TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Uso de algoritmos de aprendizaje máquina para la detección de ciberacoso en redes sociales

Estudiante: Mauro García Barro

Dirección: Diego Fernández Iglesias

Patricia Martín Rodilla

A Coruña, junio de 2023

A mis padres por haberme apoyado a lo largo de este trayecto

Agradecimientos

Agradecer tanto a mis padres como a todos aquellos familiares que me han motivado a no rendirme, y a mis tutores Diego Fernández Iglesias y Patricia Martín Rodilla por su gran ayuda con este trabajo.

Resumen

La evolución que ha presentado la tecnología en los últimos años ha permitido interacciones entre personas que se pensaban inviables, en especial la gran cantidad de comunidades que se alojan en las redes sociales.

No obstante, con ello se han trasladado muchas conductas tanto positivas como negativas, siendo una de estas últimas el ciberacoso, el cual ha aumentado de manera exponencial con el paso del tiempo, sobre todo en momentos de pandemia. Es por eso que precisamos anticipar y/o moderar estos comportamientos en la medida de lo posible.

Asimismo, debido al crecimiento de este tipo de casos, han surgido una gran cantidad de trabajos que intentan investigar, detectar o mitigar este problema mediante modelos de aprendizaje máquina.

Así pues, hemos aplicado diferentes algoritmos de *Machine Learning* a un dataset con información diversa acerca de diferentes comentarios en *posts* de *Vine*, tratando principalmente el aprendizaje supervisado e iterando sobre las distintas fases exploratorias de los datos mediante la metodología CRISP-DM.

Tras analizar y procesar dichos datos, se han contrastado los resultados de aplicar esos algoritmos, evaluando su rendimiento y deduciendo cuáles funcionan mejor bajo las condiciones dadas.

Abstract

The evolution of technology in recent years allowed many interactions among people that seemed impossible, specially the great amount of communities inside social networks.

Nevertheless, it has been accompanied with several positive as well as negative behaviours, cyberbullying being on the latter. Therefore, we need to anticipate and/or moderate this kind of behaviours as far as possible.

Likewise, due to the growth of this type of cases, a large number of works have emerged that try to investigate, detect or mitigate this problem through machine learning models.

With that in mind, we have applied different *Machine Learning* algorithms to a dataset with diverse information about different commentaries in *posts* of *Vine*, treating principally supervised learning and iterating between the distinct phases of data exploration with the methodology of CRISP-DM.

After analyzing and processing said data, we have compared the outputs of applying those algorithms, evaluating their performance and figuring out which ones work better with given conditions.

Palabras clave:

- Machine Learning
- Ciberacoso
- Aprendizaje supervisado
- Clasificación
- Grid Search
- Random Forest

Keywords:

- Machine Learning
- Ciberbullying
- Supervised learning
- Classification
- Grid Search
- Random Forest

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Durante los últimos años, especialmente en los tiempos de pandemia, se ha experimentado una evolución constante en lo que respecta a los servicios e infraestructuras tecnológicas proporcionadas con el fin de aumentar la cercanía y mejorar la interacción social entre diferentes grupos de personas. De esta forma, se pudo lograr una mayor inmediatez y simplicidad a la hora de establecer comunicaciones en múltiples ámbitos (laboral, académico, personal...), por lo que no resulta descabellado decir que en la actualidad actúan como herramientas esenciales en la mayoría de estos escenarios.

Asimismo, cabe señalar que pese a que esta masificación en el mundo digital ha sido un aspecto intergeneracional, a quienes han afectado en mayor medida son sin duda a los preadolescentes y adolescentes de hoy en día, es decir a la Generación Z, un grupo que desconoce un ambiente sin estos avances [1] y que de forma sorprendente son los que se encuentran más aislados a nivel social.

Por otro lado, esta cada vez más creciente normalización y alcance en el uso de dichas herramientas provoca que también se puedan extender ciertas conductas perjudiciales y malos hábitos, sobre todo si nos embarcamos en dominios con una menor moderación o control de estos patrones como son las redes sociales, siendo la más influyente los casos de ciberacoso o *cyberbullying*. Este se trata de la reiteración de comportamientos abusivos orientados hacia una o varias personas, el cual tiene como soporte principal los medios electrónicos. Al igual que sucede con su variante tradicional, es un problema con una gran afectación psicológica en quienes lo sufren (ya sean adolescentes o adultos), que por desgracia siempre está presente en mayor o menor medida y que resulta en ocasiones difícil de solventar [2]. Esto último se debe principalmente a que los encargados de aportar soluciones contra este problema no presentan la misma perspectiva que la mayor parte de los usuarios de estas plataformas, lo cual desemboca en una ligera desvinculación de estos con respecto a las nuevas generaciones.

Si bien es cierto que con la llegada de la pandemia y los confinamientos el acoso presencial se redujo de forma considerable, esto no provocó sino el efecto contrario en el ámbito del *cyberbullying* [3], dado que el tiempo de conexión a Internet por persona aumentó de forma significativa, derivando a su vez en una mayor probabilidad de estar expuestos a esta clase de problemas.

Además, cuenta con el añadido de que presenta una mayor diversificación e impacto sobre la víctima. Esto viene motivado tanto por la gran cantidad de opciones que tienen los agresores para atacarla como a la rapidez con la que se puede publicar información sensible sobre la misma, otorgando en el proceso una mayor visibilidad. En consecuencia, logran agravar el daño psicológico hacia la persona en cuestión e incluso llegando a permitir que, a diferencia de en su vertiente presencial, los acosadores consigan en la mayor parte de los casos salir impunes, gracias en gran medida a la capacidad de los nombres de usuario de otorgar un cierto anonimato.

Por ello, en la actualidad resulta muy relevante el poder detectar y monitorizar estos comportamientos, con el objetivo de visualizar qué tipo de publicaciones son más propensas a recibir una gran cantidad de comentarios ofensivos o, en su defecto, conocer qué clase de usuarios son potencialmente acosadores y tratar de controlar su aparición, además de asegurar un nivel adecuado de protección de las víctimas afectadas.

Una muestra de la importancia de la temática de este trabajo es el hecho de estar ligado a varias de las líneas estratégicas enmarcadas en el *Plan Estatal de Investigación Científica* [4], de las que se pueden destacar ciberseguridad e Inteligencia Artificial y robótica.

1.1 Relación entre ciberacoso y Machine Learning

Los problemas que implican el uso de procesamiento de lenguaje natural (NLP) siempre han sido relevantes en el área del aprendizaje máquina, ya que resultan demasiado complejos como para ser resueltos mediante la programación tradicional. Por otra parte, tenemos que en el ciberacoso el principal medio de propagación del mismo es mediante la publicación en redes sociales de comentarios con cierta connotación negativa, por lo que el tratar de asociar las palabras utilizadas en ellos para clasificarlos bajo un espectro de comportamientos es fácilmente encajable dentro de la clase de problemas mencionados previamente.

En consecuencia, nos encontramos con un escenario en el que se pueden reutilizar ciertos

conceptos de esta disciplina para derivar diferentes características que nos permitan aplicar una clasificación más precisa.

1.2 Objetivos

En este trabajo la finalidad principal reside en, partiendo de un dataset con unas características ya establecidas, contrastar el funcionamiento de diferentes modelos de *Machine Learning* para clasificar si ciertos comentarios de una red social (en nuestro caso, *Vine*) son ciberacoso o no. De esta forma, se pretende:

- Aplicar distintos métodos de análisis y exploración de datos para conocer su estructura básica.
- Utilizar mecanismos de Ingeniería de Características para reducir la dimensionalidad del conjunto y obtener aquellos atributos que mejor lo definan. Se trata de una serie de métodos de selección y/o transformación de los datos para maximimizar su relevancia respecto al aprendizaje máquina.
- Seleccionar, implementar y comparar diferentes algoritmos de *Machine Learning*.
- Evaluar el rendimiento de cada uno de los modelos por medio de una serie de métricas.

Asimismo, con el fin de extender esta investigación, se generarán otros tres datasets derivados del anterior, de tal forma que en cada uno se eliminará un determinado grupo o grupos de características para posteriormente someterlos a los procedimientos mencionados previamente. Esto nos permitirá reflejar en qué medida el prescindir de una serie de atributos influye o no en el resultado de los modelos, y en consecuencia cuáles pueden contener una información más valiosa a priori.

A la hora de familiarizarnos con el entorno y las tecnologías a utilizar, se ha empleado como principal referencia el libro *"Introduction to Machine Learning with Python"* [5].

1.3 Estructura del documento

Durante los siguientes capítulos detallaremos el cómo se ha llevado a cabo la realización de este proyecto, dando en primer lugar una descripción de la metodología en sí (en nuestro caso, CRISP-DM) y explicando brevemente en qué consisten cada una de sus fases[1], para posteriormente entrar en profundidad en el tratamiento de las mismas y en los procedimientos

[1] Nótese que el motivo de que los nombres de los capítulos relativos a cada fase de CRISP-DM estén en inglés es para hacer referencia a la nomenclatura original de cada etapa del ciclo.

que se han seguido para la compleción de cada tarea. Por último, se expondrán las principales conclusiones y lecciones aprendidas que se han extraído a partir del desarrollo de este trabajo, y se comentará de forma resumida una secuencia de pasos a seguir en caso de querer continuarse en un futuro.

Concretamente, seguiremos esta estructura:

- **Capítulo 2: Metodología CRISP-DM**
  - Explicación de su funcionamiento.
- **Capítulo 3: *Business Understanding***
  - Desglose de los principales conceptos y objetivos de negocio
  - Comentario acerca de las tecnologías a emplear
  - Planificación del proyecto
- **Capítulo 4: *Data Understanding***
  - Exploración y comprensión de los datos
- **Capítulo 5: *Data Preparation***
  - Explicación de los métodos empleados para limpiar, seleccionar y amoldar los datos conforme a los algoritmos de modelado.
- **Capítulo 6: *Modeling***
  - Selección e hiperparametrización de los modelos a entrenar.
- **Capítulo 7: *Evaluation***
  - Mecanismos principales para la medición del rendimiento de los diferentes algoritmos.
- **Capítulo 8: Conclusiones**
  - Análisis del grado de éxito del proyecto en base a los resultados obtenidos, y extracción de lecciones aprendidas.

1.4 Código asociado

En el siguiente repositorio de Github se pueden visualizar las principales implementaciones que se utilizaron como referencia para validar los resultados de este proyecto, así como este documento y otros archivos pkl con los modelos de entrenamiento de cada fase:

https://github.com/mgbarro/tfg-machinelearning

Capítulo 2

Metodología: CRISP-DM

Una metodología está formada por un conjunto de técnicas y actividades a seguir en un determinado proceso para obtener una serie de resultados esperados dentro de un campo concreto, independientemente de los medios o tecnologías empleadas en el transcurso del mismo [6].

En lo que respecta a la disciplina de *Data Science*, existen multitud de técnicas a abordar, entre las que incluimos KDD (*Knowledge Discovery in Databases*, un proceso automático de extracción de patrones en forma de reglas o funciones), SEMMA (*Sample, Explore, Modify, Model and Assess*; una secuencia de pasos centrada en las tareas de modelado más que en los objetivos de negocio) y CRISP-DM (*Cross Industry Standard Process for Data Mining*, focalizada principalmente en integrar de manera correcta la perspectiva de negocio), que es la que hemos tratado en nuestro caso al adaptarse de una mejor forma a los procesos a realizar en este trabajo.

CRISP-DM es un modelo de estándar abierto y uno de los más usados en lo que respecta a la exploración o minería de datos [7]. Tiene su origen en el año 1996 como un proyecto dirigido bajo la asociación de cinco empresas (SPSS, Teradata, Daimler AG, NCR y Ohra) que, tres años después, publicarían una guía paso a paso acerca de la puesta en marcha de un proceso de minería de datos. En la actualidad el proyecto original ya no se encuentra activo, aunque se han desarrollado diferentes herramientas que tratan de mejorar dicho modelo, teniendo a IBM (quien adquirió SPSS) como la principal promotora del mismo a través de su herramienta SPSS Modeler.

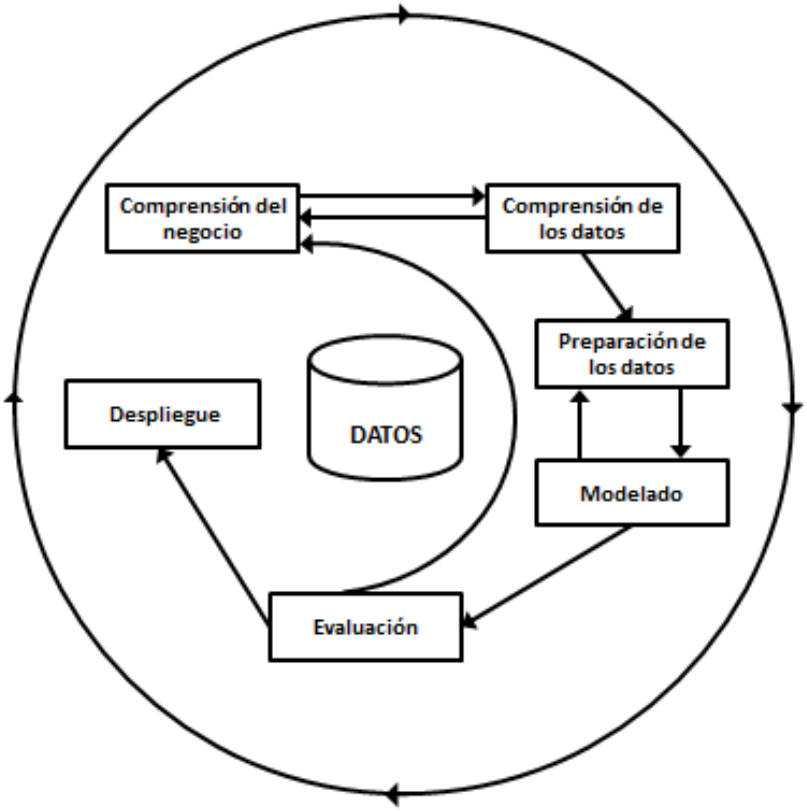

Figura 2.1: Ciclo de CRISP-DM.

En la figura 2.1[1] se puede ver que consiste en un ciclo de seis fases [8]:

- ***Business Understanding***: Su objetivo es comprender las metas a alcanzar tanto desde una perspectiva de negocio (lo que el cliente busca) como desde el punto de vista de la minería de datos (cómo lograr lo que busca el cliente). Así pues, se deben determinar los criterios de éxito, analizando la situación actual y los recursos a nuestra disposición, y en consecuencia elaborando un plan acorde al problema a solucionar.

- ***Data Understanding***: Se pretende explorar los datos extraídos para garantizar su calidad, así como conocer su estructura y relevancia. Para ello, una vez se recogen dichos datos, debemos averiguar la naturaleza de cada una de sus características, comprobando si ha habido problemas durante el proceso de recopilación de las muestras (valores atípicos, nulos, erratas...), y con ello decidir si cumplen con lo requerido para abordar los objetivos de negocio.

- ***Data Preparation***: Implica la selección y limpieza de los datos, además de una posible construcción de nuevas características, para acomodarlos a los modelos de Machine

[1] Imagen extraída de ResearchGate

Learning. También se incluyen posibles tareas de escalado y normalización para adaptar los valores a lo que puedan precisar ciertos modelos.

- ***Modeling***: Se escogen los modelos que se van a aplicar y se optimizan sus parámetros para poder proporcionar los mejores resultados. Posteriormente, usando un conjunto de entrenamiento, debemos estimar cuáles se comportan mejor en base al tipo de problema que tengamos (clasificación, regresión...).

- ***Evaluation***: Se evalúan los algoritmos seleccionados utilizando un conjunto de datos prueba y se revisa su rendimiento en base a ciertas métricas. De esta forma, se pretende observar los beneficios e inconvenientes de utilizar un modelo u otro, teniendo en cuenta su capacidad de compleción de los objetivos de negocio estipulados en fases anteriores.

- ***Deployment***: Se planifica la distribución del modelo en base a las condiciones dadas por ciertos escenarios reales y se monitoriza su ejecución, dando un informe final sobre la calidad del mismo. Al no contar con un escenario adecuado, hemos decidido prescindir de su puesta en marcha.

Cada una de estas fases se estructura a su vez siguiendo un esquema jerárquico de actividades genéricas y específicas, lo cual desemboca en diferentes instancias resultado de las tareas del segundo grupo. Una representación más clara de esta situación se puede ver en la figura 2.2[2].

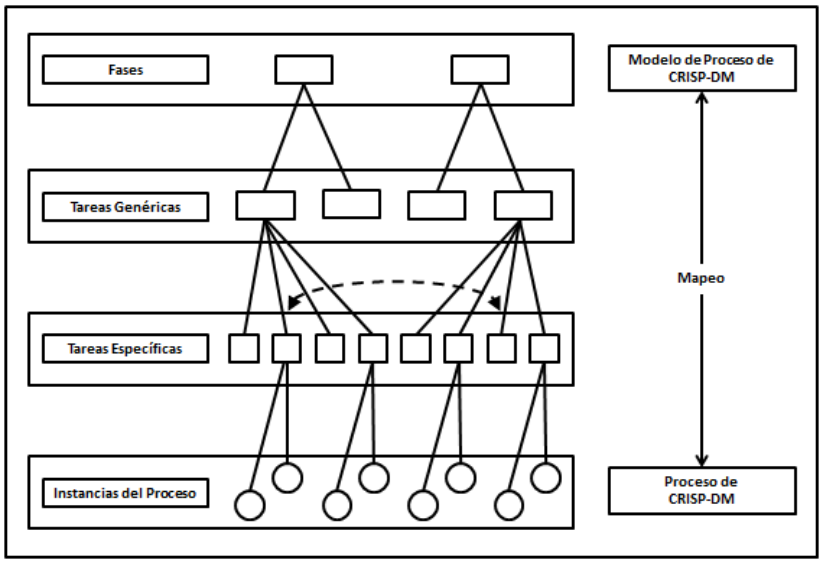

Figura 2.2: Jerarquía de actividades.

[2] Ídem figura 2.1

Asimismo, como se puede comprobar en la figura 2.1, este proceso presenta diferentes dependencias entre sus fases y no sigue una estructura rígida ni lineal, ya que en función del problema a resolver, pueden requerirse varias iteraciones con el fin de comprender verdaderamente el objetivo a seguir, permitiendo además que en cualquier momento se pueda volver a una fase previa para revisar los resultados obtenidos en otras tareas [7] [8].

A continuación, pasaremos a explicar en detalle las actividades que se han llevado a cabo en cada fase del ciclo.

Capítulo 3

Business Understanding

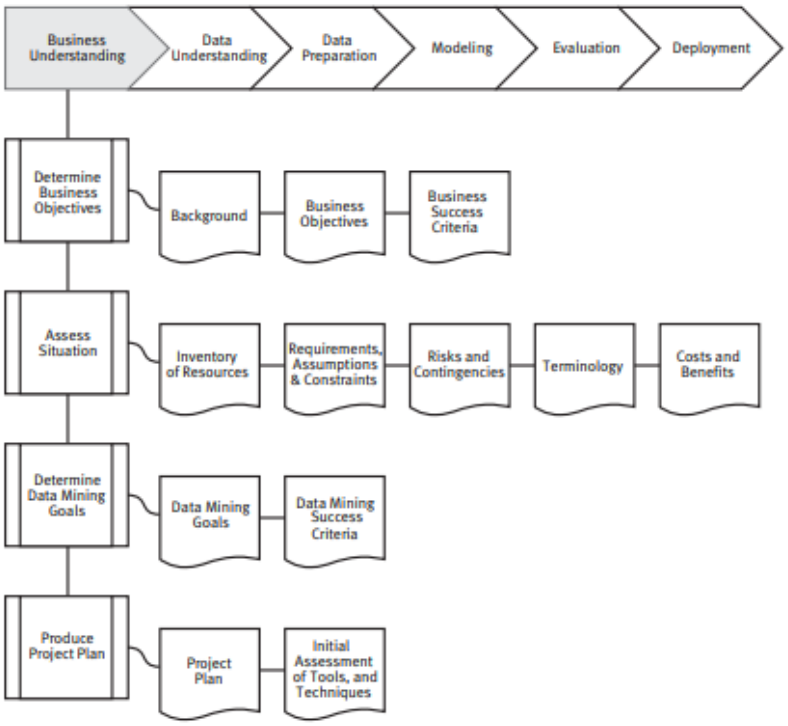

Figura 3.1: Tareas de la fase *Business Understanding*.

La comprensión de los diferentes retos y metas a alcanzar desde una perspectiva de negocio y de la minería de datos, así como de las tecnologías a utilizar es sin duda una tarea de gran importancia dentro de nuestro proceso, ya que el hecho de tener claro desde un inicio los requerimientos y las estrategias a llevar a cabo para cumplirlos nos garantiza tener una base sólida sobre la que sustentar el resto del desarrollo. Además, nos sirve como una referencia a seguir en caso de querer comprender mejor la finalidad de algunas de las características de

nuestro conjunto de datos.

En este capítulo describiremos cada una de las principales actividades de esta fase de CRISP-DM:

- **Determinar los objetivos de negocio.**
- **Evaluar la situación.**
- **Determinar los objetivos de la minería de datos.**
- **Elaborar un plan de proyecto.**

3.1 Objetivos de negocio

En este paso, se trata de descubrir qué es lo que realmente se quiere conseguir con el proyecto, identificando aquellos factores que resulten influyentes de cara al éxito del mismo. Para ello, tal y como se puede observar en la figura 3.1, debemos asegurarnos primero de conocer el trasfondo, dentro de lo cual se incluyen los datos y herramientas de las que disponemos para llevarlo a cabo y cómo funcionan. No obstante, teniendo en cuenta lo descrito en capítulos previos, en este escenario ya tenemos predefinidos los objetivos a cumplir, por lo que a partir de ellos deberemos deducir el trasfondo.

Para nuestro caso particular, debemos averiguar cómo se puede llevar a cabo un método de procesamiento del lenguaje natural (para disponer de una mejor comprensión a posteriori de algunas de las características de nuestro dataset), qué es exactamente *Machine Learning*, los tipos de modelos que hay y cuáles son los que más nos interesan a priori.

3.1.1 NLP: Cómo se procesa el lenguaje

NLP o *Natural Language Processing* representa al conjunto de técnicas utilizadas para permitir una comunicación eficiente entre humanos y máquinas por medio del lenguaje natural, es decir, aquel que ha evolucionado con el tiempo con fines para la comunicación humana tales como el español o el alemán [9], aplicando ciertas transformaciones sobre diferentes textos para obtener representaciones formales que puedan ser interpretables por un computador [10]. Asimismo, podemos dividirla en dos partes [11]:

- **NLU** (*Natural Language Understanding*): También conocido como *Lingüísticas*, es la parte que permite a las máquinas comprender el lenguaje natural mediante conceptos clave desde diferentes niveles (fonético, morfosintáctico, léxico semántico…).

- **NLG** (*Natural Language Generation*): Es el proceso responsable de generar fragmentos de texto que presenten cierta lógica, organizando la información entrante en base a la gramática y diferentes recursos lingüísticos.

Pese a que durante los años 80 y 90 hubo diferentes campos de investigación acerca de estos conceptos, no fue hasta los años 2000 cuando empezó a popularizarse esta disciplina, gracias sobre todo a la utilización de redes neuronales, dando lugar a algoritmos como *Word Embedding* o el mecanismo de *Attention*.

Por otra parte, esta disciplina tiene una gran cantidad de aplicaciones, tales como la recuperación de información, traducción automática o análisis de sentimientos. En esta sección vamos a repasar algunas de las principales técnicas utilizadas en este ámbito.

Sentiment analysis

También denominado comúnmente como *Opinion Mining*, consiste en la identificación de aspectos subjetivos dentro de un texto concreto, ya sea a nivel de palabras, frases o incluso documentos enteros, con el fin de determinar si este presenta un tono (sentimiento) positivo o negativo. Este análisis se suele desglosar en tres tareas [12]:

- Una detección de sentimientos u opiniones, que puede ser considerado como un problema de clasificación binario para ver si un texto es objetivo o subjetivo. Para ello, es habitual observar la influencia de determinadas palabras (normalmente adjetivos) en el resultado de una opinión.
- Una clasificación de la polaridad de dichas opiniones, que trata de ubicarlas dentro del rango de positividad o negatividad. Esta tarea está fuertemente ligada a la anterior, ya que por lo general un texto más subjetivo tiende a mostrar de una forma más notoria la polaridad asociada.
- Opcionalmente, se puede complementar con una tarea de descubrimiento del objetivo, con el fin de realizar este análisis de manera más precisa al intentar encuadrar cada opinión dentro de un contexto concreto.

Aplicado a nuestro escenario, este método nos permite reflexionar acerca de las emociones que pueda albergar un comentario concreto, pudiendo enmarcarlo de una forma más precisa dentro del ámbito del ciberacoso.

Para llevar todo este proceso a cabo, se puede seguir tanto una aproximación basada en *Machine Learning* (con algoritmos como *Tf-idf* o *Bag of Words*) como una basada en la semán-

tica de las palabras y frases (como es el caso de los *lexicons* o el *stemming*). Algunos de estos métodos se detallan a continuación.

Tf-idf

Es una evolución del algoritmo de IDF propuesto por Sparck Jones, el cual asignaba ponderaciones diferentes a cada palabra de un documento en base a su frecuencia de aparición. Sin embargo, *Tf-idf* considera que una palabra que es bastante frecuente en diferentes documentos debe presentar un peso menor que una palabra presente en una menor cantidad de ellos.

De esta forma, la parte de TF o *frecuencia de términos* nos permite determinar la afinidad de cada palabra con una cierta temática, mientras que la parte de IDF o *frecuencia inversa de documentos* nos mide la relevancia de cada palabra dentro del conjunto [13]. Esto nos resulta útil a la hora de comprobar qué elementos son más habituales dentro de un contexto de ciberacoso, pudiendo realizar una predicción más sencilla de dichos casos.

Modelado de tópicos

Es una tarea en la que a partir de un conjunto de documentos se pretende obtener una serie de tópicos, de forma que para cada uno de dichos documentos se tiene una proporción sobre cómo de afín es respecto a una temática en concreto. Estos modelos se suelen utilizar como complemento de otros modelos de aprendizaje para lograr predicciones más consistentes [14]. Por tanto, nos permiten definir un contexto en el que enmarcar determinados comentarios, el cual será más o menos preciso dependiendo del número de tópicos diferentes a tener en cuenta y la variabilidad de los mismos.

Dentro de estos algoritmos el más usado en investigaciones de *Machine Learning* es el LDA (*Latent Dirichlet Allocation*), un modelo generativo de connotación probabilística de un *corpus* o colección de documentos, en el cual cada uno de ellos es representado como una combinación aleatoria de diferentes tópicos [15]. A la hora de entrenar un modelo de este estilo, se tienen en cuenta dos parámetros: α, que denota la proporción de tópicos por documento; y β, que indica la cantidad de palabras por tópico a considerar.

Lematización y *stemming*

Tanto la lematización como el *stemming* son técnicas que buscan simplificar las diferentes palabras de un documento a una raíz base que permita agruparlas bajo un mismo criterio. La diferencia reside en que la primera pretende aportar el lexema de cada palabra tratando de

comprender el contexto en el que se encuentra, mientras que la segunda no utiliza esta información, sino que se basa en una serie de reglas para reducir cada palabra a un fragmento o *stem* idéntico, el cual no tiene por qué representar una palabra existente en el diccionario [16].

En lo que respecta a los algoritmos de *stemming*, podemos clasificarlos en tres grandes grupos:

- **Truncantes**: Consiste en eliminar los prefijos y sufijos de cada palabra, de forma que presenten una longitud similar.
- **Estadísticos**: Se basan en técnicas estadísticas para eliminar los diferentes afijos. Aquí se incluye el algoritmo de *n-grams*, que obtiene agrupaciones de n caracteres para obtener alguna relación entre palabras de similar estructura. No obstante, este concepto de agrupación también se puede aplicar a nivel de palabras dentro de un texto, como sucede en el caso de *Bag of Words*, donde cada palabra se tiene en cuenta como una entidad separada con el fin de ver su relevancia a partir de la frecuencia de aparición que presenta dentro de un documento. De hecho, este algoritmo se puede considerar un caso particular de *n-grams*, donde n = 1.
- **Mixtos**: Aquí se incluyen algoritmos que requieren de un corpus de un tamaño considerablemente superior.

3.1.2 Qué es Machine Learning

El aprendizaje máquina, según palabras de Arthur L. Samuel (uno de los pioneros en la demostración práctica de la Inteligencia Artificial), es un campo de estudio que otorga a los computadores la capacidad de aprender sin estar diseñados para ello [17].

Se trata de una rama científica que se mantiene en constante evolución, y que se encarga del estudio de diferentes modelos y algoritmos de connotación estadística capaces de, a partir de un conjunto de datos, ejecutar ciertas tareas de forma automática sin tener una programación explícita para llevarlas a cabo.

Este concepto lleva manejándose desde los años 50, cuando Alan Turing mencionó en el test que lleva su nombre la posibilidad de que las máquinas puedan tener inteligencia propia [18]. En las décadas posteriores, se realizaron incontables investigaciones acerca del tema, pudiendo dividirlas arbitrariamente en tres fases:

- **Edad del Razonamiento**: Comprendida entre los años 50 y 70, se busca explotar el

razonamiento lógico de las máquinas, principalmente mediante la demostración de diferentes teoremas matemáticos.

- **Edad del Conocimiento**: Como su nombre indica, se trata de aplicar una serie de técnicas para maximizar la extracción de conocimiento. Se sitúa entre los años 80 y 90, y es aquí donde se acuña el término de Aprendizaje Máquina, convirtiéndose así en una rama de investigación independiente de la Inteligencia Artificial y dando pie a la aparición de una gran cantidad de modelos y variantes, desde los basados en simbolismos y explicaciones (EBL, *Explanation-Based Learning*), hasta otros de aprendizaje estadístico (SVM, *Supported Vector Machines*) e inspirados en redes neuronales.

- ***Deep Learning* y *Big Data***: En esta etapa, la cual se extiende desde los años 2000 hasta la actualidad, gracias a la evolución de los recursos computacionales se vuelven populares los métodos conexionistas y se refuerza la idea de utilizar redes neuronales, pues permiten procesar una gran cantidad de datos y de gran complejidad como imágenes o vídeos. A día de hoy hay ciertos autores que lo consideran un subconjunto de la rama de investigación de *Machine Learning* [19], y si bien es cierto que presenta un gran e interesante trasfondo, su análisis queda fuera del alcance de este trabajo.

Aún así, cabe resaltar que no existe un algoritmo único e inmejorable para la resolución de estos problemas ya que en última instancia dependerá, entre otros aspectos, del método de aprendizaje, el objetivo a lograr y de cómo se distribuyan los datos en cuestión. Según el método de aprendizaje podemos distinguir tres grandes grupos:

- **Aprendizaje supervisado**: Consiste en tratar de asociar una entrada con una salida basándose en pares entrada-salida ya establecidos, por lo que requieren de intervención externa para su correcto desarrollo. El objetivo es dividir los datos en un conjunto de entrenamiento, con el que intenta establecer una serie de patrones, y otro de test sobre el que se intenta predecir las salidas deseadas. Aquí se incluyen modelos como los árboles de decisión, Naïve Bayes o SVM.

- **Aprendizaje no supervisado**: A diferencia del grupo anterior, no requiere de revisión o inclusión de patrones previos, sino que es el propio algoritmo quien trata de aprender cuáles son las salidas más probables a partir de las diferentes entradas. De esta forma, cuando se introducen nuevos datos, este los clasifica en base a lo ya aprendido, logrando agrupación de los mismos, y se retroalimenta para predicciones futuras. Podemos destacar métodos como *K-Means Clustering*, otros de agrupamiento jerárquico e incluso redes neuronales.

- **Aprendizaje semisupervisado**: Es una combinación de los dos grupos anteriores, utilizado sobre todo cuando resulta complejo obtener datos etiquetados. Ejemplos de estos

modelos son el aprendizaje por conjuntos (*Ensemble Learning*) y aprendizaje por refuerzo.

Asimismo, en función del objetivo a cumplir podemos agruparlos en algoritmos de **clasificación** (KNeighbors, SVC [*Support Vector Classifier*], Naïve Bayes...), **regresión** (SGD [*Stochastic Gradient Descent*], SVR [*Support Vector Regression*], entre otros), **clustering** (K-Means, GMM) o de **reducción de dimensionalidad** (PCA [*Principal Component Analysis*] aleatorizado, RFECV [*Recursive Feature Elimination with Cross-Validation*], LLE [*Locally Linear Embedding*]...).

En la figura 3.2 se puede ver un diagrama de flujos de qué algoritmos escoger en base a las características de los datos y las metas a seguir.

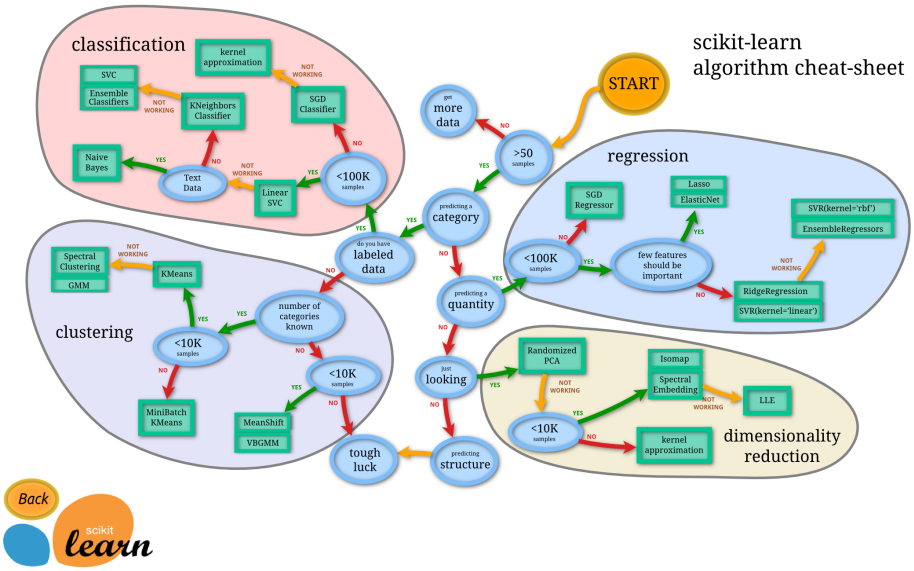

Figura 3.2: Diagrama de flujo de algoritmos.

De esta forma, para nuestro caso particular nos inclinaremos por los algoritmos de clasificación, si bien es cierto que también acudiremos a los de otros tipos para llevar a cabo ciertas fases del proyecto.

A continuación, vamos a revisar algunos de los modelos más destacados dentro de los problemas de clasificación, y que utilizaremos en este trabajo, a fin de hacernos una idea del funcionamiento básico de los mismos.

3.1.3 Naïve Bayes

Se trata de un modelo de clasificación con connotación probabilística, pues está basado en el teorema de Bayes, con la excepción de que se asume independencia entre las variables predictoras, lo que en nuestro caso se correspondería con las características del dataset. De esta forma, cada una de ellas contribuye de forma paralela a la acepción de un rasgo concreto, sin tener en cuenta la presencia del resto de atributos. Esta simplificación es la que le otorga esa ”ingenuidad”, permitiendo optimizar los tiempos de ejecución a costa de perder rendimiento en el proceso.

Asimismo, resulta un algoritmo bastante compatible con los métodos de aprendizaje supervisado y fácil de entrenar, ya que al sólo precisar conocer las varianzas de cada característica, no requiere de un elevado número de muestras para su desarrollo.

3.1.4 Random Forest

Los árboles de decisión se caracterizan por ser unos algoritmos muy similares a los sistemas de predicción basados en reglas, pues a partir de una serie de muestras y respondiendo a diferentes preguntas sobre las mismas, se desarrollan las ramificaciones que en cierto modo representan los caminos a seguir a la hora de clasificar un dato concreto, tal y como se puede ver en la figura 3.3. Estos árboles pueden presentar o no una profundidad limitada, dependiendo de con qué exactitud se busque determinar el rasgo de una muestra. Sin embargo, cabe resaltar que este modelo es susceptible de realizar modificaciones en el conjunto de entrenamiento, lo cual puede desembocar en estimaciones completamente distintas o incluso erróneas.

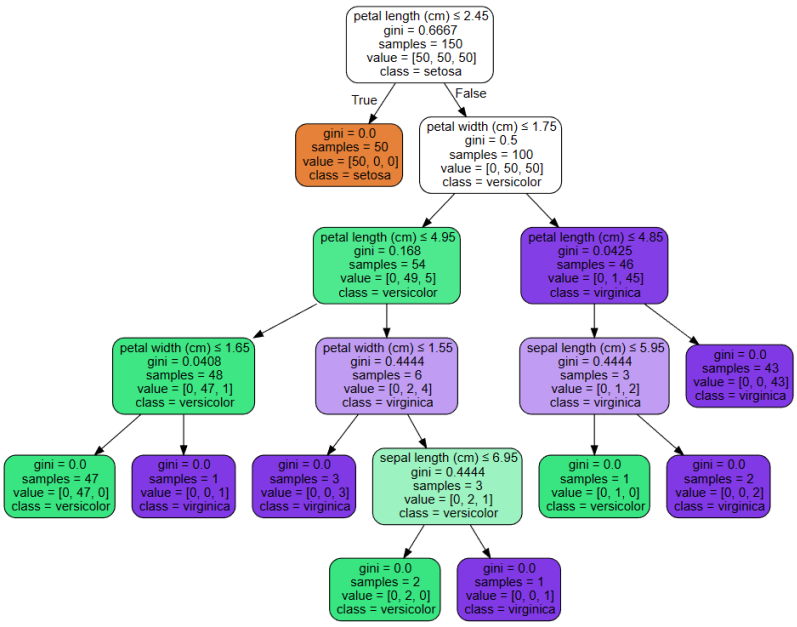

Figura 3.3: Ejemplo de un árbol de decisión.

Es por eso que se prefiere recurrir a los Random Forest, pues estos son una colección de árboles de decisión, donde cada uno se desarrolla mediante un proceso de *Bagging*. Esto último consiste en designar a cada árbol un subconjunto aleatorio de muestras, a fin de evitar las correlaciones ya mencionadas, para finalmente establecer un promedio de los mismos.

Gracias a esto se aumenta la precisión de las predicciones y se disminuye el sobreajuste causado por utilizar un sólo árbol de decisión, ya que siempre se escogerían aquellas características que permitiesen una mayor diferencia entre los conjuntos de datos existentes en cada nodo de decisión. En cambio, al utilizar un Random Forest, nos aseguramos una mayor aleatoriedad en estas decisiones.

No obstante, cabe señalar que esto de por sí no es el único factor a tener en cuenta de cara a solucionar estos problemas, ya que al igual que con el resto de modelos, se deben revisar los parámetros del estimador y tratar de ajustarlos para dar el mejor resultado posible pero sin caer en el sobreentrenamiento, en un proceso denominado *hiperparametrización*, sobre lo cual hablaremos con más detenimiento en la fase de modelado.

3.1.5 AdaBoost

Acrónimo de *Adaptative Boosting*, es un algoritmo estadístico de clasificación propuesto por Yoav Freund y Robert Schapire en el año 1995, cuyo funcionamiento se basa en la mejora

de algoritmos "débiles"[1] ya existentes mediante una combinación ponderada de sus salidas al aplicarse respecto a un conjunto de datos.

Está comprendido por tres fases [20]:

- ***Scouting***: Se testea cada modelo perteneciente al conjunto, añadiéndole un cierto coste e^{β} cada vez que comete un error en una predicción, y $e^{-\beta}$ en caso contrario.
- ***Drafting***: Se lleva a cabo un proceso iterativo en el que se escogen los algoritmos con mejor desempeño, analizando los costes obtenidos en la fase anterior y obteniendo aquellos con menor coste asociado respecto a los errores.
- ***Weighting***: Se trata de determinar el peso de cada algoritmo seleccionado, a fin de determinar su contribución en la salida final del modelo.

3.1.6 LinearSVC

Las máquinas de soporte vector (del inglés *Supported Vector Machines*, SVM) son algoritmos de clasificación supervisados no probabilísticos que tratan de ver el proceso de aprendizaje desde una perspectiva estadística y gráfica. Para ello, buscan asociar diferentes muestras de entrenamiento con puntos concretos en el espacio, de tal forma que a medida que van apareciendo nuevas predicciones están se sitúan en posiciones similares, tratando de generar así un hiperplano lineal, es decir, dos zonas fronterizas, con la mayor distancia posible [21].

Asimismo, se utilizan en el cálculo de dichos valores lo que se conoce como *funciones kernel*, las cuales permiten resolver problemas no lineales computando los productos internos de cada característica sin necesidad de construir la representación multidimensional (*kernel trick* [22]), dando lugar a un área de decisión lineal. Estas funciones presentan distintos tipos: lineales, RBF (*Radial Basis Function*), polinómicas y sigmoidales, siendo los más comunes los dos primeros.

El kernel de tipo RBF realiza los mapeos de muestras de forma no lineal dentro de un espacio con múltiples dimensiones. No obstante, cuando el número de características es muy elevado, es recomendable utilizar la aproximación de kernel lineal.

[1] Con este término nos referimos a aquellos modelos que presenten una precisión muy parecida a otros de índole aleatoria.

3.2 Evaluar la situación

El mundo del ciberacoso es tal y como comentamos en la introducción un tema de gran relevancia no solo a nivel de introspección social, sino también a nivel de estudio [2] [23], por lo que disponemos de una gran cantidad de fuentes sobre las que analizar este escenario, dentro de las cuales los comentarios en redes sociales resultan de gran relevancia.

Es por eso que el conjunto de muestras que hemos seleccionado está formado por una serie de comentarios de la red social *Vine*, sobre el cual ya se han realizado previamente diferentes estudios al respecto [24] [25]. De hecho, durante la fase de evaluación utilizaremos algunas métricas referenciadas en el artículo [24] para comparar nuestros resultados y ver si hemos conseguido alguna mejora.

3.2.1 Tecnologías a utilizar

Para el desarrollo de este trabajo se han tenido en cuenta diferentes alternativas, pero por cuestiones de comodidad y practicidad hemos empleado como base Anaconda, una distribución libre de Python y R muy utilizada en investigaciones de ciencia de datos y estadística, ya que nos permite tener múltiples entornos con diferentes versiones de Python para mantener una separación entre distintas tareas, además de disponer de una gran cantidad de paquetes para el estudio de *Data Science*, entre los cuales destacamos *scikit-learn, SciPy, Jupyter Notebook, seaborn, matplotlib, NumPy y Pandas [26].*

Jupyter Notebook

Jupyter Notebook consiste en una plataforma web para la computación interactiva de los tres lenguajes de programación que hacen referencia a su nombre: Julia, Python y R. En lo que respecta a nuestro trabajo, su uso nos resulta muy útil a la hora de poder ejecutar bloques de código de forma sencilla y comprobar de forma casi inmediata el resultado de los mismos, sin necesidad de utilizar IDEs (acrónimo de *Integrated Development Environment*) de por medio.

scikit-learn

Scikit-learn es una librería open-source que incluye una gran cantidad de modelos y documentación acerca de *Machine Learning*, siendo de las librerías más completas y utilizadas en este ámbito [27].

Entre sus principales ventajas, cuenta con un gran sustento dentro de Python, por lo que las implementaciones que ofrece de los diferentes algoritmos de aprendizaje máquina están pensadas para garantizar la mayor integración y eficiencia computacional posibles, pese a la naturaleza interpretativa de la programación en este lenguaje. Asimismo, como mencionamos previamente, la comunidad de esta librería proporciona una documentación concisa y bien mantenida acerca de cada uno de los modelos, lo cual permite conocer en mayor medida el funcionamiento de los mismos.

Este paquete cubre cuatro conceptos relacionados con *Machine Learning*: transformación de datos, evaluación y selección de modelos, y aprendizaje supervisado y no supervisado.

La transformación de datos es uno de los pasos principales en lo que respecta al **preprocesado** de un dataset, en donde se realiza la normalización o escalado de los diferentes valores de cada característica de susodicho conjunto, pero la comentaremos de forma más concreta en el Capítulo 5.

En la evaluación y selección de modelos, se busca evitar el sobreentrenamiento de los algoritmos, aplicando diferentes técnicas como la **validación cruzada** (obtener diferentes particiones de los datos y usar de forma aleatoria en cada iteración unas para el conjunto de entrenamiento y otras para el conjunto de test o validación). De ello hablaremos en los Capítulos 6 y 7.

3.3 Objetivos del Data Mining

Relacionado con la sección anterior, en este proyecto debemos cumplir dos tareas:

- Predecir si los comentarios son o no ciberacoso, es decir, estamos ante un caso de clasificación binaria.
- Desarrollar el ciclo de vida establecido por CRISP-DM hasta la etapa de evaluación de modelos, pues su cumplimiento es vital de cara a garantizar el éxito del proyecto.

3.4 Establecer el plan de proyecto

A la hora de establecer la planificación de un proyecto, debemos primero conocer la metodología que se va a seguir, y como bien hemos comentado en el Capítulo 2, hemos empleado CRISP-DM. No obstante, se ha utilizado más como una referencia que una pauta a seguir, ya que por cuestiones de tiempo se han evitado ciertos formalismos, además de que no se ha

tenido en cuenta la fase de despliegue de cara a la realización de este trabajo.

Por otro lado, a los directores del proyecto Diego Fernández Iglesias y Patricia Martín Rodilla se les ha designado el rol de supervisores, mientras que al autor del trabajo Mauro García Barro se le ha encomendado el rol de analista desarrollador. A continuación, se comentarán las etapas seguidas en el desarrollo y cómo se han ido afrontando en el transcurso del tiempo.

3.4.1 Comienzo

En la reunión inicial se presentó el proyecto en cuestión y se designaron las primeras pautas a seguir, lo cual incluía una explicación acerca del ciclo de vida de CRISP-DM, además de resolver las principales dudas acerca de las tecnologías a emplear. Asimismo, se estableció el convocar el resto de reuniones en un intervalo lo más regular posible, en torno a las dos semanas respecto a la anterior, para tratar los avances realizados y estudiar las siguientes tareas a llevar a cabo.

Posteriormente, se dio un plazo de 2-3 semanas para familiarizarse con el entorno y aprender lo básico sobre aprendizaje máquina.

3.4.2 Desarrollo de la planificación

Con el fin de facilitar el entendimiento del plan de proyecto, cada una de las fases del mismo se ha denominado de igual forma que cada etapa del ciclo de vida de CRISP-DM, de forma que en todo momento sabemos dónde se ubica nuestro trabajo y en caso de haber algún contratiempo podemos abordarlo de una mejor manera. También recalcar que para que cualquiera de estas se inicie se debe haber finalizado la anterior, y que la redacción de la memoria se ha realizado de forma paralela durante la segunda mitad del proyecto.

En la tabla 3.1 se puede ver una recopilación de la duración de cada fase, y en la figura 3.4 se incluye un *Diagrama de Gantt* con la línea base, es decir, la estimación de fechas inicial que se tuvo en cuenta de cara a la finalización del proyecto. Se puede observar que hubo ciertos retrasos en lo que respecta a las etapas de recolección de datos y de reducción de dimensionalidad, ya que tomó cierto tiempo el poder analizar el comportamiento de ciertas características, así como el procesamiento de los diferentes atributos mediante los algoritmos de RFECV y PCA, que detallaremos más adelante.

Por otro lado, el haber planificado con anterioridad cada una de las fases del proyecto y la realización de los diferentes seguimientos nos dio una estimación más precisa acerca de la duración de cada una y las dificultades que se podían llegar a manifestar.

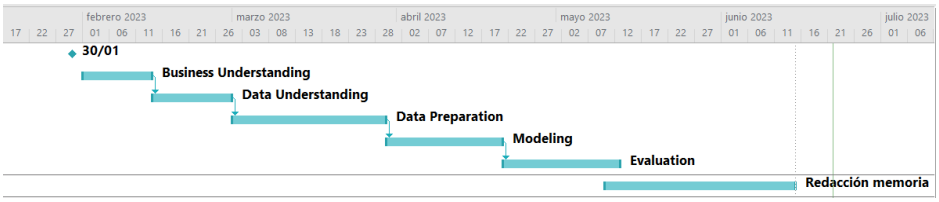

Figura 3.4: Diagrama de Gantt con línea base.

Tarea	Estimado (semanas)	Real
Business Understanding	2	2
Familiarizarse con las tecnologías	1	1
Revisar conceptos de Machine Learning	1	1
Data Understanding	2	3
Recolectar y explorar datos	2	3
Data Preparation	4	5
Limpieza, detección de anomalías y formateo	2	2
Reducción de dimensionalidad	2	3
Modeling	3	3
Selección de modelos	1	1
Diseño del entrenamiento	1	1
Hiperparametrización	1	1
Evaluation	3	3
Contraste de modelos	2	2
Revisión y líneas futuras	1	1
Redacción memoria	5	5
Núm. reuniones (se estima cada dos semanas)	6	12
Total de horas dedicadas	272	292

Tabla 3.1: Seguimiento

3.4.3 Estimación de costes

Para poder establecer un posible cálculo del coste económico del proyecto, se ha de tener en cuenta tanto aquel asociado a los recursos humanos como a los recursos materiales.

Para el primer caso se ha recurrido a la página del BOE [28] (siendo la versión más reciente que hemos podido encontrar la del día 6 de marzo de 2018), referente a los salarios en base a los diferentes niveles de profesión. Así pues, teniendo en cuenta los roles designados previamente, así como las descripciones que se aportan sobre los grupos profesionales respecto a esta área y la columna de salarios a partir de diciembre de 2019, se pueden concluir los siguientes gastos:

- Que el alumno, encargado del desarrollo del proyecto, se puede ubicar dentro del Área 3 grupo B II, al cual se le atribuiría un salario base de 23542.78 € anuales, lo cual asumiendo un año laborable de 1800 horas se traduciría en unos 13.08 €/hora.
- Que los directores del trabajo, encargados de supervisar cada fase del proyecto, se pueden enmarcar en el grupo A del área 3, lo cual se traduce en un salario base anual de 25035.54 €, o de forma equivalente 13.90 €/hora.

Teniendo en cuenta todas estas cifras, suponiendo una dedicación semanal media de 14 horas por parte del alumno, y de 1 hora como duración estimada de cada reunión, tendríamos los siguientes costes:

- Respecto al alumno, cuyo trabajo total real supondría unas 292 horas (ver tabla 3.1), tendría un coste de 3819.36 €.
- Respecto a cada supervisor, cuyo trabajo contabilizado son las reuniones establecidas con el alumno más el tiempo dedicado a la revisión de la memoria, se consideraría un coste de 176.80 €.

Con respecto a los recursos materiales, se ha empleado un equipo cuyo valor de mercado es 800 € para la realización de las diferentes pruebas. Dado que el software involucrado es de uso libre y las licencias afectadas persiguen la misma finalidad, sólo tendremos en cuenta el coste de amortizar dicho equipamiento. Así pues, sabiendo que un computador o cualquier otro equipo informático presenta un período de amortización estimado de 4 años (lo que se traduce en 48 meses) y que este se ha utilizado durante un período de 4 meses, tenemos que:

$$Amortizado = \frac{800€}{48} * 4 = 66.67€ \tag{3.1}$$

Esto nos devolvería un total de 3819.36 + 2 * 176.80 + 66.67 = 4239.63 €.

Pese a ello, en un escenario real también habría que tener en cuenta ciertos costes indirectos como gastos de luz, agua, Internet u otros relativos a la organización del proyecto.

Capítulo 4

Data Understanding

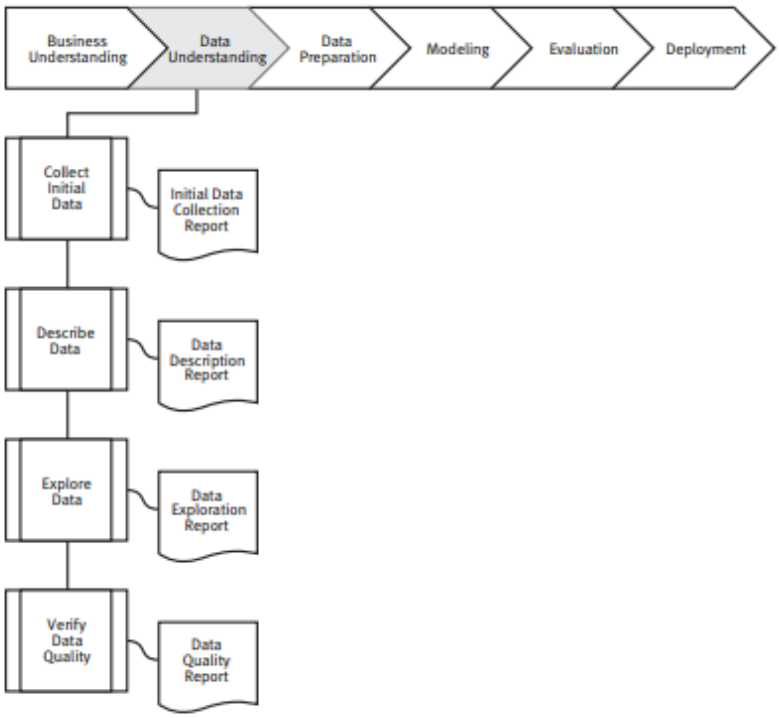

Figura 4.1: Tareas de la fase *Data Understanding*

En esta fase se pretende realizar un análisis de los datos extraídos, a fin de detectar posibles anomalías que entorpezcan o manipulen el desarrollo de las fases posteriores, de forma que garanticemos su calidad y obtengamos una explicación clara de los principales rasgos del conjunto de muestras.

De esta manera, tal y como se observa en la figura 4.1, podremos dividir esta etapa en varias tareas: recolección de los datos, descripción y exploración de los mismos, y verificación de su calidad, todo ello para dar lugar a un informe exhaustivo que recopile estos detalles.

4.1 Recolección de los datos

En este caso el elaborar un dataset propio a partir de un procedimiento de obtención de ciertas muestras no es un objetivo dentro de este trabajo por lo que, como bien se ha mencionado en el capítulo anterior, hemos decidido utilizar uno ya creado con anterioridad.

Este conjunto se ha obtenido de forma serializada como un archivo .pkl, el cual se ha transformado a un csv para obtener un vistazo completo de su contenido y permitir su mejor manipulación, y finalmente con la librería *Pandas* se ha incluido en forma de dataframe para poder facilitar la fase de exploración.

4.2 Descripción y exploración de los datos

El dataset consta de 56799 muestras, donde cada una de ellas representa a un comentario dentro de una *session o post* de Vine, la cual a su vez se puede identificar mediante el atributo de *session_id*. Asimismo, para cada uno de los datos se incluyen una serie de propiedades en forma de columnas, disponiendo así de unas 447 características (siendo la mayoría de tipo flotante y el resto con valores enteros).

Cabe señalar que este dataset es una simplificación de un conjunto de datos más grande, el cual fue extraído siguiendo un proceso de muestreo por ”bola de nieve” con información acerca de miles de usuarios, agregando diferentes *posts* con sus respectivos comentarios bajo lo que consideran como *media session*, así como una serie de atributos obtenidos mediante un proceso de LDA, tal y como se detalla en estos artículos [25] [24]. Con ello se obtuvieron cerca de 652000 sesiones, de las cuales se descartaron aquellas que tuviesen menos de 15 comentarios con el objetivo de facilitar el proceso de etiquetado, ya que uno de los fundamentos del *cyberbullying* es la frecuencia de repetición de dichos actos. Finalmente, se mantuvieron aquellas muestras con una confianza por encima del 60%.

Por otra parte, se incluyeron tres tipos adicionales que según la tesis de referencia [25] consiguen mejorar los resultados de [24]: similitud de *Bag of Words* mediante el cálculo del *Tf-idf*; una serie de atributos relacionados con el tiempo entre pares de comentarios, distinguiendo entre la diferencia con el último comentario y con el resto de comentarios de una

session, y que para los cuales se calcula a su vez el mínimo, máximo, media, mediana y desviación típica; y finalmente un conjunto de características *doc2vec*.

De entre todas las características incluidas, podemos destacar:

- ***_profane_words**: Indica el número de palabras malsonantes dentro de un comentario o una *session*.
- ***_polarity/*_subjectivity**: Hacen referencia a la positividad y subjetividad de los comentarios, obtenidas a través de un proceso de *Sentiment Analysis* mediante el uso de la librería *TextBlob* [29].
- ***_tfidf_***: Tienen que ver con la frecuencia inversa de determinadas palabras clave en los corpus o comentarios, las cuales son obtenidas mediante un método de procesamiento del lenguaje natural, que en este caso es *Bag of Words*. Con ello se obtiene la afinidad de cada comentario o *session* con respecto al resto de elementos de su mismo tipo.
- ***_lda_***: Indican cómo de relacionado se sitúa un comentario o una *session* con respecto a ciertos temas o tópicos.
- ***_doc2vec_***: Constituyen una representación vectorial de los comentarios, de forma que determinan las relaciones de adyacencia existentes entre diferentes documentos mediante el uso de palabras clave, que en nuestro caso se aplica tanto a nivel de comentarios como de *session*.
- ***label***: Es considerada la variable *target* de este conjunto, y por tanto nos permite determinar si un comentario es o no ciberacoso.

Adicionalmente, hemos agrupado aquellas que en un primer momento consideramos que pueden tener una relevancia interesante de la siguiente forma:

- **Discretas con pocos valores**: *session_id, comment_id, this_comment_n_words*.
- **Discretas con un amplio rango definido y/o que contengan valores negativos**: *this_comment_polarity, this_comment_subjectivity*, y sus variantes a nivel de *session*.
- **Continuas**: Aquellas que tienen que ver con *tfidf, lda y doc2vec*.

En las figuras 4.2, 4.3 y 4.4 se pueden observar algunos ejemplos de distribuciones de estas variables.

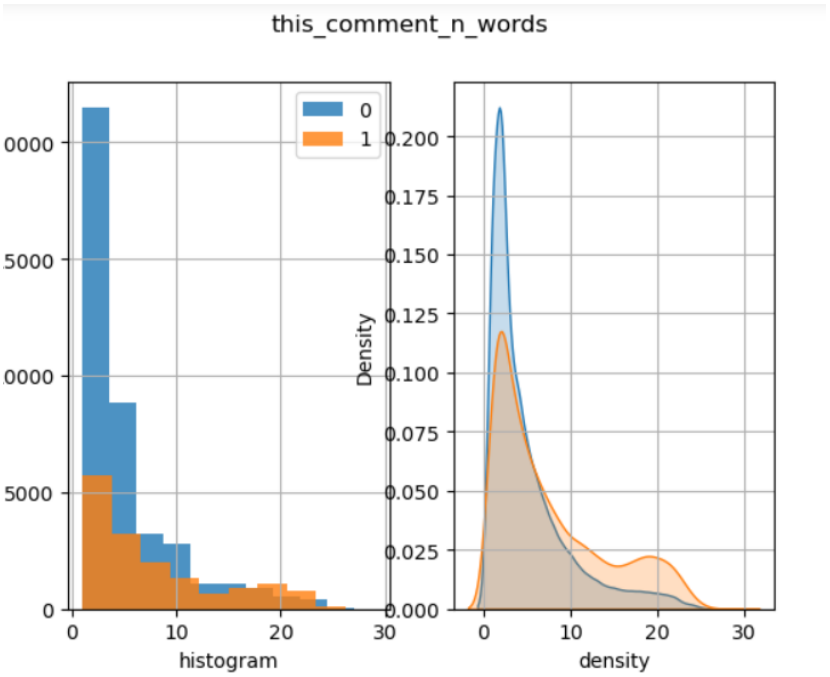

Figura 4.2: Distribución de *this_comment_n_words* frente a *label*

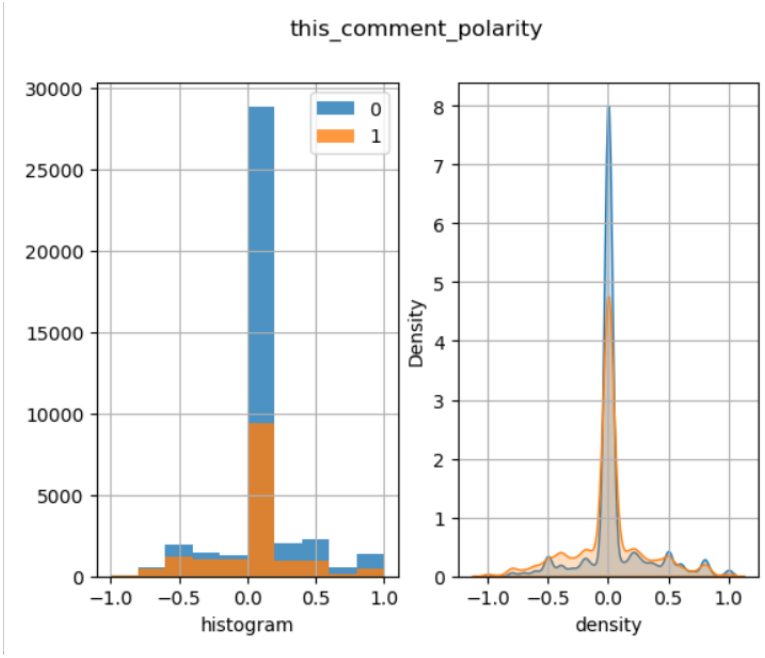

Figura 4.3: Distribución de *this_comment_polarity* frente a *label*

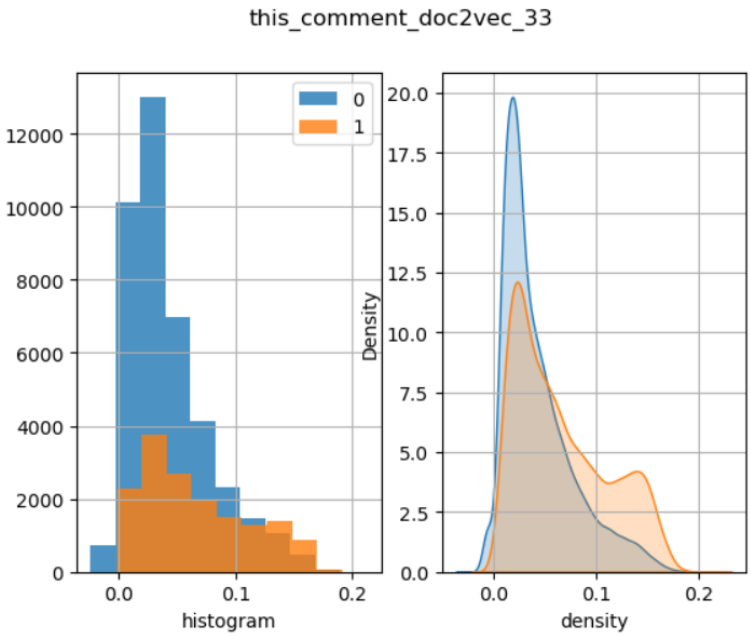

Figura 4.4: Distribución de uno de los componentes *this_comment_doc2vec* frente a *label*

Por otra parte, podemos considerar la característica *label* de tipo binaria, pues tan solo presenta valores de 0 o 1 para indicar la presencia de ciberacoso. Además, se puede ver que del total de muestras, 40389 son de *label* = 0 (**aprox. 72%**) y el resto de *label* = 1, y si bien puede aparentar una proporción ligeramente desbalanceada, es un valor asumible de cara al análisis posterior.

Con respecto a los datasets derivados, presentan los mismos atributos que el dataset original, que de ahora en adelante pasará a ser nombrado como *Baseline*, con las siguientes diferencias:

- En uno de ellos se encuentran ausentes aquellos rasgos relativos a LDA, y que a partir de este instante pasaremos a considerar como *WithoutLDA*.
- En el segundo se eliminaron aquellas columnas que tenían que ver con *doc2vec*, que denominaremos de forma análoga como *WithoutDoc2vec*.
- En el último se descartaron tanto las características de LDA como de *doc2vec*, al cual nos referiremos como *WithoutBoth*.

No obstante, todos estos datasets los tendremos en cuenta de cara a la etapa de selección de características (ver sección 5.5), y para los diferentes análisis mostrados a continuación tomaremos el caso de *Baseline*.

4.2.1 Correlaciones lineales

Además de ver de forma genérica cómo se encuentran recopilados los datos, también resulta útil conocer las diferentes relaciones entre estos, con el fin de observar si existe a priori alguna asociación interesante y, en consecuencia, hacer una primera estimación de qué características son más relevantes o se deben tener en cuenta de cara a determinar el valor de label. Para ello, disponemos de la matriz de correlaciones.

La matriz de correlaciones determina para cada par de características del dataset cuál es la linealidad que existe entre estas, mediante valores en el intervalo [-1, 1] que indican correlación negativa (decrecimiento) y positiva (crecimiento) respectivamente. De esta forma, a partir de una de las variables se puede deducir la tendencia de la otra, sin tener en cuenta su dependencia.

Dicho esto, en la figura 4.5 mostramos cuáles son las características más correlacionadas con la variable objetivo.

Si bien no hay un consenso en la forma de considerar los coeficientes de correlación, comúnmente se establecen los siguientes grupos:

- **Muy baja (0.00 - 0.19)**
- **Baja (0.20 - 0.39)**
- **Moderada (0.40 - 0.59)**
- **Alta (0.60 - 0.79)**
- **Muy alta (0.80 - 1.00)**

Por tanto, en nuestro caso se pueden ver correlaciones principalmente débiles y moderadas, lo cual en un principio puede no resultar algo conveniente de cara a intentar predecir el valor objetivo, aunque como veremos en siguientes fases la elección adecuada de los modelos puede ser un factor influyente en la compleción de los resultados.

Aquí hemos presentado correlaciones entre pares de variables, pero también habría que considerar las correlaciones entre grupos de más variables o múltiples; y si bien se han tenido en cuenta, no se han llevado a cabo al ser un proceso que consumiría mucho tiempo. Esto también aplica a aquellas correlaciones que no siguen una tendencia lineal.

```
this_comment_doc2vec_27        label   -0.464529
this_comment_doc2vec_68        label   -0.461439
this_comment_doc2vec_79        label   -0.457418
this_comment_doc2vec_19        label   -0.450446
this_comment_doc2vec_5         label   -0.435968
this_comment_doc2vec_28        label   -0.429172
this_comment_doc2vec_87        label   -0.427811
this_comment_doc2vec_85        label   -0.408196
this_comment_doc2vec_29        label   -0.401168
this_comment_doc2vec_78        label   -0.398859
this_comment_doc2vec_93        label   -0.392757
this_comment_doc2vec_39        label   -0.372089
this_comment_doc2vec_15        label   -0.366111
this_comment_doc2vec_88        label   -0.363009
this_comment_doc2vec_45        label   -0.331782
this_comment_doc2vec_54        label    0.303816
this_comment_doc2vec_99        label    0.306368
session_pct_negative_comments  label    0.306401
this_comment_doc2vec_94        label    0.322767
session_comments_lda_topic_4   label    0.335473
this_comment_doc2vec_35        label    0.355055
this_comment_doc2vec_56        label    0.373836
this_comment_doc2vec_64        label    0.382260
this_comment_doc2vec_37        label    0.390179
this_comment_doc2vec_50        label    0.398026
this_comment_doc2vec_36        label    0.407900
this_comment_doc2vec_8         label    0.433556
this_comment_doc2vec_26        label    0.459616
this_comment_doc2vec_47        label    0.465129
label                          label    1.000000
dtype: float64
```

Figura 4.5: Correlaciones más altas respecto a label.

4.2.2 *Feature importance*

Otra forma de comprobar la relevancia de las características es mediante el uso de modelos como Random Forest, que nos permiten conocer la importancia de cada una mediante el atributo *feature_importances_*.

Para ello hemos utilizado un método denominado Importancia de Gini, el cual se encarga de medir la impureza de las características, es decir, la proporción entre el número de ramificaciones en las que se utiliza cada una como referencia con respecto al número de muestras que dividen [30]. Por tanto, cuanto más pura sea, mayor relevancia tendrá. Pese a ello, existen otros métodos igual de válidos como MDA (*Mean Decrease in Accuracy*), basado en el uso de permutaciones.

Tras aplicar este modelo a nuestro dataset, obtenemos los resultados[1] indicados en la figura 4.6:

session_comments_lda_topic_1	0.004513
session_comments_doc2vec_45	0.004529
session_comments_doc2vec_96	0.004662
session_comments_doc2vec_73	0.004883
session_comments_tfidf_100	0.004939
session_comments_doc2vec_37	0.004967
session_comments_doc2vec_4	0.004969
session_comments_doc2vec_12	0.005110
session_comments_doc2vec_48	0.005126
session_comments_doc2vec_60	0.005158
session_comments_doc2vec_23	0.005163
session_comments_doc2vec_53	0.005546
session_comments_doc2vec_22	0.005660
session_pct_very_negative_comments	0.005765
session_comments_tfidf_73	0.005828
session_comments_doc2vec_88	0.006120
likes	0.006279
session_comments_lda_topic_2	0.006408
session_comments_doc2vec_47	0.006528
session_comments_lda_topic_4	0.006577
session_comments_doc2vec_100	0.006680

(a) Valores más altos de feature importance (1).

session_comments_doc2vec_15	0.006738
session_comments_doc2vec_82	0.006784
session_comments_doc2vec_33	0.006934
session_comments_doc2vec_65	0.006949
session_comments_doc2vec_6	0.007193
this_comment_doc2vec_85	0.007450
session_comments_doc2vec_92	0.007565
this_comment_doc2vec_69	0.007646
this_comment_doc2vec_93	0.008205
session_comments_tfidf_31	0.008631
this_comment_doc2vec_54	0.009530
session_comments_pct_profane_words	0.009587
session_comments_doc2vec_54	0.009975
session_comments_tfidf_6	0.009992
session_comments_tfidf_3	0.010817
session_pct_negative_comments	0.011701
session_comments_tfidf_13	0.011735
this_comment_doc2vec_26	0.011820
this_comment_doc2vec_47	0.013501
this_comment_doc2vec_8	0.013757
session_pct_rude_comments	0.015880
this_comment_doc2vec_79	0.016355
this_comment_doc2vec_68	0.016365
this_comment_doc2vec_87	0.018282
this_comment_doc2vec_78	0.018480
this_comment_doc2vec_29	0.018771
this_comment_doc2vec_27	0.021778
session_id	0.022417
this_comment_doc2vec_5	0.027350
this_comment_doc2vec_19	0.046990

(b) Valores más altos de feature importance (2).

Figura 4.6: Feature importance.

A simple vista se pueden ver características comunes con respecto a las de la figura 4.5, lo cual aporta una validación más precisa en la relevancia de las mismas. Sin embargo, este proceso nos será más útil en la fase de selección de características (ver sección 5.5), que explicaremos más adelante.

4.3 Verificación de la calidad

En este paso debemos verificar si alguna de las muestras del dataset contiene algún elemento anómalo, ya sea por no corresponderse con el rango habitual en el dominio de un determinado atributo (atípicos), por haberse medido de forma incorrecta o bien por presentar valores nulos o inexistentes. En este apartado nos centraremos en los dos últimos grupos.

Dado que partimos de un dataset que ya ha sido procesado con anterioridad, en un principio no debería haber ningún dato erróneo, pero con independencia de esta situación debemos

[1] Lo ideal sería presentar una visualización gráfica, pero dado el gran número de características esta tarea se hace bastante compleja, por lo que de aquí en adelante se aportarán valores numéricos para suplir este tipo de situaciones

revisar las muestras por si se hubiera pasado algo por alto.

Con respecto a los nulos, una manera simple de ver esto es utilizar alguna de las funciones que ofrece *Pandas* para filtrar aquellos valores que se consideren indefinidos o NaN y posteriormente contabilizarlos. Tras hacer este proceso, llegamos a la conclusión de que no había ningún valor nulo, lo cual implica que no tenemos que eliminar ninguna muestra o incluir cantidades arbitrarias.

En relación a los datos que hayan sido mal medidos, podríamos considerar principalmente la presencia de valores negativos. No obstante, en aquellas características donde esta situación ocurre contienen valores esperados. Por ejemplo, aquellas referentes a la polaridad de los comentarios se miden en el rango [-1, 1], de tal forma que los valores más bajos indican una mayor negatividad; otro caso es el de los rasgos que tienen que ver con *doc2vec*, los cuales señalan el grado de proximidad entre palabras clave, por lo que las distancias negativas indican que se sitúan a su izquierda.

Capítulo 5

Data Preparation

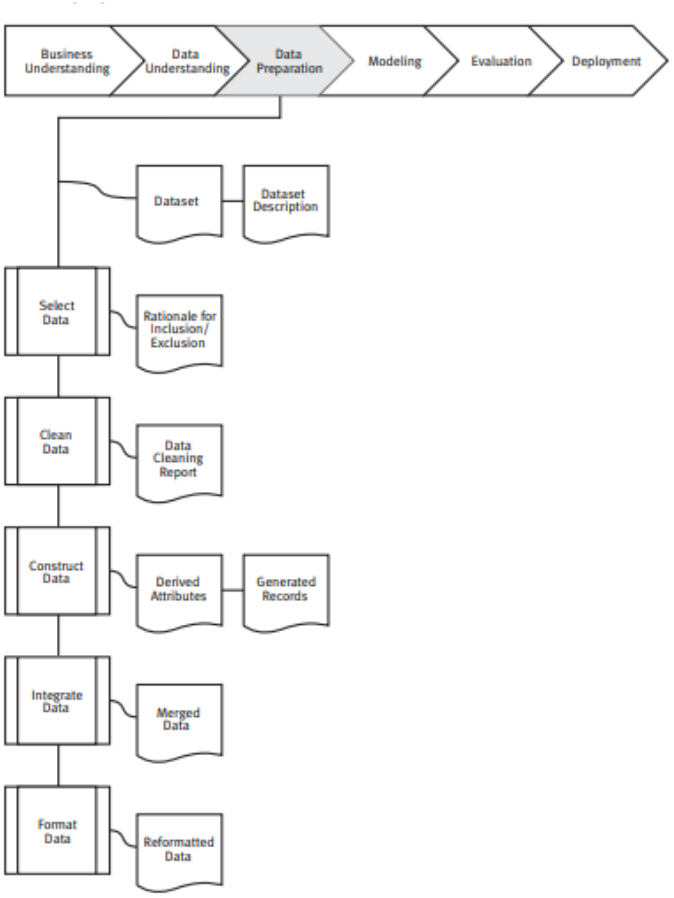

Figura 5.1: Tareas de la fase *Data Preparation*.

Esta fase comprende el proceso de mayor duración dentro de un proyecto de *Data Science*, ya que se requiere revisar que todos los datos presenten unos estándares mínimos de

cara a seleccionar las mejores características para la parte de modelado.

Así pues, tal y como se describe en la figura 5.1, en este capítulo trataremos la selección, limpieza y formateo de los datos, analizando posibles incongruencias y reflexionando acerca de diferentes métodos de selección de características.

5.1 Selección de datos

En la sección anterior ya hemos contemplado la posibilidad de eliminar ciertos datos que presentasen una medición incorrecta. No obstante, en esta fase se debe escoger tanto aquellas muestras/filas como atributos/columnas a utilizar en el posterior análisis de los modelos, lo cual dependerá de su grado de influencia en el éxito de los objetivos de negocio y de minería de datos.

En este apartado cubriremos los dos métodos principales de selección de muestras, y en la sección 5.5 hablaremos de las técnicas utilizadas para reducir el número de atributos.

5.1.1 Sobremuestreo y submuestreo

En aquellos casos en los que la proporción de las diferentes clases de un dataset resulte descompensada, puede ser interesante alterar de forma artificial estos porcentajes para disponer de un esquema más justo a la hora de comparar las distribuciones de varios parámetros. Es aquí donde entran el sobremuestreo y submuestreo.

El sobremuestreo u *oversampling* busca duplicar ciertas muestras del conjunto de entrenamiento de la clase menos representada para generar nuevos conjuntos de entrenamiento más equilibrados [31]. Sin embargo, esto puede generar sobreajuste y aumentar considerablemente los tiempos de aprendizaje, especialmente si el dataset presenta un gran tamaño [32]. El submuestreo o *undersampling*, en cambio, busca el objetivo contrario, es decir eliminar determinadas muestras de entrenamiento de la clase mayoritaria. Pese a ello, existe el riesgo de que durante el proceso prescindamos de información esencial para los modelos de clasificación.

En ambos casos es habitual seguir una aproximación aleatoria, ya que no tiene en cuenta ningún tipo de heurística y presenta una implementación sencilla y poco costosa. Pese a ello, para nuestro caso particular no ha sido necesario aplicar este procedimiento, ya que consideramos las proporciones indicadas en el Capítulo 4 como aceptables.

5.2 Limpieza de datos

Además de revisar datos erróneos o nulos, también es útil comprobar la existencia de datos anómalos, ya que si estos se encuentran en gran cantidad podrían condicionar los resultados de los modelos dando predicciones menos precisas.

En esta sección describiremos diferentes métodos que se pueden emplear para la detección de estos datos.

5.2.1 Detección de anomalías

Una anomalía o *outlier* es un tipo de muestra cuyos valores presentan una gran desviación con respecto a la mediana de un cierto atributo. Estos pueden darse a causa de la propia distribución de los datos o bien por errores humanos.

Como ya se comentó previamente, la presencia de estas observaciones puede llegar a alterar las salidas esperadas por los modelos de *Machine Learning*, influyendo de forma negativa en su rendimiento. Por eso, debemos asegurarnos de que estos no se manifiesten con gran frecuencia, a fin de preservar la calidad final de los resultados.

Esto no quiere decir que no deba existir ningún dato atípico, ya que los modelos, especialmente los de clasificación, requieren de una buena distribución de datos normales y anómalos, es decir, que conozcan un espectro de información lo más amplio posible [33], con el objetivo de permitir una generalización de las predicciones en vez de un sobreajuste.

Existen varios métodos o técnicas para identificarlos, como son:

- **Calcular el Rango Interquartílico (IQR).**
- **Emplear modelos de ML como Isolation Forest.**
- **Recurrir a variables estadísticas como el coeficiente de simetría y Z-score**. Esta última representa el número de desviaciones estándar a las que se encuentra un punto de información respecto a la media de una distribución.
- **Utilizar redes neuronales.**

En nuestro caso nos hemos centrado en los dos primeros.

IQR

IQR o Rango Intercuartílico es un método simple de detección de valores atípicos, resultado de calcular la diferencia entre el primer y tercer cuartil, es decir, los percentiles 25 y 75, respectivamente.

Posteriormente, se establece lo que denominaremos **rango de decisión**, de tal forma que cualquier muestra que se sitúe fuera del mismo se considerará atípica. Este rango presenta como límite inferior el punto del *primer cuartil - 1.5 * IQR*, mientras que el límite superior queda designado por el punto del *tercer cuartil + 1.5 * IQR*.

El motivo de que se escojan estos valores y no otros es que posibilitan aproximar de forma casi perfecta las propiedades intrínsecas de la distribución Gaussiana, en donde casi la totalidad de los datos se encuentran como muy lejos a 3 desviaciones típicas del punto medio.

Al aplicar este método en nuestro dataset, obtuvimos que de media entre el 5 y 10% de los valores eran atípicos. Como nos pareció una cifra bastante alta, decidimos aplicar también Isolation Forest para contrastar los resultados.

Isolation Forest

Es un algoritmo no supervisado inspirado en otros modelos que utilizan la combinación de árboles de decisión como principal recurso (véase el caso de Random Forest). Su funcionamiento principal radica en la creación de varios árboles de aislamiento[1] aleatorizados, con el fin de particionar las muestras de forma iterativa hasta ”aislar” aquellas que son anómalas. Estas últimas son fácilmente identificables gracias a que, al contener valores bastante desviados respecto a la media, la longitud de los caminos derivados de las mismas suele ser menor que la de aquellos derivados de otros datos normales [34].

De esta forma, el procedimiento seguido se puede considerar de la siguiente forma:

- Dado un conjunto de muestras X, se divide de forma recursiva seleccionando de forma aleatoria un atributo q y un valor de división p.
- Si se alcanza el límite de altura, se obtiene que sólo hay un elemento por nodo (es decir, cada nodo del árbol es terminal) o bien todos los datos de X presentan el mismo valor, se dará la ejecución por finalizada.

[1] Un árbol de aislamiento es en el fondo un árbol binario que solo puede presentar cero o dos hijos

Por otra parte, estamos utilizando como referencia la implementación de *scikit-learn* para este algoritmo. Entre sus parámetros de inicialización, nos encontramos con:

- ***contamination***: Un valor comprendido entre 0 y 0.5 que determina la proporción de datos atípicos que esperamos que haya en el conjunto de muestras.
- ***n_jobs***: Indica cuántos núcleos del procesador utilizar en la ejecución del modelo, con la finalidad de hacer más eficiente el proceso mediante su paralelización. En este caso al no requerir de mucho tiempo lo hemos dejado por defecto, pero veremos que en otros algoritmos es un parámetro bastante útil.
- ***random_state***: Actúa a modo de "semilla", con el objetivo de pseudoaleatorizar la división de los datos que sirven de entrenamiento.

Por otra parte, hay dos atributos de salida a tener en cuenta: la *función de decisión*, que nos dice la puntuación de anomalía de cada muestra, de modo que las inferiores a 0 son consideradas atípicas; y la variable *score_samples*, que presenta el efecto contrario.

En un primer momento, basándonos en los resultados de IQR, probamos a introducir una *contamination* del 10%. De este modo, partiendo de las *score_samples*, obtuvimos que aproximadamente unas 4500 muestras eran anómalas. Sin embargo, este número se mantenía constante aún incluso probando con otros valores de dicho parámetro.

Es por ello que decidimos recurrir a las *funciones de decisión*, y aplicando una *contamination* del 5%, obtuvimos cerca de 1500 datos atípicos, lo cual resulta un valor asumible de cara a las siguientes fases. En la figura 5.2 se puede observar cómo la línea discontinua determina el umbral a partir del cual se considera un dato como anómalo.

5.2.2 Una distribución interesante: *session_id*

Durante la fase de análisis de los diferentes datos se detectó un comportamiento interesante en lo que respecta al atributo de *session_id*. Y es que para cada uno de los *ids* existentes nos encontramos dos aspectos:

- Siempre presenta los mismos valores en la variable objetivo (*label*), ya sea todo ceros o todo unos, lo cual verifica que existe una relación de los comentarios que pertenecen a un mismo *post* y que por consiguiente el dataset se encuentra "etiquetado" a nivel de sesión.
- Cuanto menor sea el valor del *id*, mayor será la probabilidad de que contenga comentarios clasificados como *cyberbullying*. Esto último podría resultar en un sesgo a la hora

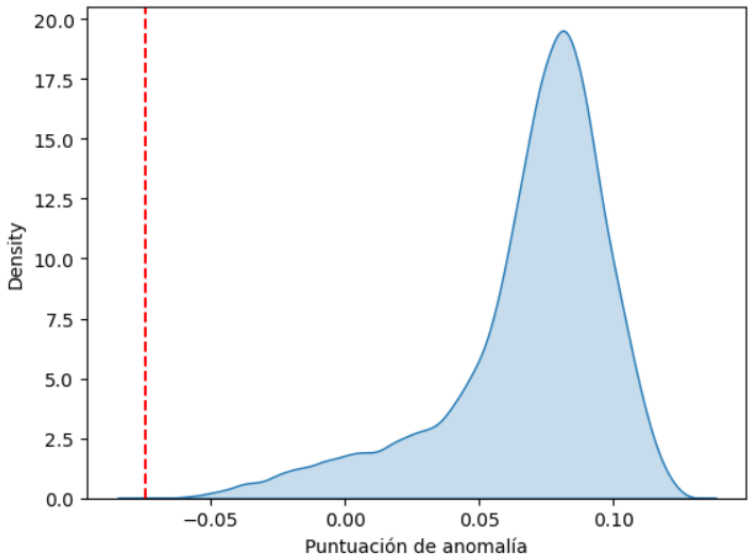

Figura 5.2: Distribución de las anomalías.

de considerar cómo de relevante es dicha característica y en última instancia condicionar las salidas de los modelos, pudiendo realizar predicciones incorrectas a partir de una secuenciación ficticia de las muestras.

Ante esto, en primer lugar decidimos estudiar la importancia de dicho atributo, y tal y como se muestra en la figura 4.6 se puede apreciar que este se encuentra entre las características más relevantes. Teniendo esto en mente, se decidió comprobar si esta distribución provoca un valor alto en la correlación entre *session_id* y *label*. Sin embargo, esta situación no llega a ocurrir, ya que es de poco más de 0.2, lo cual nos dice que es una correlación débil.

Por este motivo, y dado que los valores que adoptan los ids no siguen un patrón concreto, consideramos reemplazarlos tratando de establecer una secuencia (p.ej: los correspondientes al primer id que aparece en el dataframe asignarles un id de 1, al segundo un 2 y así sucesivamente), y analizar posteriormente su histograma y función de densidad en relación a label. Con ello obtuvimos la figura 5.3[2].

[2] Este análisis también se ha realizado teniendo en cuenta una sola muestra de cada session, y se obtuvieron las mismas conclusiones.

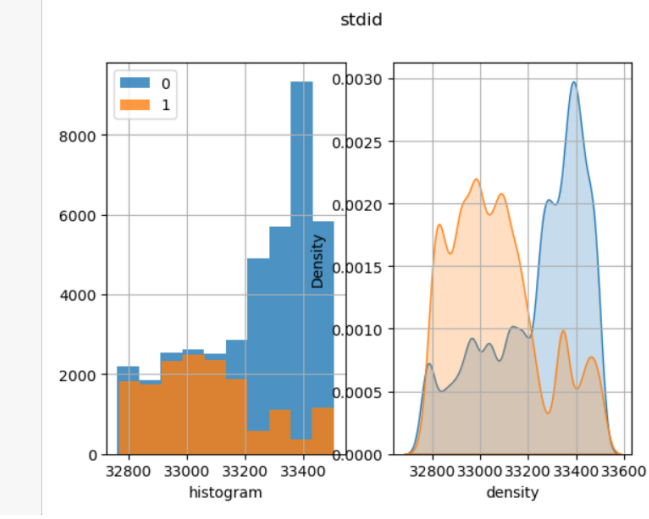

Figura 5.3: Distribución tras regenerar *session_id*

De esta forma, concluimos que el *session_id* tiene una especial influencia en la variable objetivo, por lo que hemos decidido regenerar el atributo usando valores pseudoaleatorios, a fin de evitar el escenario previamente descrito. De esta forma, la distribución de los *session_id* respecto a *label* quedaría tal y como se muestra en la figura 5.4.

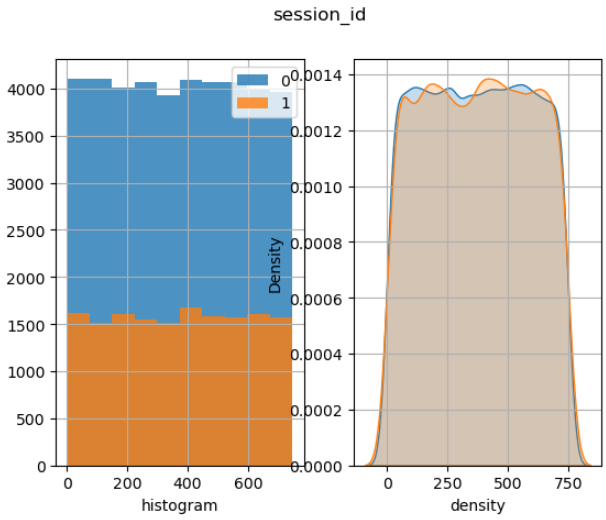

Figura 5.4: Distribución con *session_id* aleatorizado

5.3 Construcción de datos

Esta tarea consiste en analizar los diferentes atributos de cada muestra desde una perspectiva funcional, con la finalidad de poder llegar a derivar nuevas características que aporten un

mayor significado a las ya existentes, o bien permitan una mejor comprensión de las mismas.

En nuestro caso este paso se ha contemplado, pero no se ha llevado a cabo, debido sobre todo a la gran cantidad de características presentes en nuestro dataset y a que previo a este trabajo ya se había llevado a cabo un procesado que incluía atributos que se consideraban relevantes, como pueden ser *Tf-idf*, *doc2vec* o diferencias temporales en instantes de publicación entre cada par de comentarios dentro de un mismo *post*.

5.4 Formateo de datos

En esta etapa se pretende dar un formato adecuado a aquellos datos de connotación categórica o que presenten un tipo que los modelos de *Machine Learning* no sean capaces de comprender. Existen diferentes aproximaciones a la hora de encarar el codificado de estas variables, entre las que destacamos:

- ***One-hot Encoding***: Consiste en generar nuevas características a partir de nuestra variable de interés, tantas como valores distintos contenga dicho atributo. Sin embargo, para evitar problemas de multicolinealidad, es habitual generar n-1 características, ya que estas al contener valores binarios nos permiten deducir fácilmente si una cierta muestra pertenece o no a la variable restante.
- ***Label Encoding***: Para cada valor o etiqueta que se incluya dentro del conjunto de valores del atributo en cuestión, asignarle un valor numérico en base al orden alfabético. El problema con esta técnica es que puede dar lugar a dependencias inexistentes en un primer momento debido a esta ordenación artificial, por lo que sólo se debería utilizar si la propia variable objetivo contiene valores ordinales.

En relación con nuestro conjunto de datos, dado que todos los atributos presentes tienen un tipo entero o flotante (aceptado por la mayor parte de modelos de *Machine Learning*), no ha sido necesario aplicar ninguno de los métodos descritos.

5.4.1 Normalización y escalado

Uno de los principales problemas de ciertos modelos de *Machine Learning* es que son especialmente sensibles a los rangos de valores o magnitudes que pueda tener cada variable, ya que en base a los criterios que se utilicen de cara a realizar las predicciones podrían llegar a considerar más relevantes atributos que no tienen por qué ser determinantes en la resolución de las salidas, alterando en consecuencia su rendimiento.

Por tanto, debemos asegurarnos de que nuestro conjunto de datos se encuentre escalado, es decir, que cada atributo presente rangos similares de cara a evitar estos inconvenientes. Dos métodos muy utilizados son:

- **Normalización o *Min-Max Scaling***: Consiste en transformar el conjunto de valores de cada atributo en otros comprendidos entre 0 y 1. Para ello, a cada valor se le resta el mínimo del conjunto, y el resultado se divide por la diferencia entre el máximo y mínimo.
- **Estandarización**: Se trata de tipificar la distribución (centrar la media en 0 y ajustar la desviación típica en 1) que sigue el conjunto de valores, restándole a cada uno el valor de la media.

En nuestro caso hemos utilizado la estandarización (en concreto la implementación que nos proporciona *scikit-learn* denominada *Standard Scaler*), la cual nos servirá de bastante ayuda cuando apliquemos los métodos de reducción de dimensionalidad. No obstante, cabe señalar que cualquiera de las operaciones de escalado se debe realizar únicamente sobre el conjunto de entrenamiento, ya que en caso contrario estaríamos filtrando información del conjunto de test y, por tanto, alterando el resultado final.

5.5 Reducción de dimensionalidad

Si bien es cierto que disponer de un gran número de características permite realizar predicciones más precisas sobre la variable objetivo, también implica que los tiempos de ejecución aumenten de forma considerable, así como la posibilidad de provocar un cierto sobreajuste.

Así pues, en esta tarea llevaremos a cabo diferentes técnicas para tratar de minimizar el número de características, y escogeremos aquella que nos permita alcanzar un compromiso adecuado entre la cantidad de atributos a eliminar y los recursos computacionales necesarios.

5.5.1 Mediante Random Forest

Este método de selección se basa en el uso de cualquier algoritmo de aprendizaje supervisado que permita definir la relevancia de cada característica. Es por eso que se suelen utilizar modelos basados en árboles de decisión, ya que incorporan el atributo de *feature_importances_*, explicado previamente en la sección 4.2.2.

De esta forma, partiendo de la matriz de correlaciones con todas las características, el procedimiento a seguir es el siguiente:

- Primero, debemos filtrar la matriz de correlaciones para obtener aquellas que consideremos que puedan presentar información redundante. En nuestro caso, hemos utilizado un umbral de |0.95|
- Posteriormente, calculamos mediante el modelo de Random Forest las importancias de cada característica, y escogemos las correlaciones más fuertes [3].
- Se eliminan aquellas características que presenten un menor valor de importancia.
- Repetir el proceso hasta que ya que no quede ninguna correlación dentro del umbral designado.

A lo largo de este proceso, es importante tener en cuenta el recalcular las importancias de las características restantes, ya que al ir eliminando diferentes atributos su relevancia puede haberse visto afectada.

Una vez aplicado al dataset *Baseline*, se han obtenido alrededor de 100 características a eliminar, siendo la mayoría relacionadas con *this_comment_doc2vec_**, y alguna respecto a *Tf-idf*. La lista concreta de características descartadas por este método se puede ver en el código referenciado en la sección 1.4.

5.5.2 RFECV

RFECV o *Recursive Feature Elimination with Cross-Validation* es un método que tiene como objetivo averiguar el subconjunto óptimo de características a escoger, aplicando durante el proceso una validación cruzada a diferencia de lo que sucede en la implementación original de RFE. Para ello, debe utilizar como base cualquier modelo que devuelva un atributo coeficiente o de importancia de características. En un primer momento nos habíamos decantado por los árboles de decisión simples (*Decission Tree*), dado que RFECV de por sí es un método con unos tiempos de ejecución elevados y el sacrificio de rendimiento que realizamos con respecto a usar Random Forest es asumible. Sin embargo, ante la posibilidad de sobreajuste y que los resultados que se obtenían eran inconsistentes, decidimos utilizar en su lugar un algoritmo de regresión logística (*LogisticRegression* en *scikit-learn*).

En cuanto a su implementación, debemos de tener en cuenta los siguientes parámetros:

- ***estimator***: Representa el modelo de aprendizaje supervisado que se utilizará para obtener las importancias. Como ya hemos comentado, vamos a utilizar *LogisticRegression*.

[3] Idealmente se debería ir escogiendo una pareja de cada vez, pero a fin de agilizar el proceso de selección, y al no tener un especial impacto en los resultados, se decidió escoger un cierto número de parejas a la vez

- ***step***: Delimita el número de características que se descartan en cada iteración, aunque si indicamos un número flotante estaremos designando el porcentaje de atributos a eliminar. En nuestro caso asignaremos un valor de 1.
- ***cv***: Determina la estrategia de validación cruzada a emplear. Se puede especificar:
  - Un valor entero, para indicar el número de *folds* o divisiones del conjunto de entrenamiento.
  - Un iterador, como es el caso de *CV splitter*.

  Para este parámetro hemos decidido mantener el valor por defecto, que son 5 *folds*. Asimismo, en el manual de *scikit-learn* se especifica que al indicar valores enteros, si la variable objetivo es de tipo binaria o multiclase, se utilizará *StratifiedKFold*. Esta variante consiste en dividir los datos en porciones de entrenamiento y test de tal forma que conservemos la proporción original de los valores de la variable objetivo, es decir, si tenemos un 80% de una clase y 20% de otra, las divisiones se establecerán de tal forma que en el conjunto de entrenamiento y de test haya esos mismos porcentajes de cada tipo. Esto se hace para evitar que por mala suerte tengamos subconjuntos en donde estas distribuciones aparezcan modificadas de forma significativa, lo cual afectaría a la precisión final.
- ***n_jobs***: Como vimos en secciones anteriores, este parámetro nos permite acelerar los tiempos de ejecución utilizando un mayor número de procesadores en la CPU (o todos si indicamos -1). Como este algoritmo consume bastante tiempo, nos resulta de gran utilidad.

Cabe resaltar que también hemos utilizado un *random_state* para asegurarnos la posibilidad de reproducir los mismos resultados en cada nueva ejecución. Una vez aplicado, a partir del array de *rankings* lo contrastamos con respecto a la *accuracy* media, indicando a su vez un margen de error para cada valor, obteniendo así la figura 5.5:

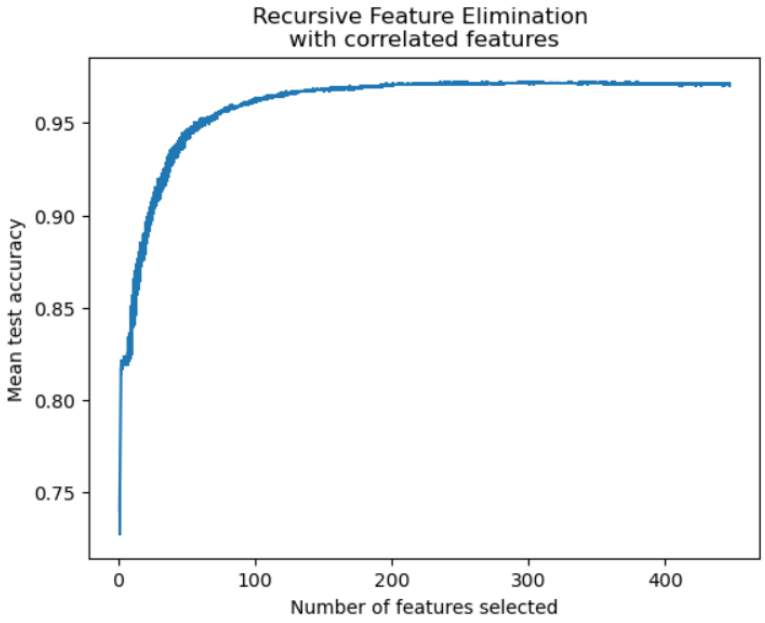

Figura 5.5: Gráfica de RFECV para *Baseline*

De esta forma, se puede ver que la precisión presenta una tendencia ascendente a medida que se van eliminando atributos, hasta que nos vamos acercando a las 50 características, momento en el cual esta precisión empieza a descender gradualmente. En consecuencia, si consideramos escoger este modelo de cara a la selección de los atributos finales, lo ideal es obtener el menor número posible de ellas sin perder demasiada precisión, por lo que seleccionaríamos las 50 mejores características en relación a la variable de *ranking*.

Por otro lado, este algoritmo también se ha aplicado a los datasets *WithoutLDA*, *WithoutDoc2vec* y *WithoutBoth*, cuyo resultado se muestra en las figuras 5.6, 5.7 y 5.8. Esto también se ha llevado a cabo para el caso de PCA.

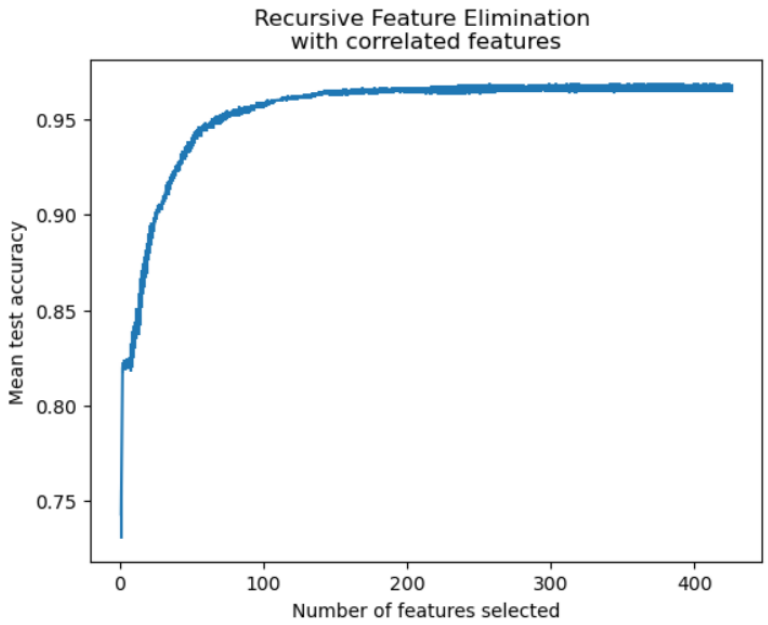

Figura 5.6: Gráfica de RFECV para *WithoutLDA*

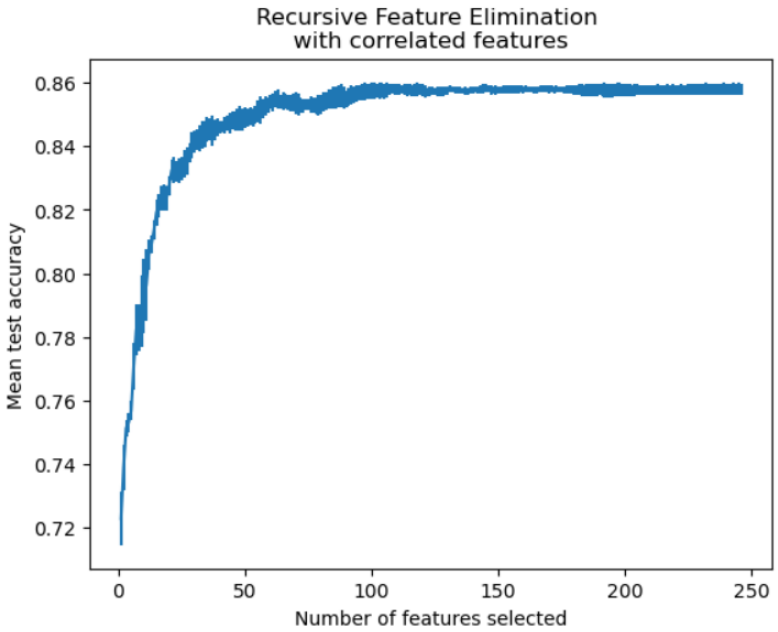

Figura 5.7: Gráfica de RFECV para *WithoutDoc2vec*

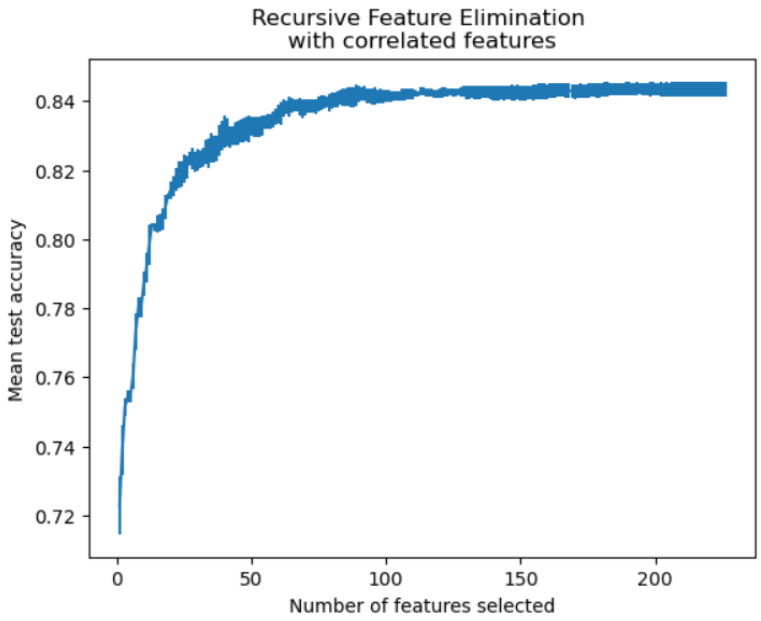

Figura 5.8: Gráfica de RFECV para *WithoutBoth*

Como vemos, en la gráfica de *WithoutLDA* se presenta una tendencia similar a la de *Baseline*, pero es en *WithoutDoc2vec* y *WithoutBoth* donde se encuentran las principales diferencias, pues para sendos escenarios el descenso se produce de forma más irregular y el punto óptimo se encuentra ahora entre las 50 y 100 características. Asimismo, sus valores medios de precisión oscilan en rangos considerablemente inferiores a los de *Baseline* y *WithoutLDA*, lo cual nos aporta un ligero atisbo de la influencia que puede tener cada dataset de cara a la fase de modelado.

5.5.3 PCA

PCA o *Principal Component Analysis* es una estrategia de tipo estadístico que consiste en extrapolar los atributos esenciales de un conjunto de datos como una serie de variables o componentes que no presentan correlación lineal entre las mismas. Para ello, se calcula la importancia de cada característica a partir de la proporción de varianza que explica cada una, pero sin tener en cuenta los atributos con un mejor coeficiente.

De esta forma, aquel con el valor más alto de variabilidad explicada se asocia con el primer componente, el siguiente con el segundo componente, y así sucesivamente. Con ello podemos obtener una matriz de correlaciones, en la que se incluye el grado de asociación de cada componente con cada una de las características de nuestro dataset, de modo que aquellas que sean las más altas de cada componente representarán los atributos a seleccionar.

En la implementación de *scikit-learn* de este modelo debemos tener en cuenta el parámetro de *n_components*, el cual puede contener valores enteros para indicar el número de componentes a utilizar en la representación de las características, o bien un número entre 0 y 1 para indicar el número de atributos a conservar para que la varianza explicada total sea como mínimo dicho valor. En nuestro caso, consideramos que un valor aceptable es del 95%, por lo que utilizaremos un *n_components* de 0.95.

Una vez aplicado este proceso, el modelo conservó alrededor de 150 características tanto para el caso de *Baseline* como de *WithoutLDA*, y cerca de las 120 tanto para el dataset de *WithoutDoc2vec* como *WithoutBoth*, por lo que se puede deducir que a mayor número de atributos que tenga inicialmente un dataset, y dependiendo de la importancia de cada característica, mayor será la proporción de rasgos descartados.

5.5.4 Selección final de características

En base a los resultados obtenidos, podemos ver que la selección por Random Forest es el método que menos características elimina, además de resultar más tedioso al tener que recalcular en cada iteración las correlaciones del dataset y las importancias de cada atributo. Es por esto que se decidió no aplicar este método a ninguno de los datasets derivados. Asimismo, RFECV representa el polo opuesto al ser el que más características elimina, pero el hecho de prescindir de una gran cantidad de atributos podría desembocar en un peor rendimiento, sobre todo si aquellos que se han descartado pueden contener alguna información valiosa.

Por ello, hemos decidido quedarnos con las características que proporcionaba PCA, ya que pese a conservar un mayor número que con RFECV presenta un comportamiento más estable para nuestro conjunto de datos, además de que con este descarte más laxo podríamos beneficiarnos de un valor de *accuracy* ligeramente mejor para los modelos que escojamos.

Capítulo 6

Modeling

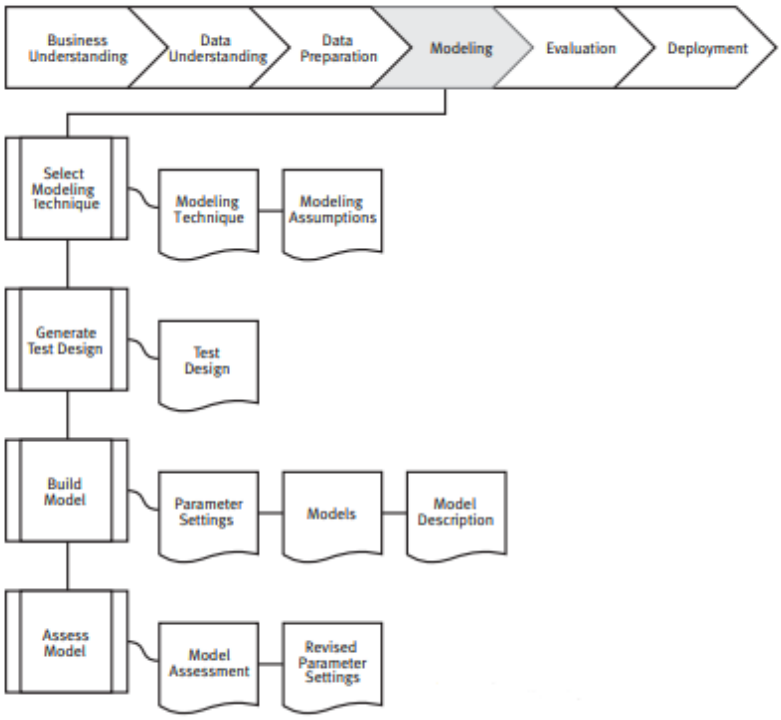

Figura 6.1: Tareas de la fase Modeling.

Partiendo del conjunto de características seleccionado en la fase anterior, debemos escoger los modelos sobre los que se va a aplicar el conjunto de entrenamiento para posteriormente evaluar su precisión mediante una serie de predicciones frente a otro grupo de datos.

Así pues, siguiendo la estructura de la figura 6.1, en esta etapa describiremos los criterios

de selección de los modelos y cómo se han diseñado las pruebas para su entrenamiento, así como el proceso de construcción de cada uno en base a sus diferentes parámetros, para finalmente realizar un pronóstico de cuáles se pueden comportar mejor teniendo en cuenta el entrenamiento seguido.

6.1 Criterios de selección

A la hora de seleccionar el mejor modelo a aplicar en un proyecto de *Data Science*, debemos considerar los siguientes factores:

- El comportamiento que presentan los datos, pudiendo tener en cuenta aspectos como las correlaciones, la cantidad de muestras extraídas y de atributos que las definan, entre otros.
- El tiempo que tarda en llevarse a cabo el entrenamiento.
- Su grado de éxito atendiendo a ciertas métricas como *accuracy* o *recall*, siendo la primera en la que nos focalizaremos de cara a la evaluación posterior.

No obstante, no nos interesa utilizar un solo modelo, ya que uno de los objetivos de negocio que persigue este trabajo es el poder contrastar los resultados obtenidos con modelos de diferente índole, para así determinar qué tipo es capaz de llevar a cabo la tarea de clasificación de la forma más efectiva posible. Con esto en mente, hemos escogido los siguientes modelos:

- **Naïve-Bayes**: En concreto usaremos la variante gaussiana, cuya implementación en *scikit-learn* es *GaussianNB*.
- **Random Forest**: Modelo que utiliza varios árboles de decisión aleatorizados. Usaremos la implementación de *RandomForestClassifier*.
- **AdaBoost**: Modelo resultado de combinar otros modelos más débiles. En este caso, su implementación es *AdaBoostClassifier*.
- **SVC linear**: Variante lineal del modelo de SVC, que en *scikit-learn* se denomina *LinearSVC*.

6.2 Proceso de entrenamiento

En primer lugar, el dataset con las características seleccionadas se dividió en un conjunto de entrenamiento que denominaremos E y que será el que utilicemos para determinar los mejores parámetros de cada modelo, y un conjunto de test (T) sobre el que cada algoritmo

realizará predicciones, obteniendo así diferentes métricas con las que evaluar su rendimiento.

Posteriormente, se tomó el conjunto de entrenamiento mencionado previamente y se dividió en 5 *folds* estratificados usando el algoritmo *StratifiedKFold*. Tal y como explicamos en capítulos anteriores, este tipo de repartición permite mantener la proporción de cada clase de la variable objetivo, con el fin de evitar sobreajustes en el proceso de entrenamiento y emular los rasgos propios del conjunto de muestras. De esta forma, para cada iteración tendremos 4 subconjuntos de entrenamiento (que consideraremos conjuntos E_i, donde i representa el conjunto i-ésimo) y 1 fold de test (T'), que irán cambiando en cada iteración.

Por último, aplicaremos sobre dichos subconjuntos E_i un proceso de *Grid Search* con validación cruzada (en *scikit-learn* implementado como *GridSearchCV*), el cual consiste en lo siguiente:

- Cada conjunto de entrenamiento se divide a su vez en N subconjuntos de entrenamiento, que pasaremos a denominar como E_{ij} (dónde j es el subconjunto j-ésimo de cada conjunto E_i), y un conjunto de validación (V) con el que obtendremos una determinada puntuación de métrica. Por defecto nos devolverá la *accuracy* del modelo, aunque se pueden utilizar otras como el *recall* modificando el parámetro de *scoring*.
  Por otro lado, si la variable objetivo es de tipo binaria o multiclase, se usará por defecto un *5-fold* estratificado (al igual que el aplicado previamente), y en caso de ser una variable regresora se usará un *k-fold* simple, siendo k el número de particiones deseadas.
- Tras llevarse a cabo todas las iteraciones, el atributo *best_params_* nos devuelve los mejores valores para los parámetros que hemos indicado mediante la variable *param_grid* para cada modelo en cuestión.

Todo ello se repite con cada distribución de conjuntos E_i y T', y de todos los resultados obtenidos escogemos los que mejor puntuación tengan (evitando siempre el sobreajuste) o, en caso de no haber mucha diferencia, aquellos parámetros que presenten resultados más consistentes.

Una visualización más clara de este proceso se puede ver en la figura 6.2:

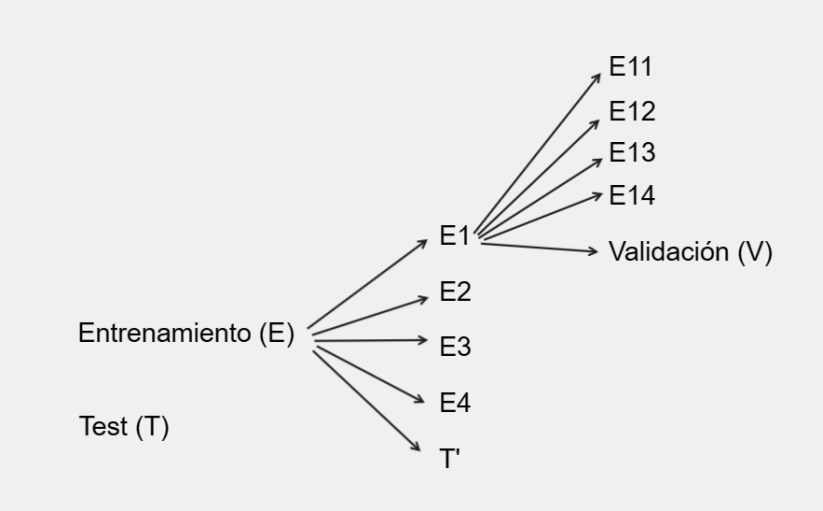

Figura 6.2: Proceso de *hiperparametrización*.

6.3 Hiperparametrización

La *hiperparametrización* es el proceso por el cual a partir de un conjunto de muestras se determinan los valores óptimos de determinados parámetros (denominados hiperparámetros) para un modelo en particular, usando técnicas como la mencionada en el apartado anterior. El escoger ciertas variables y/o valores de prueba es importante de cara a ver un correcto contraste entre las diferentes aproximaciones, disponiendo así de un panorama más claro sobre qué modelos a priori son más elegibles para la compleción de nuestros objetivos de negocio.

En esta sección desglosaremos los parámetros que se han tenido en cuenta para cada modelo, así como los distintos valores que se han tomado.

6.3.1 GaussianNB

Este modelo se basa en el análisis de la distribución gaussiana de las muestras, calculando la varianza asociada a cada atributo del dataset, asumiendo que cada característica es independiente del resto.

Su implementación en *scikit-learn* sólo contiene un parámetro relevante llamado *var_smoothing* que determina cómo de grande es el valor de la desviación típica, provocando que los elementos de la distribución queden más repartidos y permita un mejor ajuste, pero si se aumenta demasiado puede llegar a provocar un sobreentrenamiento.

Dado que el valor por defecto asociado a este atributo es de 1^{e-9}, hemos probado tanto con esta cifra como 1^{e-5}, 1^{e-6}, 1^{e-7} y 1^{e-8}.

6.3.2 Random Forest

Random Forest es un modelo que utiliza varios árboles de decisión que se entrenan con subconjuntos aleatorios de características, para finalmente obtener la media de sus resultados y evitar problemas de sobreajuste.

A diferencia del modelo anterior, contamos con una mayor variedad de parámetros a optimizar, pero nos centraremos en dos que aparecen con relativa frecuencia en la literatura referente a este ámbito:

- ***max_depth***: Determina la profundidad máxima de los árboles de decisión. Hemos utilizado los valores 3, 5 y 7.
- ***n_estimators***: Regula cuántos árboles de decisión se entrenan en el desarrollo de este modelo. Hemos seleccionado los valores 10, 100 y 500.

6.3.3 AdaBoost

AdaBoost es un algoritmo de clasificación supervisado que se basa en la combinación ponderada de las salidas de diferentes algoritmos de menor precisión o más débiles, los cuales se van reforzando con el paso de las iteraciones. Por otra parte, en la implementación de *scikit-learn* se observa que hay dos variantes posibles a utilizar: SAMME (*StageWise Additive Modeling*) y SAMME.R, una variante del anterior que incorpora lo que denominan *Real Boosting*. Ambas son implementaciones que permiten extender la aplicación de AdaBoost a modelos multiclase [35].

La diferencia entre las dos reside en la forma en la que realizan las predicciones de cada muestra; la primera es de connotación discreta, ya que sólo devuelve valores entre 0 y 1, mientras que la segunda designa probabilidades de pertenecer a cada clase. Nosotros dejaremos el valor por defecto, que es SAMME.R

Otros parámetros que resultan de nuestro interés son:

- ***learning_rate***: Indica la contribución de cada algoritmo en el resultado final. Existe una relación inversamente proporcional entre este atributo y el número de estimadores a utilizar. Hemos considerado los valores 0.1, 0.01 y 0.001.

- ***n_estimators***: De forma similar al modelo de Random Forest, este parámetro nos permite designar el número de modelos a incluir en la combinación de las salidas. Usaremos los valores 100 y 200.

6.3.4 LinearSVC

Se trata de un algoritmo de clasificación supervisado que representa una variante del modelo SVC con kernel lineal. Sin embargo, a diferencia de esta última, su implementación en *scikit-learn* utiliza otra librería (*liblinear*) que permite un escalado más eficiente conforme aumenta el número de observaciones.

En este caso, podemos destacar estos parámetros:

- ***C***: Es el parámetro de regularización del modelo; cuanto mayor sea, menos penalizaciones tendrá a la hora de realizar predicciones, si bien en última instancia dependerá de cómo se encuentren distribuidos los datos. Hemos considerado los valores 0.1, 1 y 10.
- ***tol***: Determina el valor de tolerancia a partir del cual el algoritmo finaliza su ejecución, es decir, su criterio de parada. Hemos utilizado los valores 1^{e-3} y 1^{e-4}.

6.4 Resultado del modelado

Tras aplicar el proceso de *Grid Search* con validación cruzada a cada iteración del *StratifiedKFold*, hemos obtenido para cada uno de los modelos y datasets las mejores combinaciones de parámetros y sus valores de precisión, cuyas métricas principales pueden observarse en las tablas 6.1, 6.2, 6.3 y 6.4:

6.4.1 GaussianNB

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	*var_smoothing = 1e-7*	0.8131	2.561
WithoutLDA	*var_smoothing = 1e-7*	0.8079	2.499
WithoutDoc2vec	*var_smoothing = 1e-7*	0.7726	1.970
WithoutBoth	*var_smoothing = 1e-7*	0.7765	2.053

Tabla 6.1: Tabla gridsearch (GaussianNB)

6.4.2 RandomForest

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	*n_estimators = 100, max_depth = 7*	0.9067	207.927
WithoutLDA	*n_estimators = 100, max_depth = 7*	0.9144	231.376
WithoutDoc2vec	*n_estimators = 100, max_depth = 7*	0.8739	143.291
WithoutBoth	*n_estimators = 100, max_depth = 7*	0.8910	125.986

Tabla 6.2: Tabla gridsearch (RandomForest)

6.4.3 AdaBoost

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	*learning_rate = 0.1, n_estimators = 200*	0.8852	242.377
WithoutLDA	*learning_rate = 0.1, n_estimators = 200*	0.8806	318.431
WithoutDoc2vec	*learning_rate = 0.1, n_estimators = 200*	0.8298	124.051
WithoutBoth	*learning_rate = 0.1, n_estimators = 200*	0.8315	124.936

Tabla 6.3: Tabla gridsearch (AdaBoost)

6.4.4 LinearSVC

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	*C = 1, tol = 1e-3*	0.8197	73.816
WithoutLDA	*C = 10, tol = 1e-3*	0.7786	73.328
WithoutDoc2vec	*C = 1, tol = 1e-3*	0.7645	53.182
WithoutBoth	*C = 0.1, tol = 1e-3*	0.7590	49.526

Tabla 6.4: Tabla gridsearch (LinearSVC)

Cabe señalar que a la hora de escoger las mejores combinaciones, se han tenido en cuenta los mejores resultados de cada iteración tanto en el propio *Grid Search* como en lo que respecta a los *folds* estratificados, así como la desviación típica de cada una, ya que aquellas que

tengan una menor variabilidad en sus resultados implica que son más estables ante cualquier posible distribución de los conjuntos de entrenamiento y validación.

Asimismo, en el caso de Random Forest no se escogió un *max_depth* de None, ya que únicamente se ha utilizado para contrastar su comportamiento con el resto de valores de dicho parámetro y utilizarlo como referencia para evitar un sobreajuste del modelo. En concreto, considerando esta profundidad, y en combinación con un valor de *n_estimators* igual a 500, se obtenía una *accuracy* de 0.9763 y un tiempo de 208.649 segundos en el caso de *Baseline*; 0.9805 y 229.340 segundos en *WithoutLDA*; 0.9707 y 143.291 segundos en *WithoutDoc2vec*; 0.9758 y 126.776 segundos en *WithoutBoth*.

6.5 Pronóstico para la fase de evaluación

Teniendo en cuenta los resultados descritos previamente, podemos realizar las siguientes suposiciones de cara a la fase de evaluación:

- Que el modelo de distribución gaussiana será el que menos tiempo consuma de entrenamiento, pero también de los que tendrá un valor de precisión ligeramente inferior.
- Que el modelo LinearSVC será el que menor precisión tendrá en relación al resto de modelos.
- Que AdaBoost será el que más tiempo consuma, pero tendrá un valor elevado de precisión.
- Que Random Forest tendrá un tiempo razonable y será el que mejor precisión presente.
- Que por lo general en los datasets *WithoutLDA*, *WithoutDoc2vec* y *WithoutBoth* los modelos presentan con respecto a *Baseline* un rendimiento más pobre, situación que se agrava en los dos últimos.
- Que los tiempos de ejecución de *WithoutDoc2vec* y *WithoutBoth* son considerablemente más bajos que en los otros datasets, aunque hay que tener en cuenta que el número de características que contienen es menor.

Dicho esto, pasamos a medir el rendimiento de cada modelo contra los conjuntos de evaluación.

Capítulo 7

Evaluation

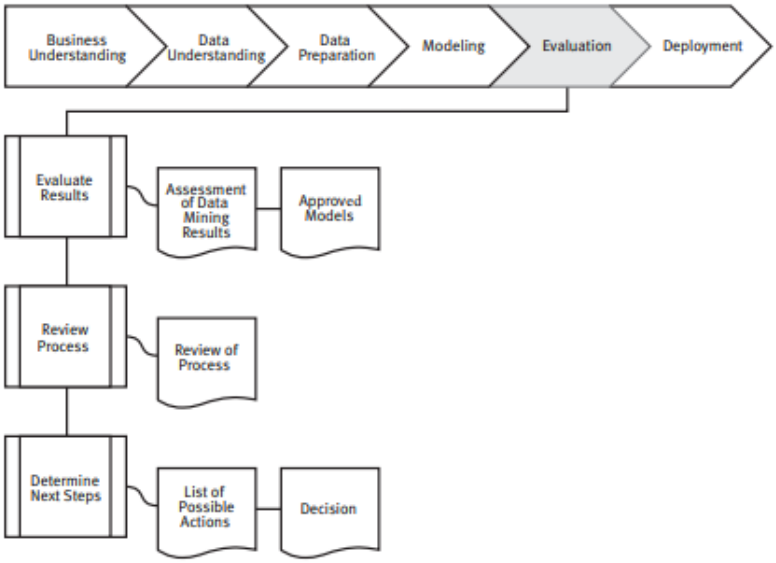

Figura 7.1: Tareas de la fase *Evaluation*.

En la fase de evaluación se entrenan los diferentes modelos considerados en base a los resultados del proceso de *hiperparametrización*, y se revisan tanto las salidas de las predicciones (supervisadas mediante diferentes métricas) como el procedimiento que se ha seguido para establecerlas (ver figura 7.1). Además, tras contrastar cada algoritmo se podrá designar una secuencia de acciones a realizar teniendo en cuenta las conclusiones obtenidas.

7.1 Evaluar resultados

Una vez realizado el proceso de *Grid Search* con validación cruzada, y habiendo considerado las mejores combinaciones siguiendo los criterios descritos en el capítulo anterior, se ha entrenado cada modelo utilizando dichos parámetros y se ha obtenido el tiempo de cada ejecución. Posteriormente, se han realizado predicciones a partir del conjunto de test original (T) y se ha obtenido la matriz de confusión de cada modelo.

7.1.1 Matriz de confusión

La matriz de confusión consiste en una representación tabular de dimensión 2x2 que permite visualizar la cantidad de predicciones con éxito y fallidas que ha realizado el modelo en cuestión. Estas se desglosan de la siguiente forma:

- En la esquina superior izquierda se encuentran los **verdaderos negativos (TN)**. En un modelo binario, se considera el 0 como valor negativo y 1 como positivo, de forma similar a la lógica booleana. De esta forma, en esta celda se indican aquellas predicciones que se asumieron negativas y realmente lo eran.
- En la esquina superior derecha se incluyen los **falsos positivos (FP)**, que engloban a aquellas predicciones que se asumieron positivas, pero que en realidad eran negativos.
- En la esquina inferior izquierda se incluyen los **falsos negativos (FN)**, que son aquellas predicciones consideradas negativas pero que en realidad eran positivos.
- En la esquina inferior derecha se reúnen los **verdaderos positivos (TP)**, que indican aquellas predicciones que se asumieron como positivas y realmente lo eran.

En la figura 7.2[1] se puede observar una representación visual de estos conceptos.

7.1.2 Métricas de supervisión

A partir de la matriz de confusión, se pueden obtener algunas de las principales métricas utilizadas para evaluar el rendimiento de los diferentes modelos. En concreto, disponemos de:

- ***Accuracy***: Nos permite saber la cantidad de predicciones que han sido correctas en relación al conjunto de test:

[1] Imagen extraída de Interactive Chaos

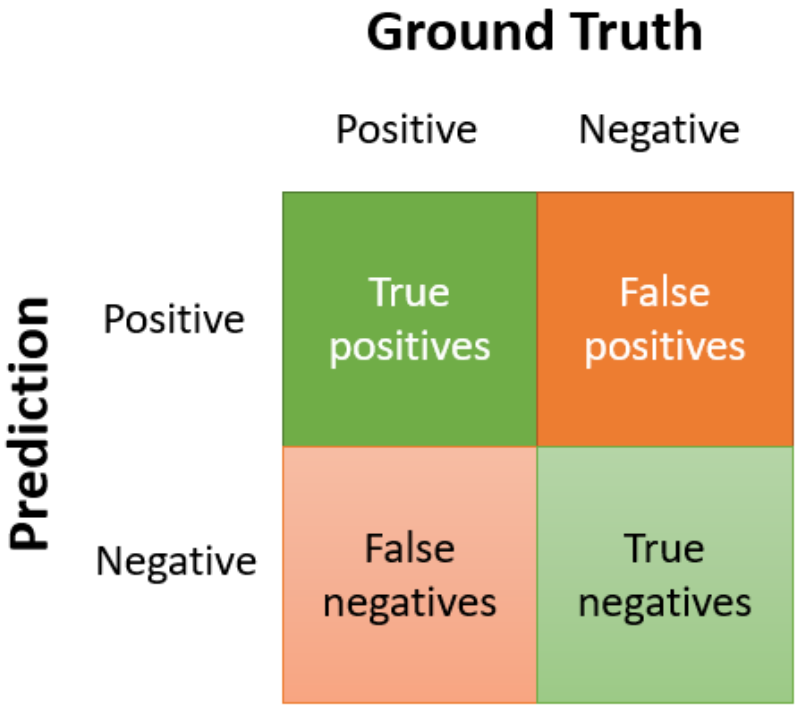

Figura 7.2: Representación de una matriz de confusión.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

- ***Recall***: Indica de los elementos considerados positivos, cuáles son realmente válidos, es decir:

$$Recall = \frac{TP}{TP + FN} \tag{7.2}$$

- ***Precision***: Mide el número de observaciones con éxito que son relevantes:

$$Precision = \frac{TP}{TP + FP} \tag{7.3}$$

- ***F-score***: Es una media armónica de los valores de *precision* y *recall*. Consideraremos su versión tradicional (F_1 *score*), que se calcula de la siguiente forma:

$$F_1 score = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2TP}{2TP + FP + FN} \tag{7.4}$$

- ***Índice Kappa de Cohen***: También conocido como *Coeficiente Kappa*, es utilizado co-

mo medición del grado de consenso entre el observador, quien determina los valores correctos de cada muestra, y el modelo de clasificación que trata de predecir dichos valores, tomando en cuenta el número de acuerdos (TP y TN) y desacuerdos (FP y FN) de ambos [36]. Este coeficiente se puede calcular de la siguiente forma:

$$K = \frac{p_0 - p_e}{1 - p_e} \tag{7.5}$$

Donde p_0 (probabilidad observada) es resultado de sumar los TP con los TN y dividirlos entre el número de muestras N, y p_e es la probabilidad esperada de que sendos calificadores estén de acuerdo por azar. Por lo general, obtener un valor de K igual o superior a 0.7 suele indicar que hay un consenso sustancial.

La elección de una u otra métrica como referente de cara al contraste final dependerá de los objetivos de negocio. En nuestro caso, queremos que se clasifiquen de forma correcta aquellos comentarios que sean o no *cyberbullying*, y que aquellos que se hayan considerado ciberacoso realmente lo sean. Esto se traduce en obtener la proporción de muestras con una predicción acertada (TP y TN), así como la cantidad de predicciones que verdaderamente eran positivas (es decir, los TP frente a los FP). Es por ello que nos centraremos en los valores de *accuracy* y *precision*, y utilizaremos el resto de métricas como refuerzo en dicho contraste.

Asimismo, en la siguiente sección, además de mostrar los resultados de cada modelo, compararemos cada valor obtenido con los mostrados de forma correspondiente en el artículo de referencia [24], y que para mayor comodidad incluimos en la tabla 7.1.

Modelo	Accuracy	Recall	Precision
Random Forest	0.86	0.88	0.88
AdaBoost	0.85	0.85	0.86
SVM linear	0.72	0.72	0.75

Tabla 7.1: Métricas de cada modelo del artículo de referencia.

7.1.3 Resultados obtenidos

Tras realizar las diferentes predicciones para cada modelo entrenado, se han recopilado los siguientes valores de cada métrica mencionada en la sección anterior, los cuales se pueden ver de forma detallada en las tablas 7.2, 7.3, 7.4 y 7.5:

GaussianNB

Dataset	Accuracy	Recall	Precision	F-Score	Kappa	Tiempo (s)
Baseline	0.808	0.553	0.701	0.618	0.492	0.103
WithoutLDA	0.805	0.456	0.753	0.568	0.451	0.108
WithoutDoc2vec	0.770	0.472	0.619	0.536	0.387	0.087
WithoutBoth	0.770	0.440	0.631	0.519	0.374	0.079

Tabla 7.2: Métricas obtenidas para GaussianNB.

Como vemos, el modelo gaussiano se presenta como uno de los modelos con menor *accuracy* (tal y como predijimos en el capítulo previo), y si bien es el algoritmo más rápido y simple de entrenar, también se aprecia que es bastante débil a nivel global. Además, se puede observar que el rendimiento general de los datasets derivados es menor que en el caso de *Baseline*, algo que también pronosticamos.

Random Forest

Dataset	Accuracy	Recall	Precision	F-Score	Kappa	Tiempo (s)
Baseline	0.907	0.679	0.988	0.805	0.747	7.039
WithoutLDA	0.915	0.704	0.993	0.824	0.770	7.723
WithoutDoc2vec	0.878	0.570	0.994	0.725	0.653	4.147
WithoutBoth	0.895	0.634	0.986	0.772	0.708	4.158

Tabla 7.3: Métricas obtenidas para RandomForest.

En este caso se ve que en cuanto a *accuracy* y *precision* es el modelo con los valores más altos, superando de forma considerable aquellos estimados en el artículo de referencia[2]. Si bien es cierto que el *recall* es ligeramente inferior, de forma conjunta se observa que es un algoritmo bastante completo.

Asimismo, se puede contemplar dos aspectos: que utilizando el dataset *WithoutLDA* se obtuvieron en todas las métricas resultados un poco superiores a los de *Baseline*, algo que en el resto de modelos no sucede; y que el dataset *WithoutBoth* presenta un mejor comportamiento que si sólo descartásemos los atributos de *doc2vec*.

[2] Para las comparaciones se están teniendo en cuenta los valores de la sección All Features + LDA.

AdaBoost

Dataset	Accuracy	Recall	Precision	F-Score	Kappa	Tiempo (s)
Baseline	0.885	0.674	0.889	0.767	0.692	46.362
WithoutLDA	0.877	0.660	0.874	0.752	0.673	50.341
WithoutDoc2vec	0.830	0.499	0.830	0.623	0.522	22.686
WithoutBoth	0.829	0.498	0.826	0.622	0.520	22.386

Tabla 7.4: Métricas obtenidas para AdaBoost.

En cuanto al caso de AdaBoost, se mantiene en unos valores similares a los de Random Forest, aunque el tiempo de entrenamiento es bastante superior, y en relación a los valores del artículo de referencia los mejora ligeramente. También se puede ver que a medida que se van eliminando ciertas características, los resultados de *recall*, *F-Score* y *Coeficiente Kappa* empeoran significativamente.

LinearSVC

Dataset	Accuracy	Recall	Precision	F-Score	Kappa	Tiempo (s)
Baseline	0.785	0.714	0.599	0.651	0.498	9.992
WithoutLDA	0.807	0.593	0.679	0.633	0.503	10.149
WithoutDoc2vec	0.683	0.581	0.451	0.508	0.280	7.236
WithoutBoth	0.764	0.194	0.855	0.316	0.237	6.800

Tabla 7.5: Métricas obtenidas para LinearSVC.

Finalmente tenemos a LinearSVC, el cual presenta los valores más bajos de *accuracy* y *precision*, pero un tiempo de entrenamiento razonable. Pese a ello, es capaz de mejorar un poco el valor de *accuracy* del artículo de referencia. Consideramos que probablemente este resultado se deba a la alta sensibilidad de este modelo a los rangos de valores de cada atributo.

Por otro lado, cabe señalar que empleando el dataset *WithoutLDA* se obtuvieron unos valores de *accuracy*, *precision* y *Coeficiente Kappa* ligeramente superiores. Asimismo, resulta interesante el resultado de *WithoutBoth*, ya que presenta unos valores de *accuracy* y *precision* similares o muy superiores respecto a *Baseline* (lo cual podría ser a priori una mejor alternativa), pero en relación al resto de valores resulta bastante inconsistente, lo cual nos permite demostrar que el hecho de que las métricas principales sean buenas no implica que una implementación sea mejor que el resto.

7.2 Revisar el procedimiento

Una vez analizados los resultados anteriores, podemos concluir que tanto Random Forest como AdaBoost han sido capaces de mejorar las métricas del artículo de referencia, y que todos han sido por lo general superiores al modelo de GaussianNB. Además, se ha observado que cuanta mayor complejidad presente el modelo a entrenar, mayor será su tiempo de ejecución, y que existen dos criterios claros de división respecto a los conjuntos de datos, así como una serie de conclusiones en cuanto a las características que se han tomado como descarte:

- Por un lado, el hecho de descartar atributos de un conjunto de datos concreto da lugar en general a rendimientos peores que si utilizásemos el conjunto original, ya que en cierto modo cuanta más información tengamos sobre cada muestra más precisa podrá ser su predicción.
- Por otro lado, el hecho de incluir o no determinadas características en un dataset puede influir en mayor medida al resultado de las métricas (ya que pueden tener información más valiosa sobre el conjunto de datos), de modo que dependiendo del tipo de muestras que tengamos puede ser recomendable utilizar unas técnicas de procesamiento de lenguaje natural u otras.
- Aquellos rasgos relativos a LDA presentan un impacto poco notorio en el resultado de las métricas. El motivo más probable es que la agrupación que realiza cada uno de los tópicos considerados es más abstracta y/o genérica de lo que en un primer momento pudiera precisar el dataset, de forma que ciertos comentarios los enmarca en temáticas que no son del todo correctas, influyendo de forma negativa en sus respectivos valores.
- La eliminación de aquellos atributos que tienen que ver con *doc2vec* presenta un efecto considerable en las métricas de evaluación, posiblemente relacionado con lo expuesto en la sección 4.2.
- El descarte de sendos grupos de características puede suponer un impacto más o menos grave, dependiendo del tipo de modelo a considerar.

Asimismo, a lo largo de todo este procedimiento se ha llevado a cabo un proceso de *hiperparametrización* con la mayor variedad posible en relación a los valores de los parámetros a optimizar, con el objetivo de poder establecer un mejor contraste respecto al comportamiento de cada modelo y poder obtener en última instancia resultados consistentes.

También hemos tenido en cuenta diversas métricas que nos permiten visualizar el rendimiento de cada algoritmo desde diferentes puntos de vista, y se han comparado con otras

medidas obtenidas en escenarios reales desarrollados en trabajos previos.

Otro detalle de especial relevancia es que todas las simulaciones se han llevado a cabo bajo las mismas condiciones, tanto a nivel de hardware como de implementación, ya que se pretende reflejar el comportamiento más puro posible, sin tener en cuenta posibles optimizaciones de los algoritmos ni en lo que respecta a la paralelización de las diferentes operaciones con el uso de librerías, como es el caso de *Intelex* [37].

7.3 Determinar siguientes pasos

Teniendo en mente todo lo descrito hasta ahora, es sensato considerar que los principales modelos a utilizar si se llegase a desplegar un escenario de similar magnitud son Random Forest y AdaBoost, ya que si bien cualquiera de los modelos descritos en este trabajo mantiene unos resultados razonables, estos dos son los más completos y versátiles a nivel de parametrización, sobresaliendo el primero por presentar un menor tiempo de ejecución. No obstante, podría ser conveniente agregar otros modelos para realizar más contrastes o, si se dispone de los recursos hardware necesarios, se puede plantear la posibilidad de incluir modelos de mayor complejidad o de *Deep Learning*.

Otro aspecto interesante podría ser el llevar a cabo un análisis exhaustivo sobre el proceso de generación de cada característica, o incluso dentro de cada tipo profundizar en su uso. Un ejemplo de esto lo conforman los atributos relacionados con LDA, en donde se puede extender el número de tópicos a tener en cuenta o incluso derivar otras temáticas más precisas a partir de los ya incluidos originalmente, y realizar un estudio acerca del impacto de esta decisión.

Además, se podrían generar adicionalmente otros datasets que descarten atributos distintos a los considerados en este trabajo, o bien combinaciones de varios de ellos, y ver cómo los diferentes modelos rinden bajo estas situaciones.

Capítulo 8

Conclusiones

PARA finalizar este trabajo, dedicaremos este capítulo a establecer un cierre extrayendo las principales reflexiones y lecciones aprendidas acerca de toda la metodología que se ha llevado a cabo, además de ofrecer una revisión sobre el grado de cumplimiento de los objetivos de negocio y la formación obtenida en el proceso, y por último estableceremos una serie de alternativas a realizar si en un futuro se pretende continuar con la investigación tomando este proyecto como base.

8.1 Grado de compleción

En relación con el nivel de éxito de este trabajo, hemos obtenido un resultado provechoso dentro del tiempo dedicado y teniendo en cuenta el nivel básico de conocimiento y formación que previamente poseía el alumno acerca de *Machine Learning* y de las diferentes técnicas a utilizar en dicho ámbito, lo cual requirió un esfuerzo extra en cuestión de aprendizaje. Por todo esto, consideramos que se han alcanzado los objetivos de negocio:

- Por un lado, en el capítulo 3 hemos expuesto de forma detallada los conceptos básicos sobre los que se sustenta este trabajo y considerado las distintas alternativas que se presentan dentro de cada disciplina.
- Por otro lado, en los capítulos 4 y 5 hemos realizado un análisis exploratorio del conjunto de muestras utilizado como punto de partida, tratando de comprender los posibles valores de cada atributo, y hemos aplicado diferentes técnicas propias de la Ingeniería de Características para detectar posibles anomalías (IQR, Isolation Forest) y procesar cada atributo, sometiéndolos a un proceso de estandarización y seleccionando aquellos que fuesen más relevantes con el fin de reducir la dimensionalidad del conjunto (matriz de correlaciones, junto con Random Forest, RFECV y PCA).

- También se han escogido y analizado los principales modelos durante el capítulo 3, y en el capítulo 6 se ha llevado a cabo un proceso de hiperparametrización (*Grid Search* con validación cruzada) para considerar las implementaciones óptimas de cada uno, además de estimar sus comportamientos en los conjuntos de test gracias a los resultados de dichas pruebas.
- Finalmente, en el capítulo 7 se ha establecido una comparativa entre los diferentes modelos y los resultados de los mismos con los de trabajos realizados con anterioridad.

8.2 Lecciones aprendidas

La realización de este trabajo ha permitido disponer de una gran cantidad de conocimiento nuevo y extender ciertos conceptos vistos tanto en la mención de *Tecnologías de la Información* como a lo largo del grado. En concreto:

- Aporta conocimientos sobre *Machine Learning*, tanto desde una perspectiva histórica como de la intrínseca existente en la implementación de diferentes algoritmos para la resolución de problemas complejos.
- Incorpora conocimientos del procesado del lenguaje natural, que permiten una mejor comprensión de los atributos del dataset.
- Manifiesta diferentes capacidades en lo referente al tratamiento estadístico de datos, al tener que reflexionar acerca de las distribuciones que siguen los mismos.
- Extiende ciertos conceptos en relación a la minería y las bases de datos.
- Se ponen en práctica varios aspectos que se han aprendido sobre el lenguaje de Python, gracias al uso de diferentes tecnologías y a la implementación de funciones propias para llevar a cabo ciertas etapas.
- Se complementa el aprendizaje que se ha adquirido acerca de la gestión de proyectos, al disponer de la oportunidad de planificar un escenario de similar magnitud, sirviendo a su vez de referencia para futuros proyectos.

En definitiva, este trabajo nos ha ayudado a consolidar varios conocimientos obtenidos durante la carrera y a formar una idea inicial de lo que comprende la rama de estudio del aprendizaje máquina, algo que lleva experimentando un gran auge en los últimos tiempos.

8.3 Líneas futuras

A continuación, ofrecemos una serie de alternativas a tomar considerando este proyecto como base:

- Profundizar en el estudio de otros modelos de *Machine Learning*, especialmente aquellos relacionados con las SVM y otros de índole lineal, ya que consideramos que los resultados que pueden alcanzar modelos de este tipo, tales como LinearSVC, son fácilmente mejorables.
- Contrastar los modelos utilizados en este trabajo con otros datasets de la misma índole (tanto ya creados como otros que puedan surgir en el futuro), con el fin de ver si este comportamiento se puede reproducir a mayor escala.
- Utilizar algoritmos de *Deep Learning*, ya que al manejarse mejor con una gran cantidad de datos podría resultar en modelos más eficientes.
- Investigar con mayor detalle acerca del impacto que tendría el construir un dataset propio mediante técnicas de NLP, y contrastarlo con los modelos de este proyecto u otros modelos distintos. También se pueden incluir modelos a gran escala similares a los *Embeddings* o bien tratar de refinar el estudio de diferentes características del dataset original, para disponer de una conclusión más definitoria sobre cuáles atributos son más influyentes a la hora de detectar ciberacoso.
- Realizar un despliegue de estos algoritmos bajo un escenario real, para ver si se comportan bien en la práctica.

Bibliografía

[1] P. Cerezo, "La generación z y la información," 2020, consultado en junio de 2023. [En línea]. Disponible en: https://www.injuve.es/sites/default/files/2017/28/publicaciones/documentos_7._la_generacion_z_y_la_informacion.pdf

[2] G. Álvarez Idarraga, "Consecuencias e impacto del ciberacoso implications and impact of cyberbullying," *BOSCO*, vol. 38, pp. 109–127, 2018.

[3] L. Bojana, V. Anca, S. Elisabeth, C. Stephane, and D. G. Rosanna, "How children (10-18) experienced online risks during the covid-19 lockdown - spring 2020," 2021, consultado en junio de 2023. [En línea]. Disponible en: https://publications.jrc.ec.europa.eu/repository/handle/JRC124034

[4] S. G. de Investigación, "Plan estatal de investigación científica, técnica y de innovación 2021 - 2023," 2021, consultado en junio de 2023. [En línea]. Disponible en: https://www.ciencia.gob.es/InfoGeneralPortal/documento/e1f1deb1-7321-4dd9-b8ca-f97ece358d1c

[5] A. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists.* O'Reilly Media, 2016.

[6] IBM, "Foundational methodology for data science," 2015, consultado en abril de 2023. [En línea]. Disponible en: https://www.ibm.com/downloads/cas/6RZMKDN8#:~:text=La%20metodolog%C3%ADa%20no%20depende%20de,para%20obtener%20respuestas%20o%20resultados.

[7] Z. Luna, "Understanding crisp-dm and its importance in data science projects," 2021, consultado en abril de 2023. [En línea]. Disponible en: https://medium.com/analytics-vidhya/understanding-crisp-dm-and-its-importance-in-data-science-projects-91c8742c9f9b

[8] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "Crisp-dm 1.0: Step-by-step data mining guide," 2000, consultado en abril

de 2023. [En línea]. Disponible en: https://web.archive.org/web/20220401041957/https://www.the-modeling-agency.com/crisp-dm.pdf

[9] L. J. P. Q. Mg. Augusto Cortez Vásquez, Mg. Hugo Vega Huerta, "Procesamiento de lenguaje natural," 2009, consultado en junio de 2023. [En línea]. Disponible en: https://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/5923/5121

[10] M. Hernández and J. Gómez, "Aplicaciones de procesamiento de lenguaje natural," 2013, consultado en junio de 2023. [En línea]. Disponible en: https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/32/pdf

[11] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia tools and applications*, vol. 82, no. 3, pp. 3713–3744, 2023.

[12] Y. Mejova, "Sentiment analysis: An overview," *University of Iowa, Computer Science Department*, 2009.

[13] W. Zhang, T. Yoshida, and X. Tang, "Tfidf, lsi and multi-word in information retrieval and text categorization," in *2008 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2008, pp. 108–113.

[14] R. Churchill and L. Singh, "The evolution of topic modeling," *ACM Computing Surveys*, vol. 54, no. 10s, pp. 1–35, 2022.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[16] A. G. Jivani *et al.*, "A comparative study of stemming algorithms," *Int. J. Comp. Tech. Appl*, vol. 2, no. 6, pp. 1930–1938, 2011.

[17] B. Mahesh, "Machine learning algorithms - a review," 2018, consultado en marzo de 2023. [En línea]. Disponible en: https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096

[18] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[19] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE, 2018, pp. 1–6.

[20] R. Rojas, "Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting," 2009, consultado en abril de 2023. [En línea]. Disponible en: http://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf

[21] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2016, consultado en mayo de 2023. [En línea]. Disponible en: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[22] E. Kim, "Everything you wanted to know about the kernel trick," *URl: http://www. eric-kim. net/eric-kim-net/posts/1/kernel_ trick. html*, 2013, consultado en mayo de 2023.

[23] M. Bégin, "El ciberacoso. una revisión de investigaciones internacionales sobre representaciones, prevalencias, efectos y explicaciones del fenómeno," *Re-presentaciones. Periodismo, Comunicación y Sociedad*, no. 10, pp. 52–78, 2018.

[24] R. I. Rafiq, H. Hosseimmardi, Mattson, Han, Q. Lv, and S. Mishra, "Analysis and detection of labeled cyberbullying instances in vine, a video-based social network," 2016.

[25] M. F. López-Vizcaíno, F. J. Nóvoa, V. Carneiro, and F. Cacheda, "Early detection of cyberbullying on social media networks," 2021, consultado en junio de 2023. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0167739X21000157

[26] NumFOCUS, "About pandas," 2023, consultado en junio de 2023. [En línea]. Disponible en: https://pandas.pydata.org/about/index.html

[27] J. Hao and T. K. Ho, "Machine learning made easy: a review of scikit-learn package in python programming language," *Journal of Educational and Behavioral Statistics*, vol. 44, no. 3, pp. 348–361, 2019.

[28] "Boe 2018," 2018, consultado en junio de 2023. [En línea]. Disponible en: https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf

[29] S. Loria, "Textblob," 2020, consultado en junio de 2023. [En línea]. Disponible en: https://textblob.readthedocs.io/en/dev/

[30] A. Perrier, "Feature importance in random forests," 2015, consultado en abril de 2023. [En línea]. Disponible en: https://alexisperrier.com/datascience/2015/08/27/feature-importance-random-forests-gini-accuracy.html

[31] J. Brownlee, "Random oversampling and undersampling for imbalanced classification," 2020, consultado en junio de 2023. [En línea]. Disponible en: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/#:~:

text=Random%20oversampling%20involves%20randomly%20selecting,them%20from%20the%20training%20dataset.

[32] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th international conference on information and communication systems (ICICS)*. IEEE, 2020, pp. 243–248.

[33] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, pp. 85–126, 2004.

[34] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[35] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multiclass adaboost," 2006, consultado en mayo de 2023. [En línea]. Disponible en: https://hastie.su.domains/Papers/samme.pdf

[36] A. Kumar, "Cohen kappa score python example: Machine learning," 2022, consultado en junio de 2023. [En línea]. Disponible en: https://vitalflux.com/cohen-kappa-score-python-example-machine-learning/

[37] Intel, "Intel(r) extension for scikit-learn," 2023, consultado en mayo de 2023. [En línea]. Disponible en: https://github.com/intel/scikit-learn-intelex