Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería de Telecomunicación

Máster Universitario en Ingeniería de
Telecomunicación

Trabajo Fin de Máster

Predicción del destino de reserva de los nuevos usuarios de USA a través del conjunto de datos de la competición de Kaggle: Airbnb

Autor: Gonzalo Abelardo Rodríguez Muñoz
Director: Felipe Alonso Atienza
Co-tutor: Carlos Figuera Pozuelo

Curso Académico 2016/2017

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi director del trabajo Fin de Máster, Felipe Alonso Atienza, por haberme introducido en este área del conocimiento, que tanto me gusta, y por haberme apoyado y orientado en la realización de este trabajo. También agradecerle por su disponibilidad y su paciencia ya que pese a que actualmente no es profesor de la Universidad Rey Juan Carlos siempre ha estado disponible para contestar todas mis preguntas y mis dudas.

También quiero agradecer a los demás profesores de la Universidad Rey Juan Carlos que pese que no han contribuido directamente en este trabajo, me han ayudado a formarme y a superar exitosamente esta etapa de mi vida.

Quiero expresar mi gratitud a toda mi familia, por el esfuerzo y apoyo que en todo momento me han brindado a lo largo de mi vida; por su apoyo incondicional y sabios consejos, porque sin ellos esto no hubiera sido posible.

No puedo olvidar a todos los compañeros y amigos que han vivido conmigo la realización de este TFM, agradecerles por haberme brindado todo su apoyo, ánimo, cariño y amistad.

Contenido

Índice de figuras

Índice de tablas

Resumen

En este Trabajo Fin de Máster se desarrolla un modelo para predecir el país destino de reserva de los nuevos usuarios de Airbnb a partir de datos demográficos y de sesión web. Los datos utilizados para el desarrollo de este modelo han sido proporcionados por la competición de Kaggle: *Airbnb New User Bookings* (25 de Noviembre de 2015 - 11 de Febrero de 2016); y provienen de la base de datos de Airbnb. Todos los usuarios del *dataset* son de USA.

Este TFM, por tanto, aborda un problema real en el ámbito de la ciencia de los datos, más conocida como *Data Science* (DS). En primer lugar se muestran los hallazgos encontrados sobre las características del conjunto de datos que proporciona Airbnb. Posteriormente se habla de la estrategia de selección de características que se ha utilizado, que está basada en estas observaciones y en maximizar la bondad de ajuste del modelo.

Dado que es un problema de clasificación supervisado, se han analizado seis tipos de clasificadores (ordenados por orden creciente en complejidad): *Naive Bayes*, *Logistic Regression*, *K-nearest Neighbors*, *Random Forest*, *Extremely Randomized Trees* y *Gradient Boosting*. Dado que se trata de un problema multiclase, se ha seguido una estrategia one vs rest; se han ajustado los parámetros libres de cada clasificador mediante técnicas de validación. Los resultados de las predicciones han sido evaluadas de acuerdo a la métrica nDCG (*Normalized discounted cumulative gain*), que es la sugerida por la empresa hosting. Los resultados del mejor clasificador desarrollado (*Gradient Boosting*), son bastante satisfactorios y me llevaron a situarme en el 10 % de los mejores participantes, consiguiendo una puntuación en la *private leaderboard* de 0,88514.

Capítulo 1
Introducción

En este primer capítulo del Trabajo Fin de Máster (TFM) se hablará de por qué surge el proyecto así como cuáles son los objetivos que se pretenden conseguir.

1.1. Motivación

Kaggle [1] es una plataforma que reúne a la mayor comunidad de personas con experiencia en el análisis de datos del mundo, actualmente de cerca de 525.000 personas, en un entorno que les permite competir para resolver problemas complejos y reales de la ciencia de los datos (DS). Kaggle es una de las contribuciones más importantes al mundo del DS, ya que permite poner en contacto a colectivos de diversos perfiles, lo que enriquece las soluciones. Estas competiciones se basan en que los participantes compiten por realizar el mejor modelo predictivo, el que mejor bondad de ajuste presenta a la variable que se necesita modelar, mediante técnicas de aprendizaje automático con el fin, dependiendo del tipo de competición, de: obtener un premio metálico, promocionarse en el mundo del DS, obtener un puesto de trabajo o aprender.

El funcionamiento de la plataforma es sencillo, un promotor contacta con Kaggle y les proporciona un conjunto de datos de su área de negocio o de su ámbito de investigación. Una parte de estos datos, los cuales están perfectamente etiquetados, son publicados en su web junto con la variable respuesta a modelar para que los participantes desarrollen sus modelos predictivos. También se da a conocer información sobre la competición y la métrica utilizada para evaluar la bondad de ajuste del modelo. Una vez los participantes han obtenido sus predicciones las envían a Kaggle, y éste automáticamente las compara, utilizando la métrica, con el valor real de la variable respuesta, que solo conoce Kaggle, para construir el *ranking*.

En este TFM se ha participado en una de las competiciones de Kaggle. Esta competición es una competición individual y de reclutamiento en la que Airbnb desafía a los participantes a predecir en qué país un nuevo usuario de USA hará su primera reserva. Los posibles lugares de destino son 12: 'US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL', 'DE', 'AU', 'NDF' (destino no encontrado), y 'other'. Anotar que la diferencia entre 'NDF' y 'other' viene dada porque 'other'significa que había una reserva, pero es

un país no incluido en la lista, mientras que 'NDF' significa que no había una reserva. Una vez finalizada la competición, los participantes que han obtenido las mejores soluciones son considerados para optar a un puesto de trabajo en el equipo *Analytics* y DS de Airbnb.

1.2. Objetivos

En esta sección se presentan los objetivos de este trabajo.

1.2.1. Objetivo General

El objetivo general de este TFM es profundizar en la disciplina de la ciencia de los datos, mediante la participación en la competición de Kaggle: *Airbnb New User Bookings*, que consiste en predecir a partir de datos de usuarios de Airbnb en que país un nuevo usuario de USA va a realizar su primera reserva.

1.2.2. Objetivos Secundarios

⇒ Revisar la teoría relacionada con el aprendizaje automático y en especial sobre los métodos de aprendizaje supervisado.

⇒ Construir un modelo predictivo.

⇒ Investigar sobre las librerías de Python que son necesarias para abordar este desafío y aprender a utilizarlas.

⇒ Realizar, para cada clasificador, ajuste de parámetro con la finalidad de maximizar la bondad de ajuste del modelo sin dejar de lado la eficiencia computacional.

⇒ Clasificar lo mejor posible dentro de la competición.

1.3. Organización de la memoria

- El capítulo 2, se da a conocer la metodología que se ha utilizado: una descripción y análisis exploratorio de los datos de los que ha dispuesto, el pre-procesado final del modelo predictivo que se utilizado, el método de validación empleado, una descripción de la métrica de evaluación utilizada y los distintos clasificadores que se ha ido probando con su respectivo ajuste de parámetros.

- El capítulo 3, se muestran y se comentan los resultados que se han obtenido con los distintos métodos, haciendo hincapié en los puntos fuertes y débiles de cada uno de ellos.

- El capítulo 4, contiene las conclusiones a las que se ha llegado tras haber finalizado el trabajo y enuncia las distintas maneras de continuar con este trabajo en futuros proyectos.

Capítulo 2
Metodología

En este capítulo del TFM se describe la metodología utilizada en este TFM. Esta metodología comprende el análisis exploratorio, el pre-procesado del conjunto de datos, la ingeniería de las características, la elección del conjunto de validación, la métrica de evaluación y la selección del modelo.

Se ha utilizado Python 2.7 con las librerías `Numpy` y `Pandas` para el tratamiento de los datos, `Pyplot` y `Seaborn` para la representación gráfica y `Scikit-learn` para utilizar los algoritmos del *machine leaning*.

2.1. Conjunto de datos

2.1.1. Descripción de conjunto de datos

El conjunto de datos con el que cuenta este desafío es proporcionado por Airbnb el cual contiene una lista de usuarios junto con sus características demográficas, sus valores de registro de la sesión web y algunos resúmenes estadísticos. El conjunto de datos contiene 5 ficheros en formato csv: `train_user_2`, `test_user`, `session`, `countries` y `age_gender_bkts`.

1) `train_user_2` y `test_user`: Ambos ficheros fueron modificaron durante la competición [2]. El fichero `train_user_2` es la versión final del fichero `train_user`, que fue modificado debido a que las últimas columnas pretenecían a usuarios que tenían destino sin haber realizado reserva. El fichero `test_user` fue modificado por los organizadores para mejorar la calidad de la competición, para ello incrementaron el número de registros y pusieron todos los datos de la variable *date_first_booking* a *null*.

   El fichero `train_user_2` contiene 213.451 muestra con 16 características:

   - *id*: identificador del usuario.
   - *date_account_create*: fecha de creación de la cuenta.
   - *timestamp_first_active*: fecha de la primera actividad. Decir que puede ser anterior a *date_account_create* y *date_first_booking* ya que un usuario puede buscar antes de estar registrado.

- *date_ first_ booking*: fecha de la primera reserva.
- *gender*: género.
- *age*: edad.
- *signup_ method*: método utilizado en el registro: Facebook, google, ...
- *signud_ flow*: de qué página un usuario se registra.
- *language*: idioma.
- *affiliate_ channel*: tipo de comercialización: direct, seo, ...
- *affiliate_ provider*: dónde se produce la comercialización: google, bing, yahoo, ...
- *first_ affiliate_ tracked*: la primera comercialización con la que el usuario interactúa antes de registrarse.
- *signud_ app*: aplicación utilizada en el registro.
- *first_ device_ type*: Hardware/Software utilizado en el registro.
- *first_ browser*: navegador utilizado en el registro.
- *country_ destination*: país de destino de la reserva.

El archivo `test_user` contiene 62.096 muestras con 15 características. Los valores de *country_ destination* se desconocen ya que es la variable respuesta que se quiere predecir.

Los conjuntos de entrenamiento y test están separados por fechas. En el conjunto de test, se espera predecir el país de destino de todos los nuevos usuarios con fecha de la primera actividad posterior a 01/07/2014.

2) `session`: El fichero `session` es un registro de las sesiones web de los usuarios. En este fichero sólo se encuentra los datos de un 49 % de los usuarios (135.483 usuarios diferentes) ya que sus datos se remontan a 01/01/2014 mientras que el conjunto de datos de usuario se remonta a 01/01/2010. Contiene 10.567.737 muestras con 6 características:

   - *user_ id*: identificador de usuario.
   - *action*: la acción realizada: mostrar, búsqueda de resultados, crear, ...
   - *action_ type*: el tipo de acción: click, view, submit, ...
   - *action_ detail*: detalles sobre la acción: contactar con el host, ver la búsqueda de los resultados, registrarse, ...
   - *device_ type*: el Hardware/Software.
   - *secs_ elapsed*: número de segundos entre acciones que se han registrado.

3) `countries`: Este fichero contiene las estadísticas y la información geográfica de los posibles países destino. Consiste en un fichero de 10 muestras con 7 características, entre los cuales se encuentra: la latitud, la longitud y el idioma del país de destino.

4) `age_gender_bkts`: Este fichero contiene las estadísticas de los usuarios por rango de edad, género y país de destino. Consiste en un fichero de 420 muestras con 5 características.

2.1.2. Análisis exploratorio del conjunto de datos

- *date_account_create* y *timestamp_first_active*: Son características discretas de tipo numérico que corresponden a valores temporales en formato fecha (año-mes-día). Como se muestra en la Figura 2.1 no hay apenas diferencia entre cuando un usuario crea su cuenta en Airbnb y cuando realiza su primera actividad. También se puede observar que el número de nuevos usuarios se incrementa con los años y que este incremento es más pronunciado en los últimos meses de verano, que corresponden con fechas entre Agosto y Octubre, como se muestra en la Figura 2.2.

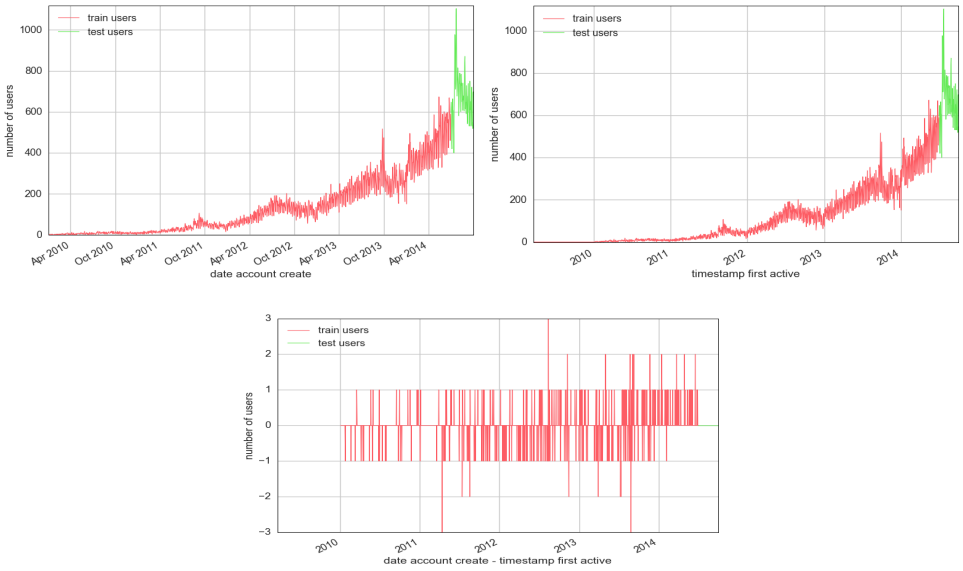


Figura 2.1. Representación de las variables account create y timestamp first active

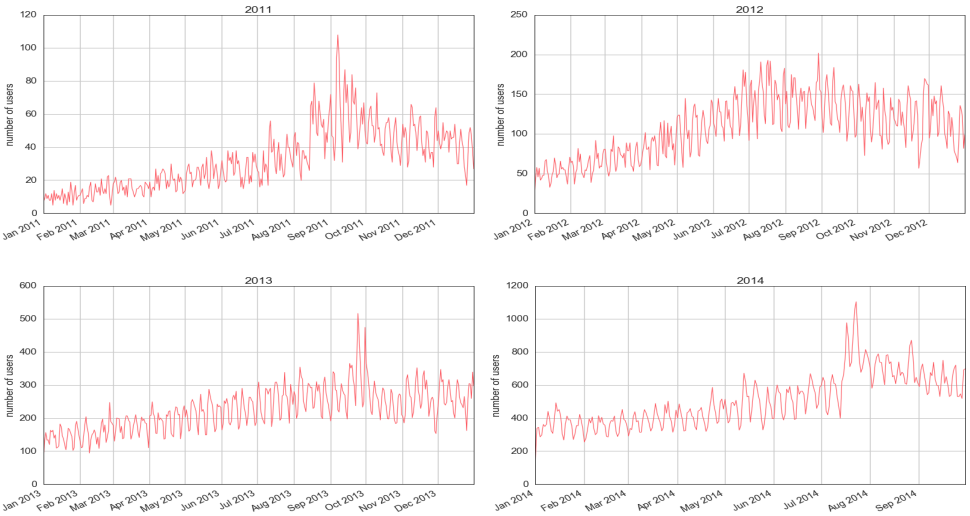


Figura 2.2. Primera actividad de los usuarios (timestamp first active) por año

- *date_first_booking*: Es una características discreta de tipo numérico que corresponde a valores temporales en formato fecha (año-mes-día). Como se muestra en el Figura 2.3 cada año se producen más reservas. La gran bajada producida a mediados del año 2014 viene dada por la gran cantidad de valores pérdidos que contiene esta característica (un 67,74 % del total): 124.543 datos pérdidos en el conjunto de entrenamiento y el total de los datos del conjunto de test. Además, se puede observar en la Figura 2.4 que los usuarios realizan principalmente sus reservas en las fechas correspondientes a días vacacionales; a excepción de navidades, donde ellos prefieren no viajar y permanecer con sus familias.

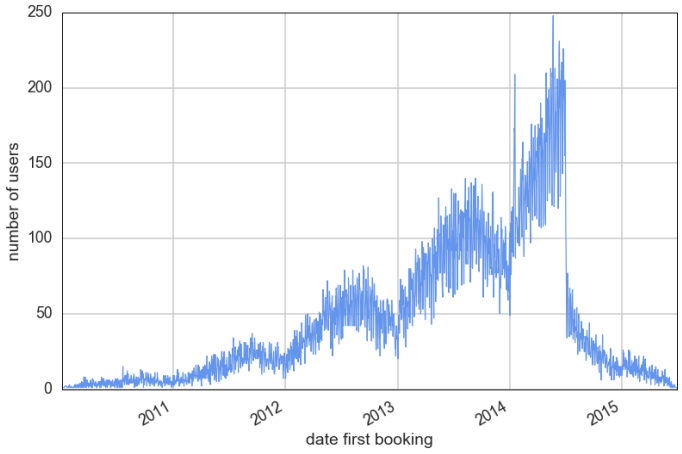


Figura 2.3. Representación de la variable date first booking

Figura 2.4. Primera reserva de los usuarios por año

- *Age*: Es una característica discreta de tipo numérico y su distribución es la que se muestra en la Figura 2.5.

Figura 2.5. Distribución de la edad de los usuarios

En la Figura, se puede observar que la mayoría de los usuarios que utilizan Airbnb son gente joven, entre 24 y 36 años.

La característica *age* tiene una gran influencia en donde los usuarios van a realizar su primera reserva. Si observamos la Figura 2.6, que enfrenta el país de destino de reserva con la edad, se puede observar que los usuarios más jóvenes tienden realizar su primera reserva en países como España y Portugal mientras que los más viejos prefieren realizar su primera reserva en Gran Bretaña, como se muestra en la imagen de la izquierda. En esta misma Figura, en la imagen de la derecha, que separa a los usuarios mayores y menores de 45 años, se puede observar que los usuarios menores de 45 años salen menos del país.

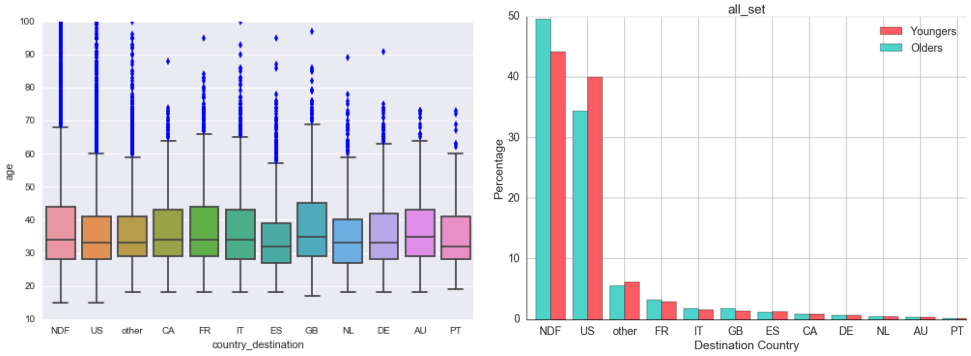


Figura 2.6. edad vs país de destino

Esta característica también posee un número significativo de valores nulos, exactamente el 42,41 % del total de los datos: 87.990 datos pérdidos en el conjunto de

entrenamiento y 28.876 datos pérdidos del conjunto de test. En las imágenes de la Figura 2.7, la imagen de la izquierda corresponde a los usuarios que especifican la edad y la de la derecha a los que poseen valores nulos. Muestran que el 55 % de los usuarios que tienen un valor no nulo, en la característica *age*, realizan reserva, mientras que solo el 25 % de los usuarios que poseen un valor nulo la realizan. Ésto sugiere que proporcionar la edad es una señal de que el usuario es más serio a la hora de realizar la reserva.

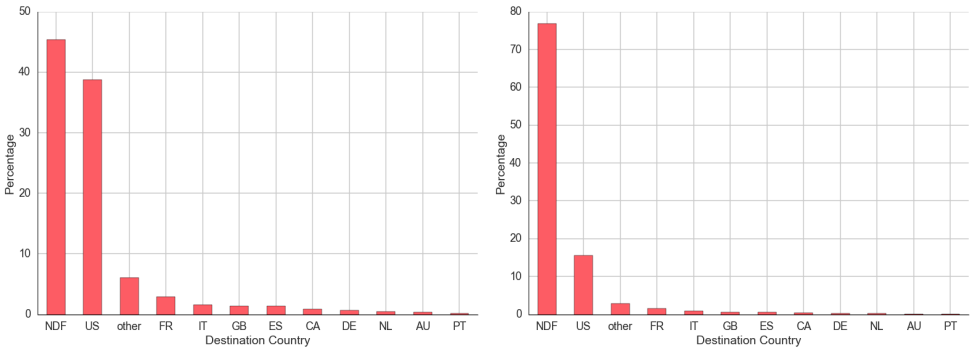


Figura 2.7. Porcentaje del país de reserva

- *Country destination*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.8.

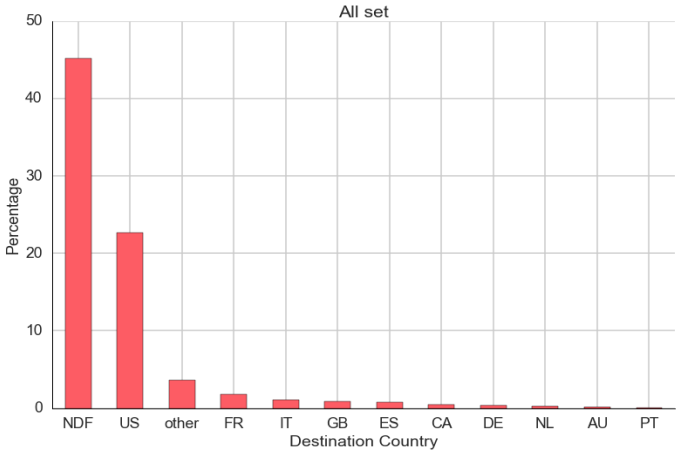


Figura 2.8. Distribución del país de reserva los usuarios

Se puede ver en la Figura 2.8 que casi la mitad de los usuarios acaba no realizando reserva. Y que entre los usuarios que han realizado reserva en Airbnb, USA es el destino más popular.

- *Language*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.9.

  En la Figura 2.9, se puede observar que la mayoría de los usuarios son de habla inglesa, concretamente el 96,74 %. No es algo que sorprenda ya que el *dataset* que

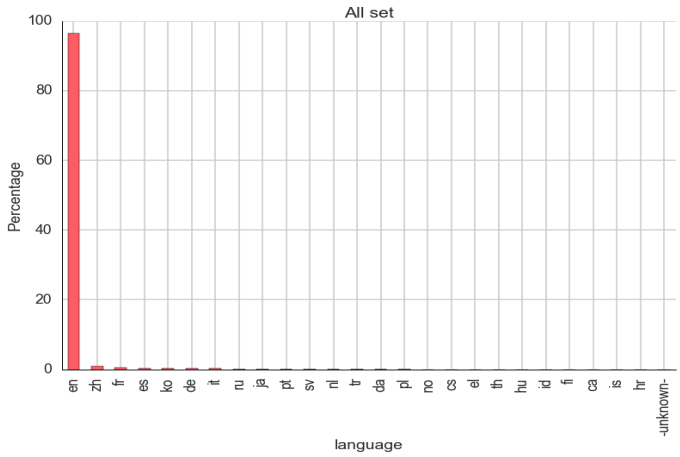


Figura 2.9. Distribución de la lengua origen de los usuarios

se ha proporcionado es de usuarios estadounidenses. Esta característica es bastante relevante ya que como se observa en la Figura 2.19 la preferencia del país de reserva de destino está muy condicionada por la lengua del usuario de origen. Por ejemplo, el usuario de idioma finés es el más fiel a la hora de realizar reservas, contemplando únicamente entre sus posibles lugares de destino opciones como: 'USA', 'Italia' u 'other'.

- *Gender*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.10.

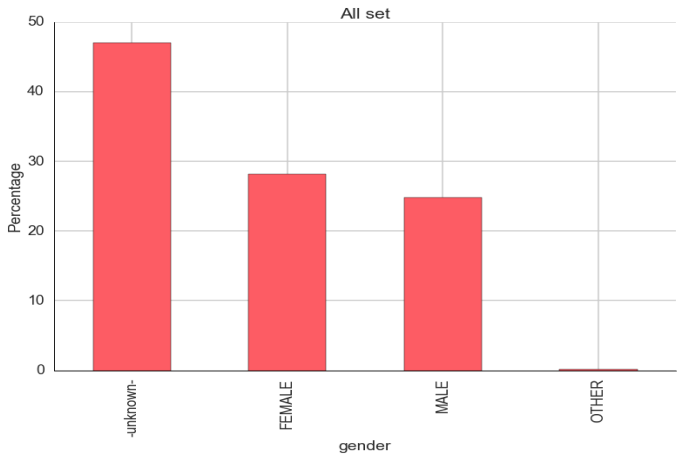


Figura 2.10. Distribución del género de los usuarios

En la Figura 2.10 se puede observar que no hay una gran diferencia entre el género de los usuarios de Airbnb, aproximadamente un 28 % de los usuarios tienen género femenino frente a un 25 % de género masculino. Pudiendose observar además en la Figura 2.19 que el género no influye mucho en el lugar de reserva ya que para ambos sexos se obtienen resultados parecidos. También se puede observar que el 46,99 % de los usuarios en este conjunto de datos no especifico su género en su

perfil de usuario y que son este grupo los más propensos a no realizar reserva.

- *Signup method*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.11.

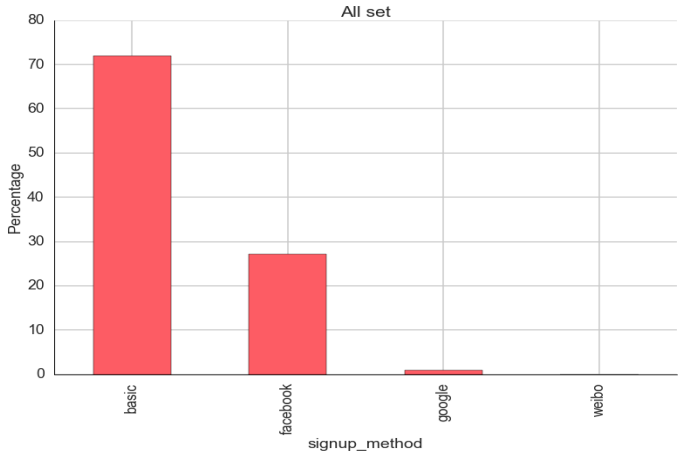


Figura 2.11. Distribución de la característica Singup method

En la Figura 2.11 y Figura 2.19, se puede observar que el método más utilizado en el registro por los usuarios de Airbnb, casi un 72 %, es el 'basic'. También se puede observar que los usuarios que utilizaron el método de registro 'google' son menos propensos a realizar reservas que los que utilizaron 'basic' o 'facebook'.

- *Signup app*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.12.

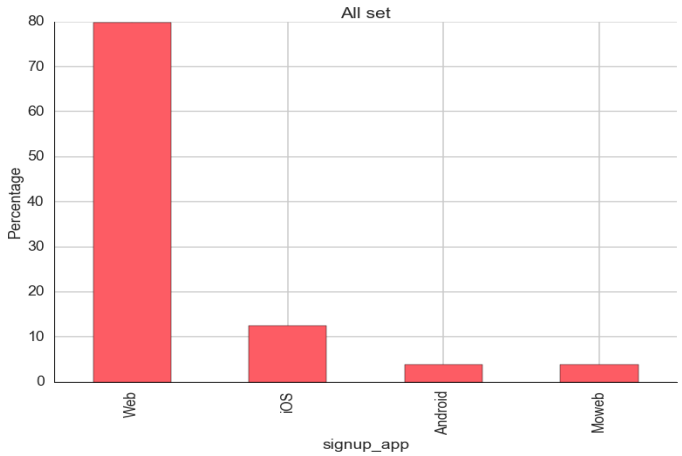


Figura 2.12. Distribución de la característica Singup app

En la Figura 2.12 y Figura 2.19, se puede observar que los usuarios prefieren registrarse vía web (un 80 %) y que son este grupo de usuarios los que con más frecuencia realizan una reserva. También se observa que los usuarios que se registran mediante aplicaciones móviles son los más propensos a no realizar reservas.

- *Signup flow*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.13.

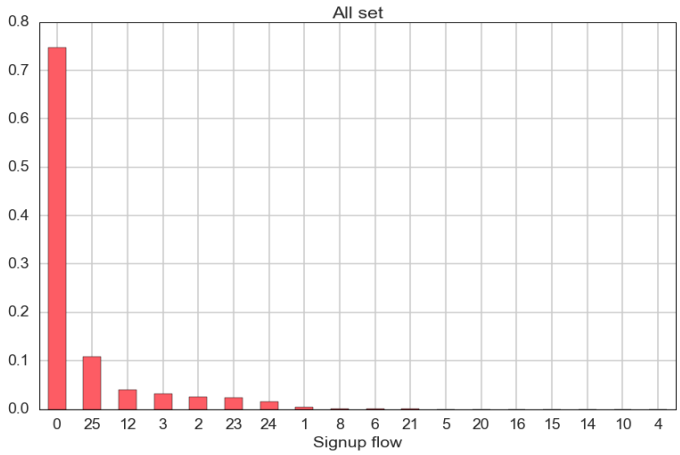


Figura 2.13. Distribución de la característica Singup flow

En la Figura 2.13, se puede observar que la gran mayoría de los usuarios, casi el 75 %, se registran a través de la página identificada con el identificador 0. También se puede ver en la Figura 2.19 que los usuarios que acceden a través de la página identificada con el id 3 son los más fiables a la hora de realizar una reserva.

- *Affiliate channel*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.14.

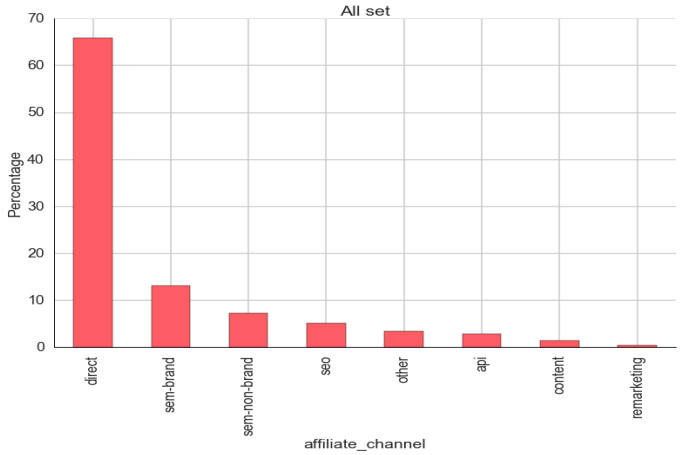


Figura 2.14. Distribución de la característica Affiliate channel

En la Figura 2.14 y Figura 2.19, se puede observar que el canal de distribución más utilizado por los usuarios, en un 65 %, es el 'direct' y que 'content', un tipo de canal utilizado con muy poca frecuencia, es el que menos proporción de reserva realiza.

- *Affiliate provider*: Es una característica de tipo categórica y su distribución es la

que se muestra en la Figura 2.15.

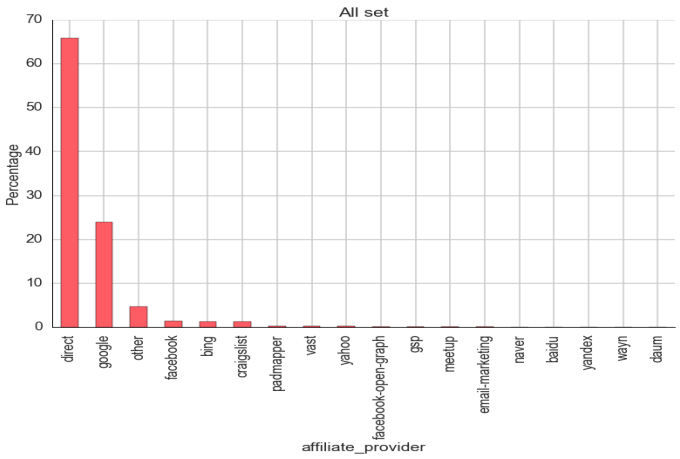


Figura 2.15. Distribución de la característica Affiliate provider

En la Figura 2.15, se puede observar tal y como se ha visto con la característica *Affiliate channel* que la opción más utilizada, con un 65 %, es la 'direct'. También se puede ver en la Figura 2.19 que los provedores más comunes, tales como 'google', 'direct', 'bing', 'yahoo' y 'facebook'; son los que más garantía presentan a la hora de que el usuario realice una reserva.

- *First affiliate tracked*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.16.

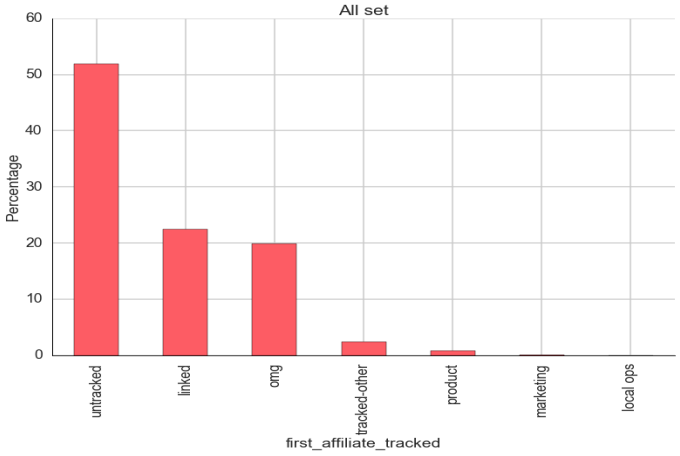


Figura 2.16. Distribución de la característica First affiliate tracked

En la Figura 2.16, se puede observar que la opción más utilizada es la 'untracked' (un 52 % de los datos). También se puede ver en la Figura 2.19 que la opción 'local ops', que es la menos utilizada, es la que menos reservas produce. Por último, nombrar que esta característica posee una pequeña cantidad de valores nulos, el 2,2 % del total de los datos: 6.065 datos pérdidos en el conjunto de entrenamiento y 20 datos pérdidos del conjunto de test

- *First device type*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.17.

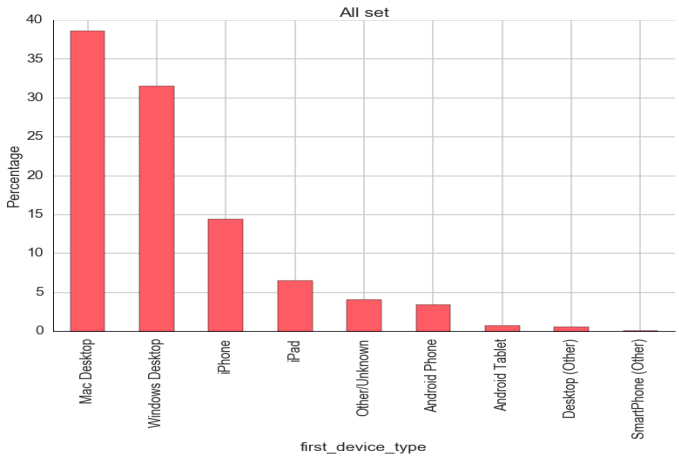


Figura 2.17. Distribución de la característica First device type

En la Figura 2.17, se puede observar que hay una gran cantidad de usuarios que utilizan el sistema operativo OS X, aproximadamente el 60 %, y que predominan los usuarios que utilizan equipos de sobremesa, un 71 %, frente a los equipos portátiles. También se puede ver en la Figura 2.19 que los usuarios que utilizan un equipo de sobremesa son más fieles a la hora de realizar una reserva.

- *First browser*: Es una característica de tipo categórica y su distribución es la que se muestra en la Figura 2.18.

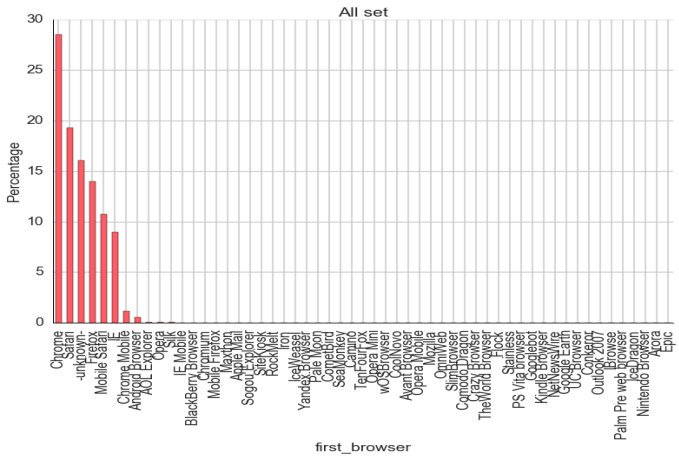


Figura 2.18. Distribución de la característica First browser

En la Figura 2.18, se puede observar que el navegador favorito de los usuarios es 'chrome', utilizado por un 28 %. La alta popularidad de navegadores como 'safari' y 'IE' está justificada debido a que son respectivamente los navegadores por defecto de los equipos OS X y Windows. De la Figura 2.19 es difícil sacar conclusiones

debido a al gran diversidad de navegadores registrados en la base de datos. Lo único a destacar es que los usuarios que utilizan algún navegador de los más conocidos tienen un comportamiento similar a la hora de realizar las reservas.

Figura 2.19. Características categóricas

- *Análisis de session.csv*: Como se hizo referencia anteriormente, este fichero solo contiene los datos de un 49 % del total del los usuarios, concretamente un 34,02 % de los usuarios del conjunto de entrenamiento (72.631) y un 98,96 % de los usuarios del conjunto de test (61.426). Por tanto, existen una gran cantidad de usuarios en el conjunto de entrenamiento sin registro. Este fichero, posee de cada usuario una cantidad de muestras limitadas que corresponden a acciones realizadas por el usuario junto a otros detalles de la acción, como son: su tipo, su duración y el

Hardware/Software donde se llevo a cabo.

El análisis de sus características es el siguiente:

— *action*: Es una característica de tipo categórica que posee un total de 359 valores diferentes, entre los que se encuentra 'nan' (valor nulo) que corresponde al 0,75 % de los datos. Las acciones más realizadas por los usuarios, que aparecen en más muestras del fichero session, son: `Show` que aparece en el 26,19 % del total de las muestras y `Index` que aparece en el 7,9837 % del total de las muestras.

— *action type*: Es una característica de tipo categórica que unicamente posee 10 valores diferentes, entre los que se encuentra 'nan' que corresponde al 10,65 % del total de los datos. Los distintos valores de esta característica y su frecuencia de aparición se muestran en la Tabla 2.1.

— *action detail*: Es una característica de tipo categórica que posee un total de 155 valores diferentes, entre los que se encuentra el mismo número de valores nulos que en la característica *action type*. El valor que aparece con más frecuencia, aparece en un 16,8142 % del total de las muestras, es `view show result` que corresponde a el tipo de acción `View`.

— *secs elapsed*: Es una característica discreta de tipo numérico. Posee 136.031 valores nulos, que corresponde a un 1,287 % del total de las muestras. Algunos de sus estadísticos son los siguientes:

- *mean*: 19.405,81
- *std*: 88.884,24
- *min*: 0,0
- *25 %*: 229,0
- *50 %*: 1.147,0
- *75 %*: 8.444,0
- *max*: 1.799.977,0

2.2. Metodología

2.2.1. Descripción de la tarea de predicción

La tarea de predicción se basa en predecir en que país un nuevo usuario de Airbnb realizará su primera reserva. Dada la naturaleza del los datos del conjunto de entrenamiento, en la que los datos están perfectamente etiquetados y la variable respuesta es categórica, la tarea de predicción consiste en la resolución de un problema de clasificación.

2.2.2. Validación de la predicción

Para validar el modelo se ha dividido el conjunto de datos de entrenamiento en dos subconjuntos:

Tabla 2.1. Distribución de frecuencias de la característica tipo de acción

Valor	Porcentaje
View	33,69 %
Data	19,90 %
Click	18,89 %
NAN	10,65 %
Unknown	9,76 %
Submit	5,89 %
Message post	0,82 %
Parter Callback	0,18 %
Booking Request	0,18 %
Modify	0,01 %
Booking Response	0,000038 %

- Subconjunto de entrenamiento (X_train, y_train): Está formado por las 165.425 primeras muestras, el 77,5 %, de los datos del conjunto de entrenamiento. Se utiliza para construir el clasificador.
- Subconjunto de validación (X_valid, y_valid): Está formado por las 48.026 últimas muestras, el 22,5 %, de los datos del conjunto de entrenamiento. Se utiliza para evaluar el clasificador.

Se ha optado por utilizar este método de validación en lugar de otros métodos más precisos de validación cruzada, como *k-fold cross-validation*, ya que para este tipo de problemas, en el que las muestras evolucionan temporalmente, no presentan buenos resultados. Las proporciones elegidas para cada subconjunto han sido elegidas respetando las proporciones del conjunto de entrenamiento y del conjunto de test, 77,47 % y 22,53 % respectivamente.

2.2.3. Pre-procesado de los datos

1) *date_first_booking*: Aunque esta característica hubiera sido interesante a la hora de predecir el lugar de destino, ya que diferencia el comportamiento de dos grupos: los que realizaron reserva y los que no; y hubiera servido para obtener más información del comportamiento de los usuarios, por ejemplo comparando ésta con otras características temporales. Para este problema se ha descartado esta característica, ya que el cambio que realizaron los organizadores, sustituyeron sus valores del conjunto de test por valores nulos, hizo que no aportara nada en las predicciones.

2) *age*: La edad se ha representado con su número directamente. Se ha considerado que todos los valores son válidos, incluso los que no tienen sentido '*outliers*' (edades < 15 años ó ≥ 100 años). Además, se ha considerado que los valores entre los años 1900 y 2010 corresponden a fechas de nacimiento, ya que presentan prácticamente la misma distribución que su respectiva edad en el país de destino de reserva. Por tanto, se han convertido a su valor correspondiente, que viene dado por $tfa\_year - age_{birthday}$.

3) *action*, *action_ type* y *action_ detail*: En estas características se ha remplazado el valor numpy.NAN por un 'NAN' de tipo string, para que sea considerado como un valor de estas variables categóricas. Se ha acotado el número de valores de la característica categórica action de 359 a 231 sustituyendo los valores menos frecuentes por el valor 'OTHER'. Este pre-procesado se ha realizado ya que por cada valor de cada acción, tipo de acción y detalle de acción se añadía una nueva característica (524 en total), lo que provoca en el equipo utilizado en este proyecto problemas de Memoria a la hora de ejecutar el script ya que el conjunto de datos se vuelve demasiado grande. De esta forma, se ha conseguido reducir el número de características añadidas a 396. Las nuevas características son discretas de tipo numérico y su valor viene dado por el número de veces que el usuario ha realizado esa determinada acción, tipo de acción y detalle de acción.

4) *Características categóricas*: Dado que la librería `Scikit-learn` de Python no trabaja directamente con variables categóricas a la hora de entrenar el clasificador [3], es necesario convertir las variables categóricas a numéricas correctamente. Esto se consigue mediante **one-hot encoding**.

5) *valores nulos*: Se ha sustituido todos los valores nulos (numpy.NAN) por el valor -1. Se ha probado sustituyendo los valores nulos de cada variable por la media, moda o mediana como se indica en [3], pero los resultados conseguidos no han sido satisfactorios.

2.2.4. Ingeniería de las características

1) *date_ account_ create* y *timestand_ first_ active*: A partir de aplicar ingeniería de datos sobre estas características se ha conseguido mejorar sustancialmente los resultados. Se han creado un total de 10 nuevas características:

   - tfa_year, tfa_month, tfa_day: Se obtienen remplazando el formato original de la característica *timestamp_ first_ active* por el año, mes y día de cada muestra.
   - dac_year, dac_month, dac_day: Se obtienen remplazando el formato original de la característica *data_ account_ create* por el año, mes y día de cada muestra.
   - dac_tfa_secs y $sign\_dac\_tfa$: Representan la diferencia entre *date_ account_ create* y *timestamp_ first_ active* de cada usuario. dac_tfa_secs contiene el valor absoluto de la diferencia, en segundos, entre las dos características; y $sign\_dac\_tfa$ contiene el signo de la diferencia, que viene dado por -1 si es negativa, por 0 si es igual a cero o por 1 si es positiva.
   - $season\_tfa$, $season\_dac$: Representan la estación del año de las características *date_ account_ create* y *timestamp_ first_ active*.

2) *secs_ elapsed*: A partir de *secs_ elapsed* se han añadido 4 nuevas características para cada usuario que poseen los valores de sus principales estadísticos: valor total, media, desviación típica y mediana.

2.2.5. Evaluación del modelo

La métrica de evaluación utilizada para evaluar las predicciones en esta competición ha sido elegida por los organizadores y se conoce como la métrica $nDCG_k$ (Normalized discount cumulative gain) donde k=5 [4]. Todos los resultados de su cálculo son valores comprendidos en el intervalo 0,0 a 1,0; éstos se obtienen de acuerdo a la ecuación (2.1).

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{2.1}$$

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{2.2}$$

Donde DCG_k es la ganancia total acumulada en un rango k, $IDCG_k$ es el valor DCG_k máximo (ideal) para un conjunto dado y rel_i es la relevancia del resultado en la posición i. El cálculo de DCG_k viene dado por la ecuación (2.2)

Este método de validación se aplica en las predicciones de cada usuario, que consisten en listar los 5 países de destino donde hay más probabilidad de que el usuario realice su primera reserva. Por tanto, una vez se obtienen las predicciones de cada usuario, se comparan con el conjunto de validación para obtener el score $nDCG_5$ de modelo. El resultado de la comparación se muestra en un array de relevancias. Este array es un array binario que relaciona para cada usuario sus k predicciones, ordenadas por mayor probabilidad, con su destino del conjunto de validación. Donde la predicción que coincide es marcado con relevancia = 1, mientras que el resto tienen relevancia = 0. Por ejemplo, si las predicciones obtenidas para un usuario son [“FR” “US” “ES” “IT” “GB”] y el lugar de destino para ese mismo usuario en el conjunto de validación es “IT”, el array de relevancia de salida viene dado por [0 0 0 1 0]. Una vez obtenido el array se utiliza la ecuación de la métrica nDCG para obtener el valor del score.

Algunos ejemplos del valor del score $nDCG_5$ para distintas predicciones de uno usuario cuyo destino es “FR” son las que se muestran a continuación:

[“FR” “US” “ES” “IT” “GB”] tiene una array de relevancia [1 0 0 0 0] y da un score nDCG = $\frac{2^1-1}{log_2(1+1)} = 1{,}0$

[“US” “FR” “ES” “IT” “GB”] tiene una array de relevancia [0 1 0 0 0] y da un score nDCG = $\frac{2^0-1}{log_2(1+1)} + \frac{2^1-1}{log_2(2+1)} = \frac{1}{1{,}58496} = 0{,}6309$

[“NDF” “US” “ES” “IT” “GB”] tiene una array de relevancia [0 0 0 0 0] y da un score nDCG = 0,0

2.3. Selección del modelo

Existe una gran cantidad de métodos para la resolución de los problemas de clasificación. Los más comunes son los siguientes:

- **Regresión Logística**.
- *Support Vector Machine*.
- ***K nearest neighbours***.
- **Naïve Bayes**.
- Árboles de decisión.
- *Ensemble methods*:
  - **Bosques aleatorios**
  - ***Boosting***
- *Neural Networks*.

Aunque esta competición presenta claramente datos desbalanceados ya que la mayoría de las muestras del conjunto de entrenamiento pertenecen a usuarios que no han elegido lugar de destino o han elegido "US", como se muestra en la Figura 2.8, lo que llevaría a construir un clasificador por niveles para separar estas muestras del resto, se ha optado por utilizar directamente un clasificador multiclase. Esto se ha hecho así como prueba para mejorar el conocimiento de los algoritmos de clasificación. Los métodos utilizados durante la competición son los marcados en negrita. A continuación se verá en que consisten estos métodos, hablando también de los árboles de decisión ya que los *ensemble methods* empleados en este proyecto lo utilizan como clasificador base, así como la configuración de parámetros que se ha empleado para cada uno de ellos con el fin de maximizar la bondad de ajuste del modelo.

2.3.1. Regresión Logística

La regresión logística [5] es un modelo de regresión en el cual la variable dependiente de respuesta posee un valor cualitativo, generalmente dicotómico. De ahí que su principal uso sea para problemas de clasificación. Su principal objetivo consiste en explicar la relación entre la variable a estimar y unas variables que se conocen a priori (a las que se denominan variables independientes o explicativas).

Dentro de estos modelos, en función del número de alternativas que presenta la variable respuesta, podemos hacer la siguiente clasificación:

- *Modelos binarios o dicotómicos*: En los que la variable respuesta sólo tiene dos alternativas, y_0 y y_1. Utilizando una estrategia *One vs rest* y categorizando la variable respuesta estos modelos pueden funcionar como un clasificador multiclase.

- *Modelos multinomiales*: El clasificador multiclase por definición. En los que la variable respuesta tiene más de dos alternativas, y_0, ..., y_j.

La ecuación de general del modelo (función sigmoidal/logística) tanto para modelos dicotómicos como multinomiales es la que se muestra a continuación:

$$P(y = j|x, \beta) = h_\beta(x) = \frac{1}{1 + e^{-\beta^T x}} \tag{2.3}$$

Donde $\beta^T x$, es la parte lineal del modelo y viene descrito por:

$$\beta^T x = \beta_{j,0} + \beta_{j,1} x_1 + \beta_{j,2} x_2 + ... + \beta_{j,k} x_k \tag{2.4}$$

Donde x_1, x_2, ..., x_k son las variables explicativas, y_j es la respuesta, y $\beta_{j,1}$, $\beta_{j,2}$, ..., $\beta_{j,k}$ son los parámetros del modelo para una determinada respuesta j.

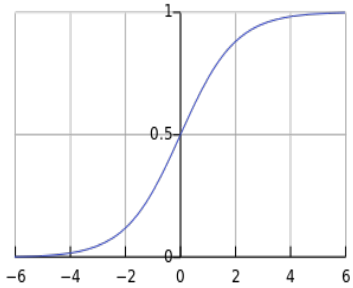


Figura 2.20. Función logística

Fuente: Wikipedia

La estimación del modelo de regresión logística se realiza por el método de máxima verosimilitud [6]. Este método consiste en encontrar, mediante un proceso iterativo, la mejor combinación de los parámetros β para el modelo. Donde la mejor combinación de los parámetros corresponde a los valores que minimizan el error entre la variable respuesta estimada, y', y la variable respuesta real de las muestras, y. Por tanto, el método consiste en minimizar el error de la función de coste, $J(\beta)$, que para problemas en los que se sigue una estrategia *one vs rest* viene dado por la siguiente ecuación:

$$\begin{aligned} J(\beta) &= \frac{1}{m} \sum_{i=1}^{m} Cost(h_\beta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} log(h_\beta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\beta(x^{(i)}))] \end{aligned} \tag{2.5}$$

Donde m es el número de instancias del conjunto de entrenamiento.

En la librería `Scikit-learn` este modelo viene dado por la función **LogisticRegression** [7]. La configuración de parámetros que se ha utilizado, ver tabla 2.2, busca maximizar la bondad de ajuste siguiendo una estrategia one vs rest.

2.3.2. Naïve Bayes

El clasificador Naïve Bayes [9] es un clasificador que está basado en aplicar el Teorema de Bayes con la asunción "naive" que asume independencia entre cada par de características.

Dada una clase y y dada una combinación de características $x_1, ..., x_k$, el Teorema de Bayes dice que la probabilidad a posteriori, $P(y|x_1, ..., x_k)$, se puede calcular a partir de las probabilidades a priori y la verosimilitud, como se observa en la siguiente relación:

$$P(y|x_1, ..., x_k) = \frac{P(y)P(x_1, ..., x_k|y)}{P(x_1, ..., x_k)} \tag{2.6}$$

Dada la independecia entre características que asume el modelo Naïve, la expresión se puede reescribir de la siguiente forma:

$$P(y|x_1, ..., x_k) = \frac{P(y)\prod_{j=1}^{k} P(x_j|y)}{P(x_1, ..., x_k)} \tag{2.7}$$

Sólo queda clasificar la muestra, y esto se logra eligiendo el valor más probable de Y dado una combinación de características. Para lograrlo se utiliza el método de máxima la probabilidad a posteriori (MAP):

$$\hat{y} = \arg\max_{y \epsilon Y} P(y) \prod_{j=1}^{k} P(x_j|y) \tag{2.8}$$

Las diferencias entre los distintos clasificadores viene dada por la distribución de $P(x_i|y)$ [10]. En este trabajo únicamente se ha utilizado la distribución Gaussiana que se implementa con la función **GaussianNB** de la librería de `Scikit-learn`. Esta función únicamente permite configurar las prioridades previas de las clases [11]. Dado que el objetivo es que las prioridades de las clases se ajusten a través del conjunto de datos de entrenamiento, se ha ejecutado la función sin ningún tipo de configuración.

A pesar de ser un método sencillo, dada su asunción de independencia, este clasificador se utiliza para situaciones reales como el filtro anti spam y la clasificación de documentos. Además presenta entre otras ventajas:

- Necesita pocos datos de entrenamiento para realizar estimaciones y generalmente consigue buenos resultados.
- Es bastante rápido si lo comparamos con otros métodos más sofisticados.

2.3.3. K nearest neighbors (KNN)

El clasificador *K nearest neighbors* es un método no paramétrico el cuál asigna a una instancia desconocida, la clase a la que pertenecen la mayoría de sus k vecinos más

próximos; donde los k vecinos más próximos a la nueva instancia vienen dados por las k instancias del conjunto de entrenamiento más próximas (en función de la distancia ya que asume que todos los datos están en un espacio métrico) [12]. Un ejemplo gráfico del proceso de clasificación KNN para un k=3 es el que se muestra en la Figura 2.21 en el cuál la instancia nueva, color verde, será clasificada en la clase 2, color azul. El método KNN es considerado un algoritmo perezoso ya que a partir de los datos de entrenamiento no crea ninguna generalización (un modelo) lo que hace que mantenga estos datos todo el tiempo en memoria durante el tiempo de ejecución.

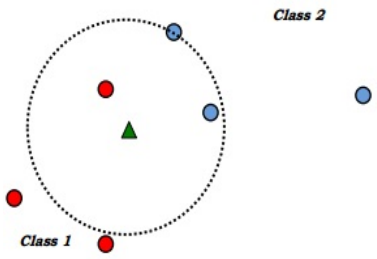


Figura 2.21. Método de clasificación Knn para k=3

Fuente: Joe Luis Villa Medina

Las características más relevantes de método KNN son las siguientes:

- **Proceso de clasificación costoso computacionalmente**. Para determinar el vecino más cercano a una instancia se tiene que calcular la distancia entre ella y las N muestras del conjunto de entrenamiento, lo que supone un coste de $O[DN^2]$. Se puede ganar eficiencia utilizando estructuras de datos eficientes, tales como: Ball-Tree y KD-tree.

- **Requiere de almacenamiento**. Al tratarse de un algoritmo perezoso, debe almacenar los datos en memoria.

- **Elección de la métrica de distancia**. Hay que elegir qué métrica de distancia hay que utilizar, ya que es crítica para el rendimiento del algoritmo. La más común en problemas en los que las variables son continuas es la distancia Euclídea, pero también se puede optar por otras métricas como: la distancia Manhattan, la distancia de Minkowski, ..; para problemas con variables categóricas la mejor métrica es la distancia de Hamming.

- **Elección del parámetro k**. La elección del parametro, k, es de gran importancia. Por norma general, k debe ser un valor muy por debajo del número de instancias del conjunto de entrenamiento para evitar que otras clases lejanas tenga mucha influencia y lo suficientemente grande para evitar la influencia del ruido.

- **Proceso de clasificación bastante sencillo**. No requiere de la construcción de un modelo previo, es una simple comparación con el conjunto de entrenamiento.

En la librería `Scikit-learn` este modelo viene dado por la función **KNeighborsClassifier** [13]. La configuración de parámetros que se ha utilizado, ver tabla 2.3 busca, utilizando la distancia euclídea como métrica de distancia, maximizar la bondad de ajuste del modelo sin perder de vista la eficiencia computacional.

2.3.4. Árboles de decisión

El árbol de decisión es un clasificador no lineal basado en un grafo con forma de árbol. Esta estructura está formada por nodos los cuales están conectados a otros nodos a través de aristas estableciendo una relación de padres e hijos.

Una vez construido un árbol de decisión, la forma de clasificar una instancia es comparándola con los nodos del árbol para encaminar la instancia a un nodo terminal (hoja) que es el nodo que contiene el resultado de la clasificación. Primero se compara con el nodo raíz en el cual se toma la primera decisión de encaminamiento, encamina a uno de sus dos hijos, y luego con los nodos descendientes por los que su respectivo nodo padre ha decidido encaminar. Como ejemplo, la Figura 2.22 muestra un árbol de decisión para un problema de clasificación con tres variables: x_1, x_2 y x_3; y dos clases de salida: A y B.

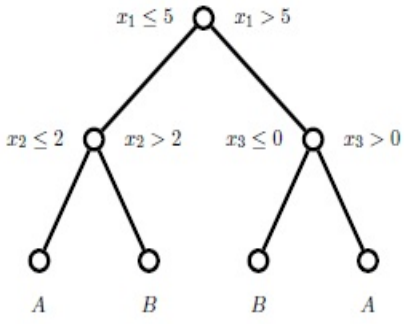


Figura 2.22. Árbol de decisión

Es de gran importancia saber en qué orden van a ser testeadas las variables explicativas. El orden viene dado por la cantidad de incertidumbre que reducen. Para cuantificar la incertidumbre y construir el árbol de decisión correctamente se utiliza técnicas como la **Información Ganada**, que utiliza la entropía, o la **Impureza de Gini** [14].

La Impureza de Gini se define como:

$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2 \tag{2.9}$$

donde t es el subconjunto de instancias para el nodo, j es el número de clases y $P(i|t)$ es la probabilidad de seleccionar un elemento de la clase i del subconjunto del nodo.

La Información Ganada se define como la diferencia entre la entropía del nodo padre, $H(T)$, y el promediado ponderado de la entropía de los nodos hijos:

$$IG(T,a) = H(T) - \sum_{v \in vals(a)} \frac{\mid \{x \in T | x_a = v\} \mid}{\mid T \mid} H(\{x \in T | x_a = v\}) \tag{2.10}$$

donde T es el conjunto de instancias, a es la variable explicativa bajo prueba, $v \in vals(a)$ es el valor del atributo a para la instancia x, $\{x \in T | x_a = v\}$ es el numero de instancias en el cual el atributo a es igual al valor de v y $H(\{x \in T | x_a = v\})$ es la entropía del subconjunto de instancias para las cuales el valor de la variable explicativa a es v.

La entropía se define como:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i) \tag{2.11}$$

2.3.5. Ensemble methods

Aunque los árboles de decisión producen buenos resultados, existen unas técnicas basadas en conjuntos que los mejoran. El principio de estas técnicas consiste en combinar un conjunto de predicciones de varios estimadores con el fin de obtener predicciones más robustas que las que se obtendría por medio de un estimador individual [15]. En este trabajo los *ensemble methods* empleados utilizan como clasificador base el árbol de decisión, y son los siguientes: *Random Forest*, *Extremely Randomized trees* y *Gradient boosting*.

2.3.5.1. Random Forest

El método *Random Forest* [16] consiste en utilizar una colección de árboles de decisión que están caracterizados por entrenarse con diferentes subconjuntos de datos de entrenamiento y diferentes subconjuntos de variables. La creación de estos subconjuntos se hace de manera aleatoria. Los subconjuntos de datos se crean mediante *bootstrapping* [17], de esta forma cada árbol se entrena aproximadamente con el 66 % del conjunto original de entrenamiento y el resto de las muestras (denominadas out of bag) se reservan para la validación del modelo. Los subconjuntos de variables se eligen de forma aleatoria para que no haya correlación entre los distintos árboles, lo que garantiza la aleatorización del modelo. Esta aleatoriedad es especialmente interesante ya que dota a este método de una gran capacidad de generalización (robusto al sobreajuste).

Una vez que el algoritmo es entrenado, la evaluación de cada nueva entrada se realiza con todo conjunto de árboles. El resultado de la variable respuesta, para problemas de clasificación, viene dado por la predicción más frecuente de entre todo el conjunto de árboles (En la librería `Scikit-learn` el resultado de la variable respuesta viene dado por el promediado).

En la librería `Scikit-learn` este modelo viene dado por la función **RandomForestClassifier** [18]. La configuración de parámetros que se ha utilizado, ver tabla 2.4 busca maximizar la bondad de ajuste del modelo sin perder de vista la eficiencia computacional.

2.3.5.2. Extremely Randomized trees

El método *Extremely Randomized Trees* consiste en utilizar una colección de árboles de decisión que están caracterizados por entrenarse con todo el conjunto de entrenamiento y diferentes subconjuntos de variables en los que el umbral de decisión de cada variable en cada nodo es elegido completamente al azar, seleccionando al mejor umbral como regla de la división [20].

Una vez el algoritmo es entrenado, las predicciones de los distintos árboles se obtienen evaluando cada nueva entrada con todo el conjunto de árboles. El resultado de la variable respuesta, para problemas de clasificación, viene dado por la predicción que más se repite.

En la librería `Scikit-learn` este modelo viene dado por la función **ExtraTreesClassifier** [21]. La configuración de parámetros que se ha utilizado, ver tabla 2.4 busca maximizar la bondad de ajuste del modelo sin perder de vista la eficiencia computacional.

2.3.5.3. Gradient boosting

El método *Gradient Boosting* está basado en el método *Ada boosting* [22]. La idea de los métodos *Ada boosting* es utilizar un algoritmo débil (por ejemplo, árbol de decisión con uno o dos nodos) secuencialmente con versiones modificadas de los datos, produciendo una secuencia de M clasificadores débiles, con el objetivo de obtener un clasificador fuerte [23]. Las modificaciones en los datos en cada paso consisten en aplicar pesos a cada una de las muestras del conjunto de entrenamiento. Inicialmente, en la primera iteración, el peso para todo el conjunto de muestras de entrenamiento es el mismo, pero conforme va iterando el modelo los pesos se van modificando, incrementando el peso a las muestras que no fueron clasificadas correctamente. De esta forma, las muestras difíciles de clasificar van tomando relevancia conforme itera el modelo. El gradiente boosting es una implementación del método Ada boosting en la que se aplica el gradiente por descenso para mejorar la robustez y la velocidad del algoritmo.

En la librería `Scikit-learn` este modelo viene dado por la función **GradientBoostingClassifier** [24]. La configuración de parámetros que se ha utilizado, ver tabla

2.5 busca maximizar la bondad de ajuste del modelo sin perder de vista la eficiencia computacional, al igual que ocurre con los otros métodos.

Tabla 2.2. Configuración de los parámetros para el modelo de regresión logística

Parámetro	Descripción	Valor	Comentario
solver	Algoritmo a utilizar en la optimización.	**'liblinear'**	Único algoritmo que soporta la estrategia one vs rest.
multi_class	Para implementar clasificación multiclase.	**'ovr' (onevsrest)**	Es la estrategía que se ha utilizado.
Penalty	Se utiliza para especificar la norma utilizada en la penalización.	**'l2' (Default)**	Se ha elegido el valor que mejor resultados produce.
Dual	Formulación dual o primaria.	**False (Default)**	Se utiliza dual=False ya que se cumple que $n\_samples > n\_features$.
C	Inversa a la fuerza de la regularización. La regularización es más fuerte conforme se acerca al valor 0.0	**0.81**	El valor utilizado, que es el que mejor resultados produce, nos indica que no es necesario aplicar un alto grado de compensación, con los pesos, en las variables, lo que nos indica que el modelo no está muy sobreajustado.
max_iter	Máximo número de soluciones para que el sorver converge.	**100 (Default)**	No es útil para el solver 'liblinear'.
class_weight	Pesos asociados a las clases. Si no se especifica, todas las clases tienen el mismo peso.	**None (Default)**	-
random_state	El seed del generador de números pseudo aleatorios.	**None (Default)**	Se ha probado con distintos valores con la misma configuración y el resultado no ha variado. Para más información, sobre porque no afecta este parámetro al resultado [8].
tol	Tolerancia al criterio de parada.	**1e-4 (Default)**	-
n_jobs	Nº de núcleos CPU utilizados durante el bucle de validación cruzada.	**-1**	El valor -1 equivale a utilizar todos los núcleos disponibles.
fit_intercept	Especifica si una constante puede ser añadida a una función de decisión.	**True (Default)**	-
intercept_scaling	Útil con el solver 'liblinear' y fit_intercept a True. Esta contante se utiliza para regular el efecto de la regularización en el peso de característica sintética.	**1.1**	Se ha ido probando y es el que más mejora la bondad del modelo.
warm_start	Cuando está a True, se vuelve a utilizar la solución anterior del método fit como inicialización.	**False (Default)**	No es útil para el solver 'liblinear'.

Tabla 2.3. Configuración de los parámetros para el modelo K-nearest neighbors

Parámetro	Descripción	Valor	Comentario
n_neighbors	Número de vecinos a utilizar por defecto en las consultas.	**50**	La elección del valor de k es bastante importante, ya que hay un compromiso entre la eficiencia computacional y la bondad de ajuste del modelo. Se ha elegido el valor de k límite en el que un incremento de su valor apenas mejora la bondad de ajuste del modelo, tal y como se puede observar en la Figura 3.1.
weights	Función de pesos utilizada en la predicción.	**'uniform' (Default)**	Para que todos los puntos del vecindario tengan la misma influencia en la toma de la decisión.
algorithm	Algoritmo utilizado para calcular los vecinos más cercanos.	**'ball_tree'**	Se ha utilizado el algoritmo BallTree porque es el que más eficiencia computacional produce $O[Dlog(N)]$, ya que este problema cuenta con un número de características (D) y un número de muestras (N) muy grande.
leaf_size	Tamaño de la hoja pasado al algoritmo BallTree o KDTree. Este puede afectar a la velocidad y de consulta, así como la memoria necesaria para almacenar el árbol.	**30 (Default)**	Se ha elegido el valor que mejor resultados produce.
metric	La métrica de distancia utilizada en el árbol.	**'Euclidean'**	Es la métrica que se ha utilizado.
p	Solo aplicable para la métrica de Minkowski. Según su valor se aplica una determinada métrica de distancia.	**1(Default)**	Otra manera de utilizar la distancia euclídea, a la utilizada; es utilizando un valor la métrica Minkowski y un p=2.
metric_params	Argumentos adicionales de las palabras clave para la función de métrica.	**None (Default)**	-
n_jobs	Nº de núcleos CPU utilizados durante el bucle de validación cruzada.	**-1**	El valor -1 equivale a utilizar todos los núcleos disponibles.

Tabla 2.4. Configuración de los parámetros para los bosques aleatorios

Parámetro	Descripción	Valor	Comentario
n_estimators	Número de árboles en el bosque.	**50**	Valor elegido para que la comparación entre los distintos ensemble methods sea válida. Su valor está limitado por el clasificador *Gradient Boosting* ya que este clasificador es bastante ineficiente computacionalmente (no dispone del parámetro n_jobs)
criterion	La función utilizada para construir el árbol. Mide la calidad de una división.	**'gini' (Default)**	Se ha elegido el criterio que mejor resultados produce.
max_features	El número de características a considerar cuando buscamos la mejor división.	**24**	Tal y como se recomienda en [19], el mejor valor viene dado por $\sqrt{n\_features}$ que en este problema es $\sqrt{566} \simeq 24$.
max_depth	Máxima profundidad del árbol.	**None (Default)**	Al ser None se expande automáticamente hasta que sean hojas puras o hasta que todas las hojas contengan menos muestras que min_samples_split.
min_samples_split	El número mínimo de muestras necesario para dividir un nodo interno.	**1**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce.
min_samples_leaf	El número mínimo de muestras necesarias para estar en un nodo de hoja.	**4**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce.
max_leaf_nodes	Máximo número de nodos hoja. Si el valor es None hay un número ilimitado de nodos hoja.	**None (Default)**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce.
min_impurity_split	Umbral para la detención temprana. Un nodo se divide si su impureza está por encima del umbral, de lo contrario es una hoja.	**1e-7 (Default)**	-
boostrap	Si se utilizan muestras boostrap para construir los árboles.	**Default**	Al utilizar *Random forest* su valor por defecto es *True* ya que utiliza boostrap, por el contrario, en los *Extra-trees*, que no utilizan boostrap, su valor por defecto es *False*.
n_jobs	Nº de núcleos CPU utilizados durante el bucle de validación cruzada.	**-1**	El valor -1 equivale a utilizar todos los núcleos disponibles.
random_state	El seed del generador de números pseudo aleatorios.	**np.random.seed(0)**	Se ha elegido un valor fijo para que los resultados de una ejecución a otra no cambien drásticamente; y así poder comparar mejor los algoritmos.

Tabla 2.5. Configuración de los parámetros para el método Gradient Boosting

Parámetro	Descripción	Valor	Comentario
loss	Optimización de la función de pérdida. Unicamente existen dos valores: 'deviance' y 'exponential'.	**'deviance' (Default)**	Se ha utilizado este valor porque la pérdida 'exponencial' sólo es utilizable en problemas de clasificación binaria.
learning_rate	La tasa de aprendizaje establece cual es la contribución de cada árbol en la solución final. Existe un trade-off entre este parámetro y n_estimators.	**0.1 (Default)**	Hay que elegir un valor pequeño para que converja hasta el mínimo global correctamente y no se produzcan oscilaciones.
n_estimators	Número de árboles débiles en secuencia a utilizar.	**50**	El valor seleccionado está sujeto a las restricciones del equipo utilizado, ya que a más clasificadores en secuencia mayor es el tiempo de computo del método. Es algo a tener en cuenta en los resultados ya que este parámetro mejora bastante los resultados de los distintos *ensemble methods*.
max_depth	Máxima profundidad de los árboles individuales.	**6**	Debido a que existe un trade-off de la profundidad de los árboles y el coste computacional, se ha tenido que limitar bastante este parámetro. Se ha elegido el valor más alto posible cuyo coste computacional no es excesivo.
criterion	La función utilizada para construir el árbol. Mide la calidad de una división.	**'friedman mse' (Default)**	Se ha elegido el criterio que mejor resultados produce.
min_samples_split	El número mínimo de muestras necesario para dividir un nodo interno.	**1**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce.
min_samples_leaf	El número mínimo de muestras necesarias para estar en un nodo de hoja.	**3**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce.
max_features	El número de características a considerar cuando buscamos la mejor división.	**24**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce. Este resultado es el mismo que se ha utilizado en los bosques aleatorios que viene dado por $\sqrt{n\_features}$.
max_leaf_nodes	Máximo número de nodos hoja. Si el valor es None hay un número ilimitado de nodos hoja.	**None (Default)**	Se ha ido probando manualmente y se ha elegido el valor que mejor resultados produce.
random_state	El seed del generador de números pseudo aleatorios.	**np.random.seed(0)**	Se ha elegido un valor fijo para que los resultados de una ejecución a otra no cambien drásticamente; y así poder comparar mejor los algoritmos.

Capítulo 3
Resultados

En este capítulo del Proyecto Fin de Máster se muestran y se comentan los resultados que se han obtenido para los distintos métodos.

3.1. Análisis de resultados obtenidos

Se han utilizado 6 clasificadores: *Naive Bayes*, *Regresión Logística*, *K-nearest Neighbors*, *Random Forest*, *Extremely Randomized Trees* y *Gradient Boosting*; con su respectiva configuración de parámetros, mostrada en la Sección 2.3, y con una estrategia *one vs rest* (OvR) para predecir el conjunto de datos de test. Tras aplicar el pre-procesamiento de los datos mostrado en la Subsección 2.2.3, se han obtenido los resultados que se muestran en la siguiente Tabla.

Tabla 3.1. Resultados obtenidos con diferentes clasificadores

Modelo	$nDCG_5$	*Public Score*	*Private Score*
Regresión Logística	0,84137	0,86649	0,87004
Naive Bayes (Gaussiana)	0,64386	0,81856	0,81943
K nearest neighbors	0,82232	0,85310	0,85611
Random Forest	0,84763	0,87452	0,88059
Extremely Randomized trees	0,84604	0,87482	0,87976
Gradient boosting	0,84859	0,87694	0,88195

En la Tabla 3.1 se muestra la probabilidad de acierto, que se ha obtenido tras evaluar cada clasificador con la métrica $nDCG_5$. La columna con la cabecera $nDCG_5$ corresponde a una evaluación local en la que se hace uso del conjunto de validación y las columnas con la cabecera *Public Score* y *Private Score* corresponden a una evaluación remota, se realiza en kaggle tras subir un fichero .csv con los 5 países con mayor probabilidad para cada instancia, en la que se hace uso de todo el conjunto de datos de entrenamiento y de test. La diferencia entre la puntuación pública y la puntuación privada viene dada por el porcentaje en uso de los datos del conjunto de test y por su utilidad [25]. Mientras la puntuación pública se obtiene de una parte de los datos del

conjunto de test (25 % al 33 %) y es utilizado para construir la tabla de clasificación pública que sirve para que los usuarios evaluen sus modelos y obtengan *feedback* del resto de competidores durante el tiempo en el que está activa la competición, la puntuación privada se obtiene del total de los datos del conjunto de test y es utilizada una vez terminada la competición para contruir clasificación final (privada), la que da acceso a los premios.

Observando los resultados obtenidos por los distintos métodos de evaluación, se puede apreciar que para cada modelo el valor del nDCG obtenido del conjunto de validación es más pequeño que el valor del nDCG obtenido del conjunto de test. También se observa que entre los distintos métodos de evaluación se respeta el orden, por puntuación (*Score*), de los distintos clasificadores. El clasificador *Gradient Boosting* es el mejor clasificador (mejor puntuación) en todos los métodos de evaluación utilizados y el *Naive Bayes* el peor (peor puntuación). De estos resultados, no hay que concluir que el único modelo válido para este problema es el clasificador *Gradient Boosting*, ya que estos resultados estan basados en un ranking de Kaggle y la diferencia entre la puntuación de este modelo y el resto, en especial los *ensemble methods*, es pequeña. Sin embargo, lo que sí se puede concluir es que el clasificador *Gradient Boosting* es el mejor modelo para predecir en que país un nuevo usuario de Airbnb hará su primera reserva. Analizando los resultados obtenidos por cada modelo, se puede decir:

El modelo de *regresión Logística* es el modelo que no utiliza *ensemble methods* que mejor resultados ha obtenido. A pesar de su simplicidad, basado en una única ecuación (la función sigma sigmoidal), se han obtenido unos resultados muy próximos a los obtenidos por los métodos de conjuntos. Añadir que no todo son ventajas, ya que es la implementación en la que más tiempo se ha gastado para configurar sus parámetros. Dispone de parámetros para manejar el *overfitting*, para configurar la norma de penalización, para elegir el algoritmo que optimiza el entrenamiento del clasificador, etc.

El modelo *Naive Bayes* como era de esperar es el modelo que peor resultados ha obtenido, ya que existe dependencias entre gran parte de las variables. Por ejemplo, hay una clara relación entre las variables *first_ device_ type* y *first_ browser* ya que el software de un equipo viene con un navegador determinado y el usuario rara vez lo cambia, bien por comodidad o bien por desconocimiento. No obstante, pese a su simplicidad este método ha conseguido unos resultados aceptables (aproximadamente un 82 % de precisión) con los métodos de evaluación que utiliza el conjunto de datos de test, no ocurre lo mismo con el método de evaluación local (un 64 % de precisión). Esta diferencia notoria entre los resultados obtenidos por los distintos métodos de evaluación probablemente es producida porque el conjunto de test dispone de muchas instancias con país de destino 'NDF'. También añadir que este clasificador es bastante eficiente computacionalmente.

El modelo *K-nearest Neighbors* es uno de los modelos que peor resultados ha obteni-

do. Hay que destacar que no se ha elegido la mejor configuración de parámetros posible, la que maximiza la bondad de ajuste, ya que este clasificador es un método bastante lento, especialmente cuando el conjunto de datos de entrenamiento es bastante grande, ya que necesita almacenarlo en memoria y posteriormente moverlo para poder clasificar las nuevas instancias, y cuando se elige un valor de k muy grande, ya que para clasificar una nueva instancia es necesario calcular la distancia con k puntos. Por esta razón, se ha utilizado el valor 50 que es el valor óptimo que cumple con el compromiso que hay entre la bondad de ajuste y la eficiencia computacional, como se indica en la Tabla 2.3. También añadir que la métrica utilizada, dada la naturaleza de los datos, no es la adecuada; lo ideal hubiera sido utilizar la distancia de Hamming (No está disponible en la librería `Scikit-learn`). No obstante, aún si se hubiera elegido la mejor configuración posible, no se hubieran alcanzado las puntuaciones alcanzadas por los *ensemble methods*.

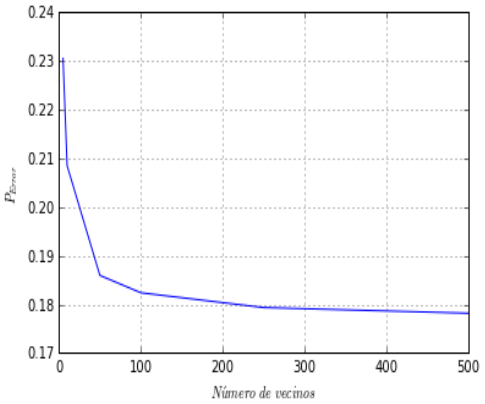


Figura 3.1. KNN: Variación del error con respecto al valor de k.

El modelo de conjuntos *Random Forest* consigue unos resultados muy cercanos a los obtenidos por el mejor modelo, clasificador *Gradient Boosting*, en mucho menos tiempo de computación (aproximadamente la ejecución tarda unos 15 segundos). Por tanto, para un escenario en el que sea muy importante el tiempo de predicción y el sobreajuste, es la mejor opción. Este modelo es muy eficiente, siempre que hardware lo permita, debido a que puede entrenar los árboles en paralelo y no de manera secuencial como lo hace el clasificador *Gradient Boosting*. Por último, destacar que el valor del parámetro *n_estimators* utilizado está lejos de ser el valor óptimo de este modelo, ver Figura 3.2, como se indica en la Tabla 2.4. Si se hubiera elegido una configuración con el valor óptimo, los resultados obtenidos por este clasificador superarían a los del modelo *Gradient Boosting*.

El modelo de conjuntos *Extremely Randomized trees* obtiene unos resultados similares a los obtenidos por el modelo *Random Forest*. Ambos modelos son muy parecidos, únicamente presentan diferencias a la hora de construir los n estimadores. Por tanto, las ventajas de este modelo frente al modelo *Gradient Boosting* son las mismas que las que presenta el modelo *Random Forest*: Más robusto al sobreajuste y más eficiente

computacionalmente (ejecución en unos 20 segundos). De igual manera, los resultado hubieran sido mejores a los conseguidos si se hubiera elegido una mejor configuración del parámetro *n_estimators*.

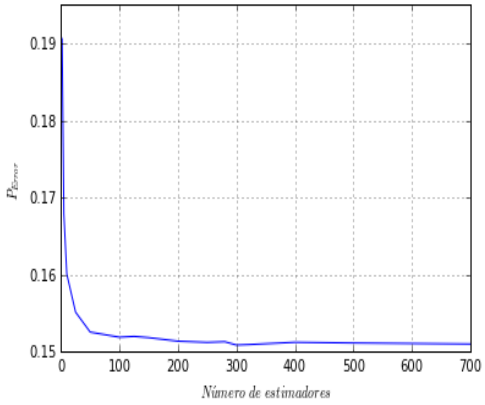


Figura 3.2. Ramdon Forest: Variación del error con respecto al número de estimadores.

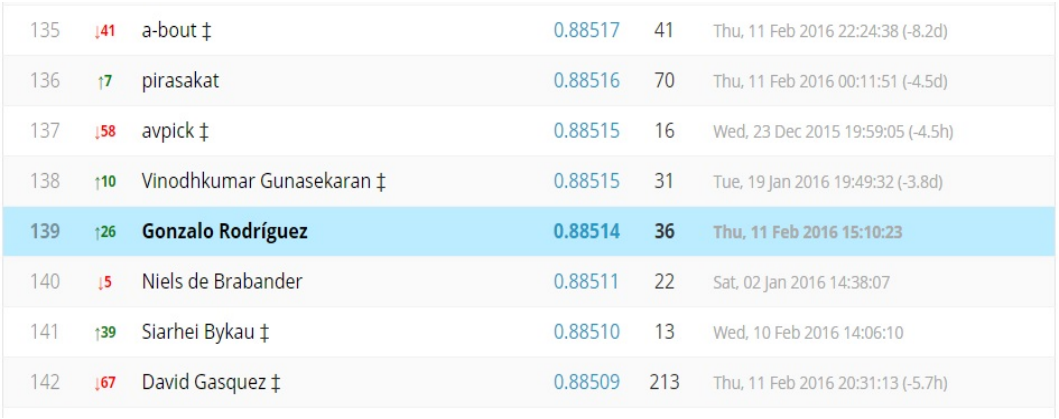
El modelo de conjuntos *Gradient boosting* es el modelo que mejores resultados ha obtenido. A pesar de sus buenos resultados, presenta una serie de desventajas que hacen que este clasificador en algunos trabajos no sea la mejor opción:

1. Entrenar los n estimadores es bastante costoso computacionalmente ya que lo hace secuencialmente. En este TFM el tiempo de ejecución ha sido aproximadamente de 1 hora y 17 minutos.

2. Se requiere de conocimiento previo para sacar el máximo rendimiento a este clasificador. Es un clasificador difícil de entrenar ya que tiene más hiperparámetros que ajustar que otros modelos: *max_depth*, *learning_rate* y *n_stimators*.

3. Es propenso al sobreajuste.

Debido a que uno de los objetivos secundarios de este TFM es clasificar lo mejor posible dentro de la competición de Kaggle, es obligatorio utilizar el modelo que mejor puntuación presenta. Para alguien nuevo en la ciencia de los datos, es difícil mejorar con un modelo tan lento ya que la experiencia se consigue a base de prueba y error. Por eso, se ha utilizado el paquete *Extremely Gradient Boosting*, [26] que es una versión escalable y eficiente de la implementación *Gradient Boosting*. Con el uso de este paquete se consiguió resolver los problemas de ineficiencia presentes en el modelo *Gradient Boosting* ya que la ejecución paso a ser de unos 5 minutos y se consiguió clasificar dentro del 10 % de los competidores (posición 139 de 1.462) con un private score de 0,88514.



135	↓41	a-bout ‡	0.88517	41	Thu, 11 Feb 2016 22:24:38 (-8.2d)
136	↑7	pirasakat	0.88516	70	Thu, 11 Feb 2016 00:11:51 (-4.5d)
137	↓58	avpick ‡	0.88515	16	Wed, 23 Dec 2015 19:59:05 (-4.5h)
138	↑10	Vinodhkumar Gunasekaran ‡	0.88515	31	Tue, 19 Jan 2016 19:49:32 (-3.8d)
139	**↑26**	**Gonzalo Rodríguez**	**0.88514**	**36**	**Thu, 11 Feb 2016 15:10:23**
140	↓5	Niels de Brabander	0.88511	22	Sat, 02 Jan 2016 14:38:07
141	↑39	Siarhei Bykau ‡	0.88510	13	Wed, 10 Feb 2016 14:06:10
142	↓67	David Gasquez ‡	0.88509	213	Thu, 11 Feb 2016 20:31:13 (-5.7h)

Figura 3.3. Captura de pantalla de la Private Leaderboard de Kaggle con mi posición final.

Capítulo 4
Conclusiones y líneas futuras

4.1. Conclusiones

Tras haber participado en la competición Kaggle: Airbnb y haber experimentado con los distintos *dataset* y los distintos clasificadores para mejorar el modelo predictivo, se ha llegado a las siguientes conclusiones:

- Las fases más importantes y que más influyen en la obtención de unos mejores resultados son el preprocesado de los datos y la ingeniería de las características. El saber entender los datos y a partir de ese entendimiento construir características útiles, es lo que realmente diferencia a los mejores científicos de los datos.
- Disponer de una métrica bien implementada y un conjunto de validación es clave para llegar lejos en la competición. Al disponer de un método de evaluación alternativo al que ofrece Kaggle, mediante su tabla de clasificación pública, se evita sobreajustar el modelo y ayuda a mejorar nuestro modelo predictivo en un corto periodo de tiempo (Kaggle a lo sumo permitía evaluar 5 predicciones al día). Con esta metodología se consiguió avanzar 26 puestos en la clasificación final, la que viene dada por la clasificación privada, con respecto a la clasificación pública y se consiguió obtener la puntuación obtenida unicamente en 36 submit. Algo a destacar ya que era el primer problema real que se afrontaba y entre los objetivos de este trabajo no estaba contemplado tratar el sobreajuste.
- La optimización de los hiperparámetros de los distintos clasificadores tiene una gran influencia en el resultado final. Aunque la librería `Scikit-learn` posee un método, llamado *GridSearchCV*, para hallar la mejor configuración de parámetros de forma óptima, no se ha utilizado ya que el único método de validación que utiliza (validación cruzada) no es el que se ha utilizado en este TFM. Por ello, la obtención de la mejor configuración se ha hallado de forma manual.
- Entre todos los modelos utilizados, los modelos basados en *ensemble methods* son los que mejor funcionan. Aunque la diferencia entre los resultados obtenidos por los distintos métodos de conjuntos es pequeña, el modelo *Gradient Boosting* es el que con más precisión predice el lugar de destino de los nuevos usuarios de Airbnb, por tanto, es el que más se acerca a nuestros objetivos perseguidos.

- Es difícil trabajar con el modelo *Gradient Boosting* ya que es un modelo ineficiente computacionalmente, es poco robusto frente al sobreajuste y posee un gran número de hiperparámetros a configurar. En consecuencia, durante la competición se trabajó con el paquete *Extremely Gradient boosting* que soluciona gran parte de estas deficiencias. Con el uso de este paquete se ha conseguido un puntuación final de 0,88514, posición 139 de 1.462, con una diferencia de nDCG con respecto al primer clasificado de 0,00183.

- Sin la ayuda Kaggle y su comunidad, no se hubiera llegado tan lejos. Kaggle es quizá la mejor herramienta para iniciarse en el mundo del análisis de datos ya que dispone de *dataset* para todo tipos de problemas reales que sirven para poner en práctica nuestro conocimientos adquiridos durante los estudios y seguir aprendiendo. Además, cada competición dispone de un foro que es una gran fuente de conocimiento en el que los competidores son muy participativos y se tratan dudas generales bastante interesantes.

4.2. Líneas futuras

Este trabajo se ha centrado en realizar un análisis predictivo para poner en práctica los conocimientos que se había adquirido en el Máster y en cursos online en el área de la ciencia de datos. De ahí que el objetivo principal fuera abordar un problema real haciendo hincapié en las distintas fases del modelado predictivo. Por lo tanto, hay muchas formas de mejorar los resultados obtenidos y de seguir aprendiendo.

Para un futuro, se podría mejorar el modelo predictivo implementado, mediante: La utilización de todos los datos de la competición, el tratamiento de datos desbalanceados y del sobreajuste, la realización de un mejor pre-procesado y de una mejor ingeniería de datos, la utilización de modelos más complejos en los que se aplican técnicas de conjuntos para aumentar aún más la precisión [27] . O se podría ampliar nuestro conocimiento realizando otro tipo de problemas de la ciencia de los datos, por ejemplo, problemas de clasificación de imágenes.

Referencias

[1] Kaggle, *Kaggle web.* Vhtisto el día 15 de Enero de 2017, de https://www.kaggle.com

[2] Kaggle, *Kaggle' Forums: Competition restart.* Visto el día 29 de Enero de 2017, de https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/forums/t/17737/announcement-competition-will-restart-shortly

[3] Pedregosa et al., *Scikit-learn Tutorial: 4.3 Preprocessing data.* Release 0.18 documentation, http://scikit-learn.org/stable/modules/preprocessing.html

[4] Kaggle, *Información de la competición: Evaluación del modelo.* Visto el día 11 de Febrero de 2016, de www.kaggle.com/c/airbnb-recruiting-new-user-bookings/details/evaluation

[5] James, G., Witten, D., Hastie, T., Tibshirani, R., *An Introduction to Statistical Learning: with Applications in R.* ISBN:1461471370 9781461471370, pp. 130-138, Springer, 2014.

[6] Alonso Atienza, Felipe, *Tema 6: Clasificación.* Máster en Ingenería de Telecomunicación: Tratamiento y gestión de información multimedia (URJC), Abril de 2014.

[7] Pedregosa et al., *Scikit-learn Methods: Logistic Regression.* Visto el día 5 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[8] Github user TomDLT., *Github Issue: is random_state in LogisticRegression useless?.* Visto el día 5 de Noviembre de 2016, de https://github.com/scikit-learn/scikit-learn/issues/4760

[9] Zhang, H., *The optimality of Naive Bayes.* In FLAIRS Conference (Journal AA, 2004).

[10] Pedregosa et al., *Scikit-learn Tutorial: 1.9. Naive Bayes.* Release 0.18 documentation, http://scikit-learn.org/stable/modules/naive_bayes.html

[11] Pedregosa et al., *Scikit-learn Methods: Naive Bayes.* Visto el día 5 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[12] Thirumuruganathan, S. (2010, 17 de Mayo), *A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm.* Visto el día 11 Marzo de 2016, de https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

[13] Pedregosa et al., *Scikit-learn Methods: K Nearest Neighbors.* Visto el día 5 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[14] Hackeling, G., *Mastering Machine Learning with Scikit-Learn.* ISBN: 978-1-78398-836-5, pp. 97-111, 1 de enero, 2014.

[15] Hackeling, G., *Mastering Machine Learning with Scikit-Learn.* ISBN: 978-1-78398-836-5, pp. 112, 1 de enero, 2014.

[16] Breiman, L., *Random forests.* Machine Learning, 45(1):5-32, 2001. doi: 10.1023/A:1010933404324.

[17] Dietterich, Thomas G., *Ensemble Methods in Machine Learning.* In J. Kittler and F. Roli, editors, Multiple Classifier Systems, pages 1-15. LNCS Vol. 1857, Springer, 2001.

[18] Pedregosa et al., *Scikit-learn Methods: Random Forest.* Visto el día 12 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[19] Pedregosa et al., *Scikit-learn Ensemble Methods: 1.11.2.3. Parameters.* Visto el día 12 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/ensemble.html

[20] Geurts, P., Ernst, D., Wehenkel, L., *Extremely randomized trees.* Machine Learning Journal (advance access: DOI 10.1007s10994-006-6226-1), pp. 1-40, 2006.

[21] Pedregosa et al., *Scikit-learn Methods: Extremely Randomized Tree.* Visto el día 12 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[22] Pedregosa et al., *Scikit-learn Tutorial: 1.11. Ensembled methods.* Release 0.18 documentation, http://scikit-learn.org/stable/modules/ensemble.html

[23] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning.* Second Edition, pages 337-388, Springer, 2001.

[24] Pedregosa et al., *Scikit-learn Methods: Gradient Boosting.* Visto el día 12 de Noviembre de 2016, de http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[25] Kaggle, *Kaggle Member FAQ.* Visto el día 20 de Noviembre de 2016, de https://www.kaggle.com/wiki/KaggleMemberFAQ

[26] XGB OFFICIAL WEBSIDE, *Extremely Gradient Boosting model.* Visto el día 30 de Diciembre de 2016, de http://xgboost.readthedocs.io/en/latest/model.html

[27] MLWAVE WEBSIDE, *Kaggle ensembling guide.* http://mlwave.com/kaggle-ensembling-guide/, 30 de Diciembre de 2016.