Grado en Ingeniería de
Tecnologías de Telecomunicación

Trabajo Fin de Grado

CLASIFICADOR DE PERFILES SOCIALES EN TWITTER

SOCIAL PROFILES CLASSIFIER ON TWITTER

Fernando Marcos-Alberca Lizcano

Cuenca, 2022

Campus Universitario | 16071 Cuenca | Telf.: (+34) 969 179 100 (+34) 902 204 100 | Fax.: (+34) 969 179 119
politecnicacuenca.uclm.es | politecnica.cu@uclm.es

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi familia por darme le oportunidad y el ánimo de estudiar, no deja de ser algo que no todo el mundo se puede permitir por muchas condiciones y, ellos siempre han estado ahí para animarme a seguir formándome sin parar. También por apoyarme, estos años en la carrera han tenido momentos en los que me he sentido muy motivado, pero también momentos de sentir que no podía dada la complejidad de algunas asignaturas y, aun así, a pesar de las adversidades han confiado en mí.

A mis amigos del pueblo, los de toda la vida, que me permiten desconectar de la vida del estudiante para charlar sobre multitud de temas, debatir y salvar el mundo con nuestras ideas. Agradecer a los amigos que he conocido en la etapa universitaria, aquellos de la carrera, donde se ha formado un grupo muy familiar con los que se comparten intereses, entre ellos académicos y profesionales y, que me han ayudado infinidad de veces con exámenes y trabajos; también al resto de amigos que se han hecho estos años, como durante la beca Erasmus, donde se han vivido experiencias muy singulares y se han creado lazos que nunca se olvidarán, o los diferentes y peculiares compañeros de piso que he tenido, cada uno a su manera me ha ayudado y enseñado mucho sobre la convivencia. Debo incluir a todas aquellas personas anónimas o desconocidas de internet que desinteresadamente suben contenido educativo y documentación a la red, que a tantas personas nos han ayudado a nivel académico y profesional, además de aquel docente que en los primeros años complementó nuestros conocimientos en física para un mejor aprendizaje, un abrazo allá donde estés.

Agradecer a todos los profesores de la Escuela Politécnica de Cuenca que me han abierto las puertas al mundo de las telecomunicaciones, a los que se preocupan por la enseñanza, por los alumnos y son capaces de ver más allá del temario, entendiendo que el mundo profesional tiene muchos matices que no se incluyen en una ecuación, en especial a mi tutor en este TFG, José Antonio Ballesteros Garrido, que con su ayuda y gran paciencia se ha conseguido elaborar este proyecto. Por último, mencionar a Telefónica y su programa Tutoría del que he sido parte, que ha puesto a disposición a profesionales del sector como Julio Gómez Ortega compartiendo sus conocimientos para poder desarrollar este TFG.

Gracias.

RESUMEN

El creciente uso de las redes sociales por particulares, empresas o gobiernos las ha convertido en el medio de intercambio social más usado. Creadas para compartir opiniones, información y elementos digitales, ofrecen una gran cantidad de información que se ha llegado a convertir en un objetivo de personas, política y economía. Extraer la información a partir de las interacciones en redes sociales puede ayudar a medir riesgos, impactos de crisis, ataques reputacionales, estudios de mercados o incluso analizar *influencers* para un negocio.

Esta gran cantidad de información que se genera crece rápidamente y requiere de métodos especializados para poder analizar su naturaleza, encontrando *bots*, perfiles falsos y cuentas con variedad de identidades e intenciones. Se puede llegar a encontrar un lugar para el crimen, estafas o manipulaciones, haciendo necesario el estudio y clasificaciones de los perfiles en las redes sociales a través del desarrollo de mecanismos automáticos.

En este Trabajo de Fin de Grado (TFG), se ha diseñado un algoritmo, basándose en mecanismos de *machine learning*, que permiten buscar patrones de comportamiento en base a unos datos de entrada. Se analizarán los mecanismos que esta tecnología ofrece, buscando el que mejor se adapte a las necesidades del proyecto teniendo en cuenta la extracción de datos, su preprocesado, clasificación y enlazando con una gran base de datos que permita almacenar toda la información.

El objetivo final será clasificar perfiles sociales de *Twitter* basándose en su comportamiento, dividiéndolos en tres grupos: “Medios”, “*Inflcuencers*” y “Otros”, almacenándolos en una base de datos y generando información relativa a la naturaleza de cada perfil, preparada para su análisis. El proyecto se centra en los perfiles de medios de comunicación: perfiles usados para entretener, formar opinión, informar, etc. Se buscará en primer lugar definir qué son, para luego caracterizar su comportamiento y finalmente diferenciarlos del resto de perfiles.

ABSTRACT

Social networks have experienced an increase in their use, becoming the most important means of social exchange for individuals, companies or governments. They were created to share opinions and digital elements, giving rise to a large amount of information that today is of personal, political and economic interest. Extracting this information, starting from the interactions in the social network, can be used to measure risks, crisis impacts, reputational attacks, market research or even analyze influencers for a particular business.

Information grows very quickly and in order to analyze it, very specialized methods are needed, leading to the discovery of bots, false profiles and accounts with different identities or intentions. Social networks can be a place for crime, fraud or manipulation and it is therefore necessary to carry out a study that classifies the profiles in these networks through the development of automatic mechanisms.

For the development of this Final Degree Project, an algorithm has been designed that makes use of machine learning mechanisms, allowing behavior to be analyzed from input data. The mechanisms of this technology will be analyzed to find the one that works best for the needs of the project, taking into account the entire process such as extraction, preprocessing, classification and connection to a database that stores all the information.

The final objective is to classify social profiles of the Twitter social network based on their behavior, differentiating between three groups: "Media", "Influencers" and "Others", and then store them in a database from where they will be analyzed. The interest group of the project is "Media", which includes the media, that is, profiles created to entertain, form opinions, inform, etc. In the first place they will be defined, to later analyze their behavior and differentiate them from the rest of the groups.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

- ATOM: Atom Web Feed Format
- API: Application Programming Interface
- BSON: Binary JavaScript Object Notation
- CSV: Comma-Separated Values
- GPS: Global Positioning System
- HTTP: Hypertext Transfer Protocol
- IDE: Integrated Development Enviroment
- IoT: Internet of Things
- JSON: JavaScript Object Notation
- NFC: Near-Field Communication
- RAM: Random Access Memory
- REST: Representational State Transfer
- TFG: Trabajo Fin de Grado
- URL: Uniform Resources Locator

PARTE I

MEMORIA

CAPÍTULO 1

INTRODUCCIÓN

ÍNDICE

En este apartado se expondrá una visión global del proyecto además de los motivos que han llevado a la elección del tema y la estructura del trabajo.

1.1 Motivación

A lo largo de toda la carrera universitaria se presentan muchos campos y el alumno empieza a darse cuenta de la alta complejidad que la tecnología conlleva y que cada asignatura es tan solo una pincelada superficial de todo lo que lleva tras de sí. Todo esto, de la mano de un punto de vista positivo, lleva a la conclusión de un enorme abanico de posibilidades que explorar y del que empaparse.

A destacar, algunos de los campos de mi interés siempre han sido la ciberseguridad, las redes de comunicación y la inteligencia del dato. Este interés, puesto que siempre ha tenido un grado de latencia, se disparó a lo largo del último curso, especialmente en una optativa relacionada con la seguridad y las redes de datos. Todo comenzó tras contar con la sugerencia y apoyo del profesor de dicha asignatura de coger como tema alguno de los retos propuestos por ElevenPaths de su programa Tutoría. Este programa plantea desafíos por parte de profesionales de Telefónica, que requieren de un proceso de solicitud, aprobación y un desarrollo tutorizado por dichos profesionales.

Las relaciones sociales, desde un punto de vista más sociológico han despertado en mí una gran curiosidad desde pequeño, analizando e intentando comprender su funcionamiento. Con el nacimiento de las redes sociales, estas relaciones han dado una vuelta de hoja y se tiene la posibilidad de analizar, para bien o para mal, las interacciones entre las personas.

Todo esto confluye en la elección de un reto del programa Tutoría de ElevenPaths que busca analizar y clasificar perfiles sociales en la red social *Twitter*, permitiendo una mejor comprensión de su comportamiento y el análisis de datos, dando paso al desarrollo de este proyecto.

1.2 La era de la información

Una red social es una web donde usuarios intercambian información y contenido multimedia, creando una comunidad virtual e interactiva [1].

Estas redes sociales se pueden usar de múltiples formas, desde usos más cotidianos como relacionarse con los amigos y familiares como si se estuviera cara a cara, pasando por conectar con personajes públicos de diferentes ámbitos como la política, música, *influencers*… hasta un uso publicitario de las empresas o informativo de los medios de comunicación. Sin embargo, existen otros usos no tan visibles de las redes sociales que están presentes como las *fakenews*, el robo de información, terrorismo, *ciberbuying* o incluso contenidos para adultos.

La mayoría de la población mundial dispone de acceso a Internet contando con opción de subir contenido a este y las redes sociales no son una excepción. Como se ha mencionado, todas las interacciones y usos que se han comentado suponen una gran fuente de información que puede ser, para bien o para mal, analizada y explotada.

1.3 Objetivos del trabajo

El objetivo principal de este trabajo consiste en desarrollar mecanismos que puedan explotar la información generada en *Twitter*, buscando clasificar perfiles de usuarios en grupos basándose en su comportamiento. En concreto, se buscará detectar y clasificar aquellos perfiles que se puedan identificar como medios de comunicación, pudiendo diferenciarlos con precisión de aquellos que no lo son.

Otros objetivos serán, la creación de una base de datos que almacene las cuentas divididas en diferentes grupos, mostrando con detalle las diferentes características que identifican la cuenta y la validación del modelo diseñado a partir del análisis de los resultados.

1.4 Estructura del trabajo

El contenido del TFG se ha distribuido en seis capítulos explicando los siguientes contenidos:

Capítulo 1: INTRODUCCIÓN

En este capítulo se exponen los motivos que han llevado a la realización de este trabajo y los objetivos a alcanzar.

Capítulo 2: MARCO TEÓRICO

En este capítulo se explicarán las bases teóricas de las que parte este proyecto, desde una visión general de *machine learning* hasta el algoritmo utilizado con sus distintas características.

Capítulo 3: DESARROLLO DEL PROYECTO

En este capítulo se expone cómo se ha programado el clasificador de perfiles. Para ello, se hace un recorrido de las diferentes partes que sigue el algoritmo y se exponen las herramientas usadas.

Capítulo 4: RESULTADOS

En este capítulo se analizarán los resultados obtenidos del entrenamiento y pruebas del algoritmo, validando en el proceso los distintos resultados obtenidos.

Capítulo 5: CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se redactan las conclusiones extraídas de todo el proceso de desarrollo del proyecto, se comentará si se han alcanzado los objetivos propuestos y se propondrán líneas futuras buscando maneras de mejorar el proyecto.

CAPÍTULO 2

MARCO TEÓRICO

ÍNDICE

En este apartado se expondrán las bases teóricas y conceptos para el correcto desarrollo del proyecto. Se hará una introducción al *machine learning* llegando al algoritmo *Random Forest*, sus características y validación.

2.1 Machine Learning

Machine Learning es un campo de la Inteligencia Artificial que busca patrones a partir de datos. Un modelo de *marchine learning* es un modelo matemático, que, a partir de unos datos de entrada hace predicciones, encuentra patrones, y proporciona una salida de información que se genera cuando el algoritmo se entrena [11]. Existen distintos tipos de algoritmos de *machine learning* [23]:

- **Aprendizaje supervisado**: algoritmo donde las entradas son etiquetadas en base a unas clases. El algoritmo aprenderá en base a los datos etiquetados anteriormente. Uno de sus modelos más usados es el modelo de regresión lineal, que permite realizar previsiones de futuro, como por ejemplo determinar cuánto cambiarán las tasas de interés en una cartera de bonos.

- **Aprendizaje no supervisado**: los datos de entrada no son etiquetados, sino que el algoritmo aprenderá reconociendo patrones para poder etiquetarlos. Los modelos se centran en reconocer anomalías, por ejemplo, en análisis de fraudes o piezas mecánicas defectuosas.

- **Aprendizaje semisupervisado**: combinación del aprendizaje supervisado y no supervisado, es decir, hace uso de datos etiquetados y no etiquetados. Puede ser útil en casos donde se hace uso de la técnica de coentrenamiento, donde se entrenan dos o más sistemas cada uno en un conjunto de ejemplos, pero cada sistema usa características diferentes.

- **Aprendizaje por refuerzo**: algoritmo que aprende por ensayo y error en base a la retroalimentación de lo que le rodea, es decir, plantea estrategias en base a una experimentación con los datos hasta dar con el comportamiento ideal. Se puede encontrar en la búsqueda de optimización de costes o beneficios de una empresa.

- **Transducción**: el aprendizaje supervisado establece una relación entre las entradas y las salidas. Este aprendizaje funciona de forma similar, pero sin construir una función, en lugar de eso, a partir de las entradas y las categorías ya existentes realiza predicciones.

- **Aprendizaje multi-tarea**: aprendizaje que parte de un conocimiento previo del sistema, entrenando una tarea a partir de un conjunto de tareas relacionadas con esta. Es un aprendizaje usado para aprender varias tareas al mismo tiempo buscando aumentar la capacidad predictiva.

Tras un análisis de los tipos de algoritmos de *machine learning*, y sabiendo que se está buscando uno que permita clasificar y determinar con exactitud la categoría de los datos, se ha llegado a la conclusión de que un **aprendizaje supervisado** es la mejor opción. Las categorías por alcanzar se conocen y están predefinidas, por lo que lo mejor sería un algoritmo que se pueda entrenar permitiendo etiquetar los datos.

2.2 Aprendizaje supervisado

En el aprendizaje supervisado, la máquina aprende por los datos de entrenamiento que está etiquetada. Un algoritmo aprende a partir de datos de ejemplo y respuestas de destinos asociadas que pueden ser valores numéricos o etiquetas de cadenas, como clases para luego predecir la respuesta correcta cuando se presenta con nuevos ejemplos [23].

Este modelo es el que más efectividad tiene, pero en casos donde se trabaja con un gran tamaño de datos etiquetados requiere de una gran capacidad de procesamiento, y, teniendo en cuenta que para la detección de anomalías se requiere de grandes conjuntos de datos etiquetados no siempre es la mejor opción. Existen diferentes algoritmos de aprendizaje supervisado: regresión lineal, regresión polinomial, regresión logística, árbol de decisión, *Random Forest*, etc.

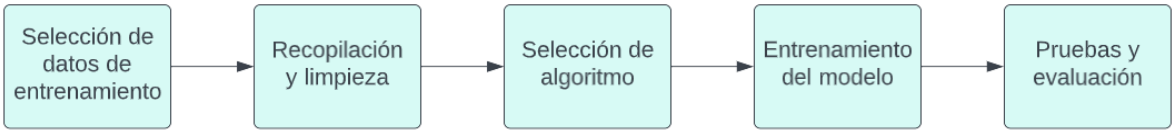


***Figura 2.1**: Diagrama aprendizaje supervisado (elaboración propia)*

Para seguir un modelo de aprendizaje supervisado, basándose en la figura 2.1 se seguirán estos pasos:

1. Se eligen los datos con los que se entrenará el modelo, tanto las variables independientes como las dependientes. Al tratarse de un aprendizaje supervisado estos datos irán etiquetados para que la máquina pueda agruparlos.

2. A partir de diversas fuentes se extraen los datos. Posteriormente y de la forma más rigurosa posible, se seleccionan los datos a utilizar, dejando una muestra preprocesada y lista para aplicarle el algoritmo seleccionado.

3. Se elige el algoritmo de aprendizaje supervisado que se adapte mejor a los datos y el uso deseado, pudiendo ser un algoritmo de regresión o de clasificación. Al elegir un algoritmo se debe tener en cuenta principalmente: la transparencia del algoritmo, el uso de la memoria y la precisión de la predicción.

4. Con los datos de entrenamiento el algoritmo va aprendiendo y creando una base de datos donde irá almacenando la información, como previamente los datos han sido etiquetados este paso se hace más sencillo. Para hacer una base de datos más fiable se deberá entrenar la máquina de manera repetida.

5. Pruebas y evaluación del modelo. Los parámetros del algoritmo se irán ajustando y optimizando, valorando los resultados y añadiendo nuevos datos.

2.3 Árbol de decisión

Se trata de un esquema que muestra las posibilidades lógicas de una secuencia de acciones, presentando los resultados en forma de árbol binario. Dependiendo de si la salida es cuantitativa o cualitativa se dirá que es un árbol de regresión o de clasificación [24].

El funcionamiento de ambos árboles de decisión es similar y se explicará a continuación. La única diferencia será la naturaleza de los nodos de decisión, para casos donde se busque predecir la pertenencia de los objetos a una clase, en base a unas características que definan dichas clases se escogerá un algoritmo que utilice árboles de

decisión cualitativos, mientras que si se trata de variables cuantitativas se escogerá un algoritmo que utilice árboles de decisión cuantitativos. En la figura 2.2 se observan los siguientes elementos de un árbol de decisión:

- **Nodo raíz**: representa la población o muestra que se podrá dividir en conjuntos homogéneos.

- **División o *split***: división de la población en subconjuntos, conocidos como nodos.

- **Nodo de decisión**: división de un subnodo en otros subnodos.

- **Hoja o Nodo final**: resultado final al no haber más divisiones de los nodos.

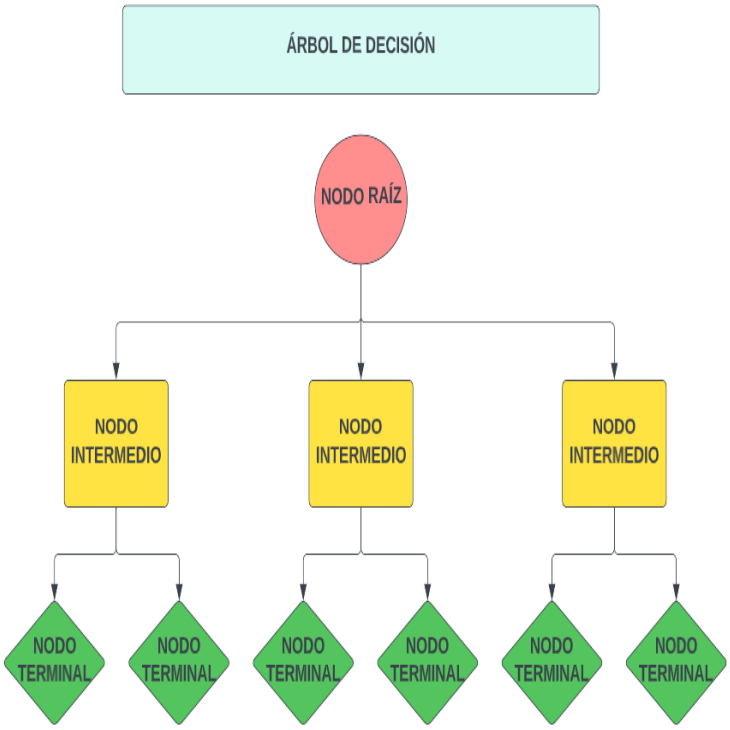


***Figura 2.2**: Esquema árbol de decisión (elaboración propia)*

El proyecto se desarrolla buscando clasificar cuentas en distintas agrupaciones, es por ello por lo que se escogerá un modelo de **árboles de decisión cualitativos**, que tendrá en cuenta las características que definen a cada clase.

2.4 Random Forest

Un algoritmo de clasificación es aquel que da como resultado una clase entre un número limitado de estas, entendiendo como clase categorías arbitrarias dado un problema. Por tanto, para el desarrollo de este proyecto en un principio había que analizar el objetivo para dar con el método más acertado. Existen varios algoritmos de clasificación, tales como el algoritmo de Bayes, el algoritmo KNN o el algoritmo *Random Forest*. Cada uno de ellos tiene sus ventajas y sus desventajas, estando enfocados a un objetivo.

Al inicio del planteamiento del proyecto se realizó una larga búsqueda de algoritmos y métodos de *machine learning* que permitieran clasificar las categorías planteadas. El algoritmo de Bayes es un clasificador probabilístico basado en la suposición de independencia de los atributos que puede ser útil en datos a pequeña escala y se trata de un algoritmo relativamente simple [16], sin embargo, necesita calcular la probabilidad previa y posee una alta tasa de error en las decisiones de clasificación. Otra opción fue el algoritmo KNN o también conocido como el algoritmo K-Vecino más cercano [21], que mide la distancia entre diferentes valores de características para la clasificación. Aunque se trata de un algoritmo muy preciso, se vuelve muy complejo cuando se tratan multitud de variables como es el caso y eleva en exceso el tiempo y el espacio computacional. Aunque se estudiaron e incluso se llegó a realizar alguna prueba sencilla se observó que no eran comparables con el algoritmo de clasificación ***Random Forest***, cuyo funcionamiento y justificación serán explicados a continuación.

Random Forest

Random Forest, es un conjunto de **árboles de decisión**, donde cada árbol depende de un vector aleatorio e independiente. La distribución de los árboles en el bosque (***forest***) es igual, pero de manera diferente, de esta forma se reduce la correlación y la varianza de las predicciones mediante la combinación de los resultados de varios árboles, siendo cada uno de ellos elaborado con distintos subconjuntos de la misma población (**proceso de *Bagging***). Todo esto se logra en el proceso de construcción de árboles a través de la selección de entradas de manera aleatoria, concretamente en la aplicación de la técnica ***Bootstrap*** (muestreo de datos con reemplazo) en la base, y dentro de cada nodo un conjunto de entradas aleatorias. Es decir, siendo M el número de entradas y m el número de muestras, se selecciona un $m<M$ de entradas antes de cada partición [25].

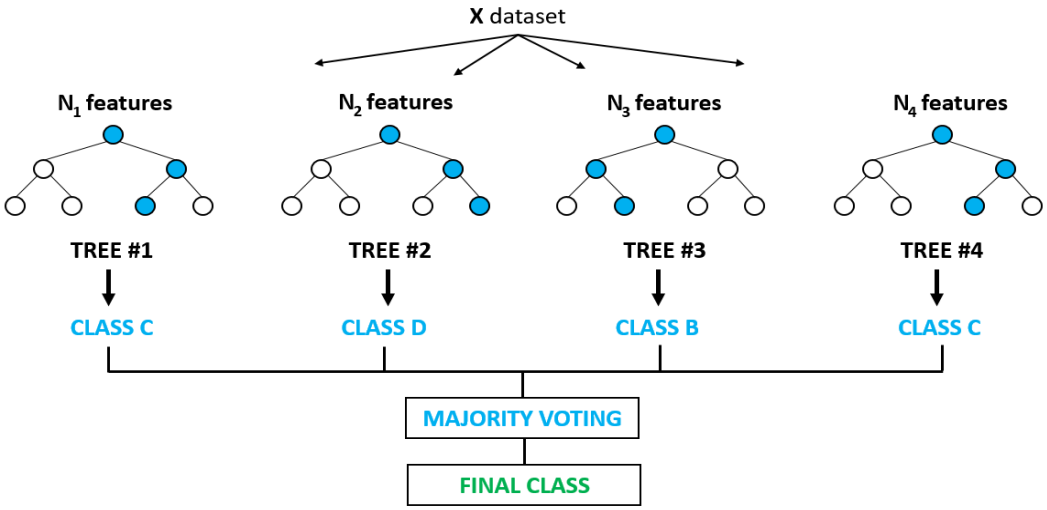


***Figura 2.3**: Esquema Random Forest [33]*

Basándose en la figura 2.3, que esquematiza el funcionamiento del algoritmo *Random Forest*, suponiendo la generación de *Ti (i=1, ..., t)* árboles, se puede explicar su funcionamiento como [26]:

1. La muestra de datos se divide en un conjunto de entrenamiento y otro de testeo.

2. Se construye un bosque aleatorio en la base de entrenamiento, donde cada árbol se construye cogiendo *"m"* datos con repetición (*Bootstrap*) de la base de forma aleatoria. Si hay *M* entradas, un número *m* de ellas de forma aleatoria se utilizará en cada nodo de decisión del árbol, con la condición de $m<M$, teniendo en cuenta que el valor de *m* será constante. Después se iteran los valores de las entradas en *m* con el fin de realizar una mejor partición y finalmente se repetirá el proceso de partición de los nodos dando lugar a subnodos hasta alcanzar el tamaño del nodo deseado, obteniendo el árbol *i*.

3. Para evaluar la capacidad predictiva del modelo entrenado con el conjunto de datos de testeo, siguiendo los criterios de partición de cada árbol desde el nodo raíz hasta el final, se asigna cada dato de testeo a cada valor estimado en los nodos terminales. El proceso se repetirá en todos los árboles y finalmente se promedia los valores predichos por cada árbol. Este último valor corresponderá al bosque.

Dado que se va a trabajar con grandes cantidades de datos y con multitud de variables a tener en cuenta, trabajar con árboles de decisión, que permiten escalar y trabajar con una *lógica booleana* que dan una respuesta fácilmente justificable es la mejor opción. Sin embargo, los árboles de decisión pueden acarrear problemas como sobreajuste, así que una combinación de ellos podría solucionar el problema, por tanto, ***Random Forest***, que realiza combinaciones de árboles de decisión, es el modelo elegido para el desarrollo del proyecto. Además, se dispone de una gran documentación e integración con herramientas como las que se van a usar. De ahora en adelante, se referirá a *Random Forest* cuando se hable de algoritmo de clasificación.

2.5 Impureza de Gini

Para la construcción de árboles de decisión se usa una métrica llamada ***Impureza de Gini***, que indica la probabilidad de clasificar erróneamente una observación. Un nodo será más homogéneo cuanta menor impureza tenga, siendo denominado puro cuando la impureza es cero.

Para calcularla se usa la siguiente ecuación [24]:

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \qquad (1)$$

Donde:

- **Gini**: Índice de Gini.

- ***Pi***: probabilidad de que un ejemplo sea de la clase *i*.

Si se tiene en cuenta la *Impureza de Gini* en la construcción de un árbol de decisión se comienza por cada división (como ya se ha mencionado anteriormente el nodo raíz se divide en otros subnodos), calculando en cada nodo hijo su impureza. Una vez se ha calculado la impureza para cada subnodo, se realiza un promediado de todos los valores, quedándose con el valor más bajo. Este proceso se repetirá hasta que se alcance una homogeneidad en todos los nodos, es decir, un valor bajo de impureza.

2.6 Entropía y Ganancia de Información

Otro factor a tener en cuenta en la formación de árboles de decisión es la ***entropía***. Esta mide cómo un árbol debe dividir los datos, pudiendo cuantificar el desorden del sistema. Para calcular la *entropía* podemos usar la siguiente ecuación [24]:

$$entropía = -\sum_{i=1}^{k} P\,(valor_i) \cdot log_2\big(P(valor_i)\big) \qquad (2)$$

Donde:

- ***entropía***: entropía

- ***P(valor$_i$):*** de la probabilidad de obtener el *i-ésimo* valor cuando se selecciona aleatoriamente uno del conjunto.

Sabiendo obtener la *entropía*, obtener la ***ganancia de información*** será importante para medir la cantidad de información que proporciona una característica y por tanto el orden de los atributos en los nodos de un árbol. La *ganancia de información* se puede medir:

Ganancia de información = entropía (padre) – entropía media (hijos)

2.7 Matriz de confusión y precisión

Para medir la efectividad de los clasificadores de *machine learning* se tienen en cuenta diferentes conceptos que surgen a partir de la clasificación que efectúa el clasificador, como lo son los verdaderos negativos, falsos negativos, falsos positivos y verdaderos positivos. Estos cuatro conceptos se recogen en la conocida **matriz de confusión** [31].

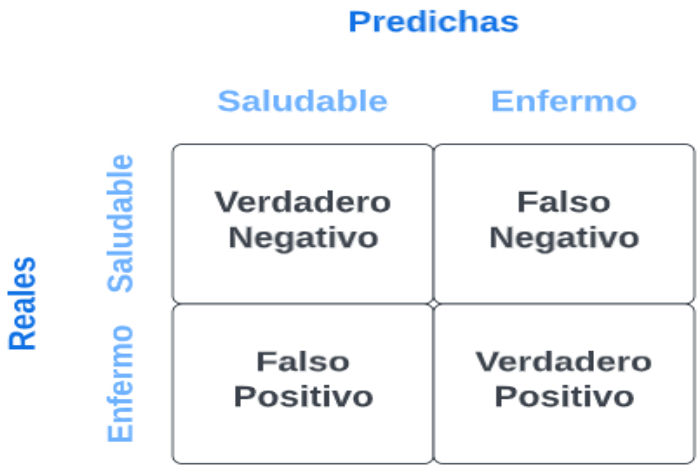


Figura 2.4: *Matriz de confusión (elaboración propia)*

Partiendo de la figura 2.4, si se diera un conjunto de datos de personas saludables y personas enfermas el objetivo del clasificador sería agrupar cada persona correctamente según sus características. Si los resultados fueran correctos, siguiendo la matriz de confusión, las personas enfermas se catalogarían como enfermos siendo verdaderos negativos, mientras que las personas saludables se colocarían como verdaderos positivos. En caso contrario, estaríamos hablando de falsos negativos y falsos positivos.

Los clasificadores ideales no existen, por lo que se encontrarán errores y es necesario poder medir los fallos de los resultados. Para ello se dispone de una métrica llamada precisión, que mide el número de predicciones correctas del total de predicciones, tomando valores en el intervalo [0, 1], siendo 0 el peor resultado y 1 el mejor. Para calcularla se hace uso de la siguiente ecuación, dividiendo el número de predicciones correctas entre el número total de predicciones:

$$Precisión = \frac{Predicciones\ correctas}{Total\ de\ predicciones} \tag{3}$$

2.8 Validación Cruzada

Cuando se realiza un modelo de *machine learning* es necesario validar y no sobrestimar su capacidad de predicción. Para la evaluación de algoritmos de *machine learning* uno de los métodos más conocidos es la **validación cruzada**. Este método consiste en un remuestreo, es decir, realizar divisiones de muestras de datos en subconjuntos que se complementan. Posteriormente se realiza un análisis dado el conjunto de entrenamiento y finalmente se valida el análisis en otro conjunto de validación. Por tanto, existirá un grupo de datos de entrenamiento y un grupo de datos de prueba [27].

Figura 2.5: *Validación cruzada 1* [30]

Dada la figura 2.5, donde cada rombo supone una instancia, es decir, un dato, si se cogieran todo el conjunto de datos (datos para crear el modelo) para validar las capacidades de predicción sería algo incorrecto, pues el algoritmo ya tendría todas las respuestas. Es por ello por lo que se divide en datos de entrenamiento con los que crear un modelo y datos de prueba con los que medir sus capacidades de predicción.

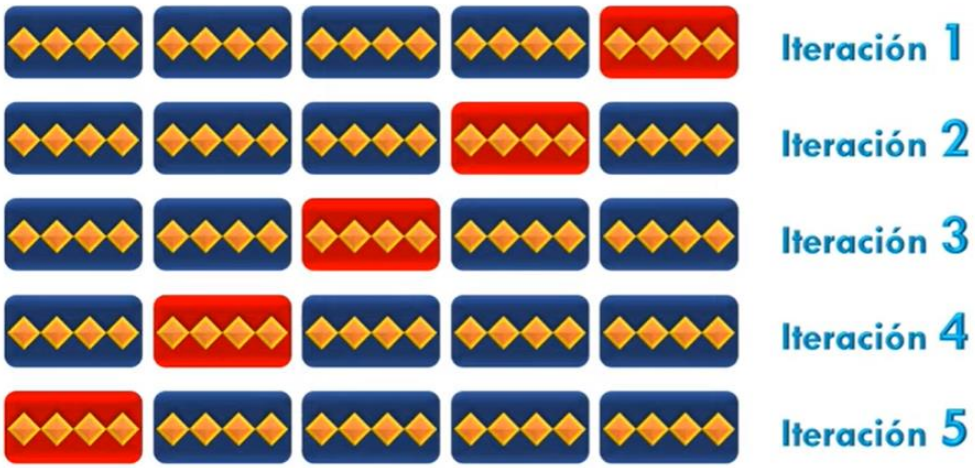


Figura 2.6: Validación cruzada 2 [30]

En el caso de la validación cruzada, que busca reducir la variabilidad, la división de los datos se realiza en partes iguales, donde cada una de ellas será usada como parte de pruebas mientras las demás forman parte del entrenamiento en cada iteración o pliegue (*k*). En el caso de la figura 2.6 se ha dividido el conjunto de datos en cinco (*k* = *5*), mostrando en rojo la parte que se usará como conjunto de datos de pruebas y cambiando para cada iteración. Finalmente se promedian los resultados obtenidos de cada iteración, obteniendo una estimación más realista de la capacidad del modelo.

CAPÍTULO 3

DESARROLLO DEL PROYECTO

ÍNDICE

En este capítulo se explican las herramientas usadas y el porqué de su elección. Se muestra el proceso seguido para el desarrollo del algoritmo incluyendo su entrenamiento, se detallan los tipos de datos analizados y cómo se preparan para su análisis.

3.1 Fuentes de datos

En internet se encuentra una gran cantidad de datos, cada segundo que pasa estos se multiplican y se requiere de un complejo control y grandes bases de datos para manejarlos. Al mismo tiempo, todos esos datos pueden suponer una fuente de información, que si se sabe gestionar dan como resultado una inteligencia capaz de predecir comportamientos tanto dentro como fuera de la red.

Se necesita en primer lugar, diferenciar entre distintas fuentes de datos [32], como los datos biométricos, basados en las características anatómicas personales incluyendo la genética, datos de las interacciones IoT, datos generados de la tecnología GPS o en los chips NFC, datos de transacciones registrados por departamentos de facturación, datos generados por los humanos como grabaciones en atención al cliente o datos generados en la web.

Una de las grandes fuentes de datos en la web son las redes sociales. El gran número de interacciones por parte de usuarios de todo el mundo, tratando temas desde lo profesional a lo personal, permiten a todo tipo de usuarios relacionarse y por tanto proporcionan una información muy variada. En este proyecto se hará uso de la red social *Twitter* como fuente de datos.

Twitter

Twitter es un servicio de *microblogging* creado por Jack Dorsey en 2006, aunque actualmente el director ejecutivo es Parag Agrawal [2].Se estima que en la actualidad tiene unos 330 millones de usuarios activos, generando más de 5000 millones de dólares anuales, con un valor de bolsa superior a los 13000 millones de dólares [3] [36]. Además, el 25 de abril de 2022 *Twitter* fue comprada por Elon Musk por 46.500 millones de dólares [37].

Twitter es uno de los medios más importantes a la hora de hablar de medios sociales. Es de destacar, que es usada por figuras públicas importantes como presidentes, actores y músicos. Esta red social permite enviar textos con un máximo de 280 caracteres conocidos como *tweets*. Estos se mostrarán en la página principal del

usuario que los publicó, pudiéndose ver por el resto de los usuarios de la red, siempre teniendo en cuenta la configuración de privacidad del usuario en cuestión.

Los usuarios pueden interactuar con otros usuarios de diversas formas, bien sea siguiendo (*followers*), marcando como favoritos *tweets* o incluso *retweetear* compartiendo la información. *Twitter* puede ser usado para muchos fines, entre ellos negocios, medios de comunicación o la simple publicación de la opinión particular de un tema. La mayoría de los usuarios son adultos mayores de edad que no han usado otra red social anteriormente [4].

3.2 APIs: Interfaces de programación de aplicaciones

Application Programming Interfaces (API) es el conjunto de reglas y protocolos dentro del software de aplicaciones que permiten la comunicación con otra aplicación sin mostrar cómo están implementadas, lo que supone una mejor administración, ahorro de tiempo y dinero, mayor sencillez en los diseños y flexibilidad [5]. Permiten a un usuario utilizar un software o una aplicación consultando, modificando o almacenando datos sin que sea necesario ingresar directamente, todo esto, dependiendo de los permisos de uso establecidos por parte del propietario de la API.

Una API tiene varios usos, como automatizar procesos o permitir nuevas funcionalidades, por ejemplo, en la rutina de las empresas en la emisión de facturas para completar el pedido de un cliente o gestionar los comprobantes y facturas del banco. El ejemplo anterior se trataría de una API privada, ya que solo sería accesible dentro de la empresa, sin embargo, también pueden ser públicas, como por ejemplo APIs de servicios en la nube como *Google, Amazon o Microsoft*. También podemos encontrar APIs locales dentro de un mismo dispositivo o remotas que permitan el acceso a un sitio diferente.

API de *Twitter*

La API de *Twitter* establece la comunicación con la red social Twitter, siendo una herramienta gratuita [8], permite la extracción de datos de la red social, el manejo de estos por parte del usuario y la conexión con otras aplicaciones. A través de la API se podrá acceder a funciones de *Twitter* como usuarios, mensajes directos, espacios, *tweets*, etc. Existen diferentes tipos de API de Twitter que se explicarán a continuación, para ello se han elaborado unas tablas que recogen a quién van destinadas, qué ofrecen, su funcionamiento y sus beneficios:

Search API

Esta API devuelve colecciones de *tweets* en base a peticiones y consultas muy concretas.

Destinada a	Desarrolladores
Ofrece	Acceso a los datos del core de *Twitter*. La API permite operaciones desde la web. Soporta formatos como XML, JSON, RSS y ATOM.
Funcionamiento	A partir de URLs por HTTP
Beneficios	Muestra un perfil completo del autor al momento de la escritura de un tweet

***Tabla 3.1**: Search API Twitter*

Streaming API

Permite el acceso en tiempo real a conjuntos de datos como ID de usuario, ubicaciones, muestreos o palabras clave.

Destinada a	Desarrolladores
Ofrece	Un *subset* de *tweets* en tiempo casi real
Funcionamiento	Por HTTP se proporciona un flujo de tweets en formato JSON.
Beneficios	Muestra un perfil completo del autor al momento de la escritura de un tweet. Se puede filtrar por palabras o usuarios y se puede obtener una muestra aleatoria.

***Tabla 3.2**: Streaming API Twitter*

API Rest

API que permite recuperar información y *tweets* dentro de un periodo de tiempo.

Destinada a	Desarrolladores
Ofrece	Dada una *query* se obtienen los *tweets* dentro de un intervalo de tiempo
Funcionamiento	A partir de URLs por HTTP
Beneficios	Información limitada, obtenida en formato ATOM o JSON. Se puede filtrar por lenguaje, localización o usuario.

***Tabla 3.3**: Rest API Twitter*

Limitaciones

La API de *Twitter* tiene limitaciones a la hora de realizar peticiones para obtener información. Dependerá principalmente del método solicitado. Las solicitudes más restrictivas son aquellas relacionadas con seguidores o seguidos, mientras que las menos restrictivas son las realizadas sobre los tweets [7].

API	Petición	Max. Datos por petición (página)	Cada 15 minutos
REST	GET statuses/user_timeline	200 *tweets*/segundo	900 · 200 = 180000 *tweets*
REST	GET user_show	1 perfil/segundo	1 · 900 = 900 perfiles
REST	GET followers_list	200 perfiles/minuto	200 · 15 = 3000 perfiles
REST	GET followers_ids	5000 ids de usuario/minuto	5000 · 15 = 75000 ids
Search	GET search/tweets	12 *tweets*/minuto	12 · 15 = 180 *tweets*
Streaming	POST statuses_filter	-	Máximo de 45 *tweets*

***Tabla 3.4**: Limitaciones APIs Twitter 1*

La tabla 3.4 muestra las limitaciones señalando el tipo de API en la primera columna. La segunda columna, cuando habla de peticiones, se refiere al tipo de dato que está solicitando, por ejemplo *followers_ids* se refiere al *id* del usuario. La última columna muestra los resultados extraídos en el intervalo de tiempo de 15 minutos a modo de ejemplo.

Cada API permite realizar peticiones para un intervalo de tiempo. La tabla 3.5 muestra qué límites temporales tiene cada una, comprendiendo que dependiendo de los datos que se quieran obtener se recurrirá a una API u otra.

API	Limitación temporal
REST	NO
Search	7 días
Streaming	Solo tiempo real

***Tabla 3.5**: Limitaciones APIs Twitter 2*

¿Cómo se accede?

Para usar la API de *Twitter* se requiere en primer lugar de un registro en el portal de desarrolladores, mostrado en la figura 3.1, para poder usar sus herramientas, pudiendo acceder a través de https://developer.twitter.com/en/docs/twitter-api. Posteriormente se podrá crear un proyecto, este será único y privado, pudiendo nombrarlo como el usuario deseé y necesario para obtener las claves de acceso que permitan la conexión con la API.

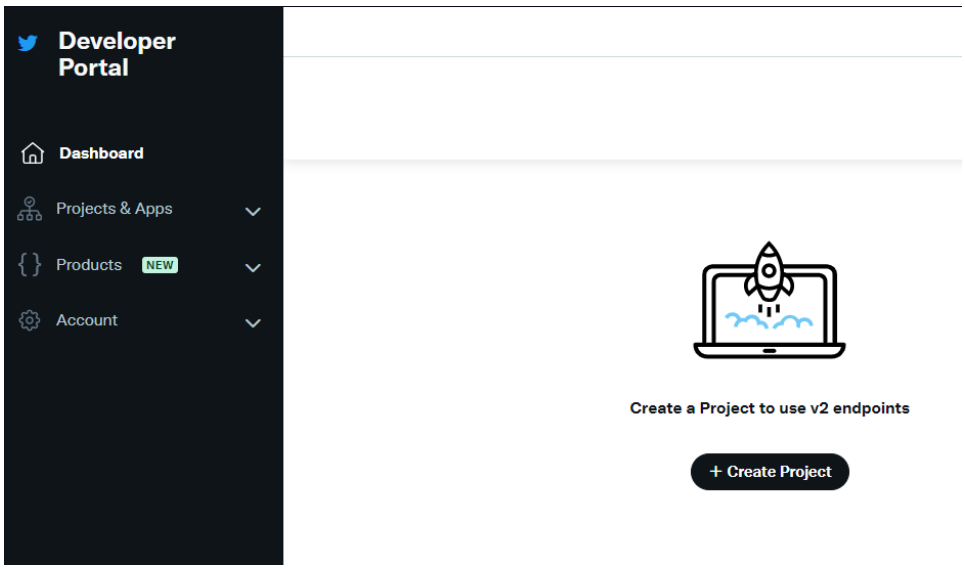


***Figura 3.1**: Portal del desarrollador*

Durante el registro, se deberá rellenar cierta información indicando los fines con los que se usará la API y aceptar las condiciones de uso, en mi caso se ha realizado explicando superficialmente los objetivos del proyecto. Además, *Twitter* proporciona materiales y guías para aprender a utilizar la API, a parte de una plataforma de desarrolladores para solucionar problemas con el uso de la plataforma [6].

3.3 Herramientas

Para el desarrollo del proyecto se han usado una serie de herramientas que ayudarán a la recolección y análisis de información, desde un entorno donde poder programar el algoritmo hasta una base de datos donde almacenar todos los datos de forma ordenada. A continuación, se explicarán y justificarán las herramientas escogidas.

3.3.1 MongoDB

Bases de datos

Una base de datos es un sistema computarizado para llevar registros. Se trata de una colección de archivos de datos, donde los usuarios pueden operar con dichos archivos: agregando, insertando, recuperando, modificando y eliminando los datos almacenados [9].

Podemos diferenciar entre bases de datos SQL y NoSQL, entendiendo las bases de datos NoSQL como estructuras que permiten almacenar información en un formato clave-valor, grafos o mapeo de columnas, mientras que las bases de datos SQL almacenan la información relacionándola entre sí. Las bases de datos NoSQL no siguen el esquema entidad-relación que puedan seguir las bases de datos relacionales (SQL), por lo que permiten un mayor rendimiento y escalabilidad [10].

Dado que las bases de datos NoSQL permiten manejar grandes cantidades de datos de una forma más flexible, algo que será necesario en el desarrollo del proyecto y

en especial con vistas al futuro de que la base de datos posibilite manejar enormes estructuras de datos, la elección del tipo de bases de datos será **NoSQL**.

MongoDB

Una de las principales bases de datos NoSQL de código abierto es MongoDB, esto es debido a que permite trabajar con series de documentos BSON, similares a JSON, un formato de texto que se almacena fácilmente en esta herramienta. Funciona con un sistema-cliente servidor que ejecuta el servidor con un binario llamado *mongod* y el cliente con mongo [11].

***Figura 3.2**: MongoDB logo*

Como característica importante para el desarrollo de este proyecto, MongoDB permite la indexación de un campo libremente, que posteriormente permite realizar consultas *ad hoc* en dichos campos. Además, integra muchos lenguajes de programación como *C#*, *Java*, *NodeJS*, y, entre ellos, Python, empleado en el desarrollo del proyecto.

Se ha mencionado antes la importancia de la escalabilidad en el desarrollo del proyecto y que para ello las bases de datos NoSQL son la mejor opción. MongoDB no solo es una herramienta que permite trabajar de manera óptima con este tipo de bases de datos, además mejora el rendimiento de la herramienta con su sistema de creación de nuevos nodos y su relación con el uso de memoria y CPU, permitiendo que la escalabilidad sea más efectiva. Por otra parte, al ser una de las herramientas más usadas en proyectos de *Big Data*, se ha elegido esta herramienta para almacenar la base de datos NoSQL porque dispone de alta documentación oficial, pudiendo resolver dudas de manera sencilla.

Los datos obtenidos se han ido almacenando en una base de datos en MongoDB. El proyecto ha incluido 1000 cuentas de *Twitter*, guardando cada una de ellas con todos los datos extraídos de la API de *Twitter*. Dentro de la base de datos, se pueden hacer consultas para realizar comprobaciones, cambios para corregir el algoritmo, o información para sacar estadísticas.

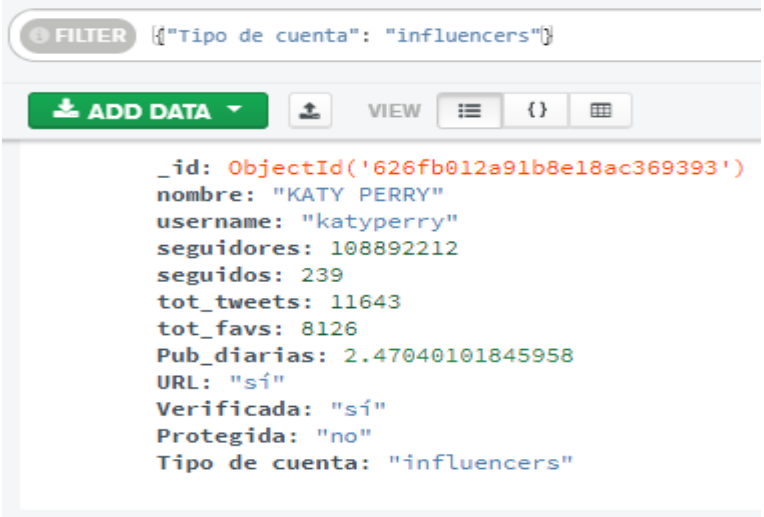


Figura 3.3: Visualización de datos en MongoDB

La figura 3.3 es un ejemplo de una consulta para las cuentas que pertenecen al grupo *Influencers*. Se puede observar la cuenta perteneciente a *katyperry* con todos sus los datos correspondientes que aunque serán explicados en siguientes puntos, se puede observar que recoge información relativa a la cuenta de *Twitter* como su nombre de usuario, sus seguidores o seguidos, *tweets* y favoritos totales, publicaciones diarias, si la cuenta está o no protegida y verificada o si dispone de una URL en el perfil, además de disponer de un *id* (un número) que proporciona MongoDB que identifica a cada cuenta dentro de la base de datos.

3.3.2 Visual Studio

Se trata de un IDE (entorno de desarrollo integrado). Aunque es compatible con muchos lenguajes de programación como *C++*, *PHP* o *Ruby*, para el proyecto será de interés tan solo el uso del lenguaje de alto nivel Python [12].

***Figura 3.4**: Visual Studio logo*

VisualStudio permitirá la comunicación entre el código programado con la API de *Twitter* y la base de datos MongoDB. Una de las razones por las que se ha escogido este IDE y no otro es por la disponibilidad que tiene para trabajar con Python, permitiendo integrar en concreto una librería llamada PyMongo, que establecerá la conexión entre la base de datos en MongoDB y Visual Studio.

Otra de las ventajas de usar este IDE, aparte de ser gratuito, es la disponibilidad de usar su *debugger*, sobre todo sabiendo que el código programado iba a ser largo y era necesario analizar su comportamiento localizando errores o buscando la manera de mejorarlo.

3.3.3 Spyder

Es un IDE centrado en Python a través de *Anaconda*. Será útil en el análisis de resultados, ya que dispone de bibliotecas como *NumPy* o *Matplotlib* usadas en el proyecto, posibilitando visualizar los resultados en figuras, gráficos, esquemas o tablas [13].

***Figura 3.5**: Spyder logo*

Aunque se trata de un IDE que maneja el lenguaje de programación usado en este proyecto, Python, su elección está justificada por la posibilidad de visualizar los datos o como apoyo para desarrollar el código, pero dando el peso de la programación a la herramienta VisualStudio.

3.3.4 Python

Python es un lenguaje de programación multiparadigma y de código abierto que permite programación orientada a objetos, funcional e imperativa [14]. Ha sido elegido porque actualmente es un lenguaje muy empleado en *machine learning* y dispone de multitud de librerías que se ajustan a las necesidades del proyecto, aparte de que es un lenguaje que tiene una gran cantidad de documentación oficial.

Las librerías son implementaciones que permiten codificar este lenguaje con el objeto de crear una interfaz independiente, cada una con diferentes módulos y sus funciones específicas. En Python, estas librerías se declaran en el código y a partir de ahí pueden ser usarse. Algunas de las librerías usadas en este proyecto son: Pandas, Sckikit-learn, Numpy, Tweepy y Pymongo, dependiendo de las funcionalidades que se busquen se hará uso de unas u otras. Por ejemplo, para manipular estructuras de datos en ficheros que serán usados como CSV o Excel, o estructuras de datos como *arrays* que proporciona la librería **Numpy** [19], se usará la librería **Pandas** [15]. Para el uso de algoritmos de *machine learning*, se usará la librería **Sckikit-learn** [17], que será de gran importancia, aportando algoritmos preprogramados, herramientas de preprocesamiento de datos y métricas para evaluar los modelos entrenados. Por último y muy importante, para las distintas conexiones, como con la API de Twitter se usará la librería **Tweepy** [20] mientras que para la conexión con MongoDB se usará la librería **Pymongo** [18].

3.4 Requisitos del sistema

Para poder comenzar el desarrollo, es necesario saber primero qué requisitos se necesitan, como qué datos extraer, su preprocesamiento para poder discernir de los datos de interés y determinar un modelo que permita llevar a cabo los objetivos planteados. Además, buscando el correcto funcionamiento del algoritmo es necesario validar los datos, por lo que se necesitarán métodos que permitan comprobar la capacidad de predicción del modelo, garantizando así su funcionamiento.

3.4.1 Selección de datos

En primer lugar, es necesario saber qué tipo de datos se están buscando. Se ha mencionado que el objetivo es poder clasificar perfiles según su comportamiento y para poder caracterizar cada perfil se necesitan una serie de datos elegidos justificadamente. Se comenzará por mostrar qué datos se han tenido en cuenta por cada perfil de *Twitter* para posteriormente establecer los diferentes grupos en los que se clasificarán los perfiles analizados. En primera instancia estos datos se extraen sin filtro, será después cuando sean preprocesados y preparados para su análisis.

Figura 3.6: Perfil de BarackObama en Twitter [29]

La figura 3.6 muestra el perfil de Twitter de Barack Obama a modo de ejemplo. En ella se han rodeado distintas partes, con un número asociado para poder identificarlas mejor:

- **Nombre (1):** puede ser reutilizado por cualquier cuenta y cambiado en cualquier momento.

- **Nombre de usuario (2):** es exclusivo para cada cuenta, aparece en la URL del perfil y es utilizado para identificarse y poder ser encontrado en búsquedas. Comienza por el símbolo "@".

- **Seguidores (3):** número de cuentas que siguen a la cuenta objetivo.

- **Seguidos (4):** número de cuentas que sigue la cuenta objetivo.

- **Fecha de creación (5):** indica la fecha en la que se creó la cuenta. En la captura se muestra el mes y el año, sin embargo, a la hora de extraer este dato se obtendrá el día exacto de la creación de la cuenta.

- **Verificación de cuenta (6):** se trata de una insignia azul que indica si la cuenta está o no verificada. Esto confirma la autenticidad de la cuenta objetivo buscando mantener la confianza entre los usuarios de la plataforma. Se necesitan una serie de requisitos para obtenerla [28], como ser una cuenta activa y relevante, no tener contenido dañino o de incitación al odio o ser una cuenta parodia.

- **Existencia de URL en perfil (7):** en el perfil se puede mostrar una URL, que normalmente enlaza con una web relacionada directamente con el propietario de la cuenta de *Twitter*, aunque puede ser simplemente un sitio de interés.

- **Protección de cuenta:** indica si la cuenta está o no protegida. *Twitter* permite proteger los *tweets* de una cuenta haciendo que estos sean privados [22], esto quiere decir que sólo los podrán ver los seguidores de la cuenta protegida. En el caso de la captura anterior, la cuenta no se encuentra protegida, si no, aparecería el icono de un candado al lado del nombre y los *tweets* no estarían visibles ya que como se puede observar en el botón de "Seguir", no se sigue la cuenta dando la opción a ello, sino aparecería como "Siguiendo".

- ***Tweets* totales:** indica el número total de *tweets* publicados. El perfil de una cuenta no muestra el número total de *tweets*, aunque en anteriores versiones sí lo hiciera. Este dato se obtendrá a través de la API de forma directa.

- **Publicaciones diarias**: es el resultado de dividir el número de publicaciones totales entre el número de días de vida de la cuenta.

3.4.2 Preprocesamiento y limpieza de datos

Como se ha mencionado anteriormente, los datos seleccionados se extraen en primer lugar sin filtro de la API de *Twitter*, lo que quiere decir que requieren de una limpieza que deje solo los datos de interés y se dejen preparados para procesar por el algoritmo de clasificación elegido (*Random Forest*). Para un tratamiento controlado de los datos se ha decidido hacer tres grupos: datos no procesados, procesados no condicionales y datos procesados condicionales.

Datos no procesados

Estos datos serán extraídos, pero no procesados por el algoritmo de clasificación. Se trata del nombre y el nombre de usuario, que servirán para almacenarlos en la base de datos, proporcionando un nivel de información mayor una vez los datos han sido procesados. El nombre de usuario, al no poderse cambiar, servirá para llevar un control de las cuentas introducidas, evitando repeticiones y permitiendo actualizaciones.

Datos procesados no condicionales

Los seguidores, los seguidos, el número de *tweets* totales y las publicaciones diarias serán extraídos y procesados tal cual se hayan obtenido desde un principio, ya que se trata de datos que expresan un valor numérico que serán tratado como tal.

Datos procesados condicionales

Son datos que se tratarán como “unos” o “ceros” en función de si se encuentran en el perfil objetivo, aunque finalmente para su evaluación se transformen a una forma más legible, mostrando un “sí” en lugar de “uno” y un “no” en lugar de “cero”. Se incluyen entre estos datos la URL, la verificación y la protección.

Resumiendo, podemos recoger los diferentes tipos de datos en la tabla 3.6, indicando los grupos de datos en la primera columna y en la segunda columna los datos que pertenecen a cada grupo.

Datos no procesados	nombre, nombre de usuario
Datos procesados no condicionales	seguidores, seguidos, número de tweets totales, número de publicaciones diarias
Datos procesados condicionales	URL, verificación, protección

Tabla 3.6: Tipos de datos

3.4.3 Determinación del modelo

El modelo seleccionado para este proyecto ha sido *Random Forest* de clasificación, explicado y justificado en anteriores puntos. A través de los bosques aleatorios creados se determinará si la cuenta de *Twitter* analizada pertenece a un grupo u otro.

Para cumplir con el objetivo principal, es decir, diferenciar cuentas de *Twitter* que pertenezcan a medios de comunicación de otros, se ha decidido crear tres grupos: “Medios”, “*Influencers”* y “Otros”. La razón de esta división es que toda aquella cuenta que no corresponda a una cuenta de medios obviamente sería “otra” clase de cuenta. En este punto se podrían haber establecido dos grupos, sin embargo, tras un análisis de las redes sociales, se observó que había otra clase de cuentas que, salvo por ligeras diferencias, se comportaban de manera similar con las cuentas de medios, las cuentas de *influencers*. Esta es la razón por la que se ha decidido añadir un grupo más, buscando conseguir una clasificación más exacta y realista.

Los datos que se procesarán han sido comentados anteriormente, tratándolos como características, mientras que las hojas de los árboles, es decir, las clases, serán el tipo de grupo al que pertenece cada cuenta. Los árboles tendrán diferentes estructuras, con diferente número de nodos, cogiendo diferentes características para elegir la clasificación, por ello alguno podría equivocarse. Es aquí donde *Random Forest* toma la mayoría de los resultados en cuenta para tomar la decisión final, resolviendo los problemas de variabilidad y el sobreajuste que pueden tener los árboles de decisión.

Finalmente, el modelo de clasificación pondrá a cada cuenta en uno de los tres grupos. Para ello, el algoritmo debe entender qué caracteriza a cada grupo:

Medios

Cuentas que realizan el proceso de comunicación orientado a las masas. Se dedican a informar sobre la actualidad cotidiana, como medio de filtraje, pudiendo ser objetivas o subjetivas. Sus fines pueden ser varios como entretener, formar opinión, informar, etc. Ejemplo, *BBC News* en la figura 3.7:

Figura 3.7: Perfil de BBCWorld en Twitter [29]

Se caracterizan por tener un alto número de *tweets*, publicaciones diarias y seguidores, en teoría interesados en lo que estas cuentas comunican. Por otro lado,

suelen tener un bajo número de seguidos, o interacciones con otras cuentas como dar favoritos o *retweet*. Suelen proporcionar una URL a su *web*, están verificadas para generar confianza en sus comunicaciones y no tienen sus *tweets* protegidos.

Influencers

Este tipo de cuentas se comportan muy similar a las de medios, sin embargo, su fin es distinto, y tienen características diferenciadoras. Este tipo de cuentas engloban a los personajes públicos o conjuntos como un grupo musical. Ejemplo, *Kim Kardashian* en la figura 3.8:

Figura 3.8: Perfil de KimKardashian en Twitter [29]

Al igual que los medios, tienen un alto número de seguidores, buscan la atención de otros usuarios no protegiendo sus *tweets* y con la verificación. Sin embargo, pueden tener un número mayor de seguidos, no tener ninguna URL, una menor publicación de *tweets* y más interacción con otros usuarios.

Otros

Se refiere a cualquier otro tipo de cuentas. Dado que el interés del proyecto son los medios de comunicación y, los *influencers* son los que más confusión pueden causar, el resto, como *bots*, empresas de otros sectores o particulares se han puesto en el mismo grupo. Ejemplo, *AuraSupertramp* en la figura 3.9:

Figura 3.9: Perfil de AuraSupertramp en Twitter [29]

Este grupo puede comportarse de diversas formas ya que engloba muchos tipos de cuenta, pero tienen en común que se diferencian de los medios e *influencers*. Cuentas que protegen sus *tweets*, sin verificación, sin URL, con bajos seguidores distan de pertenecer a cualquier de los anteriores grupos. Aunque, por otro lado, es posible que tengan bajos seguidos, o un alto número de publicaciones diarias, su interacción con otros usuarios es mucho mayor.

3.5 Implementación del sistema

Una vez que se conocen los requisitos del sistema, sabiendo qué algoritmo de clasificación se usará, en qué grupos dividirá las cuentas y qué características tomará para diferenciarlas, se procede al desarrollo de un algoritmo permita alcanzar el objetivo de clasificar y almacenar los datos. Para ello, es necesario establecer un orden a seguir que cumpla con todos los requisitos marcados, entrenarlo y validarlo. En los siguientes puntos se explicará cómo se ha logrado todo esto, mostrando y explicando en detalle el funcionamiento del algoritmo.

3.5.1 Análisis de datos

El primer paso para poner en marcha el algoritmo es la selección de datos de entrenamiento, datos que han sido revisados y servirán en un inicio para introducir al algoritmo qué clase de grupos existen y qué características tienen.

Se han seleccionado un total de 300 perfiles de *Twitter*, 100 para cada grupo. Los perfiles seleccionados a mano, uno a uno, son perfiles claramente pertenecientes a su grupo, de manera que el algoritmo comprenda de manera clara cuales son las características de cada grupo. Una vez extraída la información de estos perfiles y preprocesada, se almacena en la base de datos en MongoDB, a partir de la cual trabajará el algoritmo.

Como base datos final, se han almacenado un total de mil cuentas. Como se explicó en el funcionamiento del algoritmo de clasificación, *Random Forest*, la formación de los árboles se va retroalimentando con la entrada de nuevos datos y, gracias a la evaluación y ajuste de estos se va creando un algoritmo funcional. En este apartado se irán mostrando los resultados obtenidos a lo largo del proceso, dividiendo este en cuatro partes:

- Base de datos con 100 cuentas (Datos de entrenamiento)

- Base de datos con 250 cuentas.

- Base de datos con 500 cuentas.

- Base de datos con 1000 cuentas.

A continuación, los resultados se van a representar en unos gráficos donde el eje horizontal mostrará los distintos grupos en cada ingreso de cuentas separados por barras, mientras que el eje vertical mostrará la característica correspondiente. Cada uno de ellos contará con la información de los tres grupos para cada inyección de datos, pudiendo así visualizar el proceso y comparación de una manera más amigable. Es necesario mencionar que los gráficos van mostrando un avance relacionado con el incremento de datos. Esto quiere decir que los primeros datos son muy representativos de esos primeros datos, mientras que, con la introducción de nuevos, se va promediando y diluyendo las anomalías que pueda haber. En especial, en una primera búsqueda de

cuentas, donde se buscaba encontrar las más representativas de cada grupo para que al algoritmo de clasificación realizara las agrupaciones de una forma más acertada.

Seguidores

Los seguidores indican el número de cuentas que siguen a la cuenta en cuestión. Dada la naturaleza de las cuentas del grupo de "Medios" e "*Influencers"*, se esperaba que estas tuvieran un número elevado de seguidores que buscan recibir las publicaciones de estas cuentas.

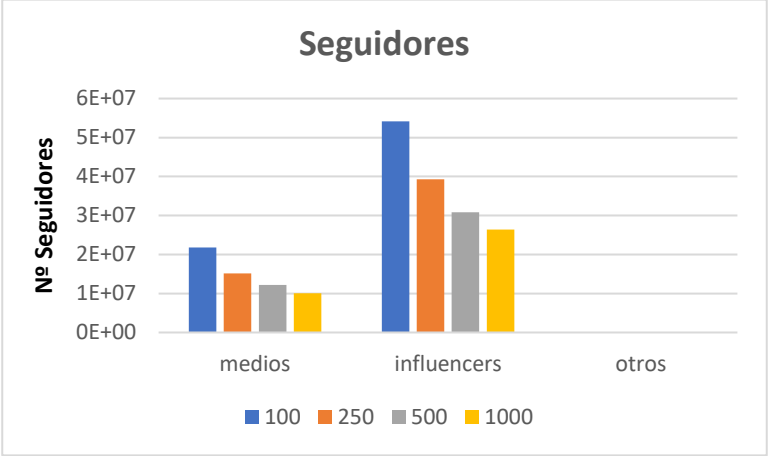


***Figura 3.10**: Resultados finales de seguidores en los diferentes grupos (elaboración propia)*

En la figura 3.10 se muestra la media en el eje vertical de seguidores de todas las cuentas analizadas en cada parte y, analizándola, se observa que el número de seguidores es con diferencia mayor en los *"Influencers"*, seguido de los "Medios" que no llegan ni a la mitad de los *"Influencers"*. Los seguidores del grupo de "Otros" son insignificantes en comparación con el resto de los grupos. De la misma forma que se comentó al principio del apartado, inicialmente el número de seguidores es muy abultado, reduciéndose y estabilizándose conforme se añaden nuevas cuentas debido a la selección inicial de las cuentas más representativas de cada grupo.

Seguidos

Los seguidos indican el número de cuentas que sigue la cuenta en cuestión. Este número puede ser muy variable, ya que depende de los intereses de cada cuenta el seguir a otras, aunque en el caso de los "Medios", cuyo máximo objetivo es difundir es raro que el número de seguidos sea alto.

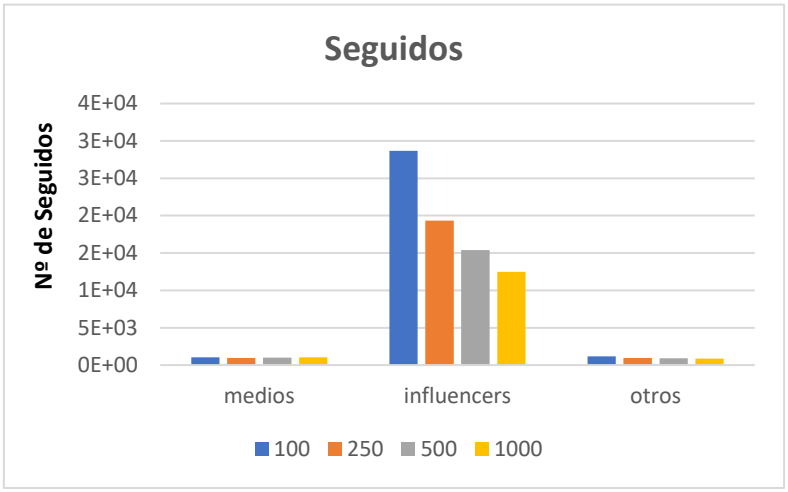


***Figura 3.11**: Resultados finales de seguidos en los diferentes grupos (elaboración propia)*

En la figura 3.11 se muestra la media en el eje vertical de seguidos de todas las cuentas analizadas en cada parte y, tras el análisis, se observa que mientras que el grupo de "Medios" y "Otros" se mantienen similares con números bajos, el grupo de "*Influencers"* se encuentra con un gran número de seguidos debido a su presencia en la red social, aunque cada uno la puede tener por diversas razones, como política, música u opiniones relevantes, tienen en común la influencia que ejercen sobre muchas personas. Al igual que en el caso de los seguidores se vuelve a ver un proceso de estabilización en el número de seguidos conforme se añaden nuevos datos.

Publicaciones totales

Se refiere al número total de *tweets* publicados.

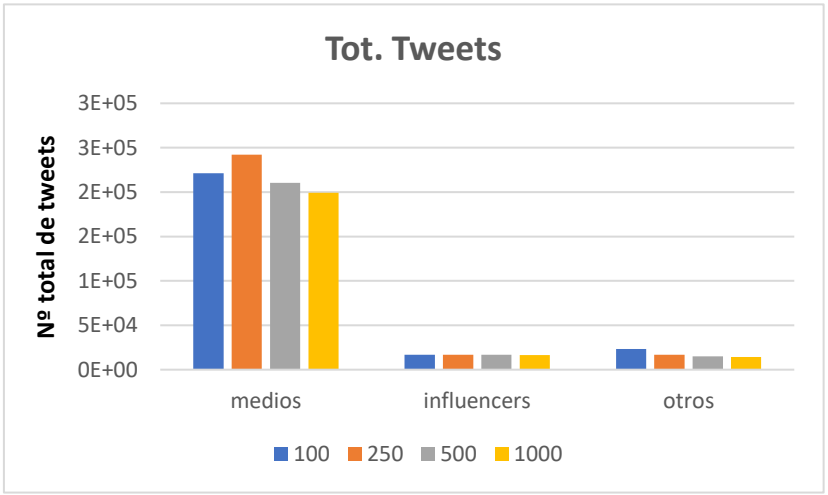


***Figura 3.12**: Resultados finales de tweets totales en los diferentes grupos (elaboración propia)*

En la figura 3.12 se muestra la media en el eje vertical de *tweets* totales de todas las cuentas analizadas en cada parte y, tras el análisis, se obtiene que el número de *tweets* totales de los "Medios" es alto y muy similar tras cada inyección de datos dada su finalidad, sin embargo, los otros dos grupos mantienen un bajo número.

Favoritos totales

Las cuentas de *Twitter* pueden interaccionar entre sí de diversas formas, como escribirse mensajes de forma privada o reaccionando a las publicaciones con un *retweet* o un favorito. Estas reacciones se reflejarán en el perfil que ha realizado la interacción, por lo que pueden indicar relación o interés entre las cuentas implicadas. Se mostrarán en el siguiente gráfico el número total de favoritos realizados.

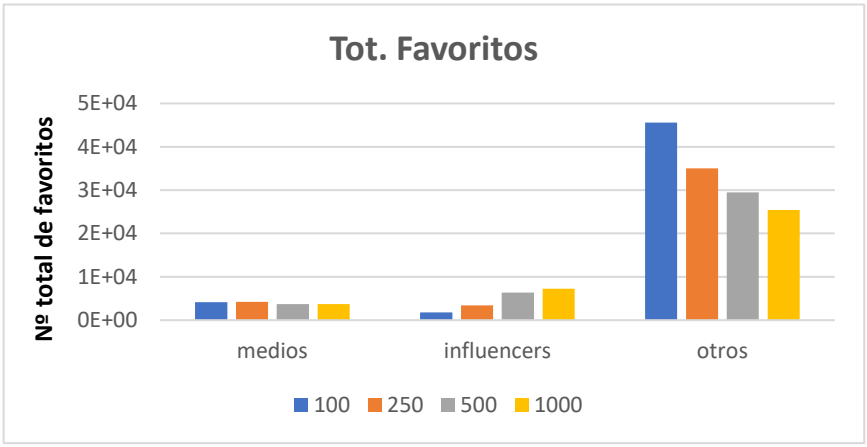


***Figura 3.13**: Resultados finales de favoritos totales en los diferentes grupos (elaboración propia)*

En la figura 3.13 se muestra la media en el eje vertical del total de favoritos de todas las cuentas analizadas en cada parte y, tras el análisis, se obtiene que ni el grupo de "Medios" ni el de "*Influencers"* llega a los mil favoritos totales, mientras que el grupo de "Otros" llega a triplicar esta cifra. Esto se debe a que los "Medios" e *"Influencers"* tienen una finalidad más orientada a crear contenido de diferente índole para sus consumidores en lugar de dedicarse a interaccionar con otros usuarios, mientras que el grupo de "Otros" incluyen perfiles que interactúan entre ellos como pueden amigos, familiares o incluso perfiles que reaccionan a las publicaciones de los dos primeros grupos.

Publicaciones diarias

El número de publicaciones diarias podría causar confusión ya que es posible que una cuenta publicara una gran cantidad de *tweets* en un corto periodo de tiempo y luego dejara de tener actividad, por ello se ha calculado este número como el resultado de dividir las publicaciones totales y el tiempo (en días) que ha pasado desde que se creó la cuenta. Todo esto es importante en especial para las cuentas de *bots*, cuentas preprogramadas que pueden actuar con diversos fines como spam, realizando un gran número de publicaciones en un corto periodo de tiempo, o con el simple fin de aumentar los seguidores de una cuenta objetivo sin realizar ninguna publicación.

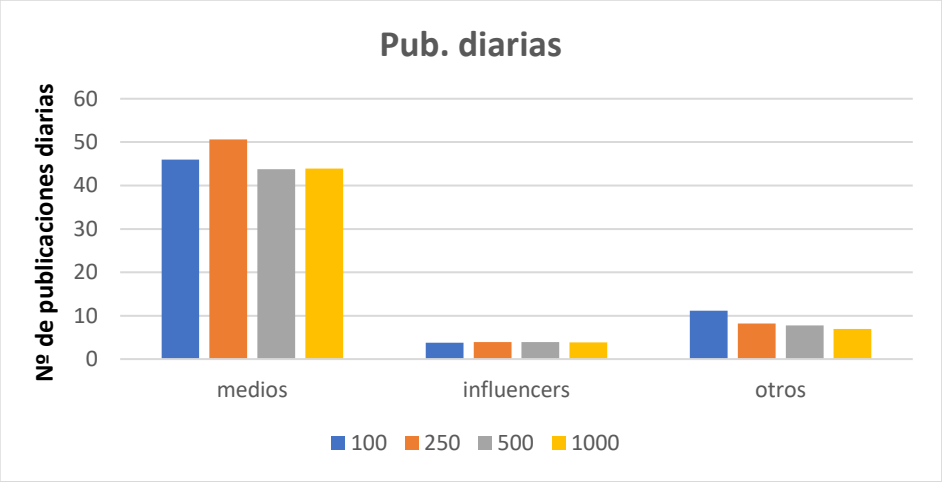


***Figura 3.14**: Resultados finales de publicaciones diarias en los diferentes grupos (elaboración propia)*

En la figura 3.14 se muestra la media en el eje vertical de las publicaciones diarias de todas las cuentas analizadas en cada parte y, tras el análisis, teniendo en cuenta que el grupo de "Medios" tenía un alto número de *tweets* totales y su naturaleza de transmitir información de manera regular, se obtiene un elevado de publicaciones diarias de este grupo. Al contrario, otras cuentas tienen un número de publicaciones menor, los *"Influencers* o el grupo de "Otros" recogen cuentas que generalmente están manejadas por una sola persona, por lo que es menos probable que mantengan un flujo alto de publicaciones, sobretodo en el caso de los *"Influencers"* donde sus comentarios pueden tener un peso importante en sus seguidores y suelen estar más medidos, en especial si son publicaciones con un fin publicitario.

URL

Para realizar la siguiente gráfica se ha calculado el porcentaje de cuentas que tenían la URL en el perfil, además de que en el caso de "Medios" o *"Influencers"* se ha comprobado que enlaza con su propia página web.

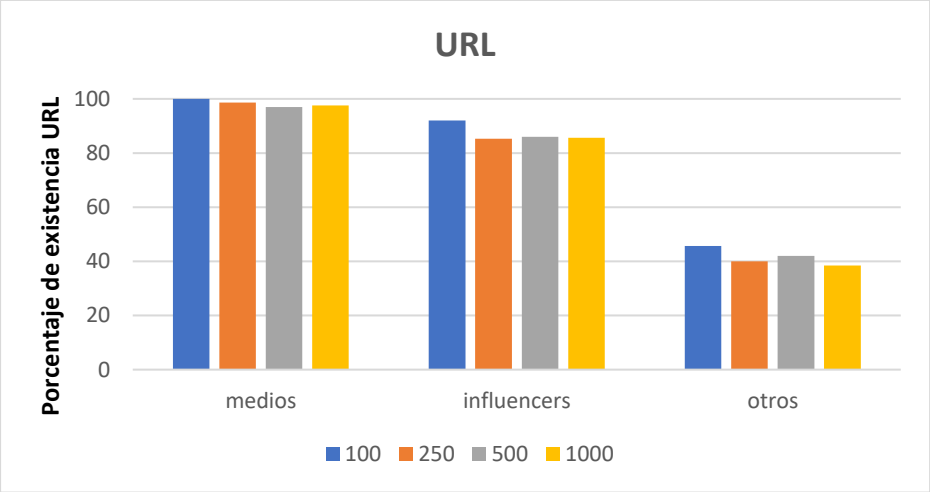


***Figura 3.15**: Resultados finales del porcentaje de existencia de URL en los diferentes grupos (elaboración propia)*

En la figura 3.15 se muestra la media en el eje vertical del porcentaje de todas las cuentas analizadas en cada parte que muestran una URL en su perfil y, tras el análisis, se observa que la existencia de URL en las cuentas de "Medios" e *"Influencers"* se encuentra entre el 80% y 100%, mientras que el grupo de "Otros" ronda el 40%.

Se puede encontrar también la existencia de URL en una cuenta que enlace con otra cuenta, por ejemplo en casos de medios que crean perfiles en diversos idiomas pero que su URL enlaza con la cuenta original, por lo que se seguiría entendiendo como una cuenta de "Medios" que enlaza con la URL de su web; también el caso de usuarios que establecen la URL de su cuenta enlazando con cuentas de "Medios" o *"Influencers"* para mostrar su gusto o acuerdo de opiniones con esas cuentas, pero como se muestra en los resultados, de igual forma los "Medios" e *"Influencers"* muestran una URL con mayor probabilidad.

Verificación

Para realizar la siguiente gráfica se ha calculado el porcentaje de cuentas que tenían la verificación en el perfil, esperando que tanto "Medios" como *"Influencers"* la tuvieran activa.

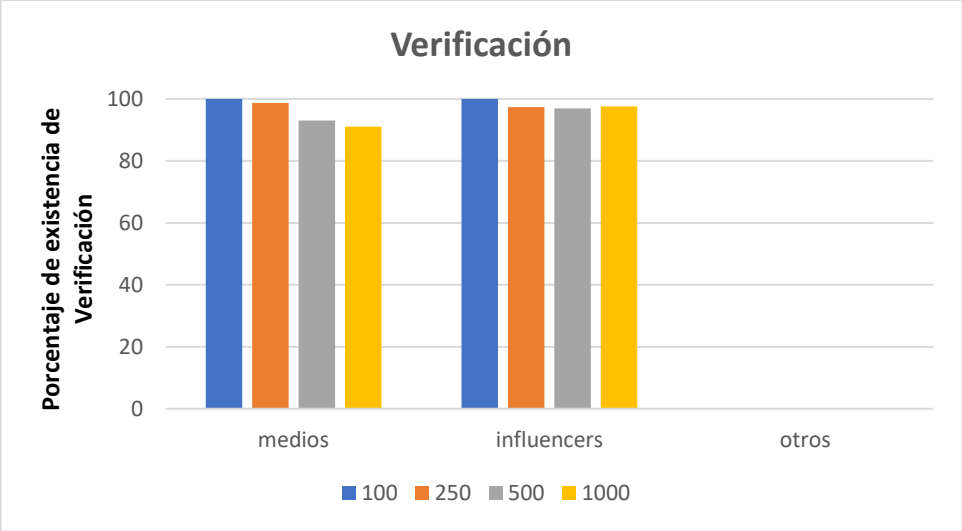


***Figura 3.16**: Resultados finales del porcentaje de existencia de verificación en los diferentes grupos (elaboración propia)*

En la figura 3.16 se muestra la media en el eje vertical del porcentaje de todas las cuentas analizadas en cada parte que se encuentran verificadas y, tras el análisis, se obtiene que la existencia de verificación en cuentas de "Medios" e *"Influencers"* es casi del 100% y en "Otros" prácticamente del 0%. Esto se debe a que la verificación muestra que es cuenta realmente pertenece al *influencer* o medio en cuestión y por tanto garantiza que las publicaciones proceden de ellos y no se trata de una cuenta falsa o de cualquier persona.

Protección

Para realizar la siguiente gráfica se ha calculado el porcentaje de cuentas que tenían la protección activada en el perfil, esperando que ni "Medios" ni *"Influencers"* la tuvieran activa.

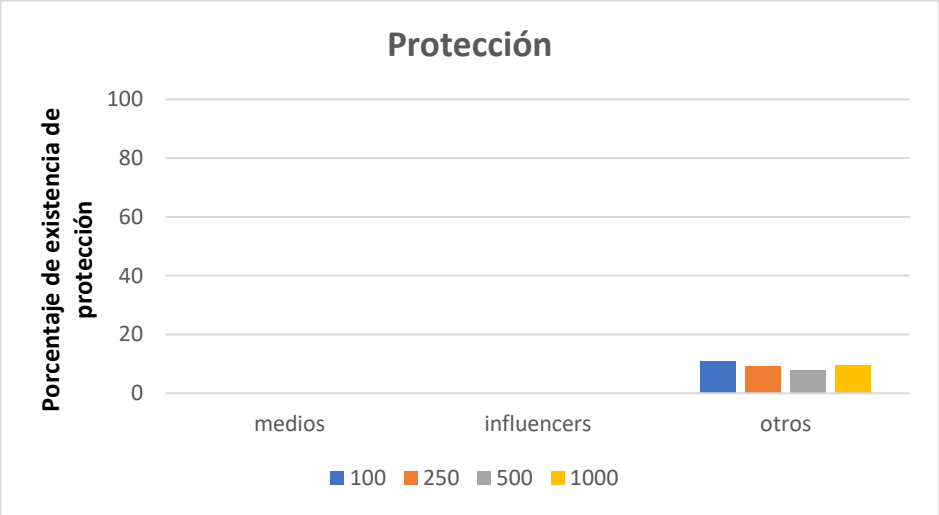


Figura 3.17: Resultados finales del porcentaje de existencia de protección en los diferentes grupos (elaboración propia)

En la figura 3.17 se muestra la media en el eje vertical del porcentaje de todas las cuentas analizadas en cada parte que se encuentran protegidas y, tras el análisis, según el gráfico se observa que proteger la cuenta es algo exclusivo del grupo de "Otros", ya que la existencia de cuentas de "Medios" e *"Influencers"* protegidas es del 0%. Si una cuenta está protegida imposibilita que otras cuentas puedan visualizar su contenido a excepción de que sean seguidores, dado que los "Medios" e *"Influencers"* buscan visibildiad, es totalmente esperable que los resultados sean tan claros, mientras que para el grupo de "Otros" dependerá de cada caso, pero garantizando que si una cuenta está protegida corresponde a este grupo.

3.5.2 Funcionamiento algoritmo

El algoritmo es la base del proyecto, ya que es el encargado de extraer los datos requeridos de *Twitter*, preprocesarlos, agruparlos según sus características y almacenarlos en la base de datos.

En la figura 3.18 se muestra el diagrama del algoritmo programado. Esta muestra el proceso que se sigue desde que se seleccionan las cuentas, pasando por el algoritmo programado en VisualStudio y su conexión con la API de Twitter hasta el paso de datos procesados a MongoDB y su retroalimentación con la base de datos generada. En primer lugar, representa en amarillo el cuadro las cuentas que se analizarán, estas cuentas han sido escogidas a mano una a una, almacenándose en un archivo TXT para que puedan ser usadas posteriormente.

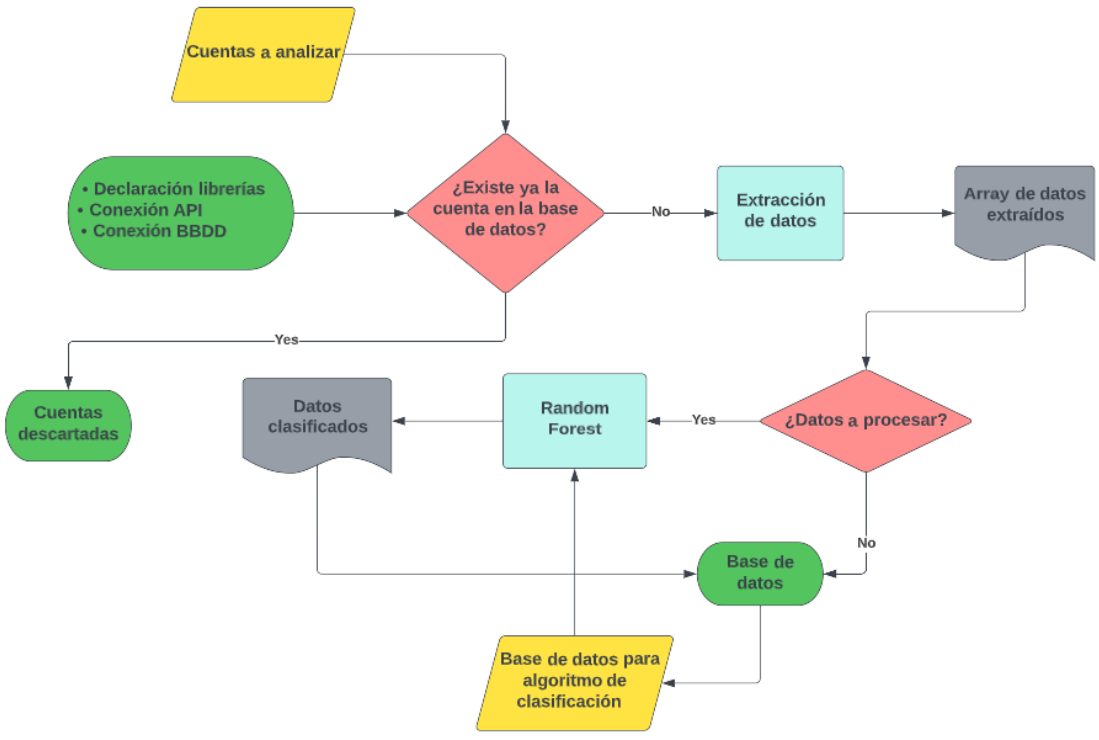


***Figura 3.18**: Diagrama de algoritmo (Elaboración propia)*

Antes de ir a la extracción de datos, será necesario declarar las librerías que permitan llamar a las distintas funciones que se usarán a lo largo de todo el proceso. Si

se llamara a una función sin estar su librería, la consola de VisualStudio devolvería un error, deteniendo la ejecución del algoritmo.

Tras realizar las conexiones con la API de *Twitter* y la base de datos, el algoritmo recibirá el nombre de las cuentas a analizar, realizando una comprobación para descartar las que ya existen obteniendo un *array* de datos. A partir de este *array*, se separarán los datos que se utilizarán en el algoritmo de clasificación *Random Forest* de los que no, llevando los que no directamente a la base de datos.

La librería **Sckikit-learn,** proporciona acceso a funciones con algoritmos de *machine learning*. Se hará uso de esta librería para llevar a cabo el algoritmo de clasificación *Random Forest,* que tras recibir los datos preprocesados los clasificará según sus características. Para el *Random Forest* se usará en primera instancia los datos de entrenamiento, pero conforme se vayan clasificando nuevos datos estos serán realimentados y usados en futuros procesos. Los datos que no se usen para el *Random Forest* tendrán un fin informativo a la hora de realizar análisis y visualizar resultados.

Ahora se entrará en detalle de algunas partes del algoritmo programado, comenzando por la declaración de librerías, cómo conectar con la API de Twitter y la base de datos en MongoDB hasta la extracción de datos, su paso por el algoritmo de clasificación y la validación final de estos.

Declaración de librerías

Las librerías de Python se declaran al principio del algoritmo, proporcionando el acceso a un conjunto de funciones ya programadas. Al tratarse de un proyecto relacionado con *machine learning* y teniendo en cuenta que existen librerías pensadas para el tratamiento de grandes cantidades de datos, será de gran importancia y ayuda tenerlas en cuenta. En algunos casos será necesario instalarlas en el equipo y se podrá hacer con ayuda del instalador de Python *pip*.

Como se ha mencionado, desde que se introducen las cuentas a analizar, hasta la interacción con la base de datos se trabaja con grandes cantidades de datos, por lo que el uso de la librería *Pandas,* muy usada en *machine learning,* es de gran ayuda. Para ello hace uso de *DataFrames* (estructuras bidimensionales) con las que mover y recorrer grandes cantidades de información. Para el cálculo numérico se hace uso de la librería

Numpy, manejando matrices multidimensionales. Las conexiones se llevarán a cabo gracias a las librerías Tweepy con *Twitter* y *PyMongo* para la base de datos en MongoDB. *Scikit-learn* será usada para acceder a distintos algoritmos de *machine learning*, *Random Forest* en este caso.

Conexión API Twitter

Para conectar con la API de *Twitter* es necesario autenticarse, se podrá hacer con una cuenta ya existente que se use normalmente como usuario de la red social o se puede crear una nueva desde el portal de desarrollador. Para completar el registro será necesario justificar el uso que se le dará a la API y, una vez se tenga acceso a la API habrá que crear un proyecto desde el portal desde el que se podrá llevar un control de la conexión con la API.

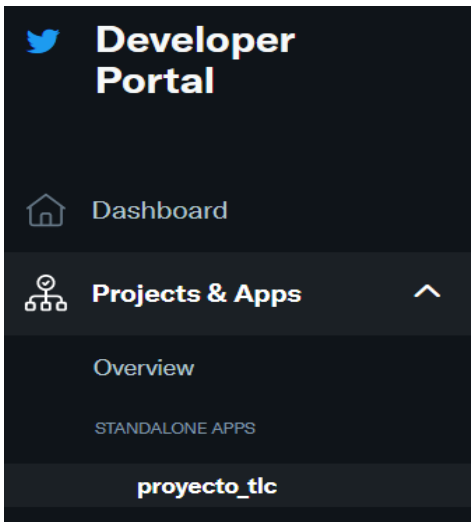


***Figura 3.19**: Portal de desarrollador de Twitter*

En la figura 3.19 se observa el portal de desarrollador de Twitter donde he creado un proyecto al que he llamado "*proyecto_tlc*". Cuando se crea un proyecto, este proporciona una serie de claves para poder conectar con él y así obtener acceso a la API.

En la figura 3.20 se muestran las claves:

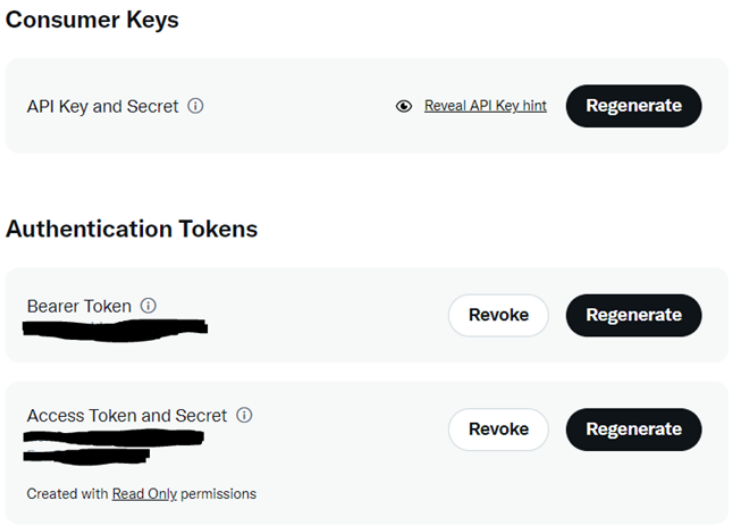


***Figura 3.20**: Keys de API de Twitter*

Se obtienen cuatro claves: *consumer key*, que actúa como nombre de usuario para el proyecto creado en el portal de desarrollador; *consumer secret*, como la clave secreta de acceso para conectar con el servidor de la API; *authentication token*, que determinará los privilegios y derechos que tiene una aplicación en particular y *authentication token secret* como contraseña de acceso del *authentication token*. Todas ellas son necesarias para realizar la conexión, siendo posible regenerar las claves en cualquier momento, invalidando las generadas anteriormente. Finalmente, una vez se han obtenido las claves se pasan al algoritmo para conectar con la aplicación a través de *Oauth2*, y se podrá empezar a extraer datos a través de las funciones de *Tweepy*.

Conexión con BBDD

Teniendo importada la librería *Pymongo* podemos empezar con la conexión, que se hará a través de un *host* y un puerto elegidos por el usuario, dependiendo de cómo se desee manejar la base de datos. Para este proyecto, como se muestra en la figura 3.21 el nombre del *host* es *"localhost"* mientras que el puerto por el que se conectará será el 27017. Pasada esta información al algoritmo se establecería la conexión.

***Figura 3.21**: Estado de servidor MongoDB*

Ahora se crean una base de datos y una colección en MongoDB con la que trabajar, la figura 3.22 muestra el nombre de la base de datos creada como *"Users_TW"* y la colección como *"Usuarios"*. Posteriormente, estos serán llamados en el código para introducir la información deseada.

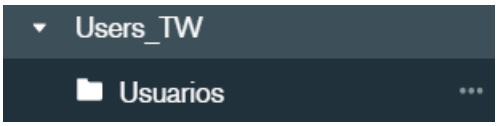


***Figura 3.22**: Base de datos y colección en MongoDB*

Extracción de datos

En este punto se pasa el nombre de cuentas deseadas al algoritmo para que se analicen a través de un archivo *TXT*. Antes de proceder a la extracción de datos se comprobará si existe la cuenta en la base de datos, se procesará sólo si no se encuentra. Al ejecutar el algoritmo en VisualStudio, se irá extrayendo cuenta por cuenta nueva su información, mientras tanto, en el terminal, se irá mostrando información de lo que va haciendo, indicando si la cuenta ya existe o por el contrario es nueva y se añadirá. Finalmente mostrará el número de cuentas analizadas, cuantas de ellas se han agregado y de qué tipo de cuenta se trata.

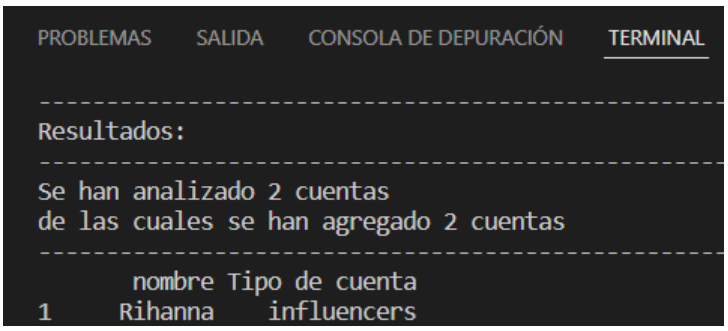


***Figura 3.23**: Introducción de nuevas cuentas vistas por terminal*

En el caso de la figura 3.23, se han introducido dos cuentas que no se encontraban aún en la base de datos, entre ellas *Rihanna*, si estas cuentas ya existen, en el terminal se mostraría un mensaje indicando que no se agregarán. Posteriormente se observa que indica el nombre de la cuenta y su grupo, en este caso de manera correcta. Si volviésemos a introducir las mismas cuentas, sabiendo que ya han sido agregadas anteriormente y que el algoritmo realiza la comprobación de su existencia en la base de datos, el terminal deberá mostrar un mensaje indicando que ya existen, tal como se muestra en la figura 3.24.

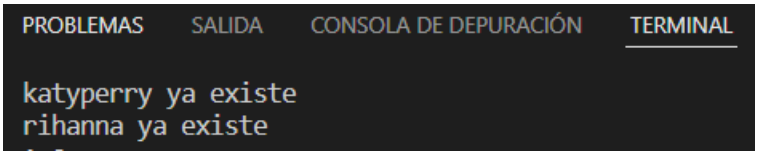


***Figura 3.24**: Introducción de cuentas ya existentes vistas por terminal*

Para extraer un dato sobre una cuenta se realiza un requerimiento a través de la librería *Tweepy,* mediante un HTTP *request* teniendo en cuenta la función ejecutada de la librería. Dependiendo del dato se seguirá un procedimiento, ya que hay funciones que extraerán la información deseada directamente y otros que seguirán un preprocesamiento.

Para entender esto mejor se ha elaborado la tabla 3.7 donde se muestra una primera columna con los datos a extraer (ya justificados anteriormente), una segunda columna con las funciones de la librería Tweepy y una tercera columna indicando al grupo que pertenece cada dato.

Dato	Funciónes Tweepy usadas	Grupo
Nombre	*name*	Datos no procesados
Nombre de usuario	*get_user*	Datos no procesados
Seguidores	*followers_count*	Datos procesados no condicionales
Seguidos	*friends_count*	Datos procesados no condicionales
Tweets totales	*statuses_count*	Datos procesados no condicionales
Publicaciones diarias	*statuses_count / tiempo_de_vida*	Datos procesados no condicionales
URL	*url*	Datos procesados condicionales
Verificación	*verified*	Datos procesados condicionales
Protección	*protected*	Datos procesados condicionales

***Tabla 3.7**: Tipos de datos extraídos y funciones usadas Tweepy*

Como se muestra en la figura 3.18, la extracción de datos da como resultado un *array* de todos los datos. Sin embargo, es necesario separar la información en datos a procesar y datos que se guardarán directamente en la base de datos. Para ello se creará un *DataFrame*, es decir, una estructura bidimensional, con la librería *Pandas* que contendrá los datos que se analizarán en el algoritmo *Random Forest*. Sin embargo, para procesar mejor la información, antes de analizar estos datos se creará un CSV, poniendo

los datos en un formato de tabla, a partir del *DataFrame* que finalmente será pasado al algoritmo de clasificación.

Random Forest

Con los datos preprocesados anteriormente y los datos de entrenamiento se procederá a la aplicación del algoritmo de *machine learning*, *Random Forest*. Esto será posible gracias a el objeto de *RandomForestClassifier* de la librería *Sklearn*. Como criterio se ha seleccionado *Gini*, como número máximo de atributos se utiliza la raíz del número de características y se realizará un muestreo aleatorio (*bootstrap*) a partir de los datos de entrenamiento.

Al introducir los datos de una nueva cuenta, pasarán por los árboles de decisión que se generan de forma aleatoria, cada uno tomando diferentes características para tomar su resultado final a través de la lógica booleana, logrando una mayor aleatorización, combinándolas para tener múltiples estimadores y clasificadores teniendo una perspectiva más global. *Spyder* permite visualizar los diferentes árboles de decisión que se han ido creando en el proceso, como se muestra en la figura 3.25, pudiendo observar en la pestaña de plots, los diferentes árboles creados.

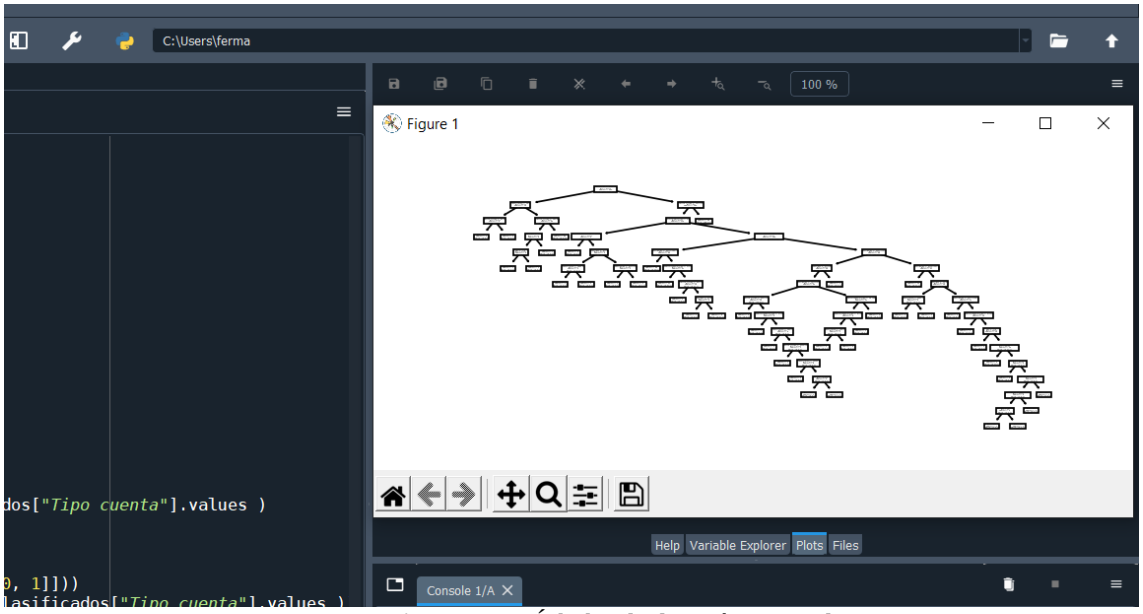


***Figura 3.25**: Árboles de decisión en Spyder*

Siguiendo el funcionamiento de *Random Forest* se tomará la mayoría como criterio para determinar el resultado, esto ayudará a solucionar uno de los grandes

problemas de los árboles de decisión, su alta variabilidad y su tendencia a sobreajustarse a los datos. Esto es válido tanto para predicciones como para clasificaciones.

Validación

Llevar todo el proceso a cabo sin un método que lo valide no valdrá de nada, pues lo resultados podrían ser erróneos o el algoritmo de clasificación se puede estar sobreajustando. Por ello, es necesario realizar ciertas validaciones que garanticen el buen funcionamiento y la capacidad de predicción del algoritmo de clasificación.

Para validar todo el proceso se calcularán los porcentajes de acierto en la clasificación, se calculará la precisión y se realizará validación cruzada. Para la validación cruzada se hará uso de la librería *Sklearn* importando la función *cross_val_score*, que nos dará un valor medio de que como de buena ha sido la capacidad de predicción del modelo. Estos métodos serán desarrollados en el punto de resultados.

CAPÍTULO 4

RESULTADOS

ÍNDICE

En este capítulo se presentan los resultados que se han obtenido tras las consecutivas inyecciones de datos al algoritmo. Posteriormente, se comentarán y validar

4.1 Validación del algoritmo

Llegados a este punto, donde se ha implementado un algoritmo, se han utilizado datos de entrenamiento y se han realizado sucesivas inyecciones de datos, se debe validar la capacidad de predicción del algoritmo de clasificación. La evaluación se realizará no solo en el estado final del algoritmo, sino que se evaluarán los resultados que se han ido obteniendo, pudiendo ver el progreso y viendo si ha mejorado o por el contrario empeora y por qué.

En primer lugar, hay que tener en cuenta conceptos teóricos explicados anteriormente que se han utilizado para realizar las distintas clasificaciones. Cuando se ejecuta el algoritmo de clasificación *Random Forest* proporcionado por la librería Sckikit-learn, se indica en código que se debe seguir el criterio de Gini y entropía para crear los árboles de decisión, lo que hará que se lleve a cabo un cálculo de la **Impureza de Gini** y de la **Entropía** en la construcción de cada árbol y teniendo en cuenta los resultados para la elaboración del bosque. Este proceso se realiza de forma automática al indicar en código que se tenga en cuenta el criterio de Gini y el de entropía, por lo que con el simple hecho de añadirlo se mejora la capacidad de clasificación del algoritmo.

Para la validación, se seguirá un proceso de tres partes (explicados en el Capítulo 2) por cada bloque de datos agregado a la base de datos. En primer lugar, se calculará el porcentaje de aciertos en la clasificación de las cuentas, siendo posible gracias a que se conocen de antemano el grupo al que pertenece cada cuenta por haber sido elegidas una a una. Este cálculo se realizará a mano, pues el algoritmo de clasificación etiquetará cada cuenta según realice los cálculos sin saber realmente si ha acertado. Posteriormente será necesario corregir los errores para realizar las siguientes validaciones, teniendo una base de datos fiable. El siguiente paso es calcular la precisión, es decir, el porcentaje de aciertos que tendrá el algoritmo de clasificación a partir de la base de datos ya corregida. Como se ha mencionado anteriormente la validación de la precisión es un poco "ingenua" ya que es como si un alumno conociera todas las respuestas antes del examen. Por tanto, es necesario un tercer paso que evalúe

de forma más realista la capacidad de predicción del algoritmo de clasificación. Este tercer paso se trata de la validación cruzada, que irá dividiendo los datos en *k* grupos para realizar las pruebas sobre el algoritmo de predicción. El valor de *k*, es decir, el número de iteraciones o pliegues que realiza la validación cruzada, según la bibliografía estudiada es válida con un valor de 5 ó 10 [34] [35] para estas cantidades de datos, por lo que realizaré las pruebas para ambos casos.

Al igual que en el punto anterior, se han dividido las inyecciones de datos en cuatro partes, que servirán como guía para el análisis de la validación del algoritmo de clasificación. Se hablará por tanto de "rondas" por cada nuevo ingreso de cuentas, entendiendo como:

- Ronda 1: 100 cuentas (Datos de entrenamiento)
- Ronda 2: 250 cuentas
- Ronda 3: 500 cuentas
- Ronda 4: 1000 cuentas

Las estadísticas de los datos recopilados ya han sido mostrados y analizados anteriormente, sin embargo, queda saber la validez de estos. Siguiendo los métodos de validación para las distintas rondas se podrá evaluar la capacidad de predicción del algoritmo concluyendo si funciona correctamente o por el contrario necesita mejorar, dando más peso a los resultados obtenidos de la última ronda, aunque se recojan los resultados de todas, cuando se tengan las 1000 cuentas clasificadas. Todos los métodos usados para validar el algoritmo son tenidos en cuenta, sin embargo, la validación cruzada es la que tiene más peso, si se tiene en cuenta que se llega al 90% de aciertos por lo menos se concluirá que el funcionamiento de este es positivo.

4.2 Porcentajes de acierto y precisión

Para poder trabajar, el algoritmo de clasificación necesita de una fuente de información donde los datos estén etiquetados (al tratarse de aprendizaje supervisado), los datos de entrenamiento son los primeros que se introducen en la base de datos.

Posteriormente se van agregando nuevas cantidades de datos cada vez mayores que van poniendo a prueba la capacidad predictiva.

	Cuentas agregadas	Total cuentas	Aciertos en "Medios" [%]	Aciertos en "*Influencers*" [%]	Aciertos en "Otros" [%]	Media aciertos [%]	Precisión [0,1]
Ronda 1	100	100	100%	100%	100%	100%	1.0
Ronda 2	150	250	92 %	24%	100%	72%	0.97
Ronda 3	250	500	48%	72%	100%	73.3%	0.97
Ronda 4	500	1000	84%	88%	96%	89.3%	1.0

***Tabla 4.1**: Porcentajes de acierto y precisión*

En la tabla 4.1 se presentarán los porcentajes de acierto por cada grupo, su media y la precisión, recordando que se mueve en un intervalo de [0,1]. Como se puede observar los resultados son excelentes, del 100% en todos los casos para los datos de entrenamiento, sin embargo, no tiene mucho sentido dar peso a estos valores a la hora de validar, ya que al ser los primeros datos (no hay más) que se introducen y a los que recurre el algoritmo de clasificación, ya sabe las respuestas y es totalmente normal que ocurra tal porcentaje de aciertos.

En la primera ronda se han introducido cuentas lo más características posibles para mostrar al algoritmo de clasificación que determina cada grupo. Hay que tener cuidado con esto pues puede llevar a un sobreajuste del algoritmo de clasificación y hacer que funcione erróneamente. Por ello, a partir de la segunda ronda se comienzan a introducir cuentas que pueden llevar a más confusión, más difíciles de clasificar, aunque recordando que se conoce previamente su clasificación para no perdernos. Esta es la razón por la que los resultados de las predicciones bajan considerablemente en especial en medios e *influencers* donde se puede encontrar más similitud y conforme se avanza, en la última ronda se puede observar que el algoritmo de clasificación ha conseguido aclarar con un porcentaje de hasta 89.3% de aciertos esas diferencias. Además, como se mencionó anteriormente justificando la división en estos tres grupos, los "Otros" se

diferencian claramente de los grupos de "Medios" e *"Influencers"* y se respalda en el alto porcentaje obtenido de manera constante, siendo 100% en todas las rondas menos en la última en la que se ha obtenido un 96% en la clasificación de las cuentas de medios.

Finalmente, la precisión, que indica la relación entre las predicciones correctas y el total de predicciones se observa bastante positiva. Podría parecer que el algoritmo de clasificación está funcionando de manera casi perfecta ya que se mantiene constante a 1 durante las cuatro rondas, pero recordemos que este valor no es del todo real, sino que se basa en un conocimiento previo de todas las respuestas ya corregidas. También aquí se observa que para las rondas 2 y 3, donde se han introducido cuentas más confusas el resultado baja a 0.97, siendo una reducción muy pequeña. Aun así, el hecho de que sea casi 1 en todas las rondas da indicios de que va por el buen camino.

4.3 Validación cruzada

Veamos ahora qué sucede en validación cruzada con los datos de entrenamiento y en sucesivas agregaciones de datos para ir probando la efectividad de predicción cuando k tiene un valor de 5 y 10.

	Pliegue 1	Pliegue 2	Pliegue 3	Pliegue 4	Pliegue 5	Media
Ronda 1	0.87	0.97	0.83	1.00	0.90	0.90
Ronda 2	0.87	0.87	0.95	0.82	0.79	0.86
Ronda 3	0.87	0.9	0.85	0.87	0.85	0.87
Ronda 4	0.85	0.95	0.87	0.84	0.90	0.88

***Tabla 4.2**: Validación cruzada k = 5*

La tabla 4.2 muestra la validación cruzada aplicada a las distintas rondas cuando $k = 5$, es decir, que se han realizado 5 pliegues a cada conjunto de datos. Si la ronda 1 estaba formada por 100 datos, estos se han dividido en cinco partes iguales, evaluando por cada pliegue una de estas partes en base al resto.

El resultado obtenido, un valor comprendido entre 0 y 1, será más próximo a 1 cuanto mayor sea la capacidad de predicción del algoritmo de clasificación. La ronda que ha obtenido un mayor resultado es la primera dado que los datos eran muy característicos, algo que se podía observar también en la validación por porcentajes de acierto. Sin embargo, en otras rondas, podemos encontrar algunos pliegues que se han acercado a 1, mientras que tienen otros pliegues por debajo de 0.8. Esto se debe a que han cogido fragmentos de datos que al algoritmo de clasificación le han sido más fácil de catalogar siendo más complejos. Por ejemplo, en el caso de la segunda ronda, se encuentran la media más baja a causa, una vez más, de introducir cuentas con comportamientos que pueden dar lugar a más confusión, pero que a pesar de ello el algoritmo de clasificación acaba por aprender.

Para k = 10:

	Pliegue 1	Pliegue 2	Pliegue 3	Pliegue 4	Pliegue 5	Pliegue 6	Pliegue 7	Pliegue 8	Pliegue 9	Pliegue 10	Media
Ronda 1	0.87	0.93	0.93	1.00	0.86	0.86	1.00	1.00	1.00	1.00	0.93
Ronda 2	0.78	0.95	0.82	0.91	0.95	0.90	0.86	0.90	0.77	0.81	0.87
Ronda 3	0.87	0.97	0.83	1.00	0.93	0.77	0.90	0.80	0.90	0.86	0.87
Ronda 4	0.89	0.84	0.94	0.90	0.90	0.98	0.78	0.83	0.97	0.90	0.89

***Tabla 4.3**: Validación cruzada k = 10*

Para la tabla 4.3, se ha realizado una validación cruzada para *k = 10*, lo que indica que se han duplicado el número de iteraciones por ronda, dividiendo cada grupo de datos en 10 partes y obteniendo resultados para cada una de ellas.

Los resultados son muy similares, moviéndose entre 0.77 como valor más bajo y llegando a obtener 1 en casos como en la cuarta iteración de la tercera ronda, pero en especial a lo largo de la primera ronda. La obtención de 1 como resultado de validación cruzada se encuentra en la ronda 1, es decir, en la ronda de los datos de entrenamiento, la cual ha obtenido los mejores resultados también anteriormente y que como se ha explicado, está justificada en que se tratan de los primeros datos que se introducen. Si nos fijamos en los valores de la última columna, que realiza la media de los distintos pliegues, observamos que de la misma forma que ocurría en *k = 5*, las rondas 2 y 3

obtienen un menor valor de validación cruzada debido a los nuevos datos, recuperándose en la última ronda donde el algoritmo de clasificación ha aprendido.

La mayor diferencia entre estos resultados y los obtenidos en la validación por porcentajes de acierto y precisión, es que estos datos son más realistas, a diferencia de los otros que tratan la información de forma más ideal. Se puede observar, además, cómo los primeros resultados son más cercanos a la perfección, dando a entender de forma errónea que el algoritmo de clasificación funciona perfectamente. Es por tanto que se tomará la validación cruzada con más peso a la hora de concluir la capacidad de predicción del algoritmo de clasificación.

Tras este análisis de los resultados, se obtiene que las cuentas introducidas en un comienzo, en los datos de entrenamiento iniciales, eran muy características, por lo que no es de extrañar que el porcentaje de acierto inicial sea alto. Conforme se iban introduciendo nuevos datos se buscaban cuentas no tan características pero que claramente pertenecieran a su grupo, es por ello por lo que en siguientes rondas el porcentaje de acierto ha disminuido, sin embargo, el algoritmo de clasificación ha ido corrigiendo los errores para que el algoritmo los tuviera en cuenta. Finalmente, en la última agregación de datos el resultado es bastante alto, de casi el 90% de aciertos en la clasificación de las cuentas, lo que indica que el algoritmo de clasificación cumple su objetivo de poder clasificar si una cuenta es de “Medios” o *“Influencers”* o si pertenece a “Otros”, concluyendo que su funcionamiento es positivo.

CAPÍTULO 5

CONCLUSIONES Y LÍNEAS FUTURAS

ÍNDICE

En este apartado se explicarán las conclusiones obtenidas una vez finalizado el proyecto. También se comentarán futuras posibles ampliaciones que busquen mejorar el proyecto.

5.1 Conclusiones

En primer lugar, como algo positivo, se concluye que se ha alcanzado el objetivo del proyecto. Se ha conseguido crear un clasificador de perfiles sociales de la red social *Twitter* basándose en su comportamiento. Esto es algo positivo no solo para poder concluir el proyecto sino también para cumplir con el reto propuesto por ElevenPaths.

Comentar de las facilidades que ha presentado trabajar con una API, en este caso la de Twitter. En un primer momento se planteó extraer los datos mediante *web scraping*, algo que hubiera dificultado y alargado enormemente alcanzar los objetivos. Una API como la usada, que permite realizar llamadas a datos muy concretas y con unas limitaciones no muy restrictivas para este caso, puede convertirse en una herramienta de gran utilidad.

Los "Medios" e *"Influencers"* tienen un gran número de seguidores. Los medios de comunicación tienen una masa de seguidores interesados en la información que estos puedan ofrecer y probablemente comparten las opiniones con el medio seguido. Los *"Influencers"* pueden ser de atractivo por destacar en un campo, por moverse en ciertos grupos sociales o simplemente en su vida cotidiana. Hay que incluir que dada su gran influencia muchas marcas también son seguidores de estas cuentas buscando la publicidad a través de ellos.

Se han encontrado características que son muy particulares de un grupo, algo que ayudará enormemente al algoritmo en su clasificación. El número de *tweets* y el número de publicaciones diarias, que para el grupo de "Medios" dada su naturaleza es siempre alto; la protección de una cuenta, siendo la intención de *"Influencers"* y "Medios" llegar a un público es lógico que no se encuentren protegidas, concluyendo que si una cuenta se encuentra protegida será del grupo de "Otros".

Por otra parte, otras características que, aunque no caracterizan un solo grupo, sí pueden ser excluyentes. Por ejemplo, la verificación, que aporta confianza una cuenta, es casi condición necesaria para la pertenencia al grupo de "*Influencers"* o "Medios"; La URL es también muy importante, ya que suele enlazar con la web del medio, del *influencer* o con una marca, se puede encontrar en el grupo de "Otros", aunque con poca

frecuencia. A favor del grupo de “Otros”, se encuentra con diferencia con el mayor número de favoritos, pudiéndose justificar en que los otros dos grupos buscan principalmente la generación de contenido y no tanto la interacción con otras cuentas, mientras que el grupo de “Otros” engloba usuarios que consumen dicho contenido.

Respecto a la clasificación de cuentas, dadas unas características iniciales, se ha trabajado en un principio con cuentas que claramente corresponden con su grupo. Conforme se ha ido avanzando algunas cuentas podían ser de clasificación dudosa a primera vista, en especial las cuentas de “Medios” e *“Influencers”*, algo que se sabía desde un principio y que es la justificación de la división de grupos elegida. Sin embargo, con el entrenamiento supervisado se ha conseguido enseñar a un algoritmo de clasificación como *Random Forest* a diferenciar unas de otras a través de los datos de entrenamiento y las sucesivas inyecciones de datos, llegando a hacerlo cada vez más preciso, llegando, de acuerdo con los resultados finales, a alcanzar una precisión de casi el 90%, un muy buen resultado.

5.2 Líneas futuras

Tras haber conseguido realizar una clasificación de varios grupos de perfiles sociales, se estima positivo seguir realizando divisiones más concretas de los perfiles para obtener una información más detallada. El objetivo ideal por alcanzar sería recoger todas las cuentas de la red social *Twitter*. Esto necesitaría de una enorme base de datos que, para cualquier consulta o elaboración de grafos sería óptimo que se tratase de una base de datos no relacional, como la usada en el desarrollo de este proyecto.

Teniendo en cuenta que se ha trabajado directamente con el código y la interfaz que proporciona la base de datos, sería más eficiente programar una interfaz gráfica que permita un manejo del algoritmo de una manera más amigable sobre todo a la hora de introducir nuevas cuentas o realizar consultas.

Aunque se han tenido en cuenta muchas características para definir un grupo de comportamiento, existen muchas otras que pueden ayudar a clasificarlas y que la API proporciona. Este proyecto se ha enfocado en las características más representativas que

han permitido clasificar las distintas cuentas, sin embargo, se han dejado otras de lado que ayudarían a obtener resultados más exactos, pero que necesitaría de un elevado estudio de las técnicas de clasificación y del funcionamiento de *Twitter*. Con esto me refiero al análisis del texto subido por las cuentas, pudiendo filtrar por idiomas o palabras clave y permitiendo identificar qué temas trata. Esto daría lugar también, a la relación directa entre las cuentas ya analizadas, estableciendo grafos para poder visualizar las agrupaciones con el fin de plantear estrategias de mercado e incluso predecir comportamientos. Sin embargo, hay que tener en cuenta que para el desarrollo de este proyecto, se han procesado cuentas que escritas dentro del alfabeto latino incluyendo caracteres diacríticos, lo que quiere decir que no se han tenido en cuenta perfiles que manejan otros caracteres como por ejemplo los cirílicos, es por tanto línea futura llegar a procesar diferentes idiomas para un completo funcionamiento del algoritmo.

Además, conforme se incluyan nuevas características se puede perfilar de mejor manera las distintas cuentas, llegando a crear nuevos grupos y subgrupos que den lugar a un algoritmo de clasificación más preciso y completo. Por un lado, crear nuevos grupos como *bots,* o uno referido a las cuentas empresariales, por otro lado, subdividir los grupos ya existentes como tipos de *influencers* según el mercado en el que se enfocan o subdividir los medios según el tipo de información que ofrezcan.

Por último, todo el proceso realizado en este proyecto se podría a extrapolar a otras RRSS, teniendo en cuenta que en este caso se ha utilizado una API junto a una librería en Python (*Tweepy*) elaborada para interaccionar con esta red social en concreto, habría que tener en cuenta cuán limitada está cada API, partiendo de que se dispusiera de una, ya que cada una funciona de distinta forma. De otro modo, existen otras formas de extraer información de una web como por ejemplo *web scrapping*, sea como sea, las bases o el proceso a seguir pueden tomar este proyecto como referencia.

6. Bibliografía

[1] «Redes-sociales» https://www.rdstation.com/es/redes-sociales/ (accedido dic. 6, 2021)

[2] Joel Comm «El poder de Twitter»

https://www.bqm.com.pe/libros/El%20poder%20de%20Twitter.pdf (accedido dic. 10, 2021)

[3] Matt Ahlgren e Equipo WSR «Estadísticas y hechos de Twitter para 2022» https://www.websiterating.com/es/research/twitter-statistics/#:~:text=¿Cuántas%20cuentas%20de%20Twitter%20hay,330%20millones%20son%20usuarios%20activos (accedido dic. 17, 2021)

[4] Rosa Fernández «Distribución porcentual de usuarios de Twitter a nivel mundial en 2021, por grupo de edad» https://es.statista.com/estadisticas/1203013/porcentaje-de-usuarios-de-twitter-por-grupo-de-edad-mudial/ (accedido dic. 17, 2021)

[5] Developer Platform «Twitter API» https://developer.twitter.com/en/docs/twitter-api (accedido dic. 22, 2021)

[6] Developer platform «Docs» https://developer.twitter.com/en/docs (accedido ene. 14, 2022)

[7] Miguel Ángel Álvarez «Introducción al API de Twitter y cURL» https://desarrolloweb.com/articulos/intro-api-twitter-curl.html (accedido ene. 14, 2022)

[8] Mariluz Congosto «Lo que siempre quiso saber del API de Twitter y nunca se atrevió a preguntar» http://www.barriblog.com/2017/10/lo-siempre-quiso-saber-del-api-twitter-nunca-se-atrevio-preguntar-actualizado-2017/#:~:text=En%20el%20Search%20API%20la,a%2050%20tuits%20por%20segundo (accedido ene. 15, 2022)

[9] C.J. Date «Introducción a los sistemas de bases de datos» https://books.google.es/books?hl=es&lr=&id=Vhum351T-K8C&oi=fnd&pg=PR17&dq=Bases+de+datos&ots=fAL4MTad7i&sig=gKJ-MOc228qI-596Vw5zZJya0aE#v=onepage&q=Bases%20de%20datos&f=false (accedido ene. 21, 2022)

[10] Acens «Bases de datos NoSQL. Qué son y tipos que nos podemos encontrar» https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf (accedido feb. 3, 2022)

[11] Stack Overflow contributors «Aprendizaje MongoDB» https://riptutorial.com/Download/mongodb-es.pdf (accedido feb. 4, 2022)

[12] «Aprender a codificar en Visual Studio» https://visualstudio.microsoft.com/es/vs/getting-started/ (accedido feb. 6, 2022)

[13] Ligdi Gonzalez «Introducción al IDE Spyder» https://aprendeia.com/ide-spyder-para-python/ (accedido feb. 6, 2022)

[14] «python» https://www.python.org (accedido feb. 12, 2022)

[15] «pandas documentation» https://pandas.pydata.org/pandas-docs/stable/index.html (accedido feb. 12, 2022)

[16] Pedro Larrañaga, Iñaki Inza, Abdelmalik Moujahud «Tema 6. Clasificadores Bayesianos» http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t6bayesianos.pdf (accedido jun. 1, 2022)

[17] «scikit-learn Machine Learning in Python» https://scikit-learn.org/stable/ (accedido mar. 7, 2022)

[18] «PyMongo 4.1.1 Documentation» https://pymongo.readthedocs.io/en/stable/ (accedido mar. 7, 2022)

[19] «NumPy documentation» https://numpy.org/doc/stable/ (accedido mar. 7, 2022)

[20] «Tweepy Documentation» https://www.tweepy.org (accedido mar. 7, 2022)

[21] Selene Hernández Rodríguez «Clasificadores rápidos basados en el algoritmo del Vecino más Similar para Datos Mezclados»

https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/386/1/HernandezRoS.pdf (accedido jun. 1, 2022)

[22] «Cómo proteger y desproteger sus Tweets» https://help.twitter.com/es/safety-and-security/how-to-make-twitter-private-and-public (accedido abr. 17, 2022)

[23] «Machine Learning | Qué es, tipos, ejemplos y cómo implementarlo» https://www.grapheverywhere.com/machine-learning-que-es-tipos-ejemplos-y-como-implementarlo/ (accedido abr. 18, 2022)

[24] Rosa Ferrero «Qué son los árboles de decisión y para qué sirven» h (accedido abr. 25, 2022)

[25] Azad Abdulhafedh «Incorporating Multiple Liner Regresion in Predicting the House Prices Using Big Real Estate Datset with 80 Independent Variables» https://www.scirp.org/journal/paperinformation.aspx?paperid=115003 (accedido abr. 25, 2022)

[26] Team redac «Random Forest: Bosque aleatorio. Definición y funcionamiento» https://datascientest.com/es/random-forest-bosque-aleatorio-definicion-y-funcionamiento (accedido abr. 26, 2022)

[27] «Sets de Entrenamiento, Test y Validación» https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/ (accedido abr. 3, 2022)

[28] «Acerca de las cuentas verificadas» https://help.twitter.com/es/managing-your-account/about-twitter-verified-accounts

[29] Twitter https://twitter.com (accedido feb. 17, 2022)

[30] «Cross-validation (o Validación Cruzada) para Evaluar Modelos de Aprendizaje de Máquina | Python» https://www.youtube.com/watch?v=Qnth2VXopLg (accedido dic. 19, 2022)

[31] Ariza, F.J.; Pinilla, C.; García, J.L. «Comparación de matrices de confusión celda a celda mediante Bootstraping» http://coello.ujaen.es/asignaturas/pcartografica/pqcarto/CPQC28/Bootstrapping_Matrices_Confusion.pdf (accedido may. 27, 2022)

[32] «Los cinco tipos de fuentes de datos» https://grupokorporate.com/los-cinco-tipos-de-fuentes-de-datos/ (accedido may. 27, 2022)

[33] Ankit Chauhan Random Forest Classifier and its Hyperparameters https://medium.com/analytics-vidhya/random-forest-classifier-and-its-hyperparameters-8467bec755f6 (accedido may 23, 2022)

[34] Payam Refaheilzadeh, Lei Tang, Huan Liu «Cross-Validation» http://leitang.net/papers/ency-cross-validation.pdf (accedido may. 5, 2022)

[35] «Lecture 13: «Cross-validation» http://leitang.net/papers/ency-cross-validation.pdf (accedido may. 5, 2022)

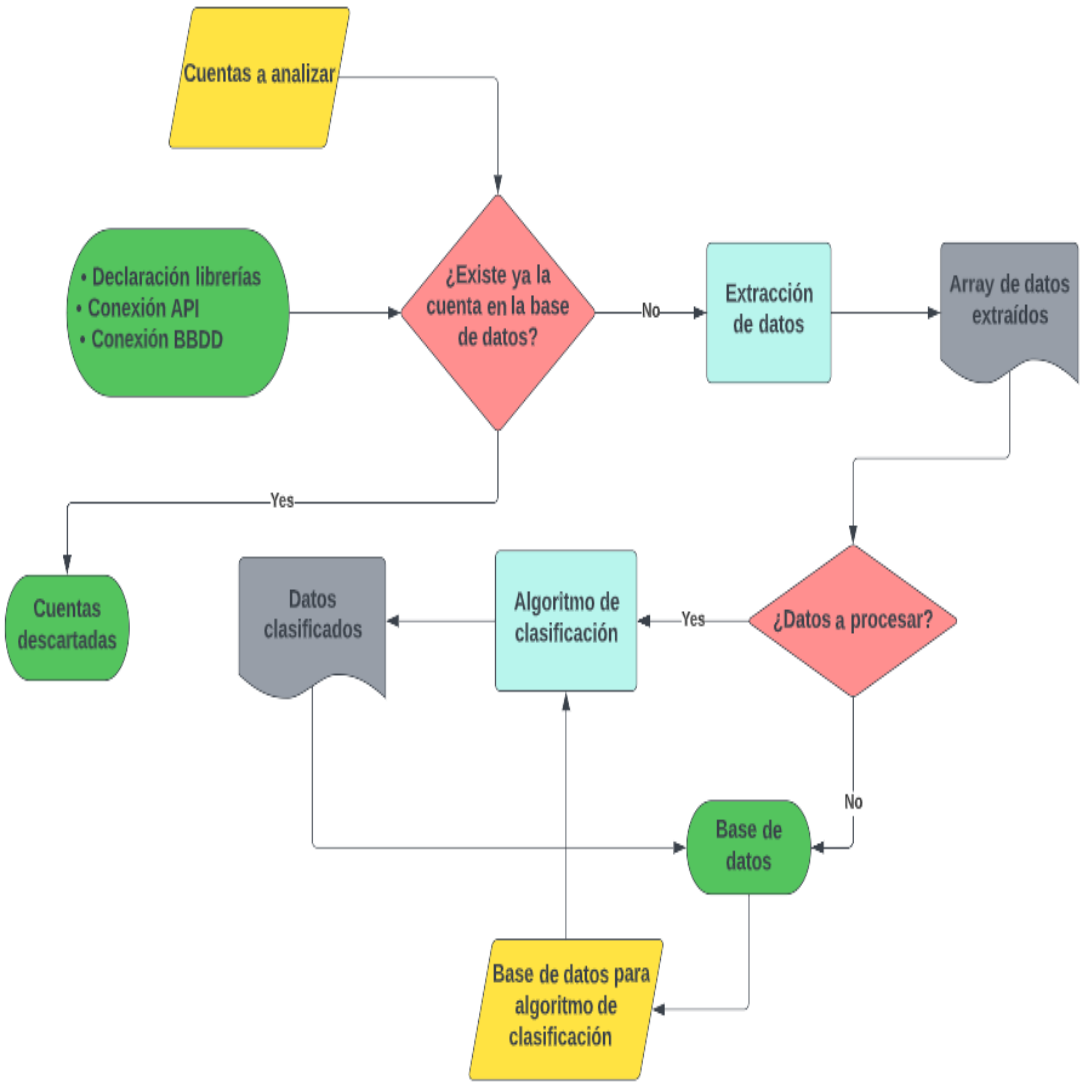
[36] «Ingresos anuales de Twitter a nivel mundial entre 2010 y 2021» https://es.statista.com/estadisticas/513538/twitter-ingresos-mundiales-anuales/ (accedido jun. 14, 2022)

[37] Musk gets Twitter for $44 billion, to cheers and fears of ‘free speech’ plan https://www.reuters.com/technology/exclusive-twitter-set-accept-musks-best-final-offer-sources-2022-04-25/ (accedido jun. 14, 2022)

PARTE II

PLANOS

PLANOS



PROYECTO	CLASIFICADOR DE PERFILES SOCIALES EN TWITTER	**PLANO Nº** **01**
PLANO	DIAGRAMA DE FLUJO DEL PROGRAMA	**FIRMA**
AUTOR	FERNANDO MARCOS-ALBERCA LIZCANO	25/05/2022

PARTE III

PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES

En este apartado se detallan las especificaciones técnicas del *hardware* y *software* utilizado en este proyecto.

SOFTWARE

Windows 10 – Sistema operativo instalado en el ordenador portátil

- Versión: Enterprise x64
- Procesador: Procesador compatible a 1 GHz
- Memoria RAM: 2 GB
- Almacenamiento: 20 GB

Microsoft Office 365 – Conjunto de programas informáticos como Microsoft Word para procesar y tratar textos o Microsoft Excel como hoja de cálculo

- Procesador. 2 núcleos a 1.6GHz o más
- Memoria RAM: 4GB
- Espacio en disco: 4GB
- Pantalla: 1280x768

Visual Studio – entorno de desarrollo compatible con Python

- Memoria RAM: 1GB
- Espacio en disco: 1GB
- Pantalla: 1024x768

MongoDB – Sistema de base de datos NoSQL

- Memoria RAM: 8GB
- Espacio en disco: 500MB

Spyder- entorno científico de desarrollo en Python para análisis de datos

- Memoria RAM: 1GB
- Python 2.7
- Anaconda

HARDWARE

Para el desarrollo del proyecto y de la memoria se ha usado un ordenador portátil MSI GP72M 7REX Leopard Pro con las siguientes especificaciones:

Ordenador portátil MSI GP72M 7REX Leopard Pro:

- Procesador: Intel core i7
- Velocidad de procesador. 2.5 GHz
- Capacidad de memoria RAM: 8GB
- Tipo de memoria RAM: DDR2 SDRAM
- Capacidad y descripción del disco duro: 1000 GB Híbrido
- Tarjeta gráfica: GTX 1050Ti
- Sistema operativo: Windows 10

PARTE IV

PRESUPUESTO

PRESUPUESTO

En este apartado del proyecto se establece un presupuesto del material usado y labores de realización. Se han establecido unos honorarios en base a unas horas empleadas de manera estimada.

Para calcular la amortización del software y hardware se usará la siguiente ecuación:

$$Coste(€) = Coste\ equipo \cdot \frac{Meses\ de\ uso}{Periodo\ de\ amortización} \cdot Porcentaje\ de\ uso \cdot Unidades \quad (4)$$

El porcentaje de uso estará en un intervalo de 0 a 1 en base al uso dado durante el periodo de tiempo empleado, mientras que el periodo de amortización será de 60 meses para el *hardware* y de 12 meses para el *software* de forma anual.

Software				
Herramienta	**Coste**	**Meses de uso**	**Uso**	**Total**
Microsoft Office Word 365	Gratuito	12	1	0,00 €
Microsoft Office Excel 365	Gratuito	12	0.6	0,00 €
MongoDB	Gratuito	10	0.5	0,00 €
Visual Studio	Gratuito	12	0.9	0,00 €
Spyder	Gratuito	12	0.3	0,00 €
		SUBTOTAL		**0,00€**

Tabla IV.1: *Presupuesto Software*

El *software* utilizado es gratuito, por eso su coste es de 0€. En el caso de las herramientas Office es a través de una licencia proporcionada por la Universidad de Castilla-La Mancha los estudiantes.

Hardware				
Herramienta	**Coste**	**Meses de uso**	**Uso**	**Total**
Portátil GP72M 7REX Leopard Pro	1050	12	1	210,00 €
		SUBTOTAL		**210,00 €**

Tabla IV.2: *Presupuesto Hardware*

Honorarios	**Coste (€/hora)**	**Horas**	**Total**
Desarrollo y memoria	40	450	18.000,00 €
	SUBTOTAL		**18.000,00 €**

Tabla IV.3: *Presupuesto honorarios*

Concepto	**Coste**
Hardware	210,00 €
Software	0,00 €
Honorarios	18.000,00€
SUBTOTAL	**18.210,00 €**

SUBTOTAL	18.210,00 €
% IVA	21 %
IVA	3.824,10 €
TOTAL	**22.034,10 €**

Tabla IV.4: *Presupuesto final proyecto*