará en la siguiente sección.

Una vez más, `cruce1` no presenta resultados debido a la dificultad que supone para las

herramientas su tamaño. R manda un mensaje, transcurrido un tiempo, de que le es imposible seguir con la ejecución del algoritmo ya que se queda sin memoria. Spark, por su parte, procesa como máximo 64,8MB del contenido para el parámetro de 5 máximas iteraciones, siendo este un valor muy bajo. Los resultados que arroja Spark se presentan en la Tabla 43.

cruce1 parcial: Algoritmo One Versus All con Spark							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	67,53	67,58	67,59	67,48	67,56	67,56	67,54
Tasa Error (%)	32,47	32,42	32,41	32,52	32,44	32,44	32,46

Tabla 43: Resultados del algoritmo One Versus All al aplicarlo a los datos de la tabla parcial `cruce1` usando Spark.

Dado que no es posible trabajar con los 136,54MB que forman la tabla `cruce1`, estos resultados no son ni precisos ni representativos. Por este motivo, la ejecución no se tendrá en cuenta a la hora de evaluar cuáles son el mejor algoritmo y entorno para los datos.

5.7. Redes neuronales

Las redes neuronales son una técnica de *machine learning* que se basan en el modelo biológico de las neuronas (Slavio, 2017). Aunque fueron creadas en los años cuarenta, es en la última década cuando más se está demandando su uso debido a la gran precisión y exactitud que se pueden obtener con ellas (Slavio, 2017). Las redes neuronales se usan para reconocer patrones, valiendo para problemas de clasificación y regresión en el aprendizaje supervisado, para el aprendizaje no-supervisado y para el aprendizaje por refuerzo (Villalonga, 2019). Dado el potencial de esta técnica, la versatilidad para adaptarse a los problemas y la cantidad de variantes que existen (Burkov, 1956), en este TFM se va a trabajar sólo con las redes neuronales denominadas perceptrón multicapa.

Una red neuronal perceptrón multicapa está compuesta, principalmente, por tres elementos: una capa de neuronas de entrada, una de salida y una intermedia, también conocida como capa oculta (Slavio, 2017). La capa de entrada está formada por las neuronas que introducen los patrones de inicio, alimentándose directamente de los atributos de entrada del conjunto de datos (Villalonga, 2019). La capa oculta se alimenta de la capa de entrada y transmite una salida a la siguiente capa, influyendo el número de capas ocultas en la precisión de la red (a mayor cantidad de capas ocultas, mayor precisión) (Burkov, 1956). Finalmente, la capa de salida consta de las neuronas que generan la predicción (Villalonga, 2019). La Figura 26 resume

el proceso que se acaba de describir.

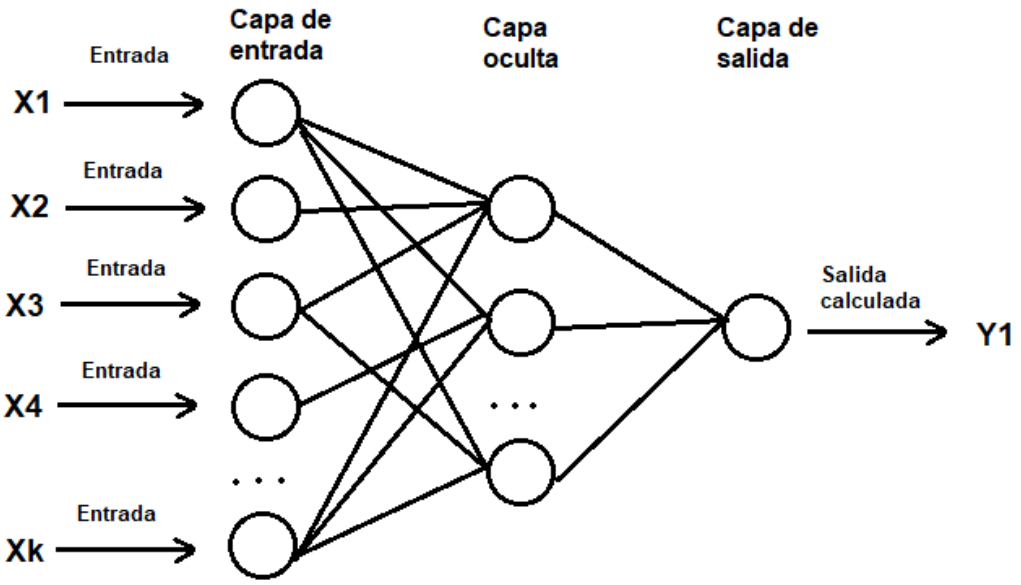

Figura 26: Representación de una red neuronal multicapa con una capa oculta.

Para que la red se active es necesario que el estímulo que le llega a la neurona, una nueva instancia, supere un umbral de excitación (Slavio, 2017). Además, el modelo de perceptrón multicapa analiza el error partiendo de la capa de salida hasta la de entrada, recibiendo el nombre de retropropagación (Villalonga, 2019). Para lograr unos buenos valores de clasificación, se han probado distintos valores para las capas ocultas, utilizando en R los paquetes *nnet* (Ripley y Venables, 2016) y *neuralnet* (Fritsch, Guenther, Wright, Suling y Mueller, 2019) y en Spark la referencia (Spark, 2019d).

Cabe comentar que, antes de la construcción del modelo, ha sido necesaria la adaptación de los datos, codificando todas las variables cualitativas usando el método One Hot Vector y normalizando las tablas con el método Min-Max, siendo estas las condiciones necesarias para poder llevar a cabo la red neuronal. Siguiendo con la explicación del modelo, los resultados que se narran a continuación corresponden a `cruce2`. En R, debido al tiempo de ejecución que conlleva, no se ha podido hacer ninguna prueba para más de una capa oculta debido a que excedía las 36 horas establecidas como tiempo de ejecución máximo. Teniendo esto en cuenta, los mejores resultados con R corresponden a una capa oculta con 11 nodos, estableciendo como falsa la linealidad del atributo de salida, *logistic* como método para suavizar los pesos y como umbral mínimo el mínimo de la base de datos. En Spark, debido a su procesamiento *in-memory*, el tiempo de ejecución es mucho menor, pudiendo hacer más ejecuciones. Los

mejores resultados con esta herramienta tienen como parámetros de entrada la construcción de una red neuronal con 3 capas ocultas de 12, 11 y 10 nodos, respectivamente, y un número máximo de 500 iteraciones. Los resultados se resumen en las Tablas 44 y 45 para cada par de entrenamiento-validación.

cruce2: Perceptrón multicapa con R							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,07	69,96	69,27	70,16	70,33	69,04	69,14
Tasa Error (%)	29,93	30,04	30,73	29,84	29,67	30,96	30,86

Tabla 44: Resultados de la red neuronal perceptrón multicapa al aplicarla sobre los datos de la tabla `cruce2` usando R.

cruce2: Perceptrón multicapa con Spark							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,04	70,11	70,57	70,41	70,89	71,07	70,24
Tasa Error (%)	29,96	29,89	39,43	29,59	29,11	28,93	29,76

Tabla 45: Resultados de la red neuronal perceptrón multicapa al aplicarla sobre los datos de la tabla `cruce2` usando Spark.

En las dos tablas superiores se aprecia que los resultados obtenidos por Spark son, en su conjunto, mejores que los de R. Esto se puede deber a la cantidad y a la profundidad de las capas ocultas que se han podido elaborar para Spark y que en R no ha sido posible. De entre todos los resultados, el de la proporción 85-15 para Spark es el que presenta las mejores métricas.

En cuanto a `cruce1`, la ejecución para R excede las 36 horas sin devolver ningún error. Con la herramienta Spark, la ejecución muestra por pantalla el comentario de que, como máximo, puede trabajar con 63,3MB de los 136,54MB que componen la tabla, ya que en su procesamiento *in-memory* se queda sin espacio y recurre al procesamiento en el disco, donde también se queda sin memoria. Los resultados que arroja Spark para `cruce1` con la cantidad de datos que consigue procesar se muestran en la Tabla 46 para 3 capas ocultas de 12, 11 y 10 nodos cada una.

cruce1 parcial: Perceptrón multicapa con Spark							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	67,56	68,12	67,79	67,80	68,14	68,03	67,54
Tasa Error (%)	32,44	31,88	32,21	32,20	31,86	31,97	32,46

Tabla 46: Resultados de la red neuronal perceptrón multicapa al aplicarla sobre los datos de la tabla parcial `cruce1` usando Spark.

De la misma forma que ha sucedido hasta este punto, en los casos en los que una herramienta no ha podido aplicar a todo el conjunto de datos el algoritmo, los resultados no han sido tenidos en cuenta. Por este motivo, esta ejecución no se contemplará en la siguiente sección para elegir la mejor técnica y entorno para el mejor conjunto de datos porque, de tenerla en cuenta, los datos no estarían reflejando ni la realidad ni el verdadero potencial de la tabla.

Tras el estudio de los resultados de las distintas ejecuciones de los algoritmos C5.0, Random Forest, Gradient Boosting, One Versus All y la red neuronal perceptrón multicapa, cada uno por su cuenta, ahora es necesario comparar estos resultados en conjunto, es decir, entre los distintos algoritmos. Esto se va a llevar a cabo en la Sección 6 junto a la elección del conjunto de datos, algoritmo y entorno que mejores resultados dan para ser el motor del sistema de recomendación basado en conocimiento que se va a construir al final de esa sección.

6. Resultados obtenidos

Llegados a este punto, ya se han ejecutado los algoritmos que se han considerado que proporcionan buenos resultados de clasificación a las dos tablas, tanto a la tabla `cruce1` como a `cruce2`. Recordar que se ha tomado la decisión de trabajar con dos tablas en vez de una porque cada una de ellas tiene sus lados positivos y sus lados negativos. `Cruce1` es un tabla muy rica en contenido ya que proporciona toda la información registrada cada interacción de los usuarios, habiendo aplicado sobre ella las menores reducciones posibles, pero es muy grande para el entorno y el equipo que con el que se trabaja. Por otro lado, `cruce2` es más pequeña y más manejable para las características del entorno pero no es tan rica en contenido al trabajar, en algunos casos, con variables cuantitativas resumidas.

A lo largo de la sección anterior se han estudiado los resultados obtenidos en cada entorno y con cada algoritmo en ambos cruces, correspondiendo a la sección actual el estudio de los resultados obtenidos. A continuación se presentan las métricas que se logran tras aplicar cada método mencionado en la Sección 5 sobre todos los datos de cada tabla. Cabe resaltar que, para abreviar el contenido, se han empleado en la Tabla 47 los siguientes acrónimos: RF (Random Forest), GB (Gradient Boosting), OvA (One Versus All), NN (Neural Network), BI (*Business Intelligence*) y BD (*Big Data*).

Resumen de los algoritmos							
Resultados cruce1 en el entorno BI con el algoritmo C5.0							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	98,03	98,04	98,08	98,14	98,14	98,14	98,16
Tasa Error (%)	1,97	1,96	1,92	1,86	1,86	1,86	1,84
AUC	0,996	0,996	0,997	0,997	0,998	0,998	0,998
Resultados cruce2 en el entorno BI con el algoritmo C5.0							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	68,59	69,63	70,30	68,18	68,45	69,1	68,34
Tasa Error (%)	31,41	30,37	29,7	31,82	31,55	30,9	31,66
AUC	0,506	0,504	0,500	0,510	0,516	0,506	0,524
Resultados cruce2 en el entorno BI con el algoritmo RF							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	69,62	69,26	68,85	69,07	68,89	68,92	70,18
Tasa Error (%)	30,38	30,74	31,15	30,93	31,11	31,08	29,82
AUC	0,519	0,517	0,519	0,528	0,512	0,512	0,518
Resultados cruce2 en el entorno BD con el algoritmo RF							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	69,53	69,96	68,86	68,86	69,80	69,63	69,69
Tasa Error (%)	30,47	30,04	31,14	31,14	30,2	30,37	30,31
Resultados cruce2 en el entorno BI con el algoritmo GB							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,08	69,92	69,56	69,75	70,12	70,10	69,38
Tasa Error (%)	29,92	30,08	30,44	30,25	29,88	29,9	30,62
AUC	0,524	0,517	0,520	0,516	0,521	0,524	0,519
Resultados cruce2 en el entorno BI con el algoritmo OvA							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,07	69,96	69,27	70,60	70,94	69,04	69,14
Tasa Error (%)	29,93	30,04	30,73	29,4	29,06	30,96	30,86
AUC	0,519	0,516	0,514	0,523	0,525	0,525	0,520
Resultados cruce2 en el entorno BD con el algoritmo OvA							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	69,68	69,69	69,61	70,11	70,12	69,20	70,30
Tasa Error (%)	30,32	30,31	30,39	29,89	29,88	30,8	29,7
Resultados cruce2 en el entorno BI con las NN							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,07	69,96	69,27	70,16	70,33	69,04	69,14
Tasa Error (%)	29,93	30,04	30,73	29,84	29,67	30,96	30,86
Resultados cruce2 en el entorno BD con las NN							
%Train - %Val	**60-40**	**65-35**	**70-30**	**75-35**	**80-20**	**85-15**	**90-10**
Exactitud (%)	70,04	70,11	70,57	70,41	70,89	71,07	70,24
Tasa Error (%)	29,96	29,89	39,43	29,59	29,11	28,93	29,76

Tabla 47: Macrotabla con todos los resultados concluyentes comentados en la Sección 5, resaltando en verde los que mejor se comportan en cada algoritmo y cada tabla.

En la Tabla 47 se encuentran resaltados en verde los mejores resultados para cada cruce y cada algoritmo, independientemente del entorno. Para poder comparar cada uno de los algoritmos más en detalle y seleccionar el que más se ajusta a los datos, se presenta la Tabla 48.

Resumen de los mejores resultados de los algoritmos						
Tabla	Cruce1	Cruce2	Cruce2	Cruce2	Cruce2	Cruce2
Entorno	BI	BI	BI	BI	BI	BD
Método	C5.0	C5.0	RF	GB	OvA	NN
%Train - %Val	85-15	80-20	90-10	85-15	80-20	85-15
Exactitud (%)	98,14	68,45	70,18	70,10	70,94	71,07
Tasa Error (%)	1,86	31,55	29,82	29,9	29,06	28,93
AUC	0,998	0,516	0,518	0,524	0,525	-

Tabla 48: Resumen de los mejores resultados de cada algoritmos aplicado sobre los datos de las tablas `cruce1` y `cruce2` en la Sección 5.

En la tabla superior aparece un sólo resultado en el que se emplean los datos de `cruce1` mientras que la tabla `cruce2` es la más recurrente. Esto se debe al tamaño de `cruce1`, pues contiene 100 veces más registros que `cruce2`, necesitando más tiempo y memoria en cada ejecución. Se recuerda que con muchos algoritmos no se ha podido trabajar usando todos los registros de `cruce1` al trabajar con un ordenador de 4GB de RAM, descartando estos resultados para el análisis final. Por otro lado, el entorno con mejores resultados en la mayoría de los algoritmos ha sido *Business Intelligence*, empleando la herramienta especializada en tratamiento de datos estadístidos R (R, 2019b). A pesar de ello, el entorno *Big Data* es el de mejores soluciones para las redes neuronales (en concreto, con la red neuronal perceptrón multicapa, la construida en esta memoria). Analizando las métricas para estudiar la ejecución de cada algoritmo al aplicarlos en ambos cruces, destaca un resultado sobre el resto: el algoritmo C5.0 con la tabla `cruce1`.

En el caso del algoritmo que mejor se ha adaptado a los datos, se ha trabajado con el conjunto de datos más completo de entre los dos, usando herramientas con antigüedad y madurez, y se han obtenido muy buenos resultados. A continuación se muestra su matriz de confusión al completo junto a otros datos proporcionados por R para hacer un estudio más detallado de este caso.

```
Attribute usage:

100.00% affiliate_provider
```

```
 99.93% date_first_booking_2009
 99.78% age
 98.34% device_type
 97.49% language
 96.46% first_browser
 83.02% first_affiliate_tracked
 80.86% gender
 73.73% month_account_created
  4.76% action_detail
  1.26% secs_elapsed
```

```
Confusion Matrix and Statistics
```

	Reference										
Prediction	AU	CA	DE	ES	FR	GB	IT	NL	other	PT	US
AU	1697	0	0	0	6	0	1	0	2	0	14
CA	0	5464	0	0	0	24	9	0	37	0	83
DE	0	4	3370	7	14	1	9	0	20	0	31
ES	5	2	2	9576	59	4	6	3	35	0	139
FR	6	13	4	34	20832	3	27	12	80	1	243
GB	0	1	2	2	14	9225	7	0	24	0	135
IT	1	3	16	1	26	18	14484	0	45	6	293
NL	0	0	0	0	14	0	1	3251	14	0	62
other	0	31	13	34	58	20	46	9	45831	1	663
PT	0	0	0	0	0	1	0	0	6	890	3
US	87	219	146	344	823	370	525	109	1678	46	247872

```
Overall Statistics

               Accuracy : 0.9814
                 95% CI : (0.981, 0.9819)
    No Information Rate : 0.6756
    P-Value [Acc > NIR] : < 2.2e-16
```

```
                Kappa : 0.964

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: AU Class: CA Class: DE Class: ES Class: FR Class: GB
Sensitivity           0.944878   0.95241  0.948494   0.95779   0.95358   0.95438
Specificity           0.999937   0.99958  0.999765   0.99929   0.99878   0.99949
Pos Pred Value        0.986628   0.97276  0.975116   0.97406   0.98010   0.98034
Neg Pred Value        0.999731   0.99925  0.999500   0.99883   0.99709   0.99877
Prevalence            0.004863   0.01553  0.009620   0.02707   0.05915   0.02617
Detection Rate        0.004595   0.01479  0.009124   0.02593   0.05640   0.02498
Detection Prevalence  0.004657   0.01521  0.009357   0.02662   0.05755   0.02548
Balanced Accuracy     0.972407   0.97600  0.974130   0.97854   0.97618   0.97693
                     Class: IT Class: NL Class: other Class: PT Class: US
Sensitivity            0.95825   0.960697      0.9594   0.942797    0.9933
Specificity            0.99885   0.999751      0.9973   0.999973    0.9637
Pos Pred Value         0.97254   0.972771      0.9813   0.988889    0.9828
Neg Pred Value         0.99822   0.999637      0.9940   0.999853    0.9858
Prevalence             0.04092   0.009162      0.1293   0.002556    0.6756
Detection Rate         0.03921   0.008802      0.1241   0.002410    0.6711
Detection Prevalence   0.04032   0.009048      0.1265   0.002437    0.6829
Balanced Accuracy      0.97855   0.980224      0.9783   0.971385    0.9785
```

En primer lugar, las variables que más influyen en el destino son el proveedor de afiliación, la fecha de la primera reserva, la edad del usuario, el tipo de dispositivo con el que se ha accedido a la web, el lenguaje del usuario y el primer explorador empleado. La matriz de confusión muestra que los elementos de su diagonal principal, los que corresponden al número de instancias correctamente clasificadas por el algoritmo, son mucho mayores que los demás valores de la matriz. Los destinos donde más falla el algoritmo son ”other” y ”US”, que a su vez son los dos más frecuentes en la tabla `cruce1`. En cuanto a las demás métricas de la clasificación, los resultados para predecir cada uno de los destinos también son buenos, ya que se mantienen altos los que corresponden a una buena clasificación (*sensitivity*, *specificity*,*pos*

pred values, *neg pred values*, *balanced accuracy*). Seguidamente se explican las métricas más relevantes gracias a la información de las fuentes (R, 2019a), (DataSchool, 2015), (Villalonga, 2019) y (Kuhn y col., 2008):

- *Accuracy*: Media de la exactitud del algoritmo para cada clase. La exactitud proporciona información sobre la cantidad de registros de los datos de entrenamiento que han sido clasificados en la clase correspondiente. Cuanto más cerca esté el valor de uno, mayor será la probabilidad de acierto en la recomendación del primer destino.

- *No Information Rate*: Proporción de registros que no aportan información para el modelo que construye el algoritmo. Cuanto menor sea este valor, el conjunto de datos constará de información más relevante para su fin.

- Kappa: Indicador para medir cuanto de bien ha actuado el modelo clasificador frente a lo bien que lo haría si elegiese las categorías al azar. Se tiene un valor elevado del índice Kappa si hay gran diferencia entre la exactitud y la tasa de error del algoritmo, implicando mejores resultados predictivos para los índices Kappa más cercanos a uno.

- *Sensitivity*: También conocida como TP Rate (*True Positive Rate*), se calcula por cada categoría mediante la división de la cantidad de instancias positivas correctamente clasificadas con la cantidad total de instancias positivas. Aporta información sobre la proporción de instancias positivas correctamente predichas, por lo tanto, cuanto más cerca esté el valor uno mejor será la predicción.

- *Specificity*: Métrica conocida como TN Rate (*True Negative Rate*) que se computa por cada categoría a través de la división entre la cantidad de registros negativos correctamente clasificados y la cantidad total de registros negativos. Sirve para extraer información sobre la proporción de instancias negativas correctamente clasificadas, siendo los mejores valores los más próximos a la unidad.

- *Pos Pred Values*: Métrica que se calcula a través de la siguiente operación, donde los valores más próximos a uno corresponden a los algoritmos con mejor capacidad de clasificar correctamente una nueva instancia:

$$\frac{sensitivity \cdot prevalence}{sensitivity \cdot prevalence + (1 - specificity) \cdot (1 - prevalence)}$$

- *Neg pred values*: Métrica que se computa mediante la siguiente operación, donde los valores más cercanos a uno implican mejores resultados en cuanto a las clasificaciones

correctas:

$$\frac{specificity \cdot (1 - prevalence)}{(1 - sensitivity) \cdot prevalence) + specificity \cdot (1 - prevalence)}$$

- *Prevalence*: Métrica que estudia, para cada categoría, la cantidad de veces que aparece en la tabla cada una de las diferentes categorías a predecir, mediante una proporción. En la salida se observa que en `cruce1` el destino más recurrente corresponde a los Estados Unidos, seguido de ”other” y Francia, pues a mayor prevalencia, mayor es el número de apariciones en la base de datos.

- *Detection rate*: Métrica que calcula la proporción entre la cantidad de clasificaciones correctas de cada categoría y la cantidad de instancias que han sido clasificadas en otras categorías. Cuanto mayor sea esta métrica, mayor será tanto la correcta clasificación de cada categoría como la cantidad de apariciones en los registros de la misma.

- *Detection prevalence*: Métrica que se calcula por cada categoría mediante la división entre la suma de las instancias que el algoritmo predice que corresponden a esa categoría y el total de valores de la matriz de confusión. Cuanto mayor es el resultado mayor será el número de recomendaciones que se va a hacer a ese destino. Lo más conveniente es que los valores de las métricas *Detection rate* y *Detection prevalence* estén próximos ya que, en caso de tener un conjunto de datos que refleja la realidad, si un destino aparece más frecuentemente que el resto, también debe ser recomendado con mayor frecuencia.

- *Balanced Accurary*: Métrica que se calcula mediante el promedio entre la sensibilidad y la especificidad, es decir, la media entre TP Rate y TN Rate. Cuanto más cerca esté el valor de esta métrica de uno mayor será la capacidad del algoritmo de clasificar correctamente las instancias.

El algoritmo C5.0 para el conjunto de datos `cruce1` presenta una exactitud del 98,14 % y un índice Kappa muy cercano al uno. Además, tanto el TP Rate como el TN Rate están por encima del 94 % en todas las clases, lo cual conlleva a una exactitud balanceada en torno al 97 % en cada destino. A pesar de que el ratio de no información es del 67,56 %, se concluye que el algoritmo, en general, tiene muy buenos resultados de clasificación y es una buena elección como motor del sistema de recomendación.

Una vez se ha elegido tanto la mejor base de datos como el algoritmo que mejores resultados arroja, es necesario estudiar ambos entornos, el de *Big Data* y el de *Business Intelligence*,

y la optimización de recursos y el tiempo de ejecución de cada uno de ellos. Este proceso es necesario para decidir el mejor entorno para abordar el problema.

A lo largo de las distintas ejecuciones, se ha observado como el tiempo de ejecución con Spark ha sido mucho más bajo que el de R. Para las redes neuronales, por ejemplo, mientras que R necesita de media 4 horas para cada ejecución con una capa oculta, Spark necesita 2 minutos para tres capas ocultas con el mismo conjunto de datos. Por ello, el corto tiempo de ejecución es un valioso motivo por el que el entorno *Big Data* resulta más atractivo que el de *Business Intelligence*. A pesar de ello, las herramientas de Big Data son relativamente nuevas (AdelaideX, 2018) y no ofrecen un entorno estadístico tan completo como sí que lo hacen otras herramientas, como se ha podido descubrir al elaborar esta memoria. Además, la comunidad de usuarios es mucho mayor en el caso de R que en el de Spark (Martínez, 2019). Este motivo hace que el entorno *Business Intelligence* sea más útil que el de *Big Data*.

Ante esta situación de entornos bastante igualados, resulta victorioso el entorno *Business Intelligence* debido a la comunidad de usuarios, el tiempo no excesivo para la ejecución del algoritmo C5.0 (necesita sólo 5 minutos) y el hecho de que este algoritmo no está disponible en Spark y sí en R.

En conclusión, el algoritmo de *machine learning* elegido para implementar el sistema de recomendación basado en conocimiento para un nuevo usuario es C5.0 y se llevará a cabo en un entorno de *Business Intelligence*. De este modo, el código para la implementación del sistema de recomendación se presenta en las siguientes líneas:

```
#Se recibe una nueva instancia
input <- data.frame('date_first_booking_2009'=1830, 'month_account_created'=1,
'gender'='FEMALE', 'age'=0, 'language'='en', 'affiliate_provider'='google',
'first_affiliate_tracked'='linked', 'first_browser'='Safari',
'action_detail'='null', 'secs_elapsed'=4880, 'device_type'='Mac Desktop')

#Se carga el modelo
library(C50)
data <- read.csv("E:\\Raquel\\TFM\\Machine Learning with R\\cruce1.csv",
header=TRUE)
model1<- C5.0(train[,-12], train[,12])

#Se le hace la predicción
```

```
output <- predict(model1, input)

output
```

7. Conclusiones

Hoy en día, casi todas las compañías tienen que hacer frente, en algún momento, a la presencia de un nuevo cliente (Adomavicius y Tuzhilin, 2005). Debido a la era digital actual, resulta más fácil deducir los intereses e inquietudes de este cliente gracias a la huella digital que deja al navegar por internet que con la información que proporciona a través de otros medios (Srinivasan, 2017). La empresa de alquileres vacacionales Airbnb (Airbnb, 2019b) es una de las empresas que ha tomado la iniciativa de utilizar la información que puede extraer sobre sus clientes de internet para enfrentarse al problema de hacer una recomendación a un nuevo usuario, también conocido como *cold start*. Para ello, la compañía ha proporcionado una base de datos *open source* a través de un concurso para que los analistas de datos que estén interesados presenten sus propuestas. Esta base de datos es de la que se nutre el contenido de este Trabajo de Fin de Máster.

La memoria dio comienzo con el estudio del problema al que se le quiere hacer frente, es decir, el de hacer una recomendación de un destino a un nuevo usuario de Airbnb (Airbnb, 2019b), y con el estudio también de las diferentes propuestas que había hasta la fecha de la memoria. Después, se analizó de forma detallada la base de datos que la empresa hizo pública y se hizo un procesamiento de los datos, donde se llegó hasta la creación de dos tablas con gran potencial para este propósito: una tabla con 2.462.325 registros (`cruce1`) y otra con 28.764 (`cruce2`). Después de pulir y tratar algo más estas dos tablas, se dio comienzo al proceso de *machine learning*. En él, se probaron distintas técnicas que las referencias de la bibliografía expresaban que aportaban buenos resultados, tanto en un entorno de *Business Intelligence* como de *Big Data*. En estos entornos, además, se experimentaron los problemas que puede conllevar el hacer ejecuciones de un tamaño decente en un ordenador con RAM insuficiente. A pesar de ello, se siguió adelante y, con los resulados que se recogieron, se determinó el conjunto de datos, el algoritmo y el entorno más adecuados para construir el sistema de recomendación: la tabla `cruce1`, el algoritmo C5.0 y el entorno *Business Intelligence*. Finalmente, se programó un sistema de recomendación basado en conocimiento.

Llegados a este punto, ha sido una lástima no haber podido aprovechar todas las opciones de velocidad de procesamiento de Spark debido a la limitación del equipo. A pesar de ello, se han estudiado las funcionalidades que tiene Spark a día de esta memoria, llegando a las siguientes conclusiones:

- Tiene un entorno SQL completo y fácil de usar.

- Presenta limitaciones en herramientas de análisis estadístico por parte de SparkML y SparkMLlib que otras herramientas con mayor antigüedad no tienen.
- Carece de la implementación de muchas técnicas de aprendizaje automático.
- Tiene una comunidad de usuarios reducida.

En cuanto al tipo de datos que se han manejado y las características que presentan, las dificultades que se han encontrado se tratan a continuación:

- Escasez de herramientas para el tratamiento estadístico de variables cualitativas.
- Algoritmos de aprendizaje automático orientados e implementados en su mayoría para variables de salida cuantitativas o cualitativas binomiales.
- Algoritmos de *machine learning* implementados para atributos de entrada sobre todo cuantitativos.
- Problemas de memoria para conjuntos de datos con más de dos millones de instancias, variables de salida multinomial y atributos de entrada mixtos, donde los cualitativos presentan, en algunos casos, más de 20 categorías.

A pesar de estas dificultades, tanto en las herramientas de *Big Data* como con la naturaleza de los datos con los que se ha trabajado, se ha alcanzado el objetivo: modelizar los datos con algoritmos de *machine learning* como motor de un sistema de recomendación basado en conocimiento y poder sugerir a un primer usuario de Airbnb (Airbnb, 2019b) su primer destino en función de su actividad, tipo de dispositivo, edad, sexo y otras variables ricas en información.

8. Trabajo futuro

La base de datos con la que se ha elaborado la memoria ha permitido la culminación de este Trabajo de Fin de Máster, pero también ha dejado algún objetivo sin satisfacer. Al principio de la memoria se pretendía construir un sistema de recomendación capaz de diferenciar a los usuarios con intenciones de reservar con Airbnb (Airbnb, 2019b) de los que no antes de llevar a cabo la recomendación del destino. Como se ha visto en la Sección 4 de la memoria, las tablas que constituyen la base de datos, aunque en un principio parecía que podían cumplir con este propósito, han acabado por no hacerlo. Por ello, con una base de datos más completa se podría estudiar esta posibilidad y agregar ese paso previo al sistema de recomendación.

Además, el equipo con el que se ha trabajado en un entorno simulado tanto de *Business Intelligence* como de *Big Data* ha supuesto un condicionamiento importante en ambos entornos: la limitación de la memoria. De tener un equipo más potente para ambos entornos, el estudio de las técnicas de *machine learning* habría sido más exhaustivo y completo, al tener más resultados con los que comparar.

Por otro lado, queda abierto el estudio de otras técnicas de aprendizaje automático para hacerle frente al problema como, por ejemplo, otras variables de redes neuronales, al ser éste un área de conocimiento muy amplio. Asimismo, la combinación de distintas técnicas también puede ser otra alternativa a estudiar, especializándose cada uno de los diferentes métodos en la predicción de uno o varios destinos. Esto sería muy positivo ya que, como en la base de datos la mayoría de las instancias tienen como país de destino Estados Unidos, algunos de los algoritmos de la Sección 5 eran capaces de predecir, con alta probabilidad de acierto, sólo el susodicho país junto a unos pocos más, algoritmos que han sido descartados en la Sección 6. Con esta idea se puede llegar a elaborar otro TFM y completar con estas aportaciones los conocimientos que se tienen de los sistemas de recomendación basados en conocimiento.

Finalmente, la recomendación a un nuevo usuario se ha hecho utilizando un sistema de recomendación basado en conocimiento al ser el más adecuado debido a la información de la base de datos. A pesar de ello, también resulta interesante el estudio de este mismo problema con un sistema de recomendación híbrido (a elegir la combinación de sistemas de los que estarría compuesto). De este modo, se podrían comparar ambos sistemas y decidir cuál es el que hace las sugerencias más acertadas.

Referencias

Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, (58), 240-242.

Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, (5), 157-175.

Fisher, R. (1922). On the Mathematical Foundations of theoretical Statistics. *Phil. Trans. A.* 309-368.

Jarník, V. (1930). About a certain minimal problem. *Práce Moravské Přírodovědecké Společnosti*, (6), 57-63.

Crámer, H. (1946). *Mathematical Methods of Statistics*. Princeton: Princeton University Press.

Mann, H. y Whitney, D. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, (18), 50-60.

Smirnov, N. (1948). Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, (9), 279-381.

Kruskal, W. y Wallis, W. (1952). Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, (47), 583-621.

Birnbaum, Z. (1956). On a use of the Mann-Whitney statistic. *In Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics. The Regents of the University of California.*

Burkov, A. (1956). *The Hundred-Page Machine Learning Book*.

Quinlan, J. (1986). *Induction of Decision Trees*.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Ho, T. (1995). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC*, 278-282.

Ríus, F. (1998). *Bioestadística: Métodos y aplicaciones. Málaga: Universidad de Málaga. Publicaciones*.

Abbate, J. (1999). Inventing the Internet Cambridge. MA: MIT Press.

Breiman, L. (1999). Random forests. *Machine learning*, (45), 5-32.

Cortes, C. y N., V. (1999). Support-vector networks. *Machine learning*, (20), 273-297.

Sánchez, J. (2000). *Técnicas de Análisis de Datos Nominales*. Retrieved, from UCM

Amazon. (2001). Web de Amazon. Retrieved junio 2, 2019, from https://www.amazon.es/

Linden, G., Smith, B. y York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1), 76-80.

Montaner, M., López, B. y de la Rosa, J. (2003). Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, *19*, 285-330. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.2519&rep=rep1&type=pdf

Adomavicius, G. y Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge Data Engineering*, *6*, 734-749. Retrieved from https://ieeexplore.ieee.org/document/1423975

Kuhn, M. y col. (2008). Building predictive models in R using the caret package. *Journal of statistical software*, *28*(5), 1-26.

Conesa, J. y Curto, J. (2010). *Introducción al Business Intelligence*. UOC.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., ... y D., S. (2010). The YouTube Video Recommendation System.

Guggilla, P., Gutha, S. y Chakraborty, D. (2011). Price Recommendation Engine for Airbnb.

Ricci, F., Rokach, L. y Shapira, B. (2011). Recommender System HandBook. *Proceedings of The 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, United Kingdom*.

Amatriain, X. y Basilico, J. (2012a). Netflix Recommendations: Beyond the 5 stars (Part 1). Retrieved from https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

Amatriain, X. y Basilico, J. (2012b). Netflix Recommendations: Beyond the 5 stars (Part 2). Retrieved from https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5

Field, A. (2013). *Discovering statistics using IBM SPSS statistics*.

Tech, E. (2013). Demographic Filters – Search for People by Religion, Ethnicity, Marital Status, Political Party, Annual Income and More. Retrieved junio 2, 2019, from https://www.cbinsights.com/research/demographic-people-search/

McDonald, J. (2014). Kruskal-Wallis test. Retrieved mayo 29, 2019, from http://www.biostathandbook.com/kruskalwallis.html

News, A. (2014). Web de ACR News. Retrieved junio 5, 2019, from https://www.acr-news.com/

Wu, L., Shah, S., Choi, S., Tiwari, M. y Posse, C. (2014). The Browsemaps: Collaborative Filtering at LinkedIn.

DataSchool. (2015). Simple guide to confusion matrix terminology. Retrieved julio 5, 2019, from https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Gorakala, S. K. y Usuelli, M. (2015). *Building a Recommendation System with R*.

Kabiljo, M. e Ilic, A. (2015). Recommending items to more than a billion people. Retrieved mayo 4, 2019, from https://code.fb.com/core-data/recommending-items-to-more-than-a-billion-people/

Kaggle. (2015a). Airbnb New User Bookings. Retrieved abril 23, 2019, from https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/overview/evaluation

Kaggle. (2015b). Web de Kaggle del concurso. Retrieved abril 23, 2019, from https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/overview

Mardones, I. (2015). *Aljebra Lineala eta Geometria II*. Retrieved, from UPV

Amat, J. (2016). Kruskal-Wallis test. Retrieved junio 1, 2019, from https://rpubs.com/Joaquin_AR/219504

Brownlee, J. (2016). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved julio 13, 2019, from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Covington, P., Adams, J. y Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. *RecSys '16 Proceedings of the 10th ACM Conference on Recommender Systems*, 191-198.

Ripley, B. y Venables, W. (2016). Package 'nnet". Retrieved junio 3, 2019, from https://cran.r-project.org/web/packages/nnet/nnet.pdf

Team, K. (2016a). Airbnb New User Bookings, Winner's Interview: 2nd place, Keiichi Kuroyanagi (Keiku). Retrieved abril 23, 2019, from http://blog.kaggle.com/2016/03/17/airbnb-new-user-bookings-winners-interview-2nd-place-keiichi-kuroyanagi-keiku/

Team, K. (2016b). Airbnb New User Bookings, Winner's Interview: 3rd place: Sandro Vega Pons. Retrieved abril 23, 2019, from http://blog.kaggle.com/2016/03/07/airbnb-new-user-bookings-winners-interview-3rd-place-sandro-vega-pons/

Aroztegi, I. (2017). *Estatistika Matekatikoa: Hipotesi Kontrastea*. Retrieved, from UPV

Chouksey, P. y Chauhan, A. (2017). A Review of Weather Data Analytics using Big Data. *International Journal of Advanced Research in Computer and Communication Engineering*.

Mahajan, V. (2017). Profile-based-recommendation-system-for-Airbnb-users—Big-Data-and-Machine-Learning-project-. Retrieved abril 23, 2019, from https://github.com/vasantivma%20%5C%5C%20hajan/Profile-based-recommendation-system-for-Airbnb-users---Big-Data-and-Machine-Learning-project-

Rezaul, M. y Alla, S. (2017). *Scala and Spark for Big Data Analytics*.

Slavio, J. (2017). *Deep Learning and Artificial Intelligence: A Beginners' Guide to Neural Networks and Deep Learning*.

Srinivasan, S. (2017). *Guide to Big Data Applications*.

SurveyMonkey. (2017). Why filtering your data by demographic groups is a game-changer. Retrieved junio 2, 2019, from https://www.surveymonkey.com/curiosity/why-filtering-your-data-by-demographic-groups-is-a-game-changer/

AdelaideX. (2018). Big Data Analytics. Retrieved mayo 24, 2019, from https://courses.edx.org/courses/course-v1:AdelaideX+AnalysisX+3T2017/course/

Aroztegi, I. y Barrio, I. (2018). *Aldagai Anitzetako Analisia*. Retrieved, from UPV

Bhandari, N. (2018). ExtraTreesClassifier. Retrieved julio 13, 2019, from https://medium.com/@namanbhandari/extratreesclassifier-8e7fc0502c7

Cheng, Y., Chen, K., Sun, H., Zhang, Y. y Tao, f. (2018). Data and knowledge mining with big data towards smart production. *Journal of Industrial Information Integration*, *9*, 1-13.

Grbovic, M. y Cheng, H. (2018). Real-time Personalization using Embeddings for Search Ranking at Airbnb. *Proceedings of The 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, United Kingdom*.

Grbovic, M., Cheng, H., Zhang, Q., Yang, L., Siclait, P. y Jones, M. (2018). Listing Embeddings in Search Ranking. Retrieved abril 20, 2019, from https://medium.com/airbnb-engineering/listing-embeddings-for-similar-listing-recommendations-and-real-time-personalization-in-search-601172f7603e

Grbovic, M., Wu, E., Liu, P., Tan, C. H., Wu, L., Yu, B. y Tian, A. (2018). Machine Learning-Powered Search Ranking of Airbnb Experiences. Retrieved abril 20, 2019, from https://medium.com/airbnb-engineering/machine-learning-powered-search-ranking-of-airbnb-experiences-110b4b1a0789

Haldar, M., Abdool, M., Ramanathan, P., Xu, T., Yang, S., Duan, H., ... y Legrand, T. (2018). Applying Deep Learning To Airbnb Search. *ArXiv e-prints*.

IBM. (2018). Infographics and Animations: The Four V's of Big Data. Retrieved junio 13, 2019, from https://www.ibmbigdatahub.com/infographic/four-vs-big-data

Orellana Alvear, J. (2018). Árboles de decisión y Random Forest.

Pelánek, R. (2018). Recommender Systems: Content-based, Knowledge-based, Hybrid. Retrieved abril 21, 2019, from https://www.fi.muni.cz/~xpelanek/PV254/slides/other-techniques.pdf

R. (2018). Package 'C50', 1-13.

Singh, H. (2018). Understanding Gradient Boosting Machines. Retrieved julio 1, 2019, from https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

ABC. (2019). Una peruana de 122 años reclama ser reconocida como la mujer más longeva del mundo. Retrieved abril 17, 2019, from https://www.abc.es/sociedad/abci-peruana-122-anos-reclama-reconocida-como-mujer-mas-longeva-mundo-201903141016_noticia.html

Airbnb. (2019a). Política de privacidad de Airbnb. Retrieved mayo 14, 2019, from https://www.airbnb.es/terms/privacy_policy#sec2

Airbnb. (2019b). Web oficial de Airbnb. Retrieved abril 29, 2019, from https://www.airbnb.es/

Canela, J. (2019). *Análisis e Interpretación de Datos*. Retrieved, from UNIR

Firican, G. (2019). The 10 Vs of Big Data. Retrieved mayo 24, 2019, from https://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx

Fritsch, S., Guenther, F., Wright, M., Suling, M. y Mueller, S. (2019). Package 'neuralnet'. Retrieved junio 3, 2019, from https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

Google. (2019). Web oficial de Google. Retrieved abril 17, 2019, from https://www.google.com/

Greenwell, B., Boehmke, B., Cunningham, J. y Developers, G. (2019). Package 'gbm'. Retrieved junio 2, 2019, from https://cran.r-project.org/web/packages/gbm/gbm.pdf

Hadoop. (2019a). Web oficial de Apache Hadoop. Retrieved julio 10, 2019, from https://hadoop.apache.org/

Hadoop. (2019b). Web oficial de Apache Hadoop con guía para HDFS. Retrieved julio 10, 2019, from https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

IBM. (2019). Nodo C5.0. Retrieved mayo 31, 2019, from https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/clementine/c50node_general.html

Kaggle. (2019). Web oficial de Kaggle. Retrieved abril 10, 2019, from https://www.kaggle.com/

Liaw, A. (2019). randomForest v4.6-14. Retrieved junio 1, 2019, from https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest

Martínez, R. (2019). *Privacidad y Protección de Datos*. Retrieved, from UNIR

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C. y Lin, C.-C. (2019). Package 'e1071'. Retrieved junio 2, 2019, from https://cran.r-project.org/web/packages/e1071/e1071.pdf

Olson, M. (2019). adaboost v2.1.0. Retrieved junio 1, 2019, from https://www.rdocumentation.org/packages/JOUSBoost/versions/2.1.0/topics/adaboost

R. (2019a). confusionMatrix: Create a confusion matrix. Retrieved junio 5, 2019, from https://rdrr.io/cran/caret/man/confusionMatrix.html

R. (2019b). Web oficial de R Project. Retrieved julio 10, 2019, from https://www.r-project.org/

Rajendra, N., Dewan, A. y Can Colakoglu, M. (2019). CS229 Project - Best Buy Recommendation System. *http://cs229.stanford.edu/proj2012/RajendraDewanColakoglu-BestBuy RecommendationSystem.pdf*.

Research, R. (2019). Is See5/C5.0 Better Than C4.5? Retrieved mayo 31, 2019, from https://rulequest.com/see5-comparison.html

Retail. (2019). Cómo gestionar el stock en la era del Big Data. Retrieved julio 13, 2019, from http://analiticaderetail.com/como-gestionar-el-stock-bid-data/

Solutions, S. (2019). Pearson's Correlation Coefficient. Retrieved mayo 29, 2019, from https://www.statisticssolutions.com/pearsons-correlation-coefficient/

Spark. (2019a). Ensembles - RDD-based API. Retrieved junio 1, 2019, from https://spark.apache.org/docs/latest/mllib-ensembles.html#random-forests

Spark. (2019b). Gradient-boosted tree classifier. Retrieved junio 2, 2019, from https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#gradient-boosted-tree-classifier

Spark. (2019c). Linear Support Vector Machine. Retrieved junio 2, 2019, from https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#linear-support-vector-machine

Spark. (2019d). Multilayer perceptron classifier. Retrieved junio 3, 2019, from https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

Spark. (2019e). One-vs-Rest classifier (a.k.a. One-vs-All). Retrieved junio 2, 2019, from https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#one-vs-rest-classifier-aka-one-vs-all

Spark. (2019f). Web oficial de Apache Spark. Retrieved julio 10, 2019, from https://spark.apache.org/

SparkMLib. (2019). Web oficial de Apache SparkMLlib. Retrieved julio 10, 2019, from https://spark.apache.org/mllib/

SparkSQL. (2019). Web oficial de Apache SparkSQL. Retrieved julio 10, 2019, from https://spark.apache.org/sql/

Teradata. (2019). Web oficial de Teradata. Retrieved julio 10, 2019, from https://www.teradata.com.es/

Transfinder. (2019). Web de Infofinder. Retrieved junio 5, 2019, from https://www.transfinder.com/solutions/school_transportation_communication

Villalonga, C. (2019). *Técnicas de Inteligencia Artificial*. Retrieved, from UNIR

Webmate. (2019). Web de Webmate. Retrieved junio 5, 2019, from https://webmate.me/

9. Anexos

En esta sección se van a completar los apartados que se han dejado abiertos en la memoria, para dar lugar a una memoria más completa, clara y autocontenida. Por ello, a continuación se va a completar la información sobre el contenido de las tablas `train` y `sessions`. Además, también se aporta en esta sección el código que se ha utilizado para lograr los resultados de la memoria. Cabe recordar que, debido al doble objetivo que consiste no sólo en crear un sistema de recomendación sino que también en medir la efectividad de las herramienteas de Big Data frente a las convencionales para llegar al objetivo, el código de la parte de *machine learning* está implementado tanto con Spark como con R.

9.1. Descripción de las variables restantes de la tabla train

En el planteamiento del problema, Sección 4, se ha comentado que los atributos `language`, `affiliate_channel`, `affiliate_provider`, `first_affiliate_tracked`, `signup_app`, `first_ device_type` y `first_browser` van a ser tratados en este anexo debido a la gran cantidad de categorías de las que se componen. A continuación, se da comienzo al resumen de los atributos, ordenando la tabla de mayor a menor proporción de aparición y empezando por `language` en la Tabla 49:

language	
Categorías	**Proporción de cada categoría (%)**
en	97,48
zh	0,59
fr	0,45
es	0,31
de	0,30
ko	0,27
ru	0,12
it	0,12
ja	0,09
pt	0,07
sv	0,05
nl	0,04
tr	0,02
da	0,02
pl	0,01
no	0,01
cs	0,01

fi	<0,01
el	<0,01
hu	<0,01
th	<0,01
is	<0,01
ca	<0,01

Tabla 49: Proporción de las categorías del atributo `language` de la tabla `train`, compuesta de 88.908 registros.

El lenguaje de mayor frecuencia entre los registros es el inglés (en), hecho que tiene sentido ya que todos los usuarios son estadounidenses. El resto de los idiomas se encuentran en el 2,52 % de los registros restantes, llegando a haber idiomas o categorías que sólo son elegidos por un único usuario.

affiliate_channel	
Categorías	**Proporción de cada categoría (%)**
direct	66,81
sem-brand	12,48
sem-non-brand	8,04
seo	4,45
other	4,05
api	3,13
content	0,63
remarketing	0,41

Tabla 50: Proporción de las categorías del atributo `affiliate_channel` de la tabla `train`, compuesta de 88.908 registros.

En el atributo `affiliate_channel`, representado en la Tabla 50, destaca que la mayoría de los nuevos usuarios han sido captados mediante técnicas de afiliación directa, mientras que el *remarketing* y ”content” no han obtenido tan buenos resultados. El resto de los grupos o categorías están distribuidos de forma bastante equitativa entre los demás registros.

affiliate_provider	
Categorías	**Proporción de cada categoría (%)**
direct	66,62
google	23,37
other	5,24
craigslist	1,82
bing	0,94
facebook	0,89

vast	0,30
padmapper	0,28
yahoo	0,20
facebook-open-graph	0,16
meetup	0,05
gsp	0,04
email-marketing	0,04
naver	0,02
baidu	0,01
yandex	<0,01
daum	<0,01

Tabla 51: Proporción de las categorías del atributo `affiliate_provider` de la tabla `train`, compuesta de 88.908 registros.

Para `affiliate_provider` hay una diferencia mayor entre la distribución de cada categoría en la base de datos que en la variable anterior, como se puede ver en la Tabla 51. De nuevo, el proveedor de afiliación directa es el que más usuarios atrae, seguido de google y de otros distribuidores. El resto de los grupos representan un 4,77 % del total, habiendo categorias con una única aparición. Esto significa que la muestra con la que se trabaja tiene categorías muy poco significativas.

first_affiliate_tracked	
Categorías	**Proporción de cada categoría (%)**
untracked	53,70
linked	22,61
omg	18,47
tracked-other	2,58
null	1,95
product	0,61
marketing	0,07
local ops	0,01

Tabla 52: Proporción de las categorías del atributo `first_affiliate_tracked` de la tabla `train`, compuesta de 88.908 registros.

El atributo `first_affiliate_tracked`, descrito con la Table 52, tiene como grupo más frecuente el que no ha podido ser rastreado por el sistema pero que no es un campo vacío. En segunda y tercera posición están ”linked” y ”omg” respectivamente. En el punto de vista contrario en cuanto a la frecuencia, los menos recurrentes son ”product”, ”marketing” y ”local ops”. Obsérvese que este atributo tiene valores nulos pero éstos sólo aparecen en el 1,95 % de los registros.

signup_app	
Categorías	**Proporción de cada categoría (%)**
Web	90,00
iOS	6,20
Moweb	2,41
Android	1,39

Tabla 53: Proporción de las categorías del atributo `signup_app` de la tabla `train`, compuesta de 88.908 registros.

En cuanto a la información sobre las aplicaciones con las que los usuarios se han registrado que proporciona la Tabla 53, la más frecuente es el registro a través de la web, seguido del sistema operativo iOS de Apple. Los menos usados, pero con poca diferencia respecto a iOS, son Moweb y los móviles con sistema operativo Android.

first_device_type	
Categorías	**Proporción de cada categoría (%)**
Mac Desktop	47,52
Windows Desktop	33,91
iPhone	7,22
iPad	6,15
Other/Unknown	3,23
Android Phone	0,80
Desktop (Other)	0,62
Android Tablet	0,52
SmartPhone (Other)	0,03

Tabla 54: Proporción de las categorías del atributo `first_device_type` de la tabla `train`, compuesta de 88.908 registros.

La variable `first_device_type` está liderada en la base de datos por la variable ”Mac Desktop”, seguido de cerca por ”Windows Desktop”. Esta información, junto a los dos siguientes grupos más recurrentes, transmiten que los usuarios de la muestra con la que se trabaja se ven bastante atraídos por los productos de la marca Apple. En cuanto a los menos recurrentes de la Tabla 54, están otro tipo de escritorios, las tablets de Android y otro tipo de teléfonos inteligentes.

first_browser	
Categorías	**Proporción de cada categoría (%)**
Chrome	33,60
Safari	22,18
Firefox	17,69
IE	8,69
-unknown-	8,57
Mobile Safari	8,08
Chrome Mobile	0,46
Android Browser	0,31
Opera	0,07
AOL Explorer	0,07
Silk	0,05
Chromium	0,05
BlackBerry Browser	0,02
Maxthon	0,02
Apple Mail	0,02
Mobile Firefox	0,02
RockMelt	0,01
Sogou Explorer	0,01
Iron	0,01
IceWeasel	<0,01
SiteKiosk	<0,01
Camino	<0,01
IE Mobile	<0,01
SeaMonkey	<0,01
Pale Moon	<0,01
TenFourFox	<0,01
CoolNovo	<0,01
wOSBrowser	<0,01
TheWorld Browser	<0,01
Opera Mini	<0,01
Avant Browser	<0,01
Kindle Browser	<0,01
Yandex.Browser	<0,01
Stainless	<0,01
Mozilla	<0,01
SlimBrowser	<0,01
Googlebot	<0,01
CometBird	<0,01
NetNewsWire	<0,01
Palm Pre web browser	<0,01

Tabla 55: Proporción de las categorías del atributo `first_browser` de la tabla `train`, compuesta de 88.908 registros.

El atributo `first_browser`, estudiado a través de la Tabla 55, tiene como grupo más recurrente el explorador Chrome, seguido de cerca por Safari y Firefox. La frecuencia del resto

de categorías es relativamente menor que las comentados, apareciendo también la versión móvil para los tres anteriores entre ellos. Este atributo presenta tantas categorías que es de esperar que haya grupos como CometBird con sólo una aparición entre los registros.

Recordando lo comentado al principio, todos estos atributos tienen en común la gran cantidad de categorías y la baja frecuencia de aparición de alguna de ellas. En el proceso de elaboración de la memoria se ha planteado la reducción de las categorías de las variables mediante el agrupamiento de las mismas, pudiendo aplicar uno de los dos siguientes criterios:

- Reducción por frecuencia de aparición, agrupando las categorías menos frecuentes, con una frecuencia menor al 1 % para evitar una gran pérdida de información.
- Disminución por significado, agrupando las categorías por significados más similares o cercanos en cuanto al contexto del atributo.

El primer criterio proporcionaría a `language`, `affiliate_channel`, `affiliate_provider`, `first_affiliate_tracked`, `signup_app`, `first_device_type` y `first_browser` 2, 7, 5, 6, 4, 6 y 7 nuevas categorías agrupadas, produciendo la pérdida de significado en algunas de ellas. Mientras tanto, el segundo criterio mantendría el significado pero sería más difícil de llevar a cabo dado el desconocimiento de los significados precisos de algunas de las categorías. Finalmente, se ha tomado la decisión de agrupar las categorías siguiendo el primer criterio, llevando a cabo esta acción en las tablas a las que se les aplicarán los algoritmos de *machine learning*. En la Subsección 4.7.2 de la Sección 4 es donde se puede apreciar el resultado de la agrupación de las categorías.

9.2. Descripción de las variables restantes de la tabla sessions

Al igual que ha sucedido para algunas variables de `train`, debido a la cantidad elevada de categorías, los atributos cualitativos nominales de `sessions` se van a resumir a continuación. Estas variables son `action`, `action_type`, `action_detail` y `device_type`, respetando este mismo orden en la posterior redacción y donde la información de cada atributo se resume en las Tablas 56, 57, 58 y 59, respectivamente.

action	
Categorías	**Proporción de cada categoría (%)**
show	26,11
index	7,96
search_results	6,84
personalize	6,67
search	5,05
ajax_refresh_subtotal	4,6
similar_listings	3,44
update	3,43
social_connections	3,2
reviews	3,02
active	1,77
similar_listings_v2	1,59
lookup	1,53
create	1,47
dashboard	1,44
header_userpic	1,34
collections	1,17
edit	1,03
campaigns	0,99
track_page_view	0,77
null	0,75
unavailabilities	0,74
qt2	0,61
notifications	0,56
confirm_email	0,55
requested	0,54
identity	0,51
ajax_check_dates	0,5
show_personalize	0,48
ask_question	0,42
authenticate	0,41
listings	0,41
calendar_tab_inner2	0,4
travel_plans_current	0,36
edit_verification	0,32
ajax_lwlb_contact	0,32
recommendations	0,29
other_hosting_reviews_first	0,29
impressions	0,26
manage_listing	0,25
click	0,24
ajax_photo_widget_form_iframe	0,23
complete_status	0,23
payment_instruments	0,22
verify	0,2
message_to_host_focus	0,2
payment_methods	0,19

cancellation_policies	0,18
callback	0,18
settings	0,16
custom_recommended_destinations	0,16
profile_pic	0,15
pending	0,15
populate_help_dropdown	0,14
ajax_image_upload	0,13
message_to_host_change	0,13
kba_update	0,12
view	0,12
ajax_get_referrals_amt	0,11
my	0,11
references	0,11
new	0,10
agree_terms_check	0,10
apply_reservation	0,10
connect	0,10
account	0,09
faq	0,09
populate_from_facebook	0,09
recommended_listings	0,09
qt_reply_v2	0,08
request_new_confirm_email	0,08
jumio_token	0,08
signup_login	0,08
available	0,08
phone_number_widget	0,07
handle_vanity_url	0,07
coupon_field_focus	0,07
set_user	0,07
open_graph_setting	0,07
kba	0,07
apply_coupon_click	0,06
reviews_new	0,06
faq_category	0,06
at_checkpoint	0,05
apply_coupon_error_type	0,05
login	0,05
endpoint_error	0,05
languages_multiselect	0,05
localization_settings	0,05
jumio_redirect	0,05
delete	0,05
apply_coupon_error	0,05
ajax_referral_banner_experiment_type	0,05
read_policy_click	0,04
this_hosting_reviews	0,04
status	0,04
hosting_social_proof	0,04

complete_redirect	0,04
facebook_auto_login	0,04
referrer_status	0,04
ajax_referral_banner_type	0,04
payout_preferences	0,04
uptodate	0,03
tell_a_friend	0,03
decision_tree	0,03
tos_confirm	0,03
cancellation_policy_click	0,03
push_notification_callback	0,03
phone_verification_number_submitted_for_sms	0,03
10	0,03
coupon_code_click	0,03
popular	0,02
ajax_statsd	0,02
terms	0,02
login_modal	0,02
signature	0,02
create_multiple	0,02
itinerary	0,02
recent_reservations	0,02
contact_new	0,02
transaction_history	0,02
signup_modal	0,02
phone_verification_number_sucessfully_submitted	0,02
host_summary	0,02
12	0,02
privacy	0,02
phone_verification_success	0,02
qt_with	0,02
pay	0,02
ajax_google_translate_reviews	0,01
15	0,01
11	0,01
requirements	0,01
top_destinations	0,01
add_note	0,01
glob	0,01
upload	0,01
webcam_upload	0,01
forgot_password	0,01
ajax_payout_options_by_country	0,01
agree_terms_uncheck	0,01
domains	0,01
ajax_payout_edit	0,01
spoken_languages	0,01
transaction_history_paginated	0,01
cancel	0,01
mobile_landing_page	0,01

my_reservations	0,01
listing	0,01
friends	0,01
travel_plans_previous	0,01
apply_code	0,01
ajax_google_translate_description	0,01
photography	0,01
other_hosting_reviews	0,01
faq_experiment_ids	0,01
change_currency	0,01
update_notifications	0,01
trust	0,01
phone_verification_error	0,01
signed_out_modal	0,01
how_it_works	0,01
my_listings	0,01
review_page	0,01
patch	0,01
position	0,01
phone_verification_number_submitted_for_call	0,01
update_cached	0,01
issue	0,01
change	0,01
this_hosting_reviews_3000	0,01
remove_dashboard_alert	0,01
phone_verification_modal	0,01
complete	0,01
supported	0,01
set_password	0,01
pending_tickets	0,01
authorize	0,01
travel	<0,01
guest_billing_receipt	<0,01
special_offer	<0,01
countries	<0,01
change_default_payout	<0,01
tos_2014	<0,01
ajax_photo_widget	<0,01
show_code	<0,01
weibo_signup_referral_finish	<0,01
request_photography	<0,01
overview	<0,01
add_guests	<0,01
set_minimum_payout_amount	<0,01
views_campaign_rules	<0,01
home_safety_terms	<0,01
localized	<0,01
ajax_special_offer_dates_available	<0,01
email_by_key	<0,01
phone_verification	<0,01

invalid_action	<0,01
sldf	<0,01
onenight	<0,01
update_reservation_requirements	<0,01
phone_verification_phone_number_removed	<0,01
clickthrough	<0,01
media_resources	<0,01
concierge	<0,01
respond	<0,01
reputation	<0,01
report	<0,01
deactivated	<0,01
email_share	<0,01
apply	<0,01
submit_contact	<0,01
destroy	<0,01
desks	<0,01
change_password	<0,01
reservation	<0,01
acculynk_pin_pad_error	<0,01
redirect	<0,01
slideshow	<0,01
payoneer_account_redirect	<0,01
new_session	<0,01
add_guest_colorbox	<0,01
receipt	<0,01
google_importer	<0,01
change_availability	<0,01
payout_delete	<0,01
wishlists	<0,01
mobile_oauth_callback	<0,01
plaxo_cb	<0,01
new_host	<0,01
country_options	<0,01
badge	<0,01
satisfy	<0,01
ajax_google_translate	<0,01
confirmation	<0,01
rate	<0,01
host_2013	<0,01
payout_update	<0,01
multi_message_attributes	<0,01
views	<0,01
track_activity	<0,01
reactivate	<0,01
has_profile_pic	<0,01
relationship	<0,01
maybe_information	<0,01
events	<0,01
why_host	<0,01

jumio	<0,01
airbrb	<0,01
revert_to_admin	<0,01
popular_listing	<0,01
create_ach	<0,01
nyan	<0,01
braintree_client_token	<0,01
deactivate	<0,01
apply_coupon_click_success	<0,01
acculynk_load_pin_pad	<0,01
update_friends_display	<0,01
stpcv	<0,01
zendesk_login_jwt	<0,01
ajax_worth	<0,01
life	<0,01
create_paypal	<0,01
multi_message	<0,01
hospitality	<0,01
toggle_starred_thread	<0,01
check	<0,01
sandy	<0,01
airbnb_picks	<0,01
open_hard_fallback_modal	<0,01
booking	<0,01
detect_fb_session	<0,01
friend_listing	<0,01
add_business_address_colorbox	<0,01
unread	<0,01
sync	<0,01
rentals	<0,01
unsubscribe	<0,01
social	<0,01
business_travel	<0,01
salute	<0,01
recommendation_page	<0,01
create_airbnb	<0,01
phone_verification_call_taking_too_long	<0,01
support_phone_numbers	<0,01
update_hide_from_search_engines	<0,01
multi	<0,01
become_user	<0,01
ajax_send_message	<0,01
founders	<0,01
city_count	<0,01
approve	<0,01
p4_refund_policy_terms	<0,01
locale_from_host	<0,01
acculynk_session_obtained	<0,01
questions	<0,01
departments	<0,01

guarantee	<0,01
terms_and_conditions	<0,01
host_cancel	<0,01
department	<0,01
acculynk_pin_pad_success	<0,01
about_us	<0,01
social-media	<0,01
ajax_ldp	<0,01
accept_decline	<0,01
toggle_availability	<0,01
toggle_archived_thread	<0,01
feed	<0,01
home_safety_landing	<0,01
preapproval	<0,01
acculynk_bin_check_failed	<0,01
currencies	<0,01
payoneer_signup_complete	<0,01
p4_terms	<0,01
ajax_get_results	<0,01
pricing	<0,01
print_confirmation	<0,01
press_news	<0,01
acculynk_pin_pad_inactive	<0,01
photography_update	<0,01
envoy_bank_details_redirect	<0,01
views_campaign	<0,01
signup_weibo_referral	<0,01
email_itinerary_colorbox	<0,01
hospitality_standards	<0,01
united-states	<0,01
office_location	<0,01
update_message	<0,01
place_worth	<0,01
set_default	<0,01
load_more	<0,01
rest-of-world	<0,01
signup_weibo	<0,01
sublets	<0,01
update_country_of_residence	<0,01
press_content	<0,01
envoy_form	<0,01
south-america	<0,01
clear_reservation	<0,01
refund_guest_cancellation	<0,01
email_wishlist	<0,01
recommend	<0,01
message	<0,01
press_release	<0,01
ajax_payout_split_edit	<0,01
image_order	<0,01

hard_fallback_submit	<0,01
ajax_price_and_availability	<0,01
southern-europe	<0,01
locations	<0,01
use_mobile_site	<0,01
friends_new	<0,01
reset_calendar	<0,01
book	<0,01
deauthorize	<0,01
widget	<0,01
acculynk_bin_check_success	<0,01
guest_booked_elsewhere	<0,01
disaster_action	<0,01

Tabla 56: Proporción de las categorías del atributo `action` de la tabla `sessions`.

La variable `action` consta de 359 categorías, de entre las cuales 178 tienen una presencia superior al 0,01 %. Además, sólo 18 de las categorías tienen una frecuencia de aparición por encima del 1 %. El grupo que más aparece en la Tabla 56 es ”show”, seguido de ”index” y ”search_results”, mientras que los grupos de menor aparición son un total de 181 . Se aprecia que este atributo está formado por categorías que mayormente tienen una baja frecuencia de aparición, siendo poco significativas.

action_type	
Categorías	**Proporción de cada categoría (%)**
view	33,59
data	19,84
click	18,83
null	10,63
-unknown-	9,73
submit	5,87
message_post	0,82
partner_callback	0,18
booking_request	0,18
modify	0,01
booking_response	<0,01

Tabla 57: Proporción de las categorías del atributo `action_type` de la tabla `sessions`.

El atributo que representa el tipo de acción está compuesto de categorías donde la mayoría tiene una frecuencia de aparición es superior al 1 %. Las categorías ”view”, ”data”, ”click” y ”null” son las más recurrentes entre los usuarios, mientras que la menos frecuente en la Tabla 57 es ”booking_response”.

action_detail	
Categorías	**Proporción de cada categoría (%)**
view_search_results	16,76
p3	12,98
null	10,63
-unknown-	9,73
wishlist_content_update	6,67
user_profile	6,19
change_trip_characteristics	4,60
similar_listings	3,44
user_social_connections	3,18
listing_reviews	2,54
update_listing	2,54
dashboard	1,44
user_wishlists	1,44
header_userpic	1,34
message_thread	1,25
edit_profile	1,03
message_post	0,82
contact_host	0,77
unavailable_dates	0,73
confirm_email_link	0,55
create_user	0,52
change_contact_host_dates	0,50
user_profile_content_update	0,48
user_reviews	0,47
p5	0,46
login	0,36
your_trips	0,36
p1	0,36
notifications	0,35
profile_verifications	0,32
reservations	0,31
user_listings	0,30
your_listings	0,29
listing_recommendations	0,29
update_user	0,27
create_phone_numbers	0,27
p4	0,26
update_listing_description	0,25
update_user_profile	0,25
manage_listing	0,25
payment_instruments	0,22
account_notification_settings	0,21
message_to_host_focus	0,20
signup	0,20
cancellation_policies	0,18
oauth_response	0,18
message_inbox	0,18

view_listing	0,17
message_to_host_change	0,13
list_your_space	0,13
pending	0,13
wishlist	0,12
profile_references	0,11
apply_coupon	0,10
oauth_login	0,10
view_reservations	0,09
login_page	0,09
post_checkout_action	0,08
send_message	0,08
trip_availability	0,08
signup_login_page	0,08
book_it	0,08
request_new_confirm_email	0,08
create_listing	0,07
view_locations	0,07
coupon_field_focus	0,07
apply_coupon_click	0,06
listing_descriptions	0,05
apply_coupon_error	0,05
at_checkpoint	0,05
account_payout_preferences	0,04
read_policy_click	0,04
listing_reviews_page	0,04
cancellation_policy_click	0,03
instant_book	0,03
coupon_code_click	0,03
photos	0,03
request_to_book	0,03
user_tax_forms	0,02
popular_wishlists	0,02
host_home	0,02
phone_verification_success	0,02
account_transaction_history	0,02
login_modal	0,02
terms_and_privacy	0,02
account_privacy_settings	0,02
guest_itinerary	0,02
signup_modal	0,02
lookup_message_thread	0,02
user_languages	0,01
profile_reviews	0,01
your_reservations	0,01
delete_phone_numbers	0,01
change_or_alter	0,01
account_payment_methods	0,01
wishlist_note	0,01
translate_listing_reviews	0,01

click_reviews	0,01
admin_templates	0,01
set_password_page	0,01
friends_wishlists	0,01
click_about_host	0,01
forgot_password	0,01
set_password	0,01
modify_users	0,01
previous_trips	0,01
guest_cancellation	0,01
guest_receipt	<0,01
move_map	<0,01
modify_reservations	<0,01
click_amenities	<0,01
email_wishlist_button	<0,01
toggle_archived_thread	<0,01
user_friend_recommendations	<0,01
p4_refund_policy_terms	<0,01
apply_coupon_click_success	<0,01
email_wishlist	<0,01
translations	<0,01
confirm_email	<0,01
p4_terms	<0,01
airbnb_picks_wishlists	<0,01
host_guarantee	<0,01
toggle_starred_thread	<0,01
alteration_field	<0,01
change_password	<0,01
cancellation_policy	<0,01
share	<0,01
view_security_checks	<0,01
remove_dashboard_alert	<0,01
delete_listing	<0,01
place_worth	<0,01
create_payment_instrument	<0,01
complete_booking	<0,01
view_identity_verifications	<0,01
calculate_worth	<0,01
change_availability	<0,01
alteration_request	<0,01
view_resolutions	<0,01
view_user_real_names	<0,01
phone_numbers	<0,01
create_alteration_request	<0,01
view_ghosting_reasons	<0,01
view_ghostings	<0,01
set_default_payment_instrument	<0,01
homepage	<0,01
respond_to_alteration_request	<0,01
deactivate_user_account	<0,01

delete_payment_instrument	<0,01
delete_listing_description	<0,01
booking	<0,01
host_respond	<0,01
special_offer_field	<0,01
tos_2014	<0,01
host_refund_guest	<0,01
host_respond_page	<0,01
host_standard_suspension	<0,01

Tabla 58: Proporción de las categorías del atributo `action_detail` de la tabla `sessions`.

El atributo `action_detail` consta de 155 grupos, donde sólo 16 tienen una frecuencia de aparición superior al 1 %. Por ello, para este atributo ocurre lo mismo que para `action`, está compuesto de categorías poco significativas. La acción que es más recurrente para los usuarios es ”view_search_results”, seguido de ”p3” y de los campos vacíos. En cuanto a los menos frecuentes en la Tabla 58 están compartir el contenido, moverse en el mapa de los apartamentos y visitar la página de inicio.

device_type	
Categorías	**Proporción de cada categoría (%)**
Mac Desktop	33,93
Windows Desktop	25,06
iPhone	19,84
Android Phone	7,91
iPad Tablet	6,45
Android App Unknown Phone/Tablet	2,58
-unknown-	1,99
Tablet	1,32
Linux Desktop	0,26
Chromebook	0,21
iPodtouch	0,08
Windows Phone	0,02
Blackberry	0,01
Opera Phone	<0,01

Tabla 59: Proporción de las categorías del atributo `device_type` de la tabla `sessions`.

El atributo `device_type`, Tabla 59, es similar a la variable `first_device_type` de la tabla `train`, salvo que `device_type` aporta información sobre con qué dispositivo ha llevado a cabo cada usuario cada una de las acciones mientras que `first_device_type` sólo guarda el dispositivo con el que el usuario interactuó con Airbnb por primera vez. Los dispositivos más recurrente son ”Mac Desktop”, ”Windows Desktop” y ”Iphone”, coincidiendo con los tres más

recurrentes de `first_device_type`. Las menos frecuentes son ”Backberry” y ”Opera Phone”, siendo distintas a los de `first_device_type`.

La tabla `sessions` consta de atributos con muchas categorías. Muchas de estas categorías son poco frecuentes, por lo que se plantea aplicar el criterio de agrupación comentado para la tabla `train`. Reduciendo las categorías en función de la frecuencia (agrupando las variables con menos del 1 % de proporción), se lograrían para `action`, `action_type`, `action_detail` y `device_type` 19, 7, 17 y 9 categorías, respectivamente. Como el momento donde realmente es necesaria la reducción de variables es tras preparar las tablas para aplicar los modelos de *machine learning*, Subsección 4.7.2 de la Sección 4, se llevará a cabo la agrupación después del cruce de tablas, en la Subsección 4.7.2.

9.3. Código de la memoria

Debido al doble propósito del Trabajo de Fin de Máster, se ha trabajado tanto en un entorno de *Business Intelligence* como de *Big Data*. Para ello, Teradata (SQL) y R han sido empleados en el primer entorno y SparkSQL junto a SparkML y SparkMLlib en el segundo. Seguidamente, se aporta el código empleado a lo largo de la memoria en los dos ambientes para llegar a los resultados de la Sección 6.

9.3.1. Código del entorno *Business Intelligence*

Desde que llegan los datos al *data warehouse* hasta preparar la tabla a la que se le aplican medidas estadísticas para descubrir la información y el conocimiento que residen en los datos, se ha trabajado con un lenguaje relacional, SQL. En las siguientes líneas se introduce primero el propósito o información que se quiere conocer y después el código empleado para ello.

En la Sección 4, al principio, es necesario modificar el tipo de dato de algunas variables y dar un formato homogéneo a otras. Para ello, el código empleado es el siguiente:

```
select country_destination, cast(round(lat_destination,2) as float) as
lat_destination, cast(round(lng_destination,2) as float) as lng_destination,
cast(round(distance_km,2) as float) as distance_km, cast(destination_km2 as
float) as destination_km2, substr(destination_language,1,2) as
```

```
destination_language, language_levenshtein_distance from countries;

select age_bucket, country_destination, upper(gender) as gender,
cast(population_in_thousands as float) as population_in_thousands, cast(year as
integer) as year from age_gender;

select id, date_account_created, concat(substr(timestamp_first_active,1,4), '-',
substr(timestamp_first_active,5,2), '-', substr(timestamp_first_active,7,2)) as
timestamp_first_active, date_first_booking, gender, case when age<15 then 'null'
when age>=15 or age<20 then '15-19' when age>=20 or age<25 then '20-24' when
age>=25 or age<30 then '25-29' when age>=30 or age<35 then '30-34' when age>=35 or
age<40 then '35-39' when age>=40 or age<45 then '40-44' when age>=45 or age<50
then '45-49' when age>=50 or age<55 then '50-54' when age>=55 or age<60 then '55-59'
when age>=60 or age<65 then '60-64' when age>=65 or age<70 then '65-69' when
age>=70 or age<75 then '70-74' when age>=75 or age<80 then '75-79' when age>=80
or age<85 then '80-84' when age>=85 or age<90 then '85-89' when age>=90 or
age<95 then '90-94' when age>=95 or age<100 then '95-99' when age>=100 or age<=120
then '100+' when age>120 then 'null' else 'null' end as age, signup_method,
signup_flow, language, affiliate_channel, affiliate_provider,
first_affiliate_tracked, signup_app, first_device_type, first_browser,
country_destination from train;

select id, date_account_created, concat(substr(timestamp_first_active,1,4), '-',
substr(timestamp_first_active,5,2), '-', substr(timestamp_first_active,7,2)) as
timestamp_first_active, date_first_booking, gender, case when age<15 then 'null'
when age>=15 or age<20 then '15-19' when age>=20 or age<25 then '20-24' when
age>=25 or age<30 then '25-29' when age>=30 or age<35 then '30-34' when age>=35 or
age<40 then '35-39' when age>=40 or age<45 then '40-44' when age>=45 or age<50
then '45-49' when age>=50 or age<55 then '50-54' when age>=55 or age<60 then '55-59'
when age>=60 or age<65 then '60-64' when age>=65 or age<70 then '65-69' when
age>=70 or age<75 then '70-74' when age>=75 or age<80 then '75-79' when age>=80
or age<85 then '80-84' when age>=85 or age<90 then '85-89' when age>=90 or
age<95 then '90-94' when age>=95 or age<100 then '95-99' when age>=100 or age<=120
then '100+' when age>120 then 'null' else 'null' end as age, signup_method,
signup_flow, language, affiliate_channel, affiliate_provider,
```

```
first_affiliate_tracked, signup_app, first_device_type, first_browser from test;

select user_id, action, action_type, action_detail, device_type, cast(secs_elapsed
as float) as secs_elapsed from sessions;
```

Después, se procede al estudio de los datos nulos, completando dichos valores con ”0” para las variables cuantitativas y ”null” para las cualitativas. Además, se descartan los registros de `date_first_booking` que están vacíos, reduciendo el tamaño de la tabla `train` y guardando el resultado en `train_reducido`. Además, también se eliminan por motivos de originalidad las variables `signup_flow` y `signup_method`, guardando esta cambio en `train_reducido` de nuevo. Cabe recordar que justo antes se ha tomado la decisión de prescindir de la tabla `test` temporalmente, por lo que hasta más adelante no volverá a aparecer.

```
select id, date_account_created, timestamp_first_active, date_first_booking, gender,
ifnull(age, 'null') as age, language, affiliate_channel, affiliate_provider,
ifnull(first_affiliate_tracked, 'null') as first_affiliate_tracked,
signup_app, first_device_type, first_browser, country_destination from
train where date_first_booking is not null;

select ifnull(user_id, 'null') as user_id, ifnull(action, 'null') as action,
ifnull(action_type, 'null') as action_type, ifnull(action_detail, 'null') as
action_detail, device_type, ifnull(secs_elapsed, cast('0.0'
as float)) as secs_elapsed from sessions;
```

Las tablas `country_destination` y `age_gender` son descartadas debido a que no aportan información, deshaciendo la agrupación de la tabla `train` en la variable `age` y volviendo a cargar la tabla. Tras ello, en vez del case when de las primeras líneas del código se escribe *ifnull(cast(age as integer), 0)*. A continuación, se añaden nuevas variables a la tabla `train` con el siguiente código, resultado que se guarda en `train_reducida2`:

```
select id, datediff(date_account_created, cast('2009-01-01' as date)) as
date_account_created_2009, datediff(timestamp_first_active, cast('2009-01-01' as
date)) as timestamp_first_active_2009, datediff(date_first_booking, cast('2009-01-01'
as date)) as date_first_booking_2009, datediff(date_account_created,
```

```
timestamp_first_active) as dif_first_active_create, datediff(date_account_created,
date_first_booking) as dif_create_first_booking, datediff(timestamp_first_active,
date_first_booking) as dif_first_active_booking, extract(month from
date_account_created) as month_account_created, extract(month from
timestamp_first_active) as month_first_active, extract(month from
date_first_booking) as month_first_booking, datediff(concat(substr(
date_account_created,1,8), '01'), cast('2009-01-01' as date)) as
year_month_account_created_2009, datediff(concat(substr(timestamp_first_active,1,8),
'01'), cast('2009-01-01' as date)) as year_month_first_active_2009, datediff(concat
(substr(date_first_booking,1,8), '01'), cast('2009-01-01' as date)) as
year_month_first_booking_2009, gender, age, language, affiliate_channel,
affiliate_provider, first_affiliate_tracked, signup_app, first_device_type,
first_browser, country_destination from train_reducida;
```

En cuanto a `sessions`, el siguiente código pone de manifiesto que todos los registros de actividades de usuarios corresponden a los mismos usuarios de los que se tiene información en `train` y `test`.

```
select count(*) from sessions;
select count(*) from train;
select count(*) from test;
select count(distinct(id)) from sessions join train on sessions.user_id=train.id;
select count(distinct(id)) from sessions join test on sessions.user_id=test.id;
```

Una vez se han estudiado todas las tablas, se procede a los cruces entre `train_reducido2` y las dos opciones para `sessions`: la completa (`sessions1`) y la resumida (`sessions2`). Seguidamente, en ese mismo orden, se muestra el código donde el primer cruce genera la tabla `pre_cruce1` y el segundo `pre_cruce2`.

```
select date_account_created_2009, timestamp_first_active_2009,
date_first_booking_2009, dif_first_active_create, dif_create_first_booking,
dif_first_active_booking, month_account_created, month_first_active,
month_first_booking, year_month_account_created_2009, year_month_first_active_2009,
year_month_first_booking_2009, gender, age, language, affiliate_channel,
affiliate_provider, first_affiliate_tracked, signup_app, first_device_type,
```

```
first_browser, action, action_type, action_detail, device_type,
secs_elapsed, country_destination from train_reducida2 left join sessions on
train_reducida2.id=sessions.user_id;

select date_account_created_2009, timestamp_first_active_2009,
date_first_booking_2009, dif_first_active_create, dif_create_first_booking,
dif_first_active_booking, month_account_created, month_first_active,
month_first_booking, year_month_account_created_2009, year_month_first_active_2009,
year_month_first_booking_2009, gender, age, language, affiliate_channel,
affiliate_provider, first_affiliate_tracked, signup_app, first_device_type,
first_browser, sum(secs_elapsed) as sum_secs_elapsed, count(action)
as numb_action, sum(secs_elapsed)/count(action) as secs_per_action,
country_destination from train_reducida2 join sessions on train_reducida2.id=
sessions.user_id group by id, date_account_created_2009, timestamp_first_active_2009,
date_first_booking_2009, dif_first_active_create, dif_create_first_booking,
dif_first_active_booking, month_account_created, month_first_active,
month_first_booking, year_month_account_created_2009, year_month_first_active_2009,
year_month_first_booking_2009, gender, age, signup_method, signup_flow, language,
affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app,
first_device_type, first_browser, country_destination;
```

El código SQL también se emplea en la agrupación de las variables menos frecuentes, de la siguiente manera:

```
select date_first_booking_2009, month_account_created, dif_first_active_create,
gender, age, case when language!='en' then 'others' else language end as language,
case when affiliate_channel in ('remarketing', 'content') then 'others' else
affiliate_channel end as affiliate_channel, case when affiliate_provider not in
('direct', 'google', 'other', 'bing') then 'others' else affiliate_provider end as
affiliate_provider, case when first_affiliate_tracked not in ('untracked', 'omg',
'linked', 'tracked-other') then 'others' else first_affiliate_tracked end as
first_affiliate_tracked, signup_app, case when first_device_type in ('Android
Tablet', 'Desktop (Other)', 'SmartPhone (Other)') then 'others' else
first_device_type end as first_device_type, case when first_browser not in
('Chrome', 'Safari', 'Firefox', '-unknown-', 'Mobile Safari', 'IE') then 'others'
```

```
else first_browser end as first_browser, case when action not in ('show',
'personalize', 'index', 'search_results', 'ajax_refresh_subtotal',
'similar_listings', 'null', 'search', 'update', 'lookup', 'social_connections',
'create', 'dashboard', 'header_userpic', 'reviews', 'edit', 'track_page_view',
'requested', 'active', 'qt2') then 'others' else action end as action, case
when action_type in ('booking_request', 'partner_callback', 'booking_response')
then 'others' else action_type end as action_type, case when action_detail not
in ('view_search_results', 'null', '-unknown-', 'p3', 'wishlist_content_update',
'change_trip_characteristics', 'similar_listings', 'user_profile', 'message_thread',
'user_social_connections', 'update_listing', 'dashboard', 'header_userpic',
'message_post', 'edit_profile', 'contact_host', 'user_wishlists', 'listing_reviews')
then 'others' else action_detail end as action_detail, secs_elapsed, case when
device_type not in ('Mac Desktop', 'Windows Desktop', 'iPhone', 'iPad Tablet',
'Android App Unknown Phone/Tablet', 'Android Phone', 'null', '-unknown-') then
'others' else device_type end as device_type, country_destination from pre_cruce1
group by date_first_booking_2009,  month_account_created, gender, age, language,
affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app,
first_device_type, first_browser, action, action_type, action_detail, secs_elapsed,
device_type, country_destination;

select dif_create_first_booking, month_first_active, gender, age, case when
language!='en' then 'others' else language end as language, case when
affiliate_channel in ('remarketing', 'content') then 'others' else affiliate_channel
end as affiliate_channel, case when affiliate_provider not in ('direct', 'google',
'other', 'facebook', 'bing') then 'others' else affiliate_provider end as
affiliate_provider, case when first_affiliate_tracked not in ('untracked', 'omg',
'linked', 'tracked-other') then 'others' else first_affiliate_tracked end as
first_affiliate_tracked, signup_app, case when first_device_type in ('Android
Tablet', 'Desktop (Other)', 'SmartPhone (Other)') then 'others' else
first_device_type end as first_device_type, case when first_browser not in
('Chrome', 'Safari', 'Firefox', '-unknown-', 'Mobile Safari', 'IE') then 'others'
else first_browser end as first_browser, sum_secs_elapsed, secs_per_action,
country_destination from pre_cruce2 group by dif_create_first_booking,
month_first_active, gender, age, language, affiliate_channel, affiliate_provider,
first_affiliate_tracked, signup_app, first_device_type, first_browser,
```

```
sum_secs_elapsed, secs_per_action, country_destination;
```

Los resultados anteriores se guardan en `p_cruce1` y `p_cruce2`, respectivamente. Finalmente, tras el análisis de la asocicación de las variables, se obtienen las tablas con las variables resultantes, siendo estas tablas a las que se les aplicarán los algoritmos de *machine learning*. Estas tablas reciben el nombre de `cruce1` y `cruce2`, respectivamente, donde su cógido aparece a continuación en ese mismo orden.

```
select date_first_booking_2009, month_account_created, gender, age, language,
affiliate_provider, first_affiliate_tracked, first_browser, action_detail,
secs_elapsed, device_type, country_destination from p_cruce1;
```

```
select dif_first_active_booking, year_month_first_booking_2009, gender, age,
language, affiliate_channel, first_affiliate_tracked, first_device_type,
first_browser, sum_secs_elapsed, secs_per_action, country_destination from
p_cruce2;
```

Llegados a este punto, se finaliza con la parte de procesamiento de datos que se ha llevado a cabo en Teradata. A partir de ahora, se empiezan a aplicar medidas estadísticas y técnicas de aprendizaje automático en ambas bases de datos, usando R. Una vez se han cargado los datos, el coeficiente de correlación de Pearson se calcula con el siguiente código para `pre_cruce2` (de igual manera se hace para `pre_cruce1`):

```
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\pre_cruce2.
csv", header=TRUE)
str(data)
data$gender <- NULL
data$language <- NULL
data$affiliate_channel <- NULL
data$affiliate_provider <- NULL
data$first_affiliate_tracked <- NULL
data$signup_app <- NULL
data$first_device_type <- NULL
data$first_browser <- NULL
data$country_destination <- NULL
```

```
str(data)
summary(data)
cor(data)
round(cor(data), 2)
```

Para el estudio de la asociación entre variables cualitativas, la V de Cramer se calcula de la siguiente manera sobre las variables `gender` y `language` de la tabla `p_cruce2` y, de forma análoga, sobre la tabla `p_cruce1` y el resto de las variables:

```
install.packages('vcd', dependencies=TRUE)
library(vcd)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\
p_cruce2.csv", header=TRUE)
str(data)
tabla = ftable(data1$gender, data1$language)
assocstats(tabla)
```

En cuanto a la asociación entre una variable cualitativa y otra cuantitativa, la prueba de Kruskal-Wallis se calcula con el siguiente código, habiendo estudiado la normalidad previamiente con el test de normalidad de Kolmogorov-Smirnov y la forma de la distribución para cada categoría:

```
Para la nomalidad:
install.packages('nortest', dependencies=T)
library(nortest)
lillie.test(data$dif_first_active_booking)

Para la asociación:
kruskal.test(data$dif_first_active_booking ~ data$gender)
```

Tras estos procesos de reducción de variables, se empieza con los algoritmos de machine learning y la búsqueda de los argumentos que mejor encajen con los datos. A continuación, se presenta el código empleado para `cruce2` (para la otra tabla el proceso es similar) con los algoritmos de regresión logística multinomial, C5.0, Random Forest, AdeBoosting, Gradient

Boosting, One Versus All y la red neuronal perceptrón multicapa de retropropagación, respectivamente:

```
Hipótesis de la regresión logística multinomial:
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
#Outliers
boxplot(age~country_destination,data=data, main="Detección outlier \texttt{age}",
xlab="Destinos", ylab="Distribución edad")
#linearity
install.packages('nnet', dependencies=TRUE)
library(nnet)
ml_multinom_fit <- multinom(country_destination ~ age, data = data)

C5.0:
install.packages('C50', dependencies=TRUE)
install.packages('pROC', dependencies=TRUE)
library(C50)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
str(data)
#Desordenar los datos y seperarlos entre el conjunto de entrenamiento y prueba
set.seed(1985)
g <- sample(nrow(data), 0.6*nrow(data), replace = FALSE)
train <- data[g,]
test <- data[-g,]
table(train$country_destination)
table(test$country_destination)
#Entrenar el modelo
model1<- C5.0(train[,-11], train[,11])
summary(model1)
#Validar el modelo
pred1 <- predict(model1, test[,])
tab <- table(test[,11], pred1 )
tab
```

```
#Exactitud, matriz de confusión y AUC
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
confusionMatrix(pred1, test[,11])
library(pROC)
pred1 <- predict(model1, test[,], type='prob')
roc_obj <- multiclass.roc(test[,11], pred1)
auc(roc_obj)

Random Forest:
install.packages("randomForest", dependencies=TRUE)
library(randomForest)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
str(data)
#Desordenar los datos y seperarlos entre el conjunto de entrenamiento y prueba
set.seed(1985)
g <- sample(nrow(data), 0.6*nrow(data), replace = FALSE)
train <- data[g,]
test <- data[-g,]
table(train$country_destination)
table(test$country_destination)
#Entrenar el modelo
model1 <- randomForest(country_destination ~ ., data = train, ntree = 100,
mtry = 4, importance = TRUE)
model1
#Validar el modelo
predTrain <- predict(model1, test, type = "class")
tab <- table(test$country_destination, predTrain)
tab
#La importancia de cada variable en el modelo
importance(model1)
varImpPlot(model1)
#Exactitud, matriz de confusión y AUC
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
```

```
accuracy(tab)
confusionMatrix(predTrain, test[,11])
library(pROC)
pred1 <- predict(model1, test[,], type='prob')
roc_obj <- multiclass.roc(test[,12], pred1)
auc(roc_obj)

AdaBoosting:install.packages("adabag", dependencies=TRUE)
library(adabag)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
str(data)
#Desordenar los datos y seperarlos entre el conjunto de entrenamiento y prueba
set.seed(1985)
g <- sample(nrow(data), 0.6*nrow(data), replace = FALSE)
train <- data[g,]
test <- data[-g,]
table(train$country_destination)
table(test$country_destination)
#Entrenar el modelo
model1 <- boosting(country_destination ~ ., data = train, tree_depth = 3,
n_rounds=100, boos=TRUE, mfinal=10)
model1
names(model1)
model1$trees
#Validar el modelo
predTrain <- predict(model1, test)
predTrain
tab <- predTrain$confusion
tab
#Exactitud, matriz de confusión y AUC
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
result <- cbind(ValidSet, predTrain$prob, predTrain$class)
result
```

```
library(pROC)
pred1 <- predict(model1, test[,], type='prob')
roc_obj <- multiclass.roc(test[,14], pred1)
auc(roc_obj)

Gradient Boosting:
install.packages("gbm", dependencies=TRUE)
library(gbm)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
str(data)
#Desordenar los datos y seperarlos entre el conjunto de entrenamiento y prueba
set.seed(1985)
g <- sample(nrow(data), 0.6*nrow(data), replace = FALSE)
train <- data[g,]
test <- data[-g,]
table(train$country_destination)
table(test$country_destination)
#Entrenar el modelo
model1 <- gbm(country_destination ~ ., data = train, distribution = "multinomial",
shrinkage = .01, n.minobsinnode = 10, n.trees = 100)
model1
summary(model1)
#Validar el modelo
predTrain <- predict.gbm(model1, test, n.trees=100, type = "response")
summary(predTrain)
result <- cbind(test, predTrain)
result
#Exactitud, matriz de confusión y AUC
mat <- confusionMatrix(predTrain_names, test[,11])
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(mat)
library(pROC)
pred1 <- predict(model1, test[,], n.trees=100, type='response')
roc_obj <- multiclass.roc(test[,11], pred1)
```

```
auc(roc_obj)

One Versus All:
install.packages("e1071", dependencies=TRUE)
library(e1071)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
str(data)
#Desordenar los datos y seperarlos entre el conjunto de entrenamiento y prueba
set.seed(1985)
g <- sample(nrow(data), 0.6*nrow(data), replace = FALSE)
train <- data[g,]
test <- data[-g,]
table(train$country_destination)
table(test$country_destination)
#Entrenar el modelo
svm1 <- svm(country_destination~., data=train, method="C-classification",
kernal="radial", gamma=0.1, cost=5, degree=3)
summary(svm1)
svm1
svm1$SV
#Validar el modelo
prediction <- predict(svm1, test)
tab <- table(test$country_destination, prediction)
tab
#Exactitud, matriz de confusión y AUC
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
confusionMatrix(prediction, test[,11])
library(pROC)
pred1 <- predict(svm1, test[,], type='probabilities')
roc_obj <- multiclass.roc(test[,11], pred1)
auc(roc_obj)

Red neuronal perceptrón multicapa con retropropagación:
```

```
install.packages("neuralnet", dependencies=TRUE)
install.packages("caret", dependencies=TRUE)
library(nnet)
library(neuralnet)
library(caret)
data <- read.csv("E:\\TFM\\Limpieza de datos\\Machine Learning with R\\cruce2.csv",
header=TRUE)
str(data)
#Transformar todas las variables cualitativas con one-hot vector
dmy <- dummyVars(" ~ .", data = data)
trsf <- data.frame(predict(dmy, newdata = data))
trsf
#Estandarizar los datos con el método min-max
scl <- function(x){ (x - min(x))/(max(x) - min(x)) }
trsf[, 1:36] <- data.frame(lapply(trsf[, 1:36], scl))
head(trsf)
#Desordenar los datos y seperarlos entre el conjunto de entrenamiento y prueba
set.seed(1985)
g <- sample(nrow(trsf), 0.6*nrow(trsf), replace = FALSE)
train <- trsf[g,]
test <- trsf[-g,]
#Entrenar el modelo
n <- names(train)
f <- as.formula(paste("country_destination.AU + country_destination.CA +
country_destination.DE + country_destination.ES + country_destination.FR
+ country_destination.GB +
country_destination.IT + country_destination.NL +
country_destination.other + country_destination.PT +  country_destination.US ~",
paste(n[!n %in% c("country_destination.AU","country_destination.CA",
"country_destination.DE", "country_destination.ES","country_destination.FR",
"country_destination.GB","country_destination.IT", "country_destination.NL",
"country_destination.other", "country_destination.PT","country_destination.US")],
collapse = " + ")))
f
```

```
nn <- neuralnet(f, data = train, hidden = c(30, 20, 11), act.fct = "logistic",
linear.output = FALSE, lifesign = "minimal")
#Validar el modelo
pr.nn <- compute(nn, test[, 1:36])
pr.nn_ <- pr.nn$net.result
head(pr.nn_)
#Accuracy
original_values <- max.col(test[, 37:47])
pr.nn_2 <- max.col(pr.nn_)
mean(pr.nn_2 == original_values)
```

Con todo este código se finaliza el proceso llevado a cabo con la herramienta R. Además, también se finaliza con la parte del TFM correspondiente al entorno de *Business Intelligence*, mostrando los pasos del código más relevantes para la elaboración de esta memoria y la obtención de los resultados.

9.3.2. Código del entorno *Big Data*

En el entorno de *Big Data* que se ha planteado en la Sección 3, Hadoop se emplea para el almacenamiento de los datos mientras que Spark se usa para su procesamiento. Aparte de utilizar Spark con la API de Scala, de todas las funcionalidades o librerías que ofrece, se utilizarán SparkSQL y SparkML/MLlib (SparkML se emplea para los algoritmos de *machine learning* con dataframes y SparkMLlib para RDDs, ambos tipos de datos válidos para esta base de datos). Desde que se extraen las tablas de Hadoop hasta que se emplean las técnicas estadísticas, se usa SparkSQL.

A la hora de cambiar el formato en el que se encuentran los datos, ya que todos los datos se reciben como string, el código que se usa es el siguiente para `countries`, `age_gender`, `train`, `test` y `sessions`, respectivamente:

```
import org.apache.spark.sql.SparkSession
val spark = SparkSession.builder().appName("Spark SQL TFM").config("spark.some.
config.option", "some-value").getOrCreate()
import org.apache.spark.sql
df_countries.printSchema()
```

```
val df2_countries=df_countries.withColumn("lat_destination", $"lat_destination"
.cast(sql.types.FloatType))
val df3_countries=df2_countries.withColumn("lng_destination", $"lng_destination"
.cast(sql.types.FloatType))
val df4_countries=df3_countries.withColumn("distance_km", $"distance_km"
.cast(sql.types.FloatType))
val df5_countries=df4_countries.withColumn("destination_km2", $"destination_km2"
.cast(sql.types.FloatType))
val df6_countries=df5_countries.withColumn("language_levenshtein_distance",
$"language_levenshtein_distance".cast(sql.types.FloatType))
df6_countries.printSchema()

df_age_gender.printSchema()
val df2_age_gender=df_age_gender.withColumn("year", $"year".cast(sql.types
.IntegerType))
val df3_age_gender=df2_age_gender.withColumn("population_in_thousands",
$"population_in_thousands".cast(sql.types.FloatType))
df3_age_gender.printSchema()

df_train.printSchema()
val df2_train=df_train.withColumn("signup_flow", $"signup_flow"
.cast(sql.types.IntegerType))
val df3_train=df2_train.withColumn("age2", $"age").drop("age")ç
.withColumn("age2", $"age2".cast(sql.types.IntegerType))
val df4_train=df3_train.withColumn("date_account_created", $"date_account_created"
.cast(sql.types.DateType))
val df5_train=df4_train.withColumn("date_first_booking", $"date_first_booking"
.cast(sql.types.DateType))
df5_train.printSchema()

df_test.printSchema()
val df2_test=df_test.withColumn("signup_flow", $"signup_flow"
.cast(sql.types.IntegerType))
val df3_test=df2_test.withColumn("age2", $"age").drop("age").withColumn("age2",
$"age2".cast(sql.types.IntegerType))
```

```
val df4_test=df3_test.withColumn("date_account_created", $"date_account_created"
.cast(sql.types.DateType))
val df5_test=df4_test.withColumn("date_first_booking", $"date_first_booking"
.cast(sql.types.DateType))
df5_test.printSchema()

df_sessions.printSchema()
val df2_sessions=df_sessions.withColumn("secs_elapsed", $"secs_elapsed"
.cast(sql.types.FloatType))
df2_sessions.printSchema()
```

Para modificar las variables de algunas de las tablas y redondear las unidades, agrupar variables, poner en mayúscula las categorías de unas varibles y dar el formato DD-MM-YYYY a `timestamp_first_active`, entre otros cambios, se ejecutan los siguientes comandos. Además, también se aprovecha para completar los campos nulos, asignando ”0” para los nulos en las variables cuantitativas y ”null” para las cualitativas. Recordar que la tabla `test` se omitirá por el momento y mencionar que todo cambio en las tablas se está guardando en tablas con el mismo nombre salvo en caso de que se indique lo contrario.

```
df6_countries.createOrReplaceTempView("countries")
df3_age_gender.createOrReplaceTempView("age_gender")
df5_train.createOrReplaceTempView("train")
df2_sessions.createOrReplaceTempView("sessions")
spark.sql("select country_destination, round(lat_destination,2) as lat_destination,
round(lng_destination,2) as lng_destination, round(distance_km,2) as distance_km,
destination_km2, substr(destination_language,1,2) as destination_language,
language_levenshtein_distance from countries").show()

spark.sql("select age_bucket, country_destination, upper(gender) as gender,
population_in_thousands, year from age_gender").show()

spark.sql("select id, date_account_created, concat(substr(timestamp_first_active,
1,4), '-', substr(timestamp_first_active,5,2), '-', substr(timestamp_first_active,
7,2)) as timestamp_first_active, ifnull(date_first_booking, 'null') as
date_first_booking, gender, case when age2<15 then 'null' when age2>=15 or age2<20
```

```
then '15-19' when age2>=20 or age2<25 then '20-24' when age2>=25 or age2<30 then
'25-29' when age2>=30 or age2<35 then '30-34' when age2>=35 or age2<40 then
'35-39' when age2>=40 or age2<45 then '40-44' when age2>=45 or age2<50 then '45-49'
when age2>=50 or age2<55 then '50-54' when age2>=55 or age2<60 then '55-59' when
age2>=60 or age2<65 then '60-64' when age2>=65 or age2<70 then '65-69' when
age2>=70 or age2<75 then '70-74' when age2>=75 or age2<80 then '75-79' when
age2>=80 or age2<85 then '80-84' when age2>=85 or age2<90 then '85-89' when
age2>=90 or age2<95 then '90-94' when age2>=95 or age2<100 then '95-99' when
age2>=100 or age2<=120 then '100+' when age2>120 then 'null' else 'null' end as
age, signup_flow signup_method, language, affiliate_channel, affiliate_provider,
ifnull(first_affiliate_tracked, 'null') as first_affiliate_tracked, signup_app,
first_device_type, first_browser, country_destination from train").show()

spark.sql("select ifnull(user_id, 'null') as user_id, ifnull(action, 'null') as
action, ifnull(action_type, 'null') as action_type, ifnull(action_detail, 'null')
as action_detail, device_type, ifnull(secs_elapsed, cast('0.0' as float)) as
secs_elapsed from sessions").show()
```

Las variables `signup_flow` y `signup_method` se descartan de la tabla `train` por motivos de originalidad. Por otro lado, `date_first_booking` presenta más valores nulos que datos. Debido a ello, se reduce la tabla `train` y se mantienen sólo los registros para los que `date_first_booking` tiene valor, guardando ambos cambios en `train_reducido`.

```
spark.sql("select id, date_account_created, timestamp_first_active,
date_first_booking, gender, age, language, affiliate_channel, affiliate_provider,
first_affiliate_tracked, signup_app, first_device_type, first_browser,
country_destination from train where date_first_booking is not null").show()
```

Posteriormente, como las tablas `countries` y `age_gender` no aportan información para el objetivo, son descartadas. De este modo, se vuelve a cargar la tabla `train`, para no agrupar la variable `age`, darle formato numérico y completar los valores nulos con un cero. A continuación, se añaden nuevas variables a la tabla `train`, guardando los resultados en `train_reducida2`:

```
spark.sql("select id, datediff(date_account_created, cast('2009-01-01' as date))
as date_account_created_2009, datediff(timestamp_first_active, cast('2009-01-01'
```

```
as date)) as timestamp_first_active_2009, datediff(date_first_booking, cast('2009-
01-01' as date)) as date_first_booking_2009, datediff(date_account_created,
timestamp_first_active) as dif_first_active_create, datediff(date_account_created,
date_first_booking) as dif_create_first_booking, datediff(timestamp_first_active,
date_first_booking) as dif_first_active_booking, extract(month from
date_account_created) as month_account_created, extract(month from
timestamp_first_active) as month_first_active, extract(month from
date_first_booking) as month_first_booking, datediff(concat(substr(
date_account_created,1,8), '01'), cast('2009-01-01' as date)) as
year_month_account_created_2009, datediff(concat(substr(timestamp_first_active,1,8),
'01'), cast('2009-01-01' as date)) as year_month_first_active_2009, datediff(concat
(substr(date_first_booking,1,8), '01'), cast('2009-01-01' as date)) as
year_month_first_booking_2009, gender, age, language, affiliate_channel,
affiliate_provider, first_affiliate_tracked, signup_app, first_device_type,
first_browser, country_destination from train_reducida").show()
```

En cuanto a la tabla `sessions`, las siguientes líneas de cógido demuestran como todos los usuarios de esta tabla salvo uno se encuentran también en `test` o en `train`.

```
spark.sql("select count(*) from sessions").show()
spark.sql("select count(*) from train").show()
spark.sql("select count(*) from test").show()
spark.sql("select count(distinct(id)) from sessions join train on sessions.user_id
=train.id").show()
spark.sql("select count(distinct(id)) from sessions join test on sessions.user_id
=test.id").show()
```

Tras haber estudiado las tablas, se procede al cruce entre `train_reducida2` y las dos opciones para `sessions`: la de la tabla completa (`sessions1`) y la de las variables resumen (sessions2). Seguidamente, cada uno de los dos bloques corresponde a las dos opciones de cruce, correspondiendo el primero al cruce con la tabla entera y el segundo al cruce con la tabla de las variables de resumen. Las tablas resultantes recibirán el nombre de `pre_cruce1` y `pre_cruce2`, respectivamente.

```
spark.sql("select date_account_created_2009, timestamp_first_active_2009,
```

```
date_first_booking_2009, dif_first_active_create, dif_create_first_booking,
dif_first_active_booking, month_account_created, month_first_active,
month_first_booking, year_month_account_created_2009, year_month_first_active_2009,
year_month_first_booking_2009, gender, age, language, affiliate_channel,
affiliate_provider, first_affiliate_tracked, signup_app, first_device_type,
first_browser, action, action_type, action_detail, device_type,
secs_elapsed, country_destination from train_reducida2 left join sessions on
train_reducida2.id=sessions.user_id").show()

spaqk.sql("select date_account_created_2009, timestamp_first_active_2009,
date_first_booking_2009, dif_first_active
_create, dif_create_first_booking,
dif_first_active_booking, month_account_created, month_first_active,
month_first_booking, year_month_account_created_2009, year_month_first_active_2009,
year_month_first_booking_2009, gender, age, language, affiliate_channel,
affiliate_provider, first_affiliate_tracked, signup_app, first_device_type,
first_browser, sum(secs_elapsed) as sum_secs_elapsed, count(action)
as numb_action, sum(secs_elapsed)/count(action) as secs_per_action,
country_destination from train_reducida2 join sessions on train_reducida2.id=
sessions.user_id group by id, date_account_created_2009, timestamp_first_active_2009,
date_first_booking_2009, dif_first_active_create, dif_create_first_booking,
dif_first_active_booking, month_account_created, month_first_active,
month_first_booking, year_month_account_created_2009, year_month_first_active_2009,
year_month_first_booking_2009, gender, age, signup_method, signup_flow, language,
affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app,
first_device_type, first_browser, country_destination".show()
```

El código de SparkSQL también es útil para la agrupación de las variables menos frecuentes, como se muestra a continuación, guardando los resultados en `p_cruce1` y `p_cruce2`, en ese orden:

```
spark.sql("select date_first_booking_2009, month_account_created, dif_first_active
_create, gender, age, case when language!='en' then 'others' else language end as
language, case when affiliate_channel in ('remarketing', 'content') then 'others'
else affiliate_channel end as affiliate_channel, case when affiliate_provider not in
```

```
('direct', 'google', 'other', 'bing') then 'others' else affiliate_provider end as
affiliate_provider, case when first_affiliate_tracked not in ('untracked', 'omg',
'linked', 'tracked-other') then 'others' else first_affiliate_tracked end as
first_affiliate_tracked, signup_app, case when first_device_type in ('Android
Tablet', 'Desktop (Other)', 'SmartPhone (Other)') then 'others' else
first_device_type end as first_device_type, case when first_browser not in
('Chrome', 'Safari', 'Firefox', '-unknown-', 'Mobile Safari', 'IE') then 'others'
else first_browser end as first_browser, case when action not in ('show',
'personalize', 'index', 'search_results', 'ajax_refresh_subtotal',
'similar_listings', 'null', 'search', 'update', 'lookup', 'social_connections',
'create', 'dashboard', 'header_userpic', 'reviews', 'edit', 'track_page_view',
'requested', 'active', 'qt2') then 'others' else action end as action, case
when action_type in ('booking_request', 'partner_callback', 'booking_response')
then 'others' else action_type end as action_type, case when action_detail not
in ('view_search_results', 'null', '-unknown-', 'p3', 'wishlist_content_update',
'change_trip_characteristics', 'similar_listings', 'user_profile', 'message_thread',
'user_social_connections', 'update_listing', 'dashboard', 'header_userpic',
'message_post', 'edit_profile', 'contact_host', 'user_wishlists', 'listing_reviews')
then 'others' else action_detail end as action_detail, secs_elapsed, case when
device_type not in ('Mac Desktop', 'Windows Desktop', 'iPhone', 'iPad Tablet',
'Android App Unknown Phone/Tablet', 'Android Phone', 'null', '-unknown-') then
'others' else device_type end as device_type, country_destination from pre_cruce1
group by date_first_booking_2009,  month_account_created, gender, age, language,
affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app,
first_device_type, first_browser, action, action_type, action_detail, secs_elapsed,
device_type, country_destination").show()

spark.sql("select dif_create_first_booking, month_first_active, gender, age, case
when language!='en' then 'others' else language end as language, case when
affiliate_channel in ('remarketing', 'content') then 'others' else affiliate_channel
end as affiliate_channel, case when affiliate_provider not in ('direct', 'google',
'other', 'facebook', 'bing') then 'others' else affiliate_provider end as
affiliate_provider, case when first_affiliate_tracked not in ('untracked', 'omg',
'linked', 'tracked-other') then 'others' else first_affiliate_tracked end as
first_affiliate_tracked, signup_app, case when first_device_type in ('Android
```

```
Tablet', 'Desktop (Other)', 'SmartPhone (Other)') then 'others' else
first_device_type end as first_device_type, case when first_browser not in
('Chrome', 'Safari', 'Firefox', '-unknown-', 'Mobile Safari', 'IE') then 'others'
else first_browser end as first_browser, sum_secs_elapsed, secs_per_action,
country_destination from pre_cruce2 group by dif_create_first_booking,
month_first_active, gender, age, language, affiliate_channel, affiliate_provider,
first_affiliate_tracked, signup_app, first_device_type, first_browser,
sum_secs_elapsed, secs_per_action, country_destination").show()
```

Finalmente, el uso de SparkSQL acaba con la selección de las variables no descartadas por los procesos de asociación. El resultado del siguiente código da lugar a las tablas `cruce1` y `cruce2`, a las que se les aplicarán los algoritmos de *machine learning*.

```
spark.sql("select date_first_booking_2009, month_account_created, gender, age,
language, affiliate_provider, first_affiliate_tracked, first_browser,
action_detail, secs_elapsed, device_type, country_destination from p_cruce1").show()
```

```
spark.sql("select dif_first_active_booking, year_month_first_booking_2009, gender,
age, language, affiliate_channel, first_affiliate_tracked, first_device_type,
first_browser, sum_secs_elapsed, secs_per_action, country_destination from
p_cruce2").show()
```

A partir de este punto se comienza con el procesamiento de los datos empleando SparkML/MLlib. De este modo, se aplicarán las medidas estadísticas de asociación y las técnicas de inteligencia artificial. Spark con la API de Scala tiene métodos para medir el coeficiente de correlación de Spearman pero no para el coeficiente de correlación de Pearson, la V de Cramer, la prueba de Kruskal-Wallis y el test de normalidad de Kolmogorov-Smirnov, siendo los métodos elegidos para el procedimiento llevado a cabo en paralelo en el entorno de *Business Intelligence*. Por ello, en la ejecución de la memoria se ha recurrido a R debido a la carencia de Spark en cuanto a los métodos estadísticos.

Para poder aplicar los métodos de *machine learning* implementados en Spark, el formato por defecto es "libsvm", habiendo tomado la decisión de transformar los datos del formato "csv" al mencionado. Para ello, se ha implementado el siguiente código con Python, usando Jupyter Notebook para mayor velocidad su la ejecución, tomando como ejemplo del código el correspondiente a la tabla `cruce1`.

```
cruce_30_out=open("cruce1_out.txt","w")
testsite_array = []
#Carga de los datos
with open("cruce1.csv", "r") as cruce_30:
    for line in cruce_30:
        linea=line.split(',')
        testsite_array.append(linea)
#Eliminación de la tabulación final
for i in range(len(testsite_array)):
    j=len(testsite_array[i])
    testsite_array[i][j-1]=testsite_array[i][j-1].replace('\n','')
#Eliminación de la cabecera
cruce_array=testsite_array[1:]
#Creación de listas para las variables cualitativas
bi=[]
lau=[]
bost=[]
sei=[]
zazpi=[]
zortzi=[]
hamar=[]
hamaika=[]
#Completando cada lista con las variables únicas de cada variable cualitativa y
ordenándolas
for i in range(0,len(cruce_array)):
    if cruce_array[i][2] not in bi:
        bi.append(cruce_array[i][2])
    if cruce_array[i][4] not in lau:
        lau.append(cruce_array[i][4])
    if cruce_array[i][5] not in bost:
        bost.append(cruce_array[i][5])
    if cruce_array[i][6] not in sei:
        sei.append(cruce_array[i][6])
    if cruce_array[i][7] not in zazpi:
```

```
        zazpi.append(cruce_array[i][7])
    if cruce_array[i][8] not in zortzi:
        zortzi.append(cruce_array[i][8])
    if cruce_array[i][10] not in hamar:
        hamar.append(cruce_array[i][10])
    if cruce_array[i][11] not in hamaika:
        hamaika.append(cruce_array[i][11])


bi.sort()
lau.sort()
bost.sort()
sei.sort()
zazpi.sort()
zortzi.sort()
hamar.sort()
hamaika.sort()
#Codicicación de las categorías de cada variable en la tabla por la posición que
ocupan en las listan anteriores ordenadas
for i in range(0,len(cruce_array)):
    for j in range(len(cruce_array[i])):
        if j==2:
            cruce_array[i][2]=str(bi.index(cruce_array[i][2]))
        if j==4:
            cruce_array[i][4]=str(lau.index(cruce_array[i][4]))
        if j==5:
            cruce_array[i][5]=str(bost.index(cruce_array[i][5]))
        if j==6:
            cruce_array[i][6]=str(sei.index(cruce_array[i][6]))
        if j==7:
            cruce_array[i][7]=str(zazpi.index(cruce_array[i][7]))
        if j==8:
            cruce_array[i][8]=str(zortzi.index(cruce_array[i][8]))
        if j==10:
            cruce_array[i][10]=str(hamar.index(cruce_array[i][10]))
```

```
        if j==11:
            cruce_array[i][11]=str(hamaika.index(cruce_array[i][11]))
#Codificación del nombre de la variable y reordenación de las mismas
cruce_restructurado=[[0 for col in range(len(cruce_array[0])-1)]
                     for row in range(len(cruce_array))]
for i in range(len(cruce_array)):
    for j in range (len(cruce_array[i])):
        if j==len(cruce_array[i])-1:
            cruce_restructurado[i].insert(0,(cruce_array[i][j]).replace("\n", ""))
        else:
            elem=[str(j+1),cruce_array[i][j]]
            cruce_restructurado[i][j]=":".join(elem)
#Guardat todo
for element in cruce_restructurado:
    cruce_30_out.write((", ".join(element) + "\n").replace(",", ""))
#Cerrar los documentos
cruce_30.close()
cruce_30_out.close()
```

Una vez se tienen modificadas las tablas, se procede a la aplicación de los algoritmos disponibles en Spark. Los elegidos han sido Random Forest, Gradient Boosting, One Versus All y la red neuronal perceptrón multicapa (recordar que, como Gradient Boosting sólo está implementado para un atributo de salida binomial, solamente se va a proporcionar el código de Spark). El código empleado para cada uno de ellos se muestra a continuación para la tabla cruce2:

```
Random Forest:
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{RandomForestClassificationModel,
RandomForestClassifier}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}
//Carga de los datos como un dataframe
val data = spark.read.format("libsvm").load("C:/Users/usuario/Desktop/Python/
cruce2_out.txt")
```

```
//Indexar los datos y ajustarlos al modelo
val labelIndexer = new StringIndexer().setInputCol("label")
.setOutputCol("indexedLabel").fit(data)
//Identificación automática de las variables cualitativas, siendo éstas las de
más de 7 categorías
val featureIndexer = new VectorIndexer().setInputCol("features")
.setOutputCol("indexedFeatures").setMaxCategories(7).fit(data)
//División de datos entre entrenamiento y prueba
val Array(trainingData, testData) = data.randomSplit(Array(0.9, 0.1))
//Entrenar el modelo
val rf = new RandomForestClassifier().setLabelCol("indexedLabel")
.setFeaturesCol("indexedFeatures").setNumTrees(60)
// Convertir las variables indexadas en las originales
val labelConverter = new IndexToString().setInputCol("prediction")
.setOutputCol("predictedLabel").setLabels(labelIndexer.labels)
//Preparar el modelo para un Pipeline y entrenarlo
val pipeline = new Pipeline().setStages(Array(labelIndexer, featureIndexer, rf,
labelConverter))
val model = pipeline.fit(trainingData)
// Validar el modelo
val predictions = model.transform(testData)
// Calcular la exactitud
val evaluator = new MulticlassClassificationEvaluator().setLabelCol("indexedLabel")
.setPredictionCol("prediction").setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
println(s"Accurary = ${accuracy}")
//Visualizar el modelo
val rfModel = model.stages(2).asInstanceOf[RandomForestClassificationModel]
println(s"Learned classification forest model:\n ${rfModel.toDebugString}")

Gradient Boosting:
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{GBTClassificationModel, GBTClassifier}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}
```

```
//Carga de los datos como un dataframe
val data = spark.read.format("libsvm").load("C:/Users/usuario/Desktop/Python/
cruce2_out.txt")
//Indexar los datos y ajustarlos al modelo
val labelIndexer = new StringIndexer().setInputCol("label")
.setOutputCol("indexedLabel").fit(data)
//Identificación automática de las variables cualitativas, siendo éstas las de
más de 7 categorías
val featureIndexer = new VectorIndexer().setInputCol("features")
.setOutputCol("indexedFeatures").setMaxCategories(7).fit(data)
//División de datos entre entrenamiento y prueba
val Array(trainingData, testData) = data.randomSplit(Array(0.6, 0.4))
//Entrenar el modelo
val gbt = new GBTClassifier().setLabelCol("indexedLabel")
.setFeaturesCol("indexedFeatures").setMaxIter(10)
//convertir las variables indexadas en las originales
val labelConverter = new IndexToString().setInputCol("prediction")
.setOutputCol("predictedLabel").setLabels(labelIndexer.labels)
//Preparar el modelo para un Pipeline y entrenarlo
val pipeline = new Pipeline().setStages(Array(labelIndexer, featureIndexer, gbt,
labelConverter))
val model = pipeline.fit(trainingData)
//Validar el modelo
val predictions = model.transform(testData)
//Calcular la exactitud
val evaluator = new MulticlassClassificationEvaluator().setLabelCol("indexedLabel")
.setPredictionCol("prediction").setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
println("Accuracy = " + accuracy)
//Visualizar el modelo
val gbtModel = model.stages(2).asInstanceOf[GBTClassificationModel]
println("Learned classification GBT model:\n" + gbtModel.toDebugString)

One Versus All:
import org.apache.spark.ml.classification.{LogisticRegression, OneVsRest}
```

```
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
//Carga de los datos como un dataframe
val inputData = spark.read.format("libsvm").load("C:/Users/usuario/Desktop/Python/
cruce2_out.txt")
//División de los datos entre entrenamiento y prueba
val Array(train, test) = inputData.randomSplit(Array(0.85, 0.15))
//Establecer los parámetros del clasificador
val classifier = new LogisticRegression().setMaxIter(5).setTol(1E-6)
.setFitIntercept(true)
val ovr = new OneVsRest().setClassifier(classifier)
//Entrenar el modelo
val ovrModel = ovr.fit(train)
val predictions = ovrModel.transform(test)
//Validar el modelo
val evaluator = new MulticlassClassificationEvaluator().setMetricName("accuracy")
//Calcular la exactitud
val accuracy = evaluator.evaluate(predictions)
println(s"Accuracy = ${accuracy}")
```

Red neuronal perceptrón multicapa:

```
import org.apache.spark.ml.classification.MultilayerPerceptronClassifier
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
// Carga de los datos como un dataframe
val data = spark.read.format("libsvm").load("C:/Users/usuario/Desktop/Python/
cruce2_out.txt")
//División de los datos entre entrenamiento y prueba
val splits = data.randomSplit(Array(0.6, 0.4), seed = 1234L)
val train = splits(0)
val test = splits(1)
//Especificación de las capas de la red neuronal
val layers = Array[Int](10, 12, 11, 10, 11)
//Establecer los parámetros de entrenamiento
val trainer = new MultilayerPerceptronClassifier().setLayers(layers)
.setBlockSize(128).setSeed(1234L)
.setMaxIter(500)
```

```
//Entrenar el modelo
val model = trainer.fit(train)
//Validar el modelo
val result = model.transform(test)
val predictionAndLabels = result.select("prediction", "label")
//Calcular la exactitud
val evaluator = new MulticlassClassificationEvaluator().setMetricName("accuracy")
println(s"Test set accuracy = ${evaluator.evaluate(predictionAndLabels)}")
```

Con todo el código anterior se da fin por completo al trabajo con la herramienta Spark y también al trabajo correspondiente al entorno *Big Data*. De este modo, queda a disposición de todo aquel que lo desee el código para reproducir el trabajo que se ha llevado a cabo en esta memoria y pueda incluso crear nuevas propuestas para afrontar desde un enfoque distinto el problema de hacer una recomendación de un primer destino a un nuevo usuario.