TRABAJO FIN DE GRADO

Competición de Kaggle
Airbnb New User Bookings

GRADO EN INGENIERÍA INFORMÁTICA

Autor
David Gasquez Arcos

Director
Francisco Herrera Triguero

Abstract

This thesis examines a solution to the Airbnb machine learning problem proposed in the Kaggle Data Science platform. The goal is to predict in which country a new user will make his or her first booking. The scope of the research presented here is the application of Boosting algorithms to predict where the new user will book, taking into account the performance of such prediction as it need to be fast. The project relies in Python programming language and uses diverse Open Source libraries such as Scikit-Learn and Pandas.

Abstract

Este trabajo resume la solución propuesta al problema de Machine Learning de Airbnb propuesto en la plataforma Kaggle. El objetivo es predecir el país donde un nuevo usuario va a realizar su primera reserva. Para realizar dichas predicciones se han usado algoritmos de *Boosting* y diversas formas de preprocesamiento. El código está mayormente escrito en Python, usando las diversas librerías de código abierto de su entorno como Scikit-Learn o Pandas.

Yo, **David Gasquez Arcos**, alumno de la titulación GRADO EN INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76421093M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: David Gasquez Arcos

Granada a 11 de Julio de 2016.

D. **Francisco Herrera Triguero**, Profesor del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Competición de Kaggle, Airbnb New User Bookings***, ha sido realizado bajo su supervisión por **Francisco Herrera Triguero**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 11 de Julio de 2016.

El director:

Francisco Herrera Triguero

Dedicado a las personas que están mejorando el mundo invirtiendo su tiempo en hacer y compartir abiertamente proyectos con la esperanza de contribuir a un mejor futuro.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

«You can have data without information, but you cannot have information without data.» **Daniel Keys Moran**

Actualmente vivimos en un mundo donde los negocios sienten la necesidad creciente de realizar predicciones sobre las consecuencias que puede tener una decisión arbitraria en un momento dado. Por ello, realizar predicciones certeras, proporciona a una empresa la capacidad de explorar más opciones y predecir qué resultado obtendrán antes de tomar cualquier decisión. Dada esta necesidad del mundo actual, construir modelos, técnicas y algoritmos para realizar dichas predicciones se convierte en un reto muy interesante. Este trabajo pretende abordar dicho problema.

El desarrollo se centrará, principalmente, en el ámbito de la *Ciencia de Datos* y el *Aprendizaje Automático*. Más concretamente, puede decirse que se ocupa del campo de la resolución de problemas de clasificación con múltiples clases, todas ellas diferentes entre sí.

A continuación, me gustaría presentar el proyecto elegido, realizar una breve descripción de sus objetivos, características y posibilidades de resolución, así como presentar la introducción del problema y ver una introducción al Aprendizaje Automático (en concreto al aprendizaje supervisado para clasificación) antes de desglosar la solución proporcionada y comentar el enfoque.

1.1. Definición del Problema

Para la demostración de los conocimiento adquiridos, se ha seleccionado el problema *Airbnb New User Bookings* [1]que se encontró activo en la plataforma web *Kaggle* durante 4 meses; desde el día 25 de Noviembre hasta el día 11 de Febrero de 2016.

El problema fue planteado por la empresa americana **Airbnb**[1]. Cuyo objetivo era predecir correctamente a qué país se dirigirían los nuevos usuarios al realizar su primera reserva de alojamiento.

En Airbnb, los usuarios pueden reservar alojamiento en más de 34.000 ciudades repartidas en más de 190 países. Realizando una predicción correcta del país donde los nuevos usuarios harán su primera reserva, la empresa obtiene diversas ventajas, entre las cuales cabe destacar dos con un gran impacto:

- Ofrecen a sus usuarios contenido más personalizado, incrementando así la posibilidad de que estos realicen una reserva.
- Pueden conocer que ciudades estarán más demandadas y en qué fechas, proporcionándole la capacidad de estimar y establecer precios.

[1] `https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings`

Ésta es una competición de *reclutamiento*, donde las mejores soluciones propuestas serán revisadas por trabajadores de la empresa, pertenecientes al equipo interno de Data Science. Tal cómo se publicita en la web del reto, los mejores participantes serán entrevistados para un posible trabajo en Airbnb. El objetivo de este trabajo fin de grado es tratar de conseguir el mejor puesto posible dentro de la tabla final de clasificación para poder optar a una entrevista.

El problema está bien definido y posee un método de evaluación establecido a partir de una métrica concreta con la que evaluar las soluciones aportadas. Para la clasificación final, se evaluarán un serie de datos pertenecientes a usuarios completamente nuevos para los participantes.

Tras la lectura de esta memoria, se podrá comprobar que los objetivos iniciales han sido ampliamente superados, obteniendo una buena posición en la competición. Además, gracias a este trabajo, han aumentado mis posibilidades dentro del mundo laboral.

Antes de empezar a comentar el problema con más detalle, vamos a conocer un poco más sobre las empresas que han hecho posible el desarrollo de este trabajo.

1.1.1. Kaggle

Kaggle es una plataforma online que ofrece a sus usuarios la opción de participar en distintas competiciones a nivel mundial, planteando problemas sobre análisis de gran cantidad de datos. Aquella persona o equipo que obtenga el mejor resultado puede recibir un premio en metálico, una oportunidad de trabajo para alguna empresa o el reconocimiento de la comunidad por su esfuerzo e innovación.

Cualquiera puede registrarse e intentar resolver retos muy variados. Esta sencilla idea ha involucrado cerca de 200.000 usuarios registrados, convirtiéndose en la comunidad de científicos de datos más grande del mundo. Los usuarios pertenecen a todo tipo de disciplinas profesionales, lo cual enriquece las soluciones y permite resolver problemas de diferente índole. Muchas empresas lo usan como laboratorio para resolver problemas que tengan con sus datos. La variedad de participantes hace que las empresas puedan evaluar las diversas soluciones propuestas, entra las que escogerán un ganador en cada competición. Este mérito proporciona actualmente cierto prestigio en el mundo del análisis de datos y, como se ha dicho anteriormente, las empresas también ofrecen un premio en metálico que en algunas ocasiones puede ser bastante suculento.

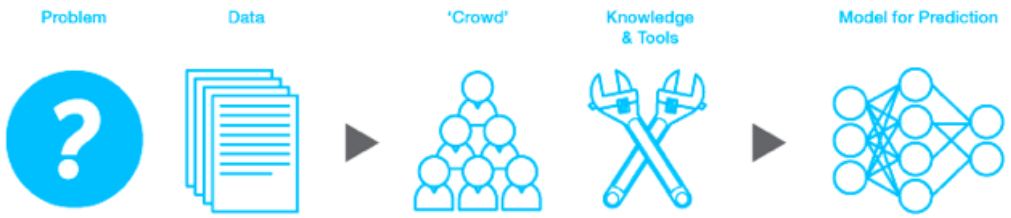

Figura 1.1: Diagrama de funcionamiento de Kaggle

1.1.2. Airbnb

Expuesto de manera simple, Airbnb es una aplicación web donde los usuarios pueden reservar alojamiento en cualquier parte del mundo. El funcionamiento es similar a las páginas de reserva para hoteles, donde el usuario elige el alojamiento y selecciona las fechas de entrada y salida. La principal diferencia radica en que las propiedades que van a reservar los usuarios no están gestionadas por profesionales, sino por otros usuarios.

Figura 1.2: Banner de Airbnb

Actualmente no existe coste para registrar un alojamiento, por lo que la variedad de opciones es inmensa. Normalmente, el anuncio cuenta con diversas fotos de la casa o apartamento, para que el futuro cliente tenga una idea de la calidad de éste.Uno de los aspectos positivos de este método de reserva es el precio, generalmente bastante más económico que un hotel, por lo que, en los últimos años, su uso está creciendo exponencialmente.

1.1.3. Motivación Personal

Mi interés en el tema de Análisis de Datos surgió una vez cursada la asignatura de introducción al mundo de la Ciencia de Datos, **Aprendizaje Automático**. Me impactó la forma tan original de resolver problemas donde no hace falta programar la respuesta del programa, ya que éste la aprende a base de enseñarle ejemplos.

Por ello decidí aplicar de forma práctica los conocimientos adquiridos en ésta y otras asignaturas relacionadas de la carrera. Podría haber elegido alguno de los problemas ya resueltos en alguna base de datos pública, pero opté por escoger un problema asociado a una competición donde no había resultados anteriores ni métodos preestablecidos para ver cuán lejos podía llegar con mis conocimientos actuales y los que desarrollaría durante el proceso.

1.1.4. Motivación de la Empresa

Como he mencionado anteriormente, hoy en día es común que las empresas registren una competición en la plataforma Kaggle para obtener ideas y soluciones a problemas a los que se enfrenten día a día o se puedan enfrentar en un futuro.

El equipo de Análisis de datos de Airbnb necesitaba saber donde iban a registrarse por primera vez los usuarios y decidió crear la competición. Estas son algunas de las ventajas que les proporciona conocer ese dato de forma precisa:

- Mejoran la calidad de las recomendaciones a otros usuarios que tengan el mismo tipo de perfil.
- Pueden mostrar sugerencias afines a cada usuario para maximizar la posibilidad de que éste reserve un alojamiento.
- Conocen en que estado se encuentra cada país y alteran los precios según requieran con un tiempo de antelación.
- Disminuyen el tiempo que los usuarios pasan buscando destino ya que les proporcionan su posible destino ideal con antelación.

Dada la cantidad de usuarios que usan esta web para reservar sus vacaciones, todo lo que esté relacionado con incrementar el número de reservas por usuario es de vital importancia para la compañía. Por muy pequeño que sea el porcentaje de gente que se incremente al aplicar algún tipo de solución usando el análisis de datos y técnicas relacionadas, tendrá un impacto muy importante en las ganancias finales de la empresa. Por eso es, a mi parecer, es una buena idea ofrecer este tipo de reto a la comunidad de científicos de datos, ya que se benefician todos los implicados.

1.2. Objetivos

1.2.1. Objetivos Personales

Cómo objetivo personal, me planteo la meta de obtener el mejor resultado posible en la clasificación final de la competición, intentando quedar entre el 10 % de los mejores competidores clasificados. Mejores resultados me otorgarán más posibilidades de obtener una entrevista en la empresa Airbnb tras la competición y un mayor reconocimiento a nivel profesional.

1.2.2. Objetivos Específicos de la Competición

El objetivo principal de esta competición es predecir con la mayor precisión posible el país que visitará un usuario que tras registrarse realiza su primera reserva. Mas concretamente, por cada usuario que nos proporciona Airbnb, podemos dar una lista de 5 países ordenados por probabilidad de visita. Por supuesto, el orden en el que se proporciona la lista es importante para el resultado y evaluación de las soluciones, como veremos posteriormente en el apartado relacionado.

Se parte de 11 países en los que un usuario podría, potencialmente, realizar una reserva. Dejando esto un problema de 12 clases distintas. La clase que completa las 11 se usa para albergar los usuarios que tras registrarse no hacen ninguna reserva.

Evaluación

La métrica de evaluación de esta competición ha sido Normalized Discounted Cumulative Gain (**NDCG**). Se suele usar para medir la calidad de las recomendaciones y es bastante acertada para este caso. NDCG se puede calcular a partir de la métrica Discounted Cumulative Gain (DCG):

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

Donde rel_i indica la relevancia del resultado en la posición i. Normalizamos usando $IDCG_k$ como el mejor valor posible (ideal) para DCG dada una serie de valores a evaluar.

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

Todos los valores que devuelva esta función deben de estar en el intervalo $[0{,}0, 1{,}0]$. Por cada usuario es posible enviar hasta un máximo de 5 predicciones para el país donde va a realizar la primera reserva. El país verdadero se marca con una relevancia de 1 y el resto con relevancia 0.

Por ejemplo, si un usuario hace su primer reserva en Francia y nosotros damos como respuesta un vector de un solo elemento, `[Francia]`, obtendríamos:

$$NDCG = \frac{2^1 - 1}{log_2(1 + 1)} = 1{,}0$$

Si, en otro caso, hubiéramos dado un vector, `[Estados Unidos, Francia]` el cálculo sería:

$$DCG = \frac{2^0 - 1}{log_2(1+1)} + \frac{2^1 - 1}{log_2(2+1)} = \frac{1}{1{,}58496} = 0{,}6309$$

Como se puede observar, lo óptimo entonces será siempre dar un vector con 5 países para cada usuario. Aunque el verdadero país no esté el primero obtendríamos algo de puntuación por el.

1.3. Ámbito de la Competición

Ya que esta es una competición pública, la empresa Airbnb no proporciona la totalidad de los datos de sus nuevos usuarios. En su lugar, se proporcionan datos de un porcentaje de usuarios aleatorios con identificadores en lugar de sus nombres para proporcionar confidencialidad.

Los datos con los que trabajamos son únicamente de ciudadanos estadounidenses que estuvieron activos después del 7/1/2014. Los destinos posibles en los que pueden reservar una habitación, casa o apartamento son: Francia, Canadá, Gran Bretaña, España, Italia, Portugal, Holanda, Alemania y Australia.

También, tal como se ha introducido previamente, cabe la posibilidad de que se queden en el mismo país (Estados Unidos) o que no hagan ninguna reserva. En este caso, los usuarios no tienen ningún destino y se les asigna la clase **NDF**(*No Destination Found*).

Como última posibilidad se contempla que el usuario no acuda a ninguno de estos países, en este caso no tendríamos que predecir el país concreto, sino especificar que ha decidido reservar en *otro* país.

Capítulo 2

Ciencia de Datos: Descripción y Herramientas

> *« A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E »*
>
> **Tom Mitchell**

Antes de empezar a describir el proceso seguido durante la competición, es interesante dar un repaso a los principales algoritmos y técnicas que se suelen emplear en este campo. Así lograremos obtener una visión más amplia del mismo y de su estado actual.

Existen tal cantidad de algoritmos que puede ser abrumador empezar a abordar cualquier problema de este tipo. Más aún cuando se tiene una cantidad inmensa de datos y no son interpretables como ocurre en muchos casos. En teoría, un buen científico de datos es capaz de clasificar estas técnicas e intuir cuál de ellas debe aplicar a cada problema. Normalmente eso no es suficiente; se necesita explorar el espacio de soluciones de un problema abordándolo de diferentes formas y con distintas técnicas. Así, además de estar más seguros de nuestra solución, tenemos una idea de como se comporta el problema con los diferentes tipos de técnicas.

En la figura 2.1 se puede ver un diagrama de flujo orientativo para orientar la decisión de la posible técnica a emplear. Ha sido creado por miembros de la comunidad científica que colaboran en la librería de código abierto Scikit-Learn [2].

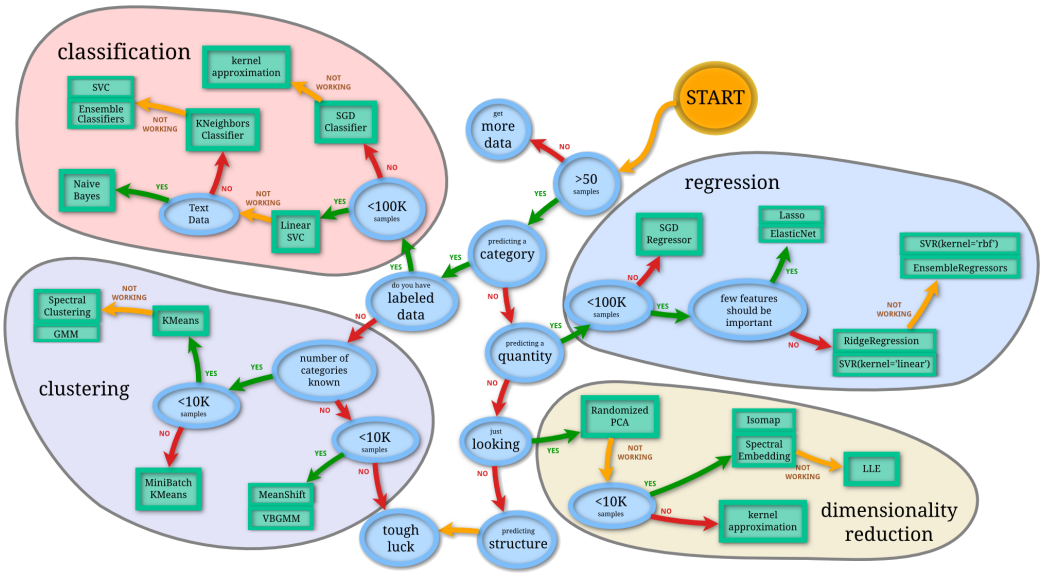

Figura 2.1: Diagrama de flujo para la decisión de técnicas de Machine Learning.

Una vez que se ha elegido un buen algoritmo y/o técnica, entran en juego una serie de factores que alteran el resultado final, generalmente con menor peso que la elección del algoritmo, aunque no en todos los casos. En general lo más importante en este tipo de problemas es la calidad y cantidad de los datos que se tienen y que se pueden obtener para enriquecerlos.

Para agrupar estas técnicas de forma simple, se suele hacer una clasificación dependiendo de como usan los datos y lo que aprenden (clases existentes o nuevas). Otra forma en la que se pueden clasificar es por las similitudes en forma o en funcionamiento, más o menos como las taxonomías en los animales.

2.1. Aprendizaje Automático

Podemos definir el aprendizaje automático como la aplicación de modelos estadísticos y otros algoritmos a conjuntos de datos con el objetivo de conocer algo más sobre ellos; predecir una respuesta, estimar una valor, conocer al grupo al que pertenecen, ...

¿Qué tipo de planta es ésta si el pétalo tiene de ancho 4 centímetros y el sépalo 1 centímetro? ¿Que número es el que está fotografiado en esta imagen? ¿Donde van los usuarios de mi empresa a realizar su primera reserva?

Dado el carácter general de las preguntas que se pueden hacer, vamos a centrarnos en las relativas a la clasificación. La clasificación es muy importante también en las empresas para poder mejorar sus ingresos aprovechando los datos de los que disponen. Las empresas están buscando mejorar en este campo cuando ofrecen los datos al público.

Me gustaría destacar alguno de los ejemplos más notables de empresas que han ofrecido *datasets*[1] para contestar algunas de las preguntas que se hacían:

- **Netflix**: Proporcionó datos históricos de usuarios para que los competidores intentaran mejorar su sistema de recomendación de películas. El ganador creó un sistema que superaba con creces el inicial, pero Netflix nunca llegó a ponerlo en producción ya que era muy complicado de implementar de forma eficiente [2].
- **Pandora**: es un servicio de música similar a Spotify que realizó algo similar usando el *feedback* de los usuarios para presentar a estos nuevas canciones basadas en las que le habían gustado anteriormente.
- **Tesla**: Buscaba cómo mejorar el sistema de reconocimiento de objetos de sus coches para facilitar la conducción automática. El sistema que ganó también era bastante más certero que el inicial, y en este caso, de mayor eficiencia.

Es interesante mencionar que el aprendizaje automático tiende a enfocarse más en la predicción que en la interpretación de los datos, como consecuencia, los modelos estadísticos suelen interpretar mejor lo que está ocurriendo con ellos. Normalmente estos modelos proporcionan coeficientes que explican las causas y el grado en el que se ven afectadas por ellas.

[1] Conjunto de datos normalmente estructurado de tal forma que exista una muestra por fila y cada columna sea un variable.

[2] `http://www.netflixprize.com/`

El problema radica en la inmensa cantidad de datos que recogen y se miden hoy en día gracias las nuevas tecnologías. Los *datasets* que se están usando en los últimos años han crecido de manera exponencial con el tiempo. Ahora, es normal manejar varios millones de muestras para un problema común. Esto hace que las técnicas estadísticas pierdan eficacia y aquí es donde entran en juego las técnicas del Aprendizaje Automático o Machine Learning.

Como consecuencia del incremento de la cantidad de datos, también se está viendo un aumento la necesidad de procesamiento de los ordenadores, es más, en la actualidad, existen hasta problemas donde no es suficiente con usar las unidades de computación (CPU) de los ordenadores sino que hay que añadir capacidad de procesamiento extra usando GPUs[3].

A pesar de la diversidad existente en esta materia, podríamos decir que la mayoría de los algoritmos de Machine Learning comparten 3 componentes básicos:

1. Una representación matemática del problema para que una máquina pueda entenderlo y aplicar operaciones sobre dicha representación de los datos.
2. Una o varias funciones de evaluación para saber cómo de buena es una solución dada a nuestro problema y en que grado afectan las modificaciones que hagamos durante el proceso al resultado de la evaluación.
3. Una serie de parámetros y pasos de procesamiento que permiten optimizarlo y mejorar el resultado de la anterior función de evaluación.

2.2. Manejo de los Datos

Una de las fases más importantes, y también la que permite una mejora considerable en el resultado final, es en la que se manejan los datos del problema.

Como sabemos o podemos imaginar, la mayoría de las veces los datos no vienen en un formato que una máquina pueda digerir fácilmente. Por ejemplo, muestras que estén en forma de imagen o audio. Aquí entra en juego la capacidad de innovar para transformar los datos, siempre intentando que no se pierda información. Esta información, también llamada señal de los datos, es la que hay que intentar capturar y extraer de los problemas que se quieran resolver.

La meta final del Aprendizaje Automático, y de cualquier técnica que deriva de él, es la **generalización**. Hay que intentar generalizar más allá del conjunto de datos inicial para poder entender el resto del problema. Una vez entendido será más fácil realizar predicciones, clasificar o agrupar dadas nuevas instancias. [4]

Tenemos que tener en cuenta que usando como objetivo la **generalización**, se pierde la posibilidad de acceder a una función global definida que optimizarla (matemáticamente), ya que existen infinitas formas de evaluar con que calidad se está generalizando. Lo que se suele hacer con estos datos es usar una parte de ellos para validar con una función establecida si estamos generalizando lo suficientemente bien. Se dará más detalles de este proceso posteriormente en la sección 2.3.

En el momento que empezamos a alterar los datos, corremos el riesgo de perder parte de la *señal* que contienen y esto volverá más difícil el problema en las etapas finales. Normalmente, hay que buscar un compromiso entre la cantidad de alteraciones que se le aplican y la mejora que se obtiene.

Claro está, el estado de los datos que tenemos es de vital importancia a la hora de manejarlos. Es muy común encontrarse, por ejemplo, fechas incorrectas, números con diferentes precisiones o países que no existen, esto ocurre por que al introducirlos en el ordenador se ha cometido un error tipográfico. Es muy importante conocer al máximo posible el ámbito y los datos con los que se está trabajando para corregir los errores que puedan tener y que estos no afecten al resultado.

Ya que lo que queremos es generalizar al máximo posible, nunca podremos decir que tenemos suficientes datos. Con un mayor numero de datos se podrán ofrecer generalizaciones mas precisas.

Por supuesto, un conocimiento previo del ámbito donde se encuentra el problema, puede ayudarnos a aplicar reglas o extraer mejor ciertas características de los datos. Pero se debe tener en mente que a mayor cantidad de datos que se tengan, mejor generalización se podrá hacer del conjunto al que pertenecen.

Un algoritmo simple con muchos datos suele dar mejores resultados que una técnica compleja con menor cantidad de datos. Aunque, más datos también implican más problemas de escalabilidad. [5]

El manejo de los datos no es una fase específica en sí y está presente en todo el problema, antes de comenzar, durante el desarrollo del mismo e incluso cuando se tienen las soluciones. Dicho proceso podemos subdividirlo en diferentes etapas, las cuales desglosaremos a continuación.

2.2.1. Preprocesamiento

Una vez que se disponga de los datos (en forma de muestras), independientemente del formato en el que se presenten, es necesario transformarlos al dominio de los ordenadores, es decir, tenemos que codificar todas las variables de manera específica (normalmente en forma de números).

Posteriormente tendremos que arreglar aquellas que no sean correctas, intentar añadir nuevas a partir de las ya existentes o eliminar algunas que no proporcionen información.

La clave de un buen resultado radica en un buen preprocesamiento. A esta fase también se le suele llamar *Feature Engineering* y es clave en casi todos los problemas que se pueden abordar con Machine Learning. A más se conozca previamente del ámbito en el que se encuentra el problema, las características que se añadan o modifiquen del mismo serán de mayor calidad.

Aunque se aborda al principio, hay que remarcar que es también un proceso iterativo donde después de cada cambio hay que evaluar y actuar en consecuencia. En mi caso, el proceso iterativo se centra en entrenar el clasificador y calcular el error, para comprobar que el cambio que hemos hecho al principio no empeora la solución.

La fase de *Feature Engineering* es crucial en esta cadena, ya que obteniendo las características correctas (independientes y correladas con lo que se va a predecir) hace el proceso de aprendizaje más fácil y rápido. No es un proceso trivial, ya que suele requerir un extenso conocimiento del dominio del problema, el cual, normalmente, no se ha capturado al completo en los datos.

A continuación veremos un pequeño resumen de varias ideas que se suelen aplicar en los set de datos.

Creación de nuevas Variables

Uno de los factores más importantes que debemos de tener en cuenta, no es otro que la hecho de que los datos que tenemos del problema no definen el problema concreto.

Un experto puede sacar información extra de las variables y saber cómo se relacionan unas con otras para proporcionar una información extra. Por ejemplo, si tenemos datos de países y estamos haciendo un estudio económico, puede que nos sea útil incluir variables conocidas que no estén en el problema inicial, como puede ser una variable que nos indique si el país está en vías de desarrollo o no.

El ejemplo anterior es bastante simple, pero esta nueva variable puede afectar de forma positiva al resultado. Teniendo esta idea en mente no es difícil comprender por qué la mayoría de Científicos de Datos no son informáticos titulados, si no que estaban previamente especializados en otra materia.

En ámbitos como la biología y la química, donde se recogen millones de parámetros para una muestra, no suele darse esta creación de nuevas variables tanto como en ámbitos más relacionados con las ciencias sociales o más genéricos.

Estandarización

Otro factor importante a la hora de procesar nuestros datos es la *estandarización* de los mismos. Es común encontrarse,por ejemplo, con errores de formato en variables numéricas.

Todos estos pequeños detalles pueden hacer el proceso más tedioso y difícil, pero es recomendable solucionarlo lo más pronto posible. Entre estos problemas de estandarización podemos incluir los siguientes a modo de ejemplo:

- Distinta precisión en diferentes muestras: Por ejemplo, cuando se recolectan datos demográficos pero no se da la misma información de todas las muestras por motivos de confidencialidad.

- Errores de escala: Casi siempre suelen ocurrir cuando se hacen mediciones en diferentes sistemas, por ejemplo, metros y en centímetros.
- Distintos símbolos para la separación de decimales: A veces se usa el símbolo '.' y otras veces ',' por ejemplo.
- Distintos valores indicando lo mismo: *ES*, *Spain* y *España* para representar al concepto de España como país.
- Fechas escritas en diferentes formatos.

En la práctica encontraremos muchos más ejemplos, ya que estamos tratando con datos provenientes de muchas fuentes y esto suele causar problemas.

La estandarización no se limita solo a controlar que el rango esté dentro del dominio que aceptamos, sino que también se encarga de transformar ese rango a uno con más sentido, o centrar los datos para que sean uniformes.

Es común que muchos algoritmos asuman que la distribución de los datos es uniforme y que están centrados en el 0. Siempre se recomienda estandarizar y escalar los datos, ya que normalmente solo aporta beneficios.

A la hora de aplicar un escalado tenemos que asegurarnos de que estamos tratando correctamente los datos *extraños*, los llamados *outliers*, son muestras que se desvían de la media y varianza estándar de una característica. Estos datos pueden presentar casos especiales o bien solo un error tipográfico. Es importante saber como van a ser tratados y si hiciera falta, eliminarlos. [6]

Codificación

Las características que se miden pueden venir definidas en formas muy diferentes. Típicamente distinguimos entre variables *categóricas* y variables *continuas*, siendo posible desglosar aún más las variables categóricas entre *ordinales* y *nominales*, según lleven orden o no.

Actualmente, casi todos los algoritmos requieren que el problema esté expresado en números (normalmente enteros), por lo que tenemos que adaptar estas variables. Las técnicas más comunes, por orden de popularidad, para realizar una codificación de variables categóricas son:

1. **Ordinal**: A cada valor categórico se le asigna un número entero.
2. **One-Hot**: Se crea una columna con un valor booleano por cada categoría.
3. **Binario**: Se convierten las categorías a ordinales y posteriormente a binario. Por último el número en binario se divide en columnas. Con esto se consigue codificar los datos en menor dimensión aunque se distorsionan las distancias.

Selección de Variables

Previamente hemos hablado de la creación de variables en el apartado 2.2.1, y hay que añadir que también es importante saber seleccionar las correctas. En muchos problemas reales ocurre lo contrario y nos encontramos con un exceso de variables.

Para seleccionar las *features* más interesantes tenemos varios métodos a nuestra disposición. Por ejemplo, podemos seleccionar las que posean más varianza (ya que teóricamente contienen más señal). Para problemas más sofisticados podemos incluso recurrir a algoritmos iterativos como un simple *Greedy* o intentar aplicar algún tipo de algoritmo genéticos que modifiquen las variables seleccionadas en cada iteración.

Datos con errores

Existen multitud de enfoques a la hora de tratar con valores que son erróneos o simplemente no están, los cuales podemos dividir en dos grandes vertientes de acción: *eliminar* los datos o *arreglarlos*. Cada una tiene sus ventajas y desventajas.

Si se opta por la eliminación, idealmente tendríamos que disponer de una cantidad suficiente de datos en el set de entrenamientos para poder permitírnoslos. En este caso, aquellos valores erróneos o que no estén completos, no son usados y así obtener un *dataset* con valores limpios. Esto puede ser aplicado tanto a muestras individuales como a características completas de la muestra.

Si no podemos permitirnos eliminar aquellas muestras que no son correctas, podemos intentar arreglar esos errores si conocemos la forma de hacerlo. Tal como se ha dicho anteriormente, esto también es un proceso iterativo y podemos ver cómo afectan las distintas correcciones al resultado final.

Si no tenemos algunos datos también podemos aplicar técnicas para inventarlos. La forma más simple podría ser remplazar los valores que faltan por la media de la columna (variable) esta idea se puede refinar si para la media solo usamos los k vecinos más cercanos a la muestra. Existen técnicas más complejas para imputar valores vacíos(*NaN*), como por ejemplo, entrenando un clasificador con el resto de muestras y haciendo la predicción de los valores de aquellas muestras en las que no están.

2.2.2. Postprocesamiento

Aunque generalmente es menos importante que el preprocesamiento, en muchos casos también es necesario hacer algunas modificaciones en las respuestas de las técnicas aplicadas.

Hoy en día, algunas codificaciones de problemas son muy complejas y hay que transformar el resultado de nuevo al dominio del problema para que un humano pueda interpretarlo.

En otros casos más complejos, quizás sea necesario aplicar otro algoritmo a los datos de salida para refinar u obtener nuevas características.

2.3. Clasificación

Una de las tareas más comunes del Aprendizaje Automático, y a su vez la base de desarrollo de este trabajo, es la clasificación.

Dada una muestra, tenemos que decidir a que grupo pertenece basándonos en una serie de características disponibles. Esta no es una tarea fácil, ya que la mayoría de las veces no sabemos cuales son las características que discriminan a una muestra para determinarle una (o varias) clase específica. En la siguiente tabla 2.1, podemos ver uno de los conjuntos de datos más famosos a la hora de mostrar un ejemplo de clasificación, se trata de los datos de las flores de Iris. Es un conjunto de datos introducido por Ronald Fisher [7] que sirve como ejemplo de la clasificación multivariable:

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Virginica
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Versicolor
5.0	3.6	1.4	0.2	Versicolor
5.4	3.9	1.7	0.4	Virginica
4.6	3.4	1.4	0.3	Setosa

Tabla 2.1: Iris Dataset

Como se puede observar, existen 3 tipos de especies de Iris; *Setosa*, *Virginica* y *Versicolor*. Cada una con unas características que la hacen diferente del resto. Si la clasificación es nuestro objetivo, tendríamos que ser capaces de, conocidos la longitud y anchura del pétalo y el sépalo, hacer una predicción de que especie es la nueva muestra. Esto es lo mismo que decir que tenemos una forma de etiquetar con una clase cualquier instancia desconocida.

La elección del algoritmo específico que deberíamos usar es un paso crítico. Para realizar esta elección es recomendable iterar varias veces con diferentes tipos de algoritmos, técnicas y diferentes preprocesamientos. La evaluación más común en la clasificación que hace un algoritmo suele hacerse en función a la precisión del mismo. Existen al menos tres técnicas que se usan para que el cálculo de la función que valora el clasificador sea lo más real posible.

Una técnica consiste en dividir los datos que tenemos en 3 partes, usamos 2 para entrenar nuestro clasificador y la tercera solo para evaluar. Así la clasificación se está realizando sobre instancias nuevas. Otra técnica, conocida como Validación Cruzada, el set de datos se divide en K partes iguales y se repite el proceso anterior; se entrena el clasificador con $K-1$ partes y se predice sobre la que no se ha usado. Posteriormente, podemos calcular el error como la media de los errores en el conjunto de datos de test.

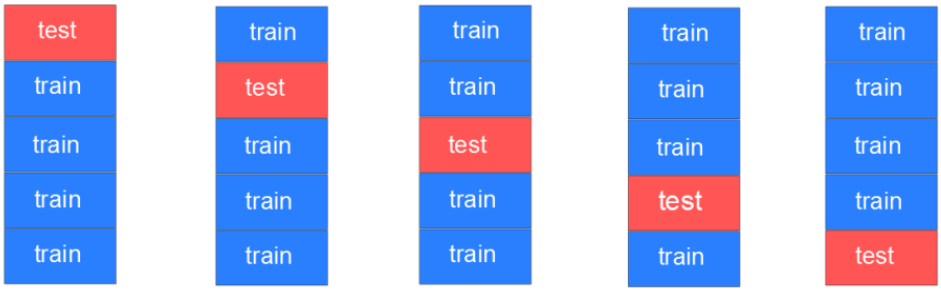

Figura 2.2: Diagrama de la Validación Cruzada con $k = 5$

Por último, si nos basamos en en esta idea, podemos dividir un conjunto de N instancias en N partes y dejar solo una fuera. A esto se le conoce como una validación Leave-One-Out y es un caso específico de Validación Cruzada. Como se puede intuir, este último caso es bastante pesado computacionalmente.

2.3.1. Métodos de Clasificación

Cuando en los problemas de clasificación contamos con una muestra de objetos ya clasificados, podemos hablar de problemas de aprendizaje supervisado. Dentro de estos, el problema de clasificación es uno de los más extendidos y para enfrentarse a este tipo de problemas existen una gran cantidad de técnicas: Técnicas estadísticas (k vecinos más cercanos, discriminadores Bayesianos, etc.), Árboles de Clasificación, Sistemas basados en Reglas, Redes Neuronales, Máquinas de Soporte Vectorial,..

Dentro de este proceso de aprendizaje, podemos distinguir (generalmente) dos etapas:

1. **Entrenamiento**: Donde se extraen las conclusiones apropiadas a partir de un conjunto de casos de entrenamiento y se debe obtener un modelo capaz de mostrar lo aprendido
2. **Prueba**: Centrada en utilizar el modelo obtenido en la fase de entrenamiento para realizar una clasificación futura, así como una estimación de la precisión del modelo, para lo cual se utiliza un conjunto de ejemplos distintos de los utilizados para la construcción del modelo (conjunto de prueba).

Árboles

Los árboles son una técnica de Machine Learning que prepara, sondea y explora los datos para sacar la información oculta en ellos. Es un algoritmo de clasificación clásico, y es muy utilizado por la fácil interpretación de sus resultados.

En la figura 2.3 podemos ver un ejemplo de árbol de decisión con el que una máquina podría decirnos si una persona en concreto es hombre o mujer conociendo su peso y altura. Es un ejemplo muy básico que a su vez muestra sus capacidades de interpretabilidad.

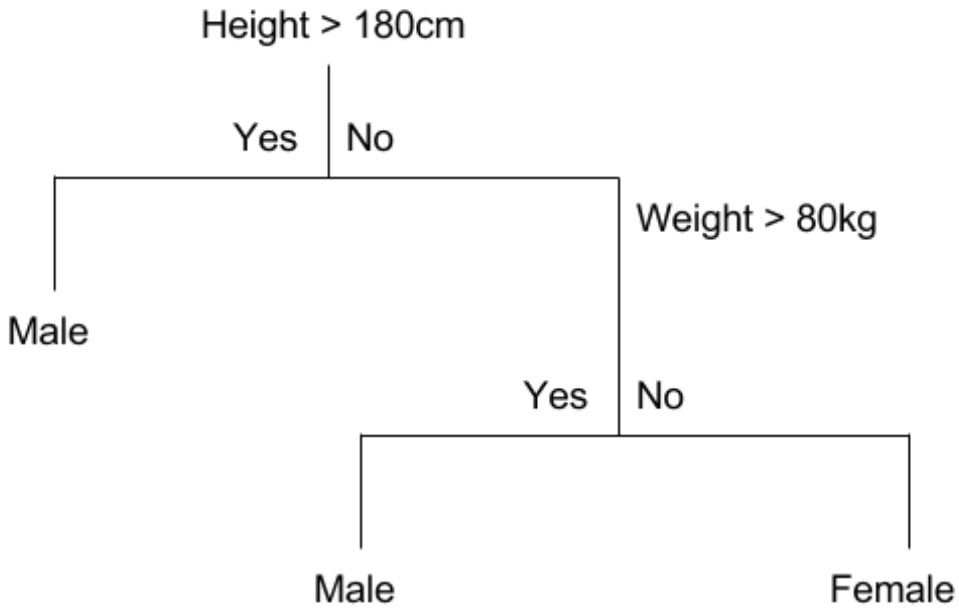

Figura 2.3: Ejemplo de árbol de decisión para decidir si una persona es un hombre o mujer.

Los árboles de decisión crean un modelo de clasificación basado en diagramas de flujo. Clasifican casos en grupos o pronostican valores de una variable dependiente (criterio) basada en valores de variables independientes (predictivas). Las ventajas de un árbol de decisión son:

- Facilita la interpretación de la decisión adoptada.

- Facilita la comprensión del conocimiento utilizado en la toma de decisiones.
- Explica el comportamiento respecto a una determinada decisión.
- Reduce el número de variables independientes.

La terminología asociada a la técnica de los árboles de decisión recurre a una terminología específica, por lo que consideramos interesante, antes de seguir adelante, clarificarla.

- **Nodo de decisión**: Nodo que indica que una decisión necesita tomarse en ese punto del proceso.
- **Nodo terminal**: Nodo en el que todos los casos tienen el mismo valor para la variable dependiente. Es un nodo homogéneo que no requiere ninguna división adicional, ya que es "puro".
- **Rama**: Nos muestra los distintos caminos que se pueden emprender cuando tomamos una decisión o bien ocurre algún evento aleatorio. Resultados de las posibles interacciones entre las alternativas de decisión y los eventos.

KNN

Los algoritmos basados en la regla del vecino más próximo son los más conocidos dentro de los clasificadores basados en instancias. Regla del vecino más próximo o *Nearest Neighbour* $(1NN)$.

Si tenemos m instancias en nuestra base de datos, para clasificar un nuevo ejemplo, se computará algún tipo de distancia con el resto de instancias y posteriormente tendremos que evaluar las clases de los n vecinos más cercanos. Normalmente la más común será la clase de nuestra instancia. Podemos ver un ejemplo gráfico en la figura 2.4.

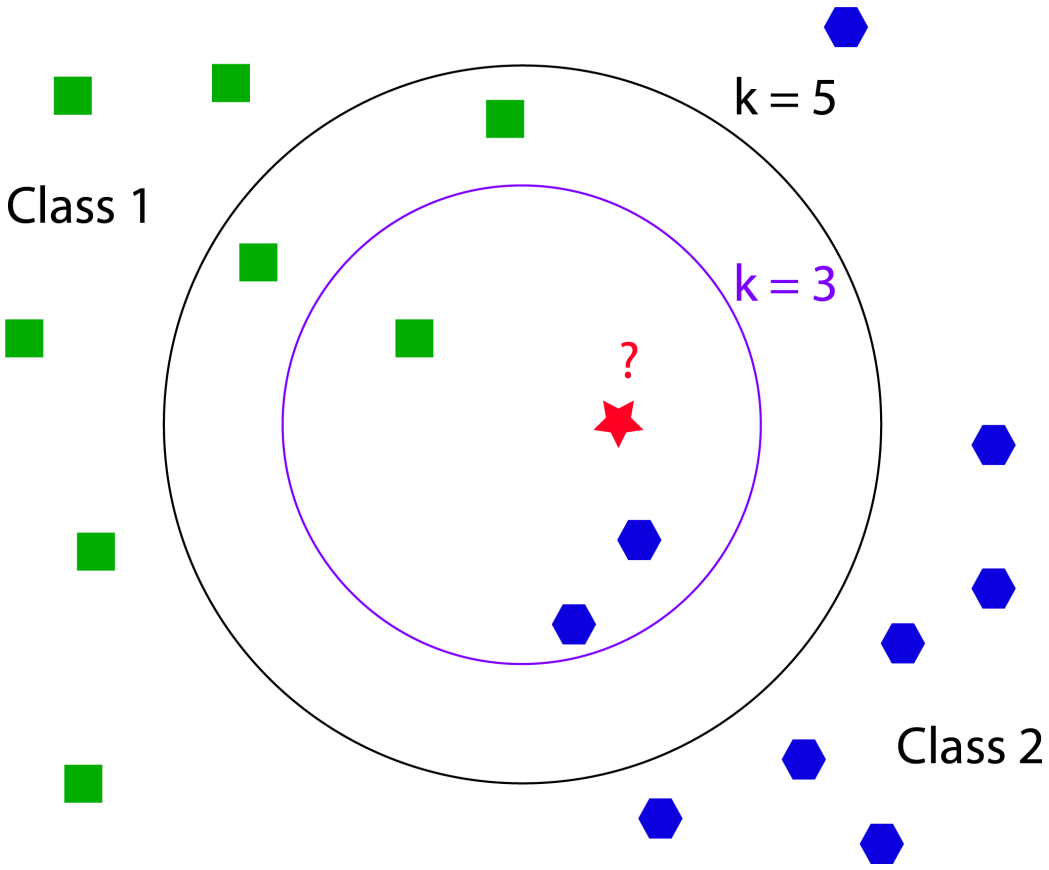

Figura 2.4: Ejemplo de una clasificación con KNN.

Debido a su simplicidad, el clasificador KNN es uno de los mas usados. El proceso de aprendizaje consiste en el almacenamiento de una tabla con los ejemplos disponibles y la clase asociada a cada uno de ellos. Cada vez que llega un nuevo ejemplo a clasificar, se calcula la distancia de dicho ejemplo a cada uno de los n ejemplos disponibles en la tabla. El nuevo ejemplo se clasificará en la clase mayoritaria de los k ejemplos más próximos. El caso más sencillo se da cuando $k = 1$, ya que a cada nuevo ejemplo se le asignará directamente la clase del ejemplo presente en la tabla más próximo. La distancia más común usada es la distancia *Euclídea*.

SVM

Las llamadas *Support Vector Machines*, o S.V.M, (máquinas de soporte vectorial) son un conjunto de algoritmos de aprendizaje supervisado. Estos métodos están propiamente relacionados con problemas de clasificación y regresión.

Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible.

Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase. Más formalmente, una SVM construye un hiper-plano o conjunto de hiper-planos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

2.3.2. Clasificación con Múltiples Clases

Hasta ahora no habíamos hecho ninguna distinción entre tipos de clasificaciones, pero es importante distinguir la clasificación binaria (si/no) de una clasificación con más de dos posibles clases, ya que existen diferentes enfoques.

El problema que se ha tratado en este trabajo consta de 12 clases diferentes como salida. Las técnicas que se usan para la clasificación multi-clase son esencialmente las mismas o con ligeras transformaciones.

Un problema multi-clase puede convertirse fácilmente en binario de diferentes formas. Volviendo al set de datos de Iris, vamos a ver algunos ejemplos de los dos grandes métodos para transformar un problema multi-clase en binario:

1. **One VS One**: Enfrentando las clases entre sí obtendríamos $\frac{n*(n-1)}{2}$ clasificadores. A la hora de predecir, la clase con el mayor número de votos será la seleccionada. También es posible la selección de la clase con la mayor probabilidad agregada.
2. **One VS All**: Consta de 3 clasificadores binarios, uno por cada clase, indicando si es o no de dicha clase. El resultado final será la clase del clasificador que dé mayor probabilidad.

2.3.3. Trabajar con Datos Desbalanceados

La mayoría de los algoritmos de clasificación solo obtendrán buenas clasificaciones cuando el número de instancias por clase sea parecido. Conjuntos de datos con grandes diferencias alteraran las predicciones.

Cómo suele ser normal en este tipo de problemas de clasificación, nuestros datos no están equilibrados y disponemos clases con mayor presencia que otras. Esto hace que los clasificadores estén sesgados y que tiendan a hacer clasificaciones solo con las clases dominantes. Cómo nuestro objetivo es que todas las clases estén igualmente modeladas es necesario aplicar algunas técnicas al *dataset*.

Una de las formas de arreglar este problema se denomina *resampling* y consiste en tomar datos aleatorios de algunas clases para generar nuevas instancias o eliminarlas, con el fin de llegar a clasificaciones más robustas. Las dos grandes técnicas de *resampling* que se usan a la hora de combatir unos datos des-balanceados son las siguientes:

1. **Undersamplig**: Tomar instancias aleatorias de las clases mayoritarias para reducir el número de muestras de estas.

2. **Oversampling**: Copiar instancias de las clases minoritarias para darle más representación.

2.4. Evaluación de las Soluciones

Cuando se tiene una solución, tanto como una regresión como una clasificación, tenemos que evaluar el modelo para saber la calidad de las predicciones realizadas con él. Existen infinidad de métricas para evaluar precisión, certeza, varianza, error medio, etc...

Lo más importante tras elegir la métrica (o métricas) ideal para nuestro caso, es saber que a la hora de realizar la evaluación lo estamos haciendo de manera correcta para que, cuando llegue el momento de iterar, mejoremos.

Ya hemos hablado anteriormente de como se puede dividir un set de datos (sección 2.3) en partes iguales para usar nuevos datos en la evaluación (Split Train/Test, Validación Cruzada, LOO). Muchas veces lo que ocurre es que el algoritmo empieza a aprender cosas específicas de los datos y la métrica mejora en nuestra evaluación, aunque realmente empeora cuando se le proporcionan nuevos datos. A esto se le llama *overfitting*.

Una forma de interpretar el *overfitting* es dividir el error a la hora de generalizar en los siguientes componentes: varianza y sesgo. El sesgo es la tendencia del clasificador a aprender las mismas cosas erróneas mientras que la varianza es la tendencia para aprender cosas aleatorias independientemente de la señal. Hay que buscar controlar el sesgo y varianza para intentar evitar el error a la hora de generalizar.

2.5. Herramientas Software Utilizadas

El número de herramientas que podría utilizar para resolver problema planteado en la competición es muy numeroso, por ello, para elegir aquellas que iba a utilizar, me he regido por el número de librerías disponibles en función de los algoritmos que planeaba usar. A partir de ahí, seleccioné el software que nos lo otorgaba de una manera más eficaz.

Para la mayoría del proyecto se ha usado el lenguaje de programación Python. Es un lenguaje de programación interpretado (utiliza tipado dinámico) y multi-paradigma. La filosofía de Python se basa en conseguir una sintaxis que favorezca un código legible. Además, se trata de un lenguaje que soporta tanto orientación a objetos como programación imperativa, y aunque en menor medida, programación funcional.

En concreto, estas han sido las librerías principales de Python que más útiles me han resultado:

- **NumPy**: Paquete esencial para la computación científica en Python:
  - Poderosa implementación del array o vector N-dimensional.
  - Avanzadas funciones matemáticas implementadas.
  - Herramientas para la integración de códigos en C, C++ o en Fortran.
  - Buen generadores de números aleatorios.

  Además, NumPy puede ser utilizado como un contenedor de varias dimensiones eficiente para datos genéricos. También permite definir tipos de datos arbitrariamente lo que convierte a NumPy en una buena herramienta para integrar sin problemas con una gran variedad de bases de datos.

- **Scikit Learn**: se trata de una librería de código abierto para aprendizaje automático en Python. Dentro de esta librería se encuentran algoritmos de clasificación, regresión y clustering entre los que se encuentra, por ejemplo, máquinas de soporte vectorial, Naïve Bayes, Random Forests, Boosting, k-medias, etc. Además, está diseñada para operar directamente con las librerías NumPy y SciPy.

- **Pandas**: es una librería de código abierto que proporciona un excelente rendimiento y una forma fácil de estructurar los datos para posteriormente analizarlos. Ha sido clave en el procesamiento de datos ya que facilita mucho todas las alteraciones que queramos realizar.

- **XGBoost**: Es una librería que implementa una forma específica de algoritmo de clasificación basado en árboles llamado *Boosting*. Últimamente está ganando tracción ya que se usa en muchas competiciones por su simplicidad y potencia a la hora de clasificar.

- **H20**: Librería alternativa a *XGBoost* con implementaciones modernas de multitud de técnicas, desde redes neuronales hasta preprocesamientos especiales para procesamientos de lenguajes.

Capítulo 3

Exploración de los Datos

Esta fase es clave para obtener una idea general y validar las ideas que se tienen durante el transcurso del proyecto, puesto que una imagen puede orientar mejor que los resultados mas técnicos.

El análisis exploratorio de datos o E.D.A., de sus siglas en inglés, *Exploratory Data Analysis* es un procedimiento estadístico que sirve de herramienta para el tratamiento de datos con el fin de encontrar relaciones en los datos obtenidos en diferentes tipos de estudios. Su metodologıa es relativamente sencilla y sistemática, permite organizar los datos para encontrar conjuntos con relaciones fuertes entre sus observaciones (clusters), observaciones que se encuentran fuera de esas relaciones (outliers) y mediante el estudio de estas ultimas comprobar si están fuera de esos rangos por errores en las mediciones o por ser casos especiales de estudio. Por lo tanto, se trata de hacer un análisis cualitativo de los datos para poder extraer información relevante de los mismos.

En primer lugar se organizan los datos para el estudio mediante una estandarización de las variables de manera que podamos aplicar las técnicas estadısticas para conseguir la media, varianza y demás variables estadísticas que definen el conjunto a estudiar. A continuación se realiza una visualización de las relaciones existentes. Y por ultimo estudiar los patrones "*especiales*" como los outliers para ver porque lo son y si nos son de utilidad en la investigación.

En resumen, podemos listar los objetivos más importantes del análisis exploratorio como:

- Maximizar la visión general de los datos.
- Descubrir la estructura en la que se encuentran.
- Extraer o descubrir las variables con mayor importancia.
- Detectar anomalías.
- Comprobar teorías previamente asumidas.
- Determinar posibles cambios y ver como afectan al problema.

3.1. Origen de los Datos

Antes de examinar los datos en profundidad vamos a explicar un poco más en detalle las diferentes fuentes que tenemos. La competición está ceñida a los archivos que Airbnb ha proporcionado a través de la plataforma Kaggle.

En total son 5 archivos en formato C.S.V. (comma separated values):

1. *train_users*: Conjunto de usuarios de entrenamiento (sabemos la clase a la que pertenecen)

2. *test_users*: Conjunto de usuarios de test, que son aquellos que tenemos que predecir. La competición se evalúa en función de la precisión con la clasifiquemos a estos usuarios.

3. *sessions*: Historial de las sesiones web de algunos usuarios.

4. *countries*: Resumen de las estadísticas de los países que aparecen en el problema

5. *age_gender_bkts*: Resumen de las estadísticas de la población en diferentes rangos de edades y países.

El primer problema con el que nos encontramos es la amplia variedad de fuentes de información. Para poder aplicar una técnica de Machine Learning, los datos deberían de estar dispuestos de manera que, cada muestra, fuera solo una fila de una matriz. Airbnb nos ofrece así los datos para que nosotros decidamos como usar cada archivo.

Los archivos más importantes son aquellos que nos muestran datos de los usuarios y de las sesiones. Cada usuario tiene un numero N de eventos en el fichero *sessions.csv*, esto hace que no podamos mezclarlos de por si y haga falta un procesamiento extra para intentar aprovechar el máximo de información.

age	country_destination	date_account_created	date_first_booking	first_browser	first_device_type	gender	language	signup_app	signup_flow	signup_method
NaN	NDF	2010-06-28	NaN	Chrome	Mac Desktop	-unknown-	en	Web	0	facebook
38.0	NDF	2011-05-25	NaN	Chrome	Mac Desktop	MALE	en	Web	0	facebook
56.0	US	2010-09-28	2010-08-02	IE	Windows Desktop	FEMALE	en	Web	3	basic
42.0	other	2011-12-05	2012-09-08	Firefox	Mac Desktop	FEMALE	en	Web	0	facebook
41.0	US	2010-09-14	2010-02-18	Chrome	Mac Desktop	-unknown-	en	Web	0	basic

Tabla 3.1: Parte de los datos de 5 usuarios aleatorios

user_id	action	action_type	action_detail	device_type	secs_elapsed
d1mm9tcy42	lookup	NaN	NaN	Windows Desktop	319.0
d1mm9tcy42	search_results	click	view_search_results	Windows Desktop	67753.0
d1mm9tcy42	lookup	NaN	NaN	Windows Desktop	301.0
d1mm9tcy42	search_results	click	view_search_results	Windows Desktop	22141.0
d1mm9tcy42	lookup	NaN	NaN	Windows Desktop	435.0

Tabla 3.2: Ejemplo de sesión de un usuario

Como podemos apreciar, tenemos una clara relación uno a muchos entre el archivo de usuarios y el de las sesiones, siendo la clave primaria en ambos la ID de los usuarios. El archivo de sesiones es de un tamaño de 55MB y contiene hasta 5000 eventos por usuario, por lo que unir los datos en forma de columna queda descartado. Abordaremos diferentes enfoques a este problema más adelante, primero comenzaremos por la descripción de las variables.

3.2. Descripción de las Variables Proporcionadas

Dentro de los archivos de usuarios tenemos las siguientes variables:

- **id**: ID del usuario. Única para cada uno de los usuarios.
- **date_account_created**: Fecha de la creación de la cuenta en Airbnb.
- **timestamp_first_active**: Estampa de tiempo de la primera actividad del usuario. Esto puede ser antes de la creación de la cuenta ya que un usuario puede buscar aunque no tenga una cuenta.
- **date_first_booking**: Fecha en la que el usuario hizo la primer reserva. [1]
- **gender**: Género del usuario.
- **age**: Edad.
- **signup_method**: Método usado para registrarse.
- **signup_flow**: Página desde la que se ha registrado.
- **language**: Preferencias de lenguaje.

[1] Esta variable se eliminó de la competición posteriormente ya que estaba totalmente correlada con reservar un viaje y se podían acertar todos los usuarios de la clase **NDF**

- **affiliate_channel**: Tipo de marketing asociado.
- **affiliate_provider**: Empresa del marketing asociado.
- **first_affiliate_tracked**: Desde que empresa de marketing vino el usuario.
- **signup_app**: Aplicación desde donde se registraron.
- **first_device_type**: Desde que dispositivo se abrió por primera vez la web.
- **first_browser**: Primer navegador usado para acceder a la web.
- **country_destination**: País de destino del usuario.

Y con respecto al archivo de sesiones, estas son las variables de las que disponemos:

- **user_id**: ID del usuario. Existen IDs repetidas dentro de este archivo ya que son eventos.
- **action**: Acción realizada.
- **action_type**: Tipo de acción.
- **action_detail**: Detalle de la acción.
- **device_type**: Dispositivo.
- **secs_elapsed**: Segundos que han pasado. Airnbnb no especifica más sobre esta variable por lo que hay que hacer y validar algunas suposiciones.

A continuación entraremos de lleno en el análisis exploratorio que se realizó al principio de la competición, para ir analizando y comprendiendo todos los detalles de los datos.

3.3. Análisis Exploratorio General

Antes de cualquier proyecto, siempre es bueno realizar una exploración de los datos. Por ello empezaremos observando los datos de los usuarios, y planteándonos algunas preguntas básicas. ¿Hay algunos errores en los datos? ¿Se comportan de alguna forma en particular? ¿Necesito arreglar o eliminar algunos datos para que sea más realista?

Para empezar, tenemos 213451 usuarios diferentes de entrenamiento y 62096 de test. Esto hace un total de 275547 usuarios. Una cantidad considerable, en la que encontramos diferentes comportamientos y tipos de usuarios.

Para comenzar, partiremos de las variables de género y edad. Para ello, vemos los valores únicos que tienen cada columna. Para la de género se esperarían solo dos valores, Male y Female.

Veamos el recuento de valores:

-unknown-	129480
FEMALE	77524
MALE	68209
OTHER	334

Tabla 3.3: Recuento de valores en la variable *género*

El 46 % de los usuarios no tiene asignado género y el 0.1 % tiene como género *otro*.

Para las edades, donde esperaríamos valores en el intervalo $[1, 100]$, vamos a describir la columna:

mean	47.145310
std	142.629468
min	1.000000
25 %	28.000000
50 %	33.000000
75 %	42.000000
max	2014.000000

Tabla 3.4: Resumen estadístico de la variable edad

Podemos ver que la media está en 47 años, pero salta a la vista que no estamos ante una desviación típica normal, haciéndonos suponer que no todos los datos son correctos. Si nos fijamos en el valor máximo, podemos suponer que se refiere a un año en concreto, es decir, a una fecha, no a la edad de ningún usuario real. Esto puede ser una inconsistencia de valores, donde algunas personas ponen la fecha de nacimiento en lugar de la edad actual.

Hay 830 usuarios con edad superior a 100 años y 188 con menos de 18. Para las muestras que están entre 1940 y 2014 se puede calcular la edad haciendo una simple resta. En el resto de muestras, descartamos el valor.

date_first_booking	67.8 %
first_affiliate_tracked	2.2 %
first_browser	16.1 %
gender	46.5 %

Tabla 3.5: Porcentaje de datos no rellenos

Continuaremos con una serie de exploraciones gráficas para intentar comprender mejor los datos y ver si encontramos algunos patrones en ellos. Visualizar las variables hace que detectar *outliers* y errores sea mucho más fácil que estudiando las gráficas.

De nuevo, exploramos primero las variables de género y edad:

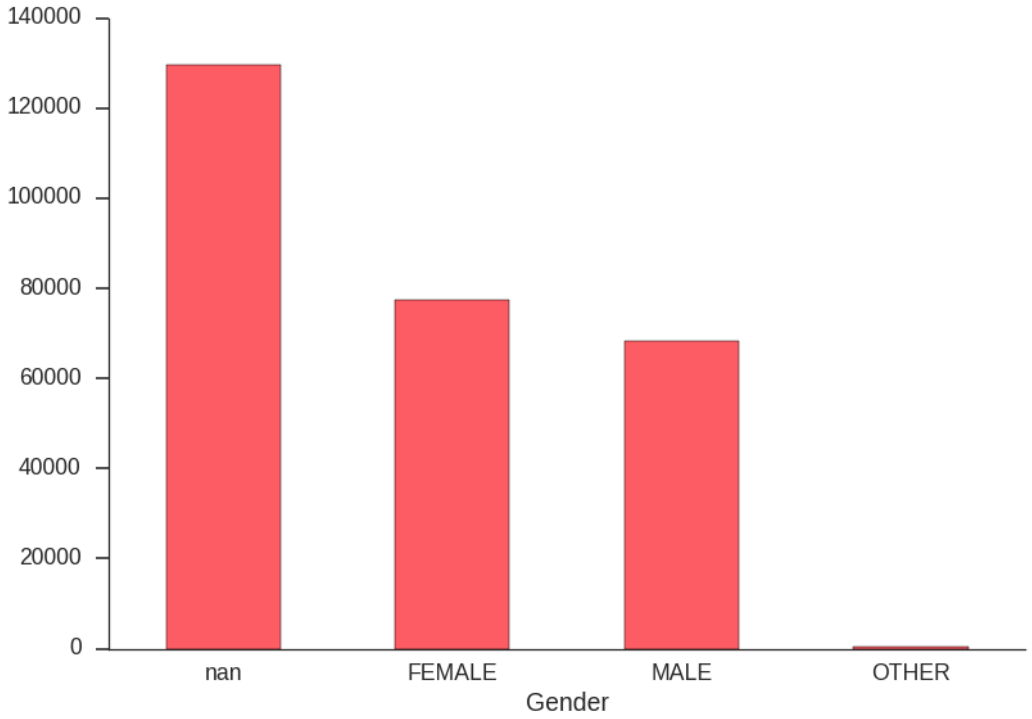

Figura 3.1: Histograma de valores para la variable de género.

Como hemos observado anteriormente tenemos una amplia variedad de usuarios que han especificado su género. La figura 3.1 nos ayuda a poner la cantidad en perspectiva, pero también nos muestra una leve diferencia en el género de los usuarios. Debemos recordar que esto puede no estar relacionado con la distribución real, ya que quizás existan más mujeres registradas y que simplemente no han dejado constancia de ello.

Vamos a ver a continuación si existen diferentes preferencias de país según el género a la hora de viajar con Airbnb:

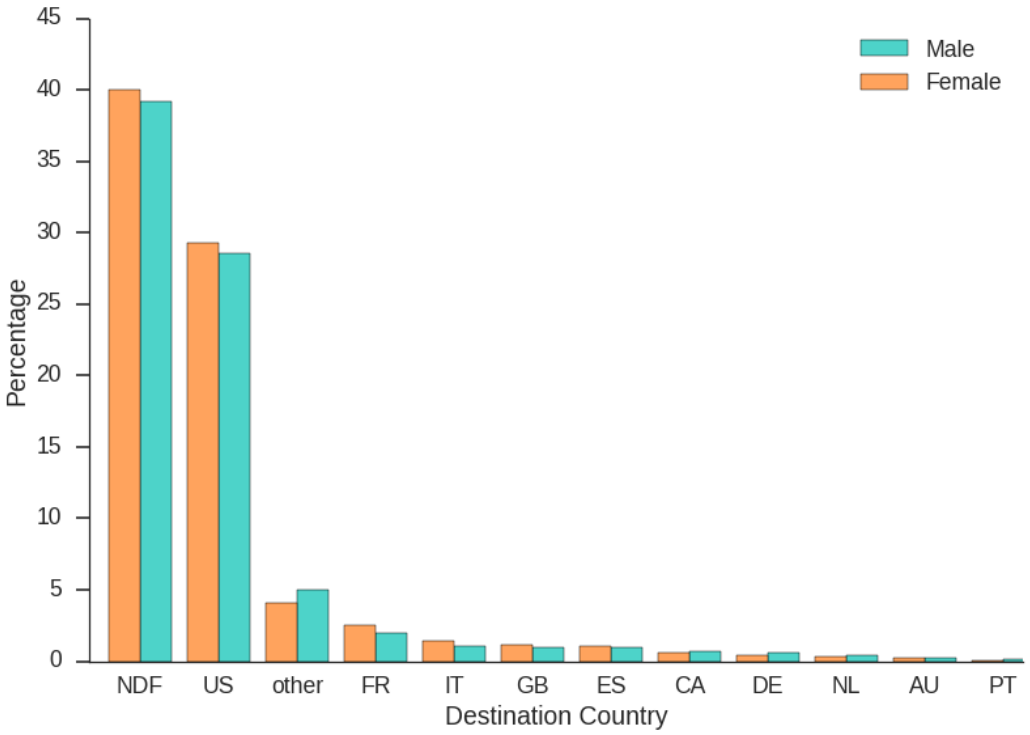

Figura 3.2: Porcentaje de genero por opción de destino.

Como podemos observar, no hay grandes diferencias entre los dos géneros principales. Este gráfico no es muy útil con esta separación, pero podemos volver a hacer a repetirlo uniendo todos los datos y ver la frecuencia real de los países de destino de forma más visual. Lo podemos ver más claramente en la siguiente figura(3.3):

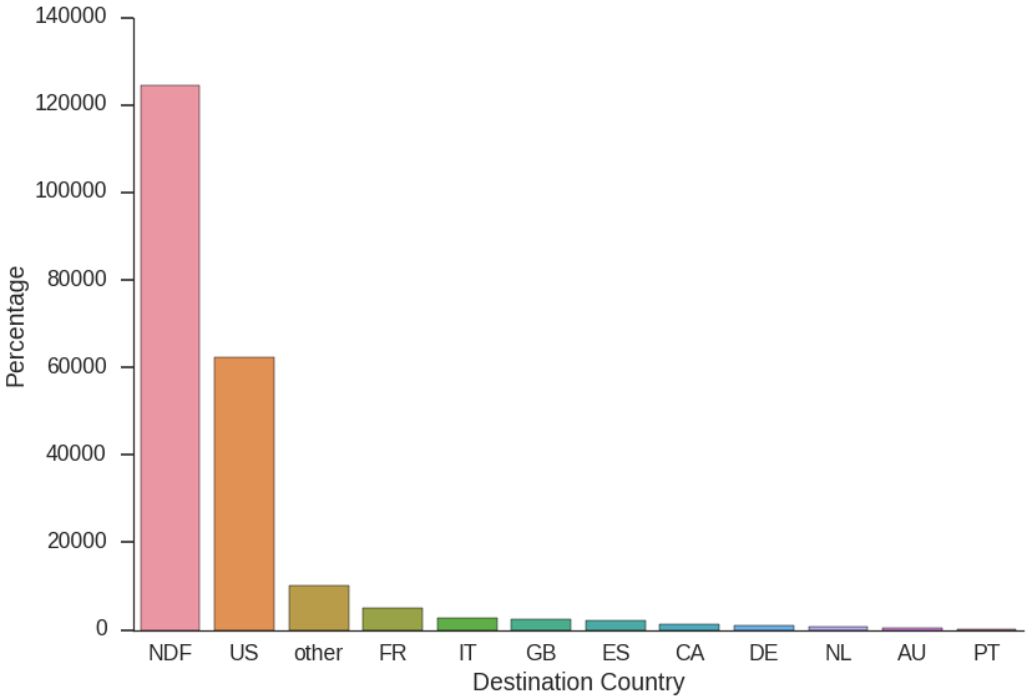

Figura 3.3: Número de usuarios por opción destino.

En relación al número de usuarios según el país podemos sacar una conclusión muy simple, si un usuario reserva, lo más probable es que no salga de Estados Unidos. Pero hay que tener en cuenta que hay un 45 % de usuarios que no llegaron a realizar ninguna reserva. Estas dos clases pueden des-balancear el problema.

Ahora que tenemos una idea de como están repartidos los géneros y la cantidad de usuarios que reservan en cada país, vamos a investigar la variable de edad. Primero, podemos ver una distribución(3.4) de las edades de los usuarios.

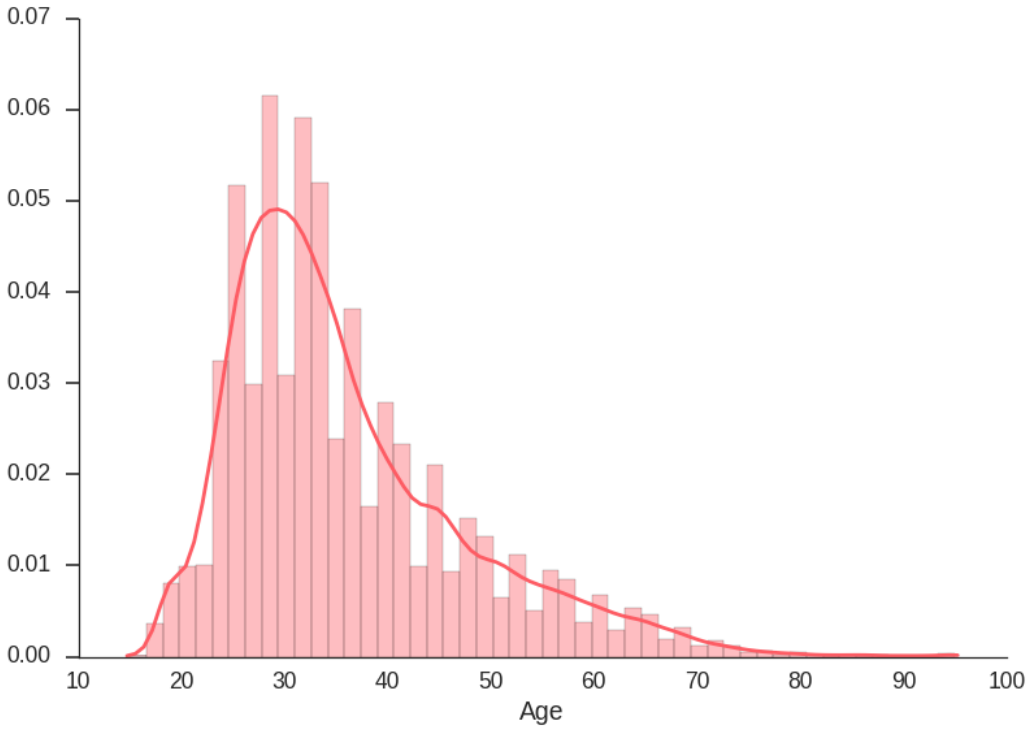

Figura 3.4: Distribución de la edad de los usuarios.

Como cabría esperar, la edad más común para viajar con Airbnb se encuentra entre los 25 y 40 años. ¿Afecta la edad al destino? Para ver si los usuarios de distintos grupos de edades van a distintos lugares vamos a agruparlos y hacer un simple gráfico que muestre los destinos de dos grupos. El primer grupo será hasta los 45 años y el segundo a partir de los 45 años.

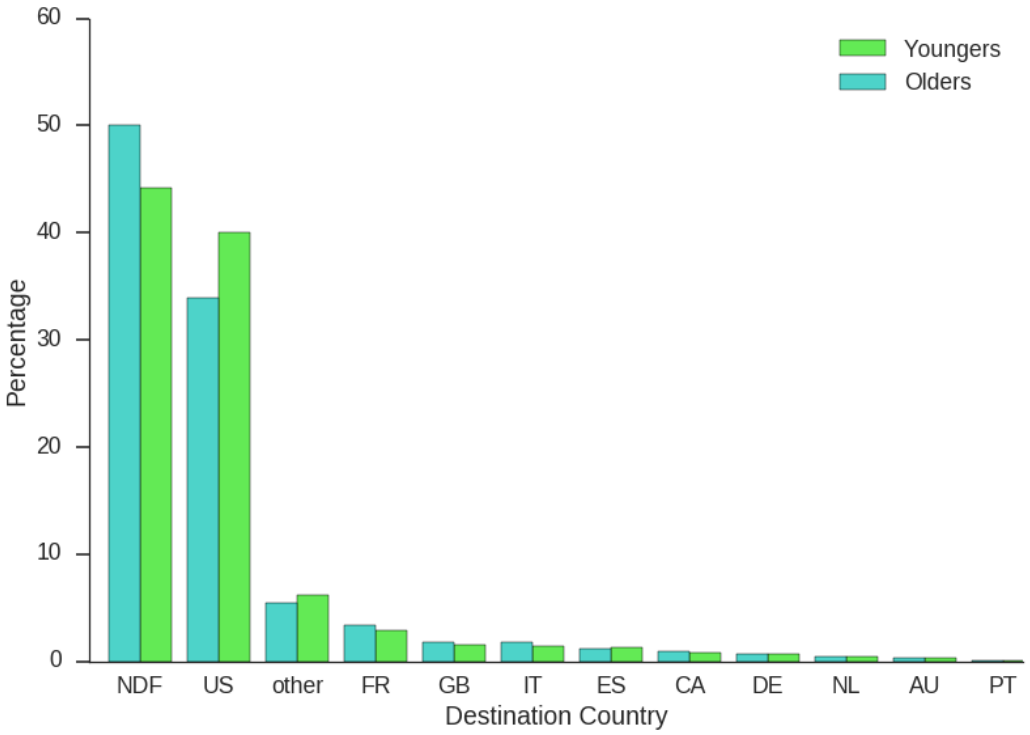

Figura 3.5: Distribución del destino según grupo de edad.

Estudiando este gráfico, podemos observar de forma más clara el comportamiento de ciertos usuarios. Podemos ver como los menores de 45 años suelen quedarse en los Estados Unidos, y la gente mayor tiende a visitar otros países. Hay que destacar que, como se ha comentado anteriormente, solo tenemos el 58 % de las edades de los usuarios.

Otro factor de suma importante en un ámbito como este, podría ser el lenguaje en el que los usuarios tienen configurada la aplicación. Al ser estadounidenses, podemos esperar que la mayoría tengan la aplicación en inglés, para ser más concretos, el porcentaje real de usuarios con la aplicación en inglés es del 96 %. Por lo tanto, no nos podemos guiar por este atributo ya que no tenemos suficientes muestras para comparar con los usuarios que lo tiene en otro idioma.

Seguramente, los usuarios tengan en cuenta la lengua del destino en el que acuden, y para saber eso tendríamos que tener una variable con los lenguajes que conoce cada usuario. Debemos recordar que existen bastantes otros factores que no estamos teniendo en cuenta y hacer suposiciones o predicciones basándonos en datos que no poseemos puede ser peligroso.

Por último, también tenemos información disponible de las fechas de registro de los usuarios, esto no estaría muy relacionado con el objetivo pero sería interesante explorar las fechas en caso de descubrir algún patrón interesante.

Vamos a realizar un gráfico con el número de registros por día de los usuarios proporcionados por Airbnb:

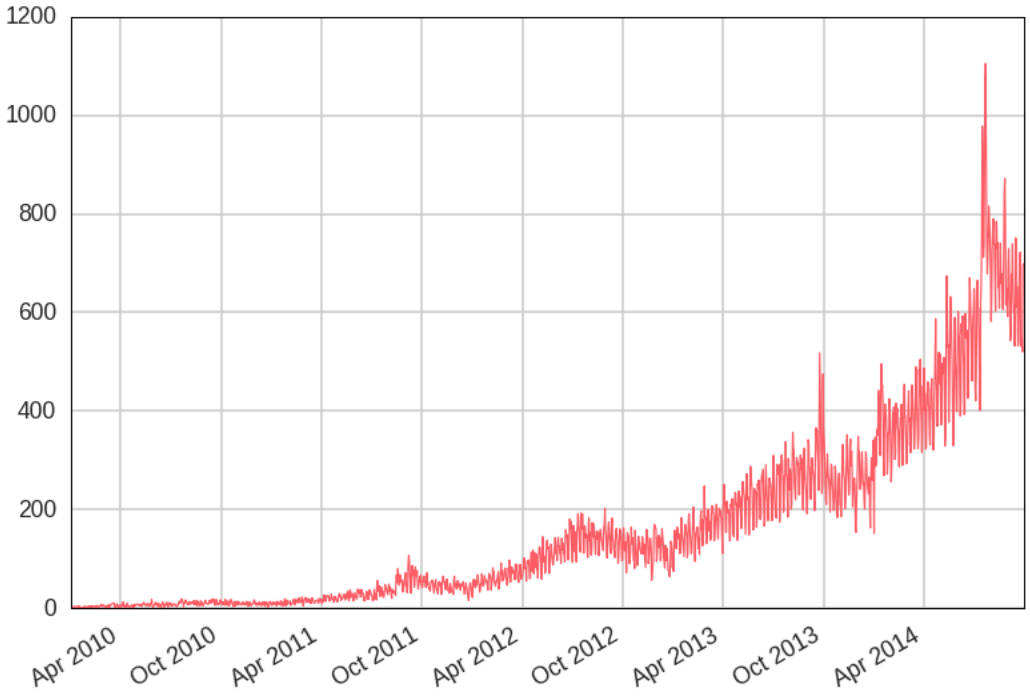

Figura 3.6: Cantidad de nuevos usuarios.

Como vemos, el número de usuarios no ha cesado de incrementar, además de hacerlo a un ritmo considerable. A pesar de ser casi una curva exponencial, vemos como es muy abrupta y posee muchos picos en todas las fechas.

La amplitud de esos picos va en aumento también con el número de usuarios. Tras varias suposiciones, la respuesta se encuentra analizando el día de la semana en el que los usuarios se registra. Es lógico que se registren usuarios todos los días, pero si miramos la siguiente gráfica, podemos ver la distribución de los registros según el día. El 0 indica el lunes y el 6 el domingo.

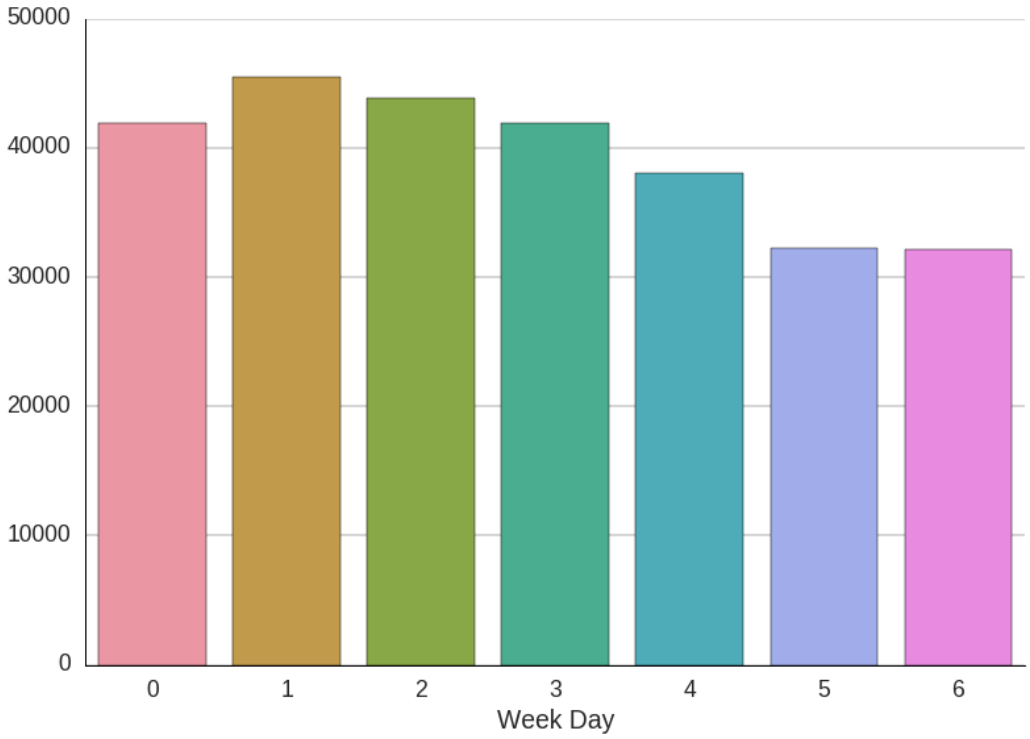

Figura 3.7: Cantidad de nuevos usuarios por día de la semana.

Además de las variables comentadas del archivo de usuario, también se ha explorado el archivo de sesiones. En este archivo tenemos 135484 IDs de usuarios. Si comparamos con los 275547 usuarios que tenemos en total, resulta que tenemos sólo un 50 % de las sesiones de los usuarios.

Vamos a explorar más a fondo las acciones que tenemos registradas de los usuarios. Empezamos con el tipo de acción este puede ser uno de los siguiente valores; `nan`, `click`, `data`, `view`, `submit`, `message_post`, `-unknown-`, `booking_request`, `partner_callback`, `booking_response` y `modify`

Lo primero que notamos es que hay dos formas de definir los valores nulos o desconocidos. Una vez realizada una sustitución para normalizar esos datos vamos a ver cuales son los tipos de acciones más comunes según el numero de veces que se repiten en el archivo de sesiones:

view	3560902
data	2103770
click	1996183
submit	623357
message_post	87103
partner_callback	19132
booking_request	18773
modify	1139
booking_response	4

Tabla 3.6: Número de repeticiones de los distintos tipos de acción en el archivo de sesiones.

Con los nombres no podemos discernir muy bien la importancia de cada tipo de acción salvo quizás booking_request y booking_response, pero dada la poca frecuencia relativa de estos no se han realizado más investigaciones relacionadas. Lo mismo ocurre con la acción y los detalles de la acción (las otras columnas relacionadas con la acción). Existen multitud de acciones posibles haciendo cualquier gráfico o tabla confuso.

Comprobaremos si disponemos de todos los datos computando el número de huecos que tenemos:

action	79626
action_type	1126204
action_detail	1126204
device_type	0
secs_elapsed	136031

Tabla 3.7: Cantidad de valores nulos del archivo de sesiones.

Siempre que falta la variable del tipo de acción, nos falta la del detalle, pero no siempre falta la de la acción en si. Otro detalle a tener en cuenta, serian los detalles sobre el dispositivo (o dispositivos) desde el que los usuarios entran en Airbnb, como refleja la tabla anterior.

Mac Desktop	3594286
Windows Desktop	2658539
iPhone	2105031
Android Phone	839637
iPad Tablet	683414
Android App Unknown Phone/Tablet	273652
-unknown-	211279
Tablet	139886
Linux Desktop	28373

Tabla 3.8: Recuento de dispositivos en las acciones de las sesiones.

De nuevo observamos como los datos no eran del todo correctos, ya que desconocemos que navegador están usando muchos de los usuarios y que vemos reflejado como (`-unknown-`).

Ahora que ya tenemos una idea de la forma y estado de los datos, vamos a proceder con la fase del preprocesado. Aplicaremos las técnicas que se han comentado en la sección de Teoría(2) y posteriormente evaluaremos su colaboración en la métrica final.

Este análisis [2] fue realizado durante los primeros días de la competición y compartido con la comunidad a través de la plataforma Kaggle. También ha sido patrocinado en diferentes sitios(blogs y repositorios de código abierto públicos) a lo largo del tiempo tras ser el *script* más votado de la competición.

[2] `https://www.kaggle.com/davidgasquez/airbnb-recruiting-new-user-bookings/user-data-exploration`

Capítulo 4

Fase de Preprocesamiento

Se ha comentado anteriormente la importancia de un buen preprocesamiento a la hora de evaluar el clasificador final. Esta, se podría decir, es la fase donde se producen más diferencias entre los competidores de una competición de esta índole.

Los algoritmos y parámetros será muy parecidos, pero cada participante aporta un preprocesamiento distinto, generalmente con ideas novedosas. En esta competición, ha sido también la fase en la que los mejores resultados han destacado. Cualquier cambio que se le aplica al set de datos que nos dan, produce un cambio en el resultado. Aunque el cambio no sea muy significativo con respecto a la evaluación inicial, nos puede proporcionar una ventaja en la clasificación final.

A continuación desarrollaremos los cambios que se le han aplicado al set de datos durante la competición que han llevado al mejor resultado. No hay que olvidar que esto ha sido un proceso iterativo y se han aplicado otra multitud de pasos que no han llevado a una mejora.

4.1. Limpieza de los Datos

Tal como hemos visto antes en el análisis exploratorio, tenemos muchos datos “sucios” y desorganizados. Esto significa que tenemos que limpiar y organizarlos de tal forma que sea lo más claro posible.

El primer cambio que se hace está relacionado con las edades. Tenemos un amplio espectro y desde el principio del análisis se eliminan aquellas edades imposibles (menores de 10 años, mayores de 100) y para aquellos valores que representan un año concreto hacemos la diferencia.

En vez de usar el valor *“NaN”* para marcar los valores nulos, se va a usar el número -1 para hacer implícito que son valores que no tenemos a aquellos algoritmos que tratan el valor *“NaN”* de forma diferente.

Para los géneros, al igual que con otras variables y como veremos posteriormente, se ha experimentado con diferentes codificaciones y al ser una variable categórica con pocos valores no ha hecho falta una limpieza.

En el análisis exploratorio nos percatamos de la existencia de varias columnas con el valor ‘-unknown-‘ indicando que era desconocido. Vamos a tratar este valor también como *“NaN”*. Y posteriormente lo rellenamos con el valor -1.

Con las transformaciones anteriores tendríamos los datos en un formato aceptable, lo único que faltaría para completar la limpieza sería asegurarnos que no existen valores repetidos de acciones, países (mayúscula/minúscula) y demás variables que sean una cadena de texto propensa a error. Una vez tengamos el set preparado en esta fase pasamos a normalizar los valores.

4.1.1. Normalización Realizada

Principalmente, dado el tipo de variables que se tienen en este problema, la normalización no es un paso obligatorio. Mi enfoque en esta fase ha sido intentar estandarizar formatos, especialmente con las fechas.

Al estar usando la librería Python *Pandas* con capacidad de manejar fechas, he encontrado muy útil transformar las cadenas de caracteres que representaban fechas a un objeto Python para poder extraer más datos de ellas posteriormente. En el siguiente fragmento de código podemos ver lo simples que son dichas transformaciones.

```
users['date_account_created'] = pd.to_datetime(
    users['date_account_created'], errors='ignore')
users['date_first_active'] = pd.to_datetime(
    users['timestamp_first_active'],
    format='%Y%m%d%H%M%S')
```

4.2. Formas de Codificación de Variables

En esta sección es importante hablar de dos tipos de codificaciones que se han aplicado durante el preprocesamiento. Una referida a las fechas y otra a las variables categóricas.

Una fecha se puede representar de diversas formas, con las cuales se ha experimentado; haciendo una estampa del tiempo en una cadena de caracteres(20151230), usando una representación UNIX del formato de fecha o aplicando algún tipo de *hashing* a la fecha para codificarla, entre otras. Según se ha experimentado, el resultado final mejora cuando la fecha se divide en diferentes variables independientes:

1. Día

2. Día de la semana
3. Número de la semana
4. Mes
5. Año

Esta codificación tiene como ventaja que los algoritmos puede abstraer mejor y comparar el mismo día de distinto mes o el mismo día de la semana entre varias semanas. Además, nos permite un enfoque más granular a la hora de seleccionar las variables importantes, por ejemplo, descartando el año si no hiciera falta.

Para las variables categóricas se ha experimentado con diferentes tipos de *encodings*. Los principales, explicados en la sección de teoría, han sido la codificación ordinal simple, One-Hot-Encoding y encoding binario. La codificación ordinal simple permite hacer una iteración rápida, ya que no aumenta el tamaño de los datos y los clasificadores tardan menos en ofrecer un resultado, aunque considerablemente menos preciso que haciendo One-Hot-Encoding. El problema que nos encontramos al hacer la codificación One-Hot-Encoding es la gran cantidad de variables finales (alrededor de 1000). Con la codificación binaria tenemos un equilibrio entre ambos enfoques, pero se ve perjudicada en el resultado fina,l así que se ha optado por el clásico One-Hot-Encoding, aunque tengamos que lidiar con *datasets* de mayor tamaño como veremos próximamente.

4.3. Manejo de los Valores Vacíos

Para los valores vacíos se han realizado multitud de técnicas y enfoques para intentar aprovechar el máximo de información. Para empezar, se rellenaron con la media/moda de la columna en particular. El problema de este enfoque es que al tener tanta cantidad de datos no rellenos, estamos distorsionando la realidad al añadir estos datos.

Posteriormente, se pasó a técnicas más complejas y se hizo una predicción de las edades y géneros usando Máquinas de soporte Vectorial entrenadas con los valores que disponíamos. Mejorando esto el primer resultado, es cierto que el proceso se volvía lento y no ofrecía una ventaja estadísticamente significativa para añadirlo al flujo.

Por último, tal y cómo se ha comentado con anterioridad, los datos vacíos han sido simplemente sustituidos con el valor -1 tras haber comprobado que sí existía una mejora tangible en la evaluación de las clasificaciones.

4.4. Enriquecimiento de los Datos

Cómo siempre, los datos que nos dan pueden ofrecernos algo más allá de lo evidente. Realizando cambios simples, y siempre siguiendo un proceso iterativo buscando mejorar, se han explorado una serie de nuevos datos. A continuación detallaremos los más importantes.

Empezamos por agrupar las edades para hacer presente los diferentes grupos de edades. Sabemos que las personas tienen diferentes hábitos de viaje según su edad, ya que tenemos las edades muy granuladas pasamos también a añadir grupos. Esto es, si hacemos 5 grupos, clasificaremos a los usuarios según la edad entre los intervalos $[0-20)$, $[20-40)$, $[40-60)$, $[60-80)$, $[80-100]$. Tras experimentar con distintos intervalos y amplitud de los mismos se ha fijado el final del número de intervalos a 5.

Otra característica que se ha podido extraer de el set de datos es el número de datos no rellenos por instancia, en este caso usuarios. Para cada usuario vamos a computar ese dato sumando la fila de la matriz final, esto puede servir de referencia del nivel de compromiso de algunos usuarios. Aquellos que tengan mayor interés, tenderán a incluir más información en su perfil y no dejar datos en blanco.

Ya que disponemos de distintas fechas, también podemos calcular la distancia entre ellas. Añadimos las distancias entre la fecha de creación de la cuenta y del primer usuario en días (también se pueden añadir meses y semanas).

Para todo esto solo se ha usado el archivo de usuarios, pero sabemos que el de sesiones también contiene información importante para la competición. Para aprovechar esa información se pensó originalmente contar la frecuencia de las acciones, tipos de acciones y detalles para cada usuario. Posteriormente también se ha experimentado con la frecuencia relativa de secuencias de acciones (n-grams), sin mejorar el resultado, por lo tanto finalmente se decidió añadir, además de la frecuencia, otros valores estadísticos como la varianza, moda, co-varianza, etc. Esto nos proporciona una información bastante personalizada de cada usuario y ha sido una parte vital para el resultado final.

Al disponer también de los segundos entre las acciones de los usuarios, hemos aprovechado esa información para también calcular valores estadísticos sobre el uso de la aplicación por cada usuario. Además se han añadido las características nuevas:

1. Número medio de acciones por sesión. Se ha definido sesión como el conjunto de aquellas acciones que difieren en menos de 2 horas tras explorar la distribución de la variable *secs_elapsed*número de sesiones.
2. Cantidad de sesiones.
3. Media de segundos entre sesiones.
4. Cantidad de pausas mayor que 1 día y menores que un día.

Además de las anteriores variables del archivo de sesiones, se añadieron varias más de los otros archivos (estadísticas de género, edad y países) sin mejorar considerablemente el resultado. Se decidió descartar esta información, ya que no proporcionaba una mejora estadísticamente significativa en la media de la función de evaluación.

Por último, se ha añadido una variable teniendo en cuenta la naturaleza del problema. Cómo sabemos, la gente reserva alojamientos durante todo el año, pero se incrementa con los periodos vacacionales o días festivos. Tras conseguir un set de datos con los días festivos de Estados Unidos, se procedió a calcular los días de diferencia con la fecha de registro. Esta nueva variable proporciona una información adicional significativa a nuestro algoritmo.

4.5. Mejoras de Rendimiento Aplicadas

Hay que tener en cuenta que, de las posibles soluciones que se podían desarrollar, muchas quedaban descartadas por la cantidad de datos con la que estábamos trabajando. Si bien es cierto que disponíamos del servidor Hércules perteneciente a la UGR, la mayoría de las funciones están pensadas para una sola hebra.

Inicialmente, el conjunto de pasos de preprocesamiento tardaba 32 horas, ya que tras las primeras iteraciones, los datos pasaban al disco duro al no tener suficiente memoria RAM (8GB). Vamos a detallar las distintas optimizaciones que se han ido realizando a lo largo del proyecto:

1. Los valores numéricos como la edad o los datos de la fecha se han transformado a enteros para ahorrar espacio en memoria RAM.
2. Al aplicar la codificación One-Hot-Encoding pasamos a tener 1311 columnas. Muchas de ellas son binarias y sólo tienen unos pocos valores como 1. Estos datos esparcidos ocupan 1.2GB en memoria RAM. Después de transformarlos a enteros y declarar la matriz como *sparse*, para obviar los valores a cero y mejorar el espacio pasamos a un tamaño de 500MB. Con esta optimización se redujo el tiempo para procesarlos a 12 horas.
3. Cómo el procesamiento de cada usuario es independiente, se paraleliza. Esto redujo el tiempo total a 4 horas.

4. Por último, en Python 3, las cadenas de caracteres están representadas como *unicode* internamente y son bastante pesadas. A la hora de trabajar con los IDS de usuario, se les ha aplicado previamente un *hashing* a números enteros. Cada usuario queda así representado por un número único ahorrando una cantidad considerable de espacio y acelerando mucho su procesamiento paralelo (ya que las hebras tienen que serializar el entero en vez de la cadena de caracteres). Con este cambio el tiempo total en aplicar todo el preprocesamiento pasó a ser 30 minutos.

4.5.1. Otras Mejoras Aplicadas

En su momento también se realizaron otro tipo de mejoras para intentar mejorar el resultado de la clasificación. Entre ellas estaba el uso de herramientas para reducir la mensualidad de los datos (P.C.A.), que quedaron descartadas al no mejorar nuestro objetivo.

En cuanto a la velocidad, también se exploró la opción de programar ciertas partes en C, ya que Python ofrece la posibilidad de hacer un enlace entre ambos lenguajes. Se usaron herramientas en C para alcanzar resultados similares.

Para hacer todo el proceso más rápido y dinámico se han creado puntos de guardado durante todo el preprocesamiento. Además, usando el *hash* del objeto en memoria podemos saber si tenemos el mismo objeto que tenia el paso X y, en ese caso, cargar el resultado en lugar de volver a calcularlo.

Por último, es importante comentar que en varias ocasiones se han generado muestras adicionales de las clases menos representadas a través de técnicas como SMOTE o Random Over-Sampling. Estas técnicas, dependiendo del algoritmo utilizado, han otorgado una leve mejora.

4.6. KISS

Aunque muchos participantes han dejado estos valores en su formato original, dada la cantidad de datos que tenemos que procesar, ha sido muy importante mantener todos los datos y el procedimiento lo más simple posible y sin grandes y complicadas modificaciones.

He querido destacar la importancia del principio K.I.S.S. (Keep It Simple Stupip) en esta competición, ya que ha sido muy importante para el resultado final. Usando Python como lenguaje principal es fácil que el programa sea pesado a la hora de ejecutarse, puesto que tiene que mantener en memoria RAM alrededor de 2GB de información relacionada solo con los usuarios y las sesiones. Si a esto añadimos toda la información adicional que necesita en memoria Python al ejecutarse, y la que se crea a la hora de hacer modificaciones, es normal que el S.O. empiece a pasar partes desde la R.A.M. al disco duro (*caching...*).

Podría decir que, una de las lecciones más importantes que he aprendido durante el desarrollo de este proyecto, ha sido comprender como haciendo las cosas simples, además de que ser más fáciles de entender, también se suelen obtener mejores resultados. En este caso, siempre que varios enfoques han dado resultados similares, me he decantado por el enfoque más simple.

Capítulo 5

Modelos Usados y Técnicas Aplicadas

Tras haber realizado las primeras modificaciones al set de datos se empezó a investigar con distintas técnicas. Las aquí reunidas son el conjunto más notable en cuanto a la evaluación específica del reto.

En general, la mayoría de los participantes han elegido algoritmos basados en árboles, donde se pueden obtener buenos resultados fácilmente. Otro de los factores, como ya hemos visto, es la cantidad de datos, por lo que no se pueden aplicar ciertas técnicas de clasificación que funcionan bien en conjuntos de datos pequeños pero tienen problemas al escalar. Si bien es cierto que, con estas restricciones, limitamos bastantes los algoritmos de clasificación que se pueden usar, tenemos aún un amplio espectro de posibilidades para hacer una solución que diverja del resto.

5.1. Enfoque General del Proyecto

Desde el comienzo de la competición, se ha querido tomar un enfoque dinámico, donde se pueda iterar rápidamente y explorar el espacio de soluciones de forma sencilla. Al no conocer en profundidad ninguno de los algoritmos y poder decantarse desde el principio por alguno, se decidió hacer una exploración con los algoritmos de clasificación presentes en la librería Scikit Learn.

Usando Scikit Learn se pueden usar más de 10 algoritmos de clasificación, y estos, a su vez, disponen de una inmensidad de parámetros ajustables. Los algoritmos seleccionados fueron: Vecino más cercano, Máquina de soporte vectorial con diferentes *kernels*, árboles de decisión, Random Forest, AdaBoost, Naive Bayes, Linear Discriminant Analysis y Quadratic Discriminant Analysis.

En cuanto se hizo la primera evaluación se obtuvieron unos resultados muy claros. Usar clasificadores potentes, como las máquinas de soporte vectorial (SVM), era muy lento y los métodos tradicionales dejaban mucho que desear. Se decidió enfocar el problema de la competición, como la mayoría de los participantes, mediante métodos basados en árboles.

5.2. Métodos Basados en Árboles

En la sección de teoría 2.3.1 se ha explicado la idea detrás de un árbol de decisión y sus ventajas. Vamos a ver un poco más en profundidad estos tipos de clasificadores y cómo nos pueden ayudar en esta competición.

Dada la naturaleza del problema, tanto en tamaño como en tipo de variables, es lógico que un árbol de decisión sea un buen clasificador, ya que está haciendo las preguntas que un humano haría. Por ejemplo: ¿Tiene el usuario el navegador en Italiano? Si la respuesta es sí, se pueden hacer preguntas para ir cercionándose de que el usuario quiere ir a Italia desde el primer momento. Otra de las ventajas que obtenemos es que, como estos árboles hacen preguntas según los datos, no tenemos que preocuparnos de escalarlos para que funcionen mejor.

Normalmente, en la práctica, no se suelen usar árboles de decisiones de por sí, los árboles de decisión suelen tener un alto nivel de varianza o un alto nivel de sesgo. Por norma general se suelen hacer varios, y según la técnica elegida, usar la información que cada uno aporta de forma particular. Podemos decir que es el equivalente al concepto de *Board Of Directors* [1] de una empresa. Cada director tiene una visión diferente del problema y en común son mejores que por separado.

Sin embargo, la razón más importante a la hora de haber elegido estos algoritmos, no es otra que su fácil paralelización. Es trivial entrenar un árbol por hebra y esto hace que sean fácilmente escalables. En este problema, aunque no se hayan utilizado grandes *clusteres* de computación, con herramientas como Hadoop o Spark, también es crucial la escalabilidad del problema. Hay que pensar que una versión del algoritmo ganador es la que posiblemente esté algún día en producción y la cantidad de datos de Airbnb puede ser muy superior a la de la competición.

[1] `https://en.wikipedia.org/wiki/Board_of_directors`

5.2.1. Random Forest

A nivel más básico, el algoritmo de Random Forest, se ocupa de crear muchos árboles ligeramente diferentes, para posteriormente, tomar el voto de la mayoría como respuesta correcta. Los árboles se crean usando un subconjunto aleatorio de las variables del problema.

El principio básico detrás de esta técnica parte de un grupo de "weak learners", los cuales pueden agruparse entre ellos y formar lo que conocemos como "strong learner" [8]. Random Forest es una herramienta estupenda para hacer predicciones, dado que no suelen ajustarse mucho al set de datos que se esté usando y son capaces de generalizar de manera exitosa. Esto son los beneficios de usar Random Forest:

- Precisión.
- Eficiencia con conjuntos de datos grandes.
- Maneja fácilmente datos con muchas variables sin necesidad de simplificar previamente los mismos.
- Se puede conocer que variables son más importantes.
- Maneja muy bien los valores vacíos.
- Puede balancear problemas donde las clases no tengan el mismo número de muestras.

Durante la primera fase de la competición, este algoritmo ha sido ampliamente utilizado para realizar una comprobación rápida de los cambios que se hacían en el preprocesamiento. Posteriormente, se usó para rellenar los valores vacíos de la edad y género, aunque sin éxito.

Un aspecto importante, que ha influido en la competición en diversas ocasiones, es la capacidad que tiene los algoritmos basado en árboles, de conocer la importancia relativa de las variables. Al obtener este dato, podemos eliminar aquellas que no superen un porcentaje mínimo de importancia, comprobando así si mejora el algoritmo en tiempo o en calidad. Scikit-Learn ofrece esta posibilidad y se ha estado usando con un umbral muy bajo para no afectar demasiado a las soluciones otorgadas.

5.2.2. Boosting

Boosting es un enfoque donde el resultado se da usando la media ponderada de varios árboles y combina las ventajas de cada árbol al darle a éste un peso en la decisión final. En este enfoque, los árboles tienen que ir construyéndose de manera lineal para intentar añadir árboles que mejoren aquello en lo que el resto ha fallado. El objetivo final es eliminar el sesgo. [9]

Para la competición en particular, se ha usado la librería XGBoost, que nos permite usar Boosting con árboles de forma paralela, eficiente y flexible. Este ha sido el algoritmo usado en el resultado final debido a su capacidad para obtener valores significativamente superiores a Random Forest, aunque sacrificando velocidad en el entrenamiento.

El tiempo de entrenamiento variaba de 3 a 4 horas, dependiendo del número de variables con las que se lanzaba el problema. Para mostrar la flexibilidad y facilidad de uso de esta librería, mostraré el código requerido para entrenar un clasificador con Validación Cruzada usando XGBoost.

```
param = {
    'max_depth': 10,
    'learning_rate': 1,
    'n_estimators': 5,
```

```
    'objective': 'multi:softprob',
    'num_class': 12,
    'gamma': 0,
    'min_child_weight': 1,
    'max_delta_step': 0,
    'subsample': 1,
    'colsample_bytree': 1,
    'colsample_bylevel': 1,
    'reg_alpha': 0,
    'reg_lambda': 1,
    'scale_pos_weight': 1,
    'base_score': 0.5,
    'missing': None,
    'silent': True,
    'nthread': 4,
    'seed': 42
}

num_round = 10
xgboost.cv(param, train_data,
           num_boost_round=num_round, metrics=['mlogloss'],
           feval=ndcg5_score)
```

Con esos simples comandos estamos lanzando un Boosting sobre nuestros datos de entrenamiento, con unos parámetros específicos y con una función de evaluación personalizada. Así podemos decidir, con mayor claridad, si descartamos o no la solución.

Los parámetros más importantes para mejorar el resultado han sido, por orden de importancia: el número de estimadores usados, la profundidad de los mismos, el coeficiente de aprendizaje, y el peso de los nodos hijos. También se ha percibido una mejora mínima alterando el número de muestras, por árbol y por nivel.

Como veremos en la sección 5.6, se han explorado una gran cantidad de posibles parámetros y combinaciones entre ellos para obtener la mejor solución.

5.3. Estrategias Usadas

Normalmente, con la elección de un buen algoritmo y un buen preprocesamiento, obtendríamos un clasificador bastante decente. Para una competición de este estilo, donde la gente está exprimiendo todo lo posible el problema, se han de aplicar más fases y técnicas.

Se ha experimentado con multitud de técnicas para modificar las predicciones del clasificador entrenado, a continuación comentaremos brevemente algunas las más interesantes. En estas técnicas se han usado algoritmos para igualar las clases, debido al gran desbalanceamiento de éstas en los datos.

En primero lugar, dada la naturaleza del reto, se implementaron una serie de funciones para hacer clasificación One vs All. Esto es, entrenaremos 12 clasificadores para decidir la probabilidad de que el usuario reserve alojamiento en un país específico.

Posteriormente se pasó a implementar un clasificador agregado One vs One, lo cual nos proporciona 66 clasificadores que enfrentan a clases. Podemos verlo como una matriz que enfrenta 12 países entre sí. Dada esta matriz, existen diferentes enfoques modernos[10] para tratarla y así intentar eliminar aquellos clasificadores no competentes. Normalmente se suele hacer la predicción con aquella clase que tenga más votos en la matriz o en el que las probabilidades agregadas sean mayores. Estas son dos alternativas que se han implementado para mejorar esto:

- Dynamic OvO: Aquellos clasificadores, en los que su salida no es relevante, son marcados como no competentes. Para determinar aquellos que no son competentes, se explora el vecindario de cada clase y se eliminan aquellas columnas de la matriz correspondientes a las clases que no están en el vecindario. Este enfoque es muy simple y ha dado buenos resultados.
- Relative Competence: Se basa en Dynamic OvO pero en vez de eliminar aquellas columnas de las clases que no están presentes en el vecindario, les asigna un peso. Las clases más cercanas a la instancia se verán menos afectadas que las lejanas. Esto hace alcanzar un equilibrio y mejora al Dynamic OvO en este problema.

Estos enfoques mejoran los clasificadores para el conjunto de entrenamiento, pero en el conjunto de test se vieron perjudicados, ya que la distribución de las clases no era exactamente igual. Se abandonó este enfoque, aunque es interesante y una buena alternativa si no se quiere mejorar apilando diferentes modelos, como veremos a continuación.

5.4. Model Stacking

Una de las ideas más novedosas que he descubierto a lo largo de la competición, ha sido cómo mejorar la generalización de los clasificadores simplemente agrupándolos con algún criterio.

Podemos empezar simplemente haciendo un meta-clasificador muy simple, donde la predicción final se realizará por votos entre los diferentes clasificadores. Esto, en general, no es buena idea, ya que estamos dando el mismo valor a todos los distintos clasificadores. Usualmente se le añade peso a cada uno y se usa la seguridad con la que ese clasificador vota a esa clase en vez del voto simple. [11]

Enfoques más complejos implican por ejemplo, entrenar un clasificador (máquina de soporte vectorial, árboles, KNN, ...) con los resultados de N clasificadores. Esto hace que no tengamos que darle peso nosotros mismos, sino que se encargará el clasificador. Además de esto, pueden aprender que clases se le dan mejor a los distintos clasificadores, y dependiendo de la entrada, cambiar los pesos dinámicamente.

Para este problema, en los últimos días se realizaron pruebas siguiendo este concepto. Inicialmente con un clasificador con peso en los votos y posteriormente entrenando un algoritmo de *Boosting* para diferente número de clasificadores usados. Esto proporciona un descenso en la varianza, pero puede aumentar el sesgo final de la respuesta.

Para intentar mejorar esta última idea y, de alguna forma, captar también la evolución de los gustos de los usuarios con el paso del tiempo, el clasificador final fue un *Boosting* con:

1. Resultado de la clasificación usando Boosting con todos los datos y el preprocesamiento estándar.
2. Resultado de la clasificación usando Boosting con los últimos 3 meses de datos y el preprocesamiento estándar.
3. Resultado de la clasificación usando Boosting con los usuarios del 2014 y el procesamiento estándar.
4. Resultado de un Random Forest con todos los experimentos añadidos que no han sido por si solos beneficiosos para el Boosting. También se incluyen los datos de las estadísticas de los países y del géneros/edad.

Este último enfoque ha sido el que ha proporcionado el mejor puesto en la clasificación, dado que usa diversa información.

5.5. Uso de la Validación Cruzada para Evaluar Soluciones

Es importante recalcar que todas las evaluaciones se han realizado mediante la técnica de Validación Cruzada. Se ha explorado también con Validación Cruzada con clases balanceadas, donde en cada sección de la división inicial, las clases están repartidas en la misma proporción en en el set de datos original. Al final se ha usado el enfoque estándar.

Para mejorar la rapidez de la Validación Cruzada, ésta se ha paralelizado entre distintos nodos en Hércules, para ello se ha realizado un script en Python que permite gestionar los resultados de cada sección de validación y calcular la media del error.

Esto permite ejecutar secciones de la Validación Cruzada (normalmente 5) en nodos independientes y con esto reducir el tiempo en obtener los resultados.

5.6. Grid Search: Exploración de los posibles Parámetros

Para la búsqueda de los mejores parámetros en cada clasificador, se aplica el concepto de Grid Search. Esto significa que podemos hacer bucles para explorar con diferentes valores y combinaciones de parámetros. Si tenemos suficiente capacidad de procesamiento, es lo ideal a la hora de optimizar los parámetros, pero, como es de esperar, es una operación muy pesada.

De nuevo, gracias al servidor Hércules, se puede aprovechar el sistema de colas Open Grid Scheduler para repartir el trabajo entre distintos nodos. Se ha creado un script[2] para repartir el entrenamiento de clasificadores entre nodos dada una serie de parámetros. Este script también controla la salida y proporciona como resultado final los parámetros que han proporcionado el mejor resultado.

Esto nos deja con el siguiente esquema de paralelización:

1. Primeramente obtenemos la lista de parámetros y se crea un clasificador por cada parámetro.
2. Por cada clasificador se lanzan 5 instancias en nodos diferentes para poder hacer una Validación Cruzada más rápida.
3. A nivel de nodo tenemos el algoritmo Boosting ejecutándose en paralelo, aprovechando el máximo número de núcleos disponibles.

Por ello, lo anteriormente mencionado, proporciona un vector con los resultados por cada clasificador. Este vector es el que finalmente guardamos, junto con un archivo que ha registrado todo lo que ha ocurrido, por si llegáramos a necesitar replicarlo.

[2] `https://gist.github.com/davidgasquez/7ccdee4361f07e0029f9`

Capítulo 6

Soluciones para Airbnb New User Bookings

La solución final al reto propuesta por mi, no ha sido demasiado compleja en relación con la de otros competidores, los cuales que han aplicado otras técnicas. Al haber desarrollado un preprocesamiento de calidad, el algoritmo no ha necesitado ser tan complejo. El proceso de preprocesamiento, entrenamiento, post-procesamiento y evaluación tarda alrededor de 4 horas. Es posible paralelizar diversos algoritmos al mismo tiempo y obtener distintas evaluaciones, lo que alivia el tiempo dedicado a ajustar los hiper-parámetros.

Aquello que me ha hecho despuntar en la tabla de clasificación de la competición ha sido:

- Una codificación correcta de las fechas en diferentes variables en lugar de una sola.
- Añadir diversas estadísticas sobre las acciones y los segundos que pasan entre ellas por usuario.

- Añadir información extra de las vacaciones para enriquecer el problema.
- Usar XGBoost en lugar de Random Forest y tener la capacidad de automatizar la búsqueda de hiper parámetros.
- Hacer un apilamiento de los resultados para eliminar la varianza final y usar diferentes modelos a la hora de hacerlo para captar distintas señales.

6.1. Resultados del a Competición

Las evaluaciones se hicieron sobre el archivo de test que se tenía desde el principio, no siendo evaluados todos los usuarios, sino solamente un porcentaje de los mismos (30 %). Posteriormente se dio un resultado final, evaluando aquellos que no habían sido evaluados previamente.

Los primeros modelos dieron como resultado de la evaluación, valores alrededor de 0.685. Este resultado fue mejorado hasta un máximo de 0.88070. Este sería un gráfico de la distribución de los resultados entre todos los participantes:

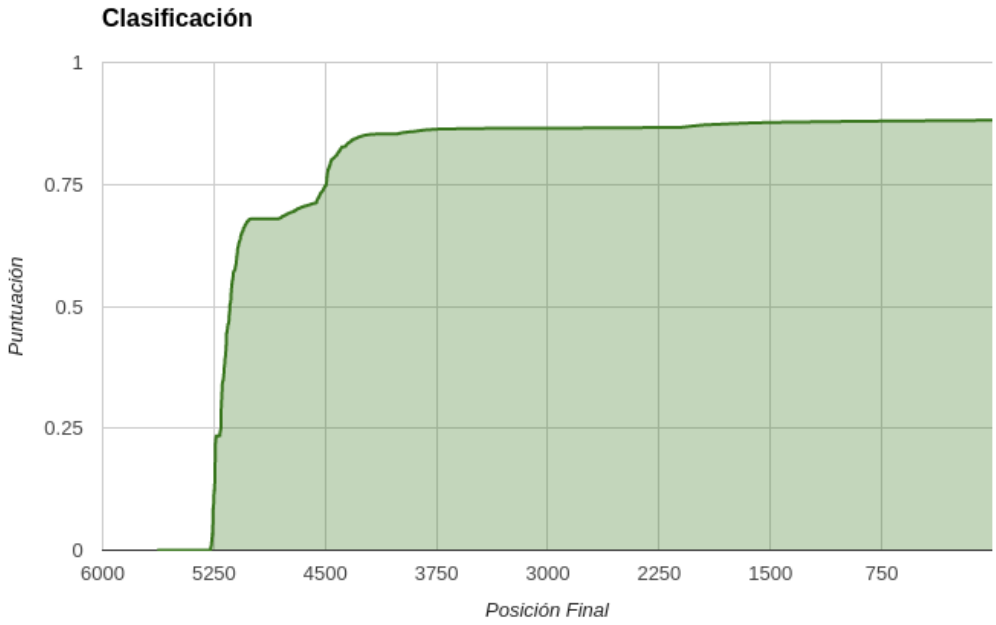

Figura 6.1: Distribución de la tabla de clasificación.

6.1.1. Clasificación en la Tabla Clasificatoria

La competición estuvo muy igualada durante los últimos días, siendo el valor de la posición 100 en la tabla de clasificaciones pública, 0.88070 y el ganador obteniendo un 0.88210.

Entre mis logros destacaría haber estado en el puesto 13 de la clasificación una semana antes de que esta finalizara y una vez finalizada alcanzar el puesto 40 con el valor 0.88090 en la tabla de clasificación pública tras haber enviado más de 200 soluciones para evaluar (con un límite de 5 al día). En total participaron 5632 personas enviando un total de 32650 entradas al concurso.

Finalmente, en la tabla de clasificación privada (usando el 70 % de usuarios que antes no habían sido evaluados) obtuve el puesto número 140 de 5632. La clasificación final fue poco previsible, y personas que se encontraban en la posición 100 han conseguido estar ahora entre los 10 primeros. Esto es debido a que existía una diferencia entre los datos con los que se valuaban inicialmente en la tabla de clasificación pública y los datos que se han usado en la tabla de clasificación privada.

A posteriori, se ha comprobado que algunos métodos usados previamente y descartados han sido mejores que los enviados finalmente en el nuevo set de datos. Habría accedido a puestos superiores si hubiera usado dichas soluciones.

6.2. Otros Enfoques Posibles

Los enfoques más comunes han sido muy similares al que se ha desarrollado en este proyecto, aunque sin las optimizaciones del mismo. Los usuarios que se han decantado por usar *XGBoost*, lo hacían en servidores con unas características técnicas muy superiores.

La otra vertiente que ha tenido buenos resultados, es la que ha seguid del ganador. Éste ha realizado una mezcla entre *XGBoost* y redes neuronales para hacer la clasificación. Podemos ver un esquema de su solución en el siguiente gráfico:

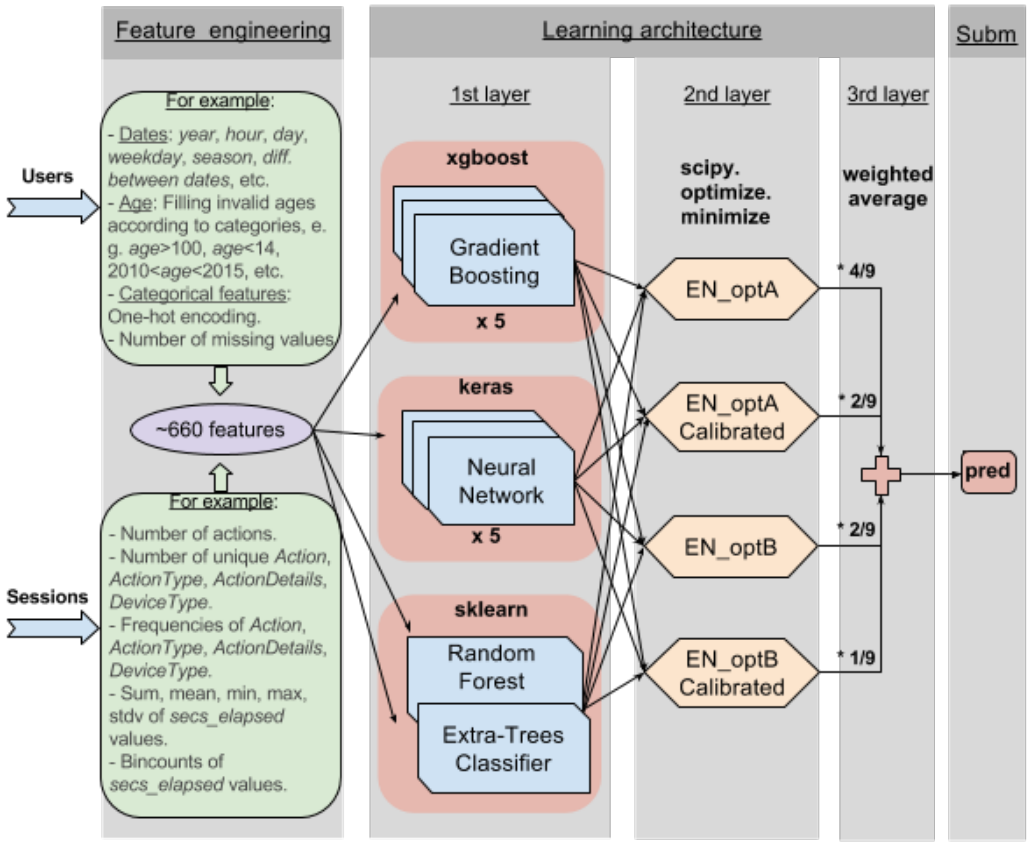

Figura 6.2: Estructura de la solución del ganador de la competición.

6.3. Reproducibilidad de las Soluciones

Para hacer todos los resultados replícales en un futuro y poder estar seguros de los métodos seguidos, se han fijado desde etapas muy tempranas del proyecto semillas para los números aleatorios. Esto es importante debido a la agrupación de personas en la clasificación. Cualquier cambio, por muy pequeño que sea, ha de estar registrado y poder ser replicado para intentar mejorarlo.

Al estar usando distintos ordenadores, y como elemento extra, se le ha añadido soporte para usar *Docker*, una plataforma que permite la virtualización y abstracción de la aplicación para el S.O. A su vez nos ofrece aún más estabilidad a la hora de generar números aleatorios.

6.3.1. Posibles Mejoras para el Futuro

Tras acabar la competición y ver las soluciones que han propuesto el resto de participantes, creo que existe, como casi siempre, espacio en el que mejorar. Para empezar, el uso de redes neuronales en este ámbito me parece muy interesante para conseguir un buena clasificación final, además de ofrecer un amplio espectro de posibilidades extra.

También valoro el gran potencial que ofrece saber agrupar modelos y usar meta-modelos para mejorar los resultados, como lo ha hecho el ganador. Quizás no es la solución más óptima para tener en producción, donde devolver un resultado rápidamente es vital, pero sí que es una solución ingeniosa y con mucho potencial.

Por otra parte, las grandes alteraciones en la clasificación final se han debido a que la generalización no se estaba realizando de forma correcta. La mayoría de usuarios que estábamos en las primeras posiciones cometíamos el mismo error, los algoritmos que estábamos enviando estaban haciendo *overfitting* involuntario a las instancias que se evaluaban en la tabla de clasificación pública. Se pude mejorar la forma de evaluar cada clasificar cada técnica usando enfoques que modifican Validación Cruzada para ofrecer unos resultados más reales y así estar más seguros de la solución final.

Capítulo 7

Lecciones Aprendidas

La competición en general me ha servido para afinar mis conocimientos en Machine Learning, así como descubrir bastantes más métodos, algoritmos y técnicas relacionadas. También he mejorado mucho en cuanto al uso del lenguaje de programación Python y su relación con Scikit Learn.

Una de las lecciones más importantes que he aprendido, es, sin duda, la de enfrentarme al problema iterativamente y estar preparado para cambiar el totalmente enfoque en caso de que fuera necesario. Esto me ha permitido gran flexibilidad a la hora de ir desarrollando el proyecto. Creo que ésta es una lección que se aprende haciendo proyectos de este tipo y es muy importante a la hora de trabajar en este ámbito.

Otra de las lecciones aprendidas, está relacionada con el principio comentado anteriormente de mantener siempre un enfoque sencillo. En diferentes etapas de la competición me he dado cuenta de que muchas de las funciones y técnicas usadas podían implementarse de manera más simple sin mucha penalización. El mejor ejemplo de ellos fue a la hora de sacar las estadísticas de los segundos. En lugar de buscar como optimizar la matriz por mi mismo, aplicando diferentes transformaciones a la hora de paralelizar usando código C u otras alternativas, se podía haber usado desde el principio el hashing y así simplificar el problema.

Por otra parte, aunque siempre se habla de la importancia del preprocesamiento, no tenía un ejemplo real donde pudiera ver como afecta cada paso al resultado final. Realizar esta competición me ha otorgado una experiencia para valorar el preprocesamiento como una de las fases más importantes, siempre que el problema esté en un ámbito parecido.

Y para finalizar, también me gustaría comentar, que además he aprendido a tener en cuenta las distintas formas de evaluar una solución para representar correctamente la generalización. Como se ha dicho anteriormente, al estar siempre intentando mejorar el resultado que Kaggle otorga y no el personal que se obtuvo de la Validación Cruzada, perdí varias posiciones.

Capítulo 8

Conclusión

He de decir que estoy bastante contento por el trabajo realizado durante este proyecto. Ha sido muy interesante enfrentarme a él y estar compitiendo con gente de mucha reputación en este ámbito, perteneciente a universidades y empresas muy famosas. Haber obtenido diversas menciones y, tras la popularización del script que presenté para la exploración de usuarios, ha hecho que varias personas contacten conmigo para debatir aspectos de la solución.

Profesionalmente también ha sido muy importante para mi, ya que meses después de la finalización del proyecto, este me ha servido como el mejor ejemplo dentro de mi portafolio personal de trabajos relacionados con el Aprendizaje Automático. He de comentar que después de unos meses, Airbnb ofreció mis datos a otra empresa americana en la cual, tras haber pasado 3 entrevistas, estoy actualmente trabajando como analista de datos. Este trabajo ha sido gracias al esfuerzo y dedicación empleados en esta competición.

Durante el desarrollo del proyecto he implementado una serie de funciones para que los clasificadores *XGBoost* puedan usar la interfaz de *Scikit Learn*, la cual permite hacer una selección inicial de las variables que el clasificador va a tener en cuenta. El código que he desarrollado está actualmente en revisión por los desarrolladores de la librería *XGBoost* y próximamente será añadida como nueva funcionalidad. Además de esta colaboración, también he participado activamente en la librería *Scikit Learn* de manera abierta a través de la plataforma para compartir código *Github*.

Todo el proyecto, así como los pasos seguidos, el historial de modificaciones, modelos usados, parámetros explorados y ficheros auxiliares se encuentran disponibles públicamente en el enlace `https://github.com/davidgasquez/kaggle-airbnb`.

Es un honor haber podido colaborar en librerías tan importantes y con gente tan capacitada, de la cual he podido aprender durante este periodo. Espero poder seguir ampliando mis conocimientos sobre Machine Learning y Data Science en un futuro y poder enfrentarme a otros problemas como este.

Bibliografía

[1] B. Chesky. (2008). Airbnb web, dirección: https://www.airbnb.com/.

[2] D. Cournapeau. (2007). Scikit learn web, dirección: http://scikit-learn.org/stable/.

[3] S. Jones. (2015). Gpu revolution, dirección: https://blogs.nvidia.com/blog/2015/12/10/machine-learning-gpu-facebook/.

[4] P. Domingos, «A few useful things to know about machine learning», *Communications of the ACM*, vol. 55, n.º 10, págs. 78-87, 2012. DOI: 10.1145/2347736.2347755.

[5] B. Wertz. (2016). Data, not algorithms, is key to machine learning success, dirección: https://blogs.nvidia.com/blog/2015/12/10/machine-learning-gpu-facebook/.

[6] D. Walter. (2015). Overfitting, regularization, and hyperparameters, dirección: http://dswalter.github.io/blog/overfitting-regularization-hyperparameters/.

[7] R. Fisher, «The use of multiple measurements in taxonomic problems», *Communications of the ACM*, 1936. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

[8] T. K. Ho, «Random decission forests», 1995.

[9] J. H. Friedman, «Greedy function approximation: A gradient boosting machine», *The Annals of Statistics*, vol. 29, n.º 5, págs. 1189-1232, 2001.

[10] M. Galar, A. Fernández, E. Barrenechea, H. Bustince y F. Herrera, «Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers», *Pattern Recogn.*, vol. 46, n.º 12, págs. 3412-3424, dic. de 2013, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2013.04.018. dirección: http://dx.doi.org/10.1016/j.patcog.2013.04.018.

[11] R. Opitz D.; Maclin, «Popular ensemble methods: An empirical study», *Journal of Artificial Intelligence Research*, vol. 11, págs. 169-198, 1999. DOI: 10.1613/jair.614.

Referencias Web de Interés

- http://amueller.github.io/sklearn_tutorial/#/
- http://nerds.airbnb.com/overcoming-missing-values-in-a-rfc/
- https://github.com/ianozsvald/data_science_delivered
- https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf
- https://github.com/rasbt/python-machine-learning-book
- http://nerds.airbnb.com/architecting-machine-learning-system-risk/
- http://nbviewer.ipython.org/github/agconti/kaggle-titanic/blob/master/Titanic.ipynb
- http://dswalter.github.io/blog/overfitting-regularization-hyperparameters/
- http://fastml.com/what-is-better-gradient-boosted-trees-or-random-forest/
- https://buhrmann.github.io/sklearn-pipelines.html
- https://michelleful.github.io/code-blog/2015/06/20/pipelines/
- https://www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/13370/how-to-combine-models-in-python
- http://nbviewer.jupyter.org/github/rasbt/pattern_classification/blob/master/machine_learning/scikit-learn/ensemble_classifier.ipynb

- https://github.com/wdm0006/categorical_encoding
- http://rasbt.github.io/mlxtend/
- https://github.com/rasbt/pattern_classification
- http://nbviewer.jupyter.org/github/rasbt/pattern_classification/blob/master/preprocessing/feature_encoding.ipynb
- http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html