Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

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

Estudiante: Andrés Abal Aldao

Dirección: Diego Fernández Iglesias

Patricia Martín Rodilla

A Coruña, septiembre de 2023.

A mi familia, por su amor y ánimo constante. Gracias por haberme apoyado en cada paso de este viaje

Agradecimientos

Mi familia merece un agradecimiento especial por su apoyo inquebrantable y sus palabras de aliento constantes, su respaldo ha sido un pilar fundamental en este viaje académico. También quiero agradecer a mis compañeros de clase por su apoyo mutuo durante esta travesía, ya que juntos hemos superado diversos desafíos y celebrado éxitos. Y, por último, agradezco a mis tutores, Diego Fernández Iglesias y Patricia Martín Rodilla, por su orientación experta y darme la posibilidad de compartir su experiencia conmigo.

Resumen

Las redes sociales han alcanzado un éxito indiscutible en nuestra sociedad, forman parte de nuestras vidas en el día a día y cuentan con innumerables virtudes. Lamentablemente, también son un soporte ideal para la propagación del ciberacoso, el cual puede acarrear un gran impacto a largo plazo, con efectos tanto físicos como psicológicos padecidos por las víctimas.

Debido al auge del uso de las redes sociales (el cual se vio potenciado por la pandemia provocada por el COVID-19) y a la cantidad ingente de datos existentes, la detección del ciberacoso haciendo uso de algoritmos de Inteligencia Artificial se hace primordial, ya que busca paliar un problema que afecta diariamente a muchas personas.

Siguiendo las fases de la metodología CRISP-DM, hemos aplicado diversos algoritmos de *Machine Learning*, centrados en el aprendizaje supervisado, a un conjunto de datos que abarca una amplia gama de información de la red social *Instagram*, con el objetivo de predecir si un comentario es acoso o no.

Una vez que los datos han sido analizados y procesados, se han utilizado para alimentar a los algoritmos correspondientes. Posteriormente, hemos comparado los resultados en función de su desempeño y eficacia, con el propósito de identificar cuáles son los más adecuados.

Abstract

Social networks have achieved undeniable success in our society; they are an integral part of our daily lifes and possess countless virtues. Unfortunately, they also serve as an ideal platform for the proliferation of cyberbullying, which can have a significant long-term impact, with both physical and psychological effects suffered by the victims.

Due to the rise in the use of social networks (which was further accelerated by the COVID-19 pandemic) and the vast amount of existing data, the detection of cyberbullying using Artificial Intelligence algorithms becomes paramount, as it aims to mitigate a problem that affects many people daily.

Following the phases of the CRISP-DM methodology, we have applied various Machine Learning algorithms, primarily focused on supervised learning, to a dataset covering a wide range of information from the social network *Instagram*. The objective was to predict whether

a comment constitutes harassment or not.

Once the data has been analyzed and processed, it has been used to feed the corresponding algorithms. Subsequently, we have compared the results based on their performance and effectiveness, with the aim of identifying which ones are the most suitable.

Palabras clave:

- Aprendizaje Máquina
- Ciberacoso
- Aprendizaje Supervisado
- Clasificación
- Grid Search
- Random Forest
- CRISP-DM

Keywords:

- Machine Learning
- Ciberbullying
- Supervised Learning
- Classification
- Grid Search
- Random Forest
- CRISP-DM

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los tiempos recientes, hemos presenciado un continuo avance tecnológico enfocado en promover la cercanía y potenciar la interacción social. Como consecuencia de estos esfuerzos, se ha logrado una notable inmediatez y facilidad en las comunicaciones. El auge de las redes sociales, especialmente durante la pandemia, ha traído un cambio drástico a nuestras vidas, brindándonos innumerables alegrías y momentos especiales para compartir.

Lamentablemente, esto tiene dos caras, ya que no todo lo que se comparte son experiencias positivas. El acoso consiste en la repetición de comportamientos abusivos llevada a cabo por un grupo o individuo contra una víctima indefensa [1]. Cuando se infringe haciendo uso de dispositivos electrónicos, se trata de ciberacoso o *ciberbullying* [2], el cual resulta ser más peligroso debido a su rápida propagación y a las frecuencias de daño permitidas.

Sus consecuencias pueden ser altamente perjudiciales. De hecho, este comportamiento puede generar un gran impacto negativo a largo plazo en las personas afectadas [3]. Los daños que provoca pueden ser tanto psicológicos como físicos, entre los que destacan el aislamiento, la depresión, comportamiento autolesivo y, en casos extremos, incluso el suicidio.

Este impacto cobra una relevancia especial en los preadolescentes y adolescentes [4], ya que este grupo demográfico desconoce cómo es la vida sin estos avances tecnológicos. Paradójicamente, a pesar de ser la generación más conectada y con mayores facilidades comunicativas, también es la que experimenta un mayor aislamiento social.

Dadas las circunstancias, la detección automática del ciberacoso se convierte en una prioridad fundamental para mitigar el impacto que está teniendo en la sociedad. Para lograrlo, aprovechamos la vasta cantidad de datos generados por las redes sociales, junto con diversas técnicas de aprendizaje automático, con el propósito de identificar estos comportamientos en

las cuentas de los usuarios. Esta aproximación permitiría la supervisión de cuentas potencialmente acosadoras y, en última instancia, el control de su ocurrencia, proporcionando un nivel adecuado de protección a las víctimas.

Un indicativo de la relevancia de la temática abordada en este trabajo, radica en su estrecha relación con diversas áreas prioritarias delineadas en el Plan Estatal de Investigación Científica (PEICTI) [5].

1.1 Relación entre ciberacoso y Machine Learning

La sinergia entre diversas técnicas de procesamiento del lenguaje natural (NLP) y la abundante información generada por las redes sociales constituye una valiosa fuente de datos en la detección del ciberacoso. Esto se debe a que las redes sirven como canales de propagación, y los comentarios son los principales encargados para transmitir el contenido negativo a sus víctimas.

El carácter del conjunto de datos resultante es especialmente relevante en el ámbito del Machine Learning. Sus dimensiones y complejidad lo hacen prácticamente inabarcable para el entendimiento humano, y resulta desafiante abordarlo mediante enfoques de programación tradicional. Por esta razón, el aprendizaje automático emerge como la solución ideal para abordar problemas en este dominio. Facilita enormemente la resolución de estos desafíos de clasificación, al tiempo que demuestra ser efectivo y eficiente.

1.2 Objetivos

El objetivo principal de este proyecto es evaluar el rendimiento de varios modelos de *Machine Learning* en la tarea de clasificar si determinados comentarios de la red social *Instagram* pueden ser identificados como acoso o no. Para ello se parte de un conjunto de datos con características previamente definidas. Entre los objetivos específicos a alcanzar, se incluyen:

- Utilizar una variedad de métodos de análisis del ciberacoso.
- Utilizar técnicas de ingeniería de características con el objetivo de simplificar la estructura del conjunto de datos y extraer los atributos que mejor representen la información relevante.
- Seleccionar, aplicar y contrastar diversos algoritmos de aprendizaje automático.
- Evaluar los modelos resultantes.

Además, con el objetivo de aumentar la diversidad de conjuntos de datos disponibles para comparar los resultados, generaremos tres nuevos a partir del conjunto de datos original del que partimos. Estos conjuntos se obtendrán mediante la eliminación de grupos de características similares. Posteriormente, cada uno será sometido al proceso previamente descrito, lo que nos permitirá evaluar de manera cuantitativa la contribución de cada grupo de características.

Como primer paso para adquirir conocimiento sobre el contexto y las tecnologías que vamos a utilizar, hemos utilizado el libro *"Introducción al Aprendizaje Automático con Python"* [6] como nuestra principal fuente de referencia.

1.3 Estructura del proyecto

La estructura de este informe se adhiere al ciclo de vida de la metodología CRISP-DM. Cada capítulo se corresponde con una de sus fases y, en cada uno de ellos, ofrecemos una breve introducción que resalta la relevancia de la fase en cuestión, seguida de una explicación detallada y desarrollo de sus subfases. Finalmente, presentamos las conclusiones obtenidas y comentamos posibles líneas de trabajo futuro que pueden aprovechar este trabajo como punto de partida. La estructura es la siguiente:

- **Capítulo 2 Metodología**: en este capítulo, presentamos en detalle la metodología CRISP-DM, ofreciendo una comprensión completa de su funcionamiento.
- **Capítulo 3 Comprensión del Negocio**: en esta fase, analizamos en detalle los conceptos y objetivos clave del negocio, evaluamos las tecnologías utilizadas y elaboramos un plan para el proyecto.
- **Capítulo 4 Comprensión de los Datos**: esta etapa se centra en explorar y comprender los datos disponibles, identificando patrones, relaciones y posibles limitaciones.
- **Capítulo 5 Preparación de los Datos**: explicamos e implementamos las técnicas utilizadas para limpiar, seleccionar y transformar los datos para su uso en los modelos de aprendizaje automático.
- **Capítulo 6 Modelado**: escogemos los modelos que utilizaremos y comenzamos el proceso de hiperparametrización.
- **Capítulo 7 Evaluación**: en esta fase, llevamos a cabo la evaluación y comparación de los modelos que hemos desarrollado, empleando diversas métricas como criterio de análisis.

- **Capítulo 8 Conclusiones**: en este capítulo final, exponemos los resultados más destacados, compartimos las lecciones aprendidas y ofrecemos recomendaciones para investigaciones futuras o posibles mejoras.

1.4 Código del proyecto

El código desarrollado para este proyecto se encuentra alojado en el repositorio ML-CyberbullingDetection-SocialNetworks de GitHub y se distribuye bajo la Licencia MIT. Esta licencia es altamente permisiva, permitiendo a otros usuarios utilizar, modificar y distribuir el software de forma gratuita, siempre y cuando incluyan la atribución de derechos de autor y conserven la licencia original en cualquier versión modificada.

Dentro del repositorio, se pueden encontrar informes detallados sobre las ejecuciones, las implementaciones de los diversos algoritmos y otros archivos, como los modelos de entrenamiento en formato binario.

Capítulo 2

Metodología

Metodología hace referencia a una serie de métodos y técnicas de rigor científico que se aplican sistemáticamente durante un proceso de investigación para alcanzar un resultado teóricamente válido. Orienta la manera en que vamos a enfocar una investigación y la forma en que vamos a recolectar, analizar y clasificar los datos, con el objetivo de que nuestros resultados tengan validez, pertinencia y cumplan con los estándares de exigencia científica.

En cuanto a la selección de metodologías para aplicar a la ciencia de datos o *data science*, hay varias opciones disponibles. Entre estas podemos destacar KDD (*Knowledge Discovery in Databases*) [7], SEMMA (*Sample, Explore, Modify, Model, Assess*) y CRISP-DM (*Cross Industry Standard Process for Data Mining*), siendo estas dos últimas consideradas como una implementación más completa de KDD [8]. Finalmente, para este trabajo elegimos utilizar CRISP-DM, aún siendo conscientes de sus limitaciones y nuevas alternativas que las solventan. Entre las razones principales en las que basamos nuestra elección, está la consideración de que el proyecto se beneficiaría del uso de una tecnología extensamente utilizada y experimentada [9].

El ciclo de vida del modelo está compuesto de las siguientes fases: Comprensión de Negocio, Comprensión de Datos, Preparación de Datos, Modelado, Evaluación y Despliegue, como se puede ver en la Figura 2.1. La ejecución de estas fases no tienen necesidad de hacerse en un orden estricto, ya que se puede volver a una fase anterior en caso de necesidad. Las dependencias más importantes entre las fases están indicadas con flechas, mientras que el círculo externo representa la propiedad cíclica de esta metodología. A continuación, describiremos los objetivos de cada fase [10]:

1. *Comprensión de Negocio*: la primera fase de la metodología se enfoca en comprender a fondo el contexto y los objetivos del proyecto. También se evalúa la situación actual y comienza la producción de un plan detallado que indique las actividades, plazos y recursos necesarios. Esto es esencial para establecer una base que garantice el éxito,

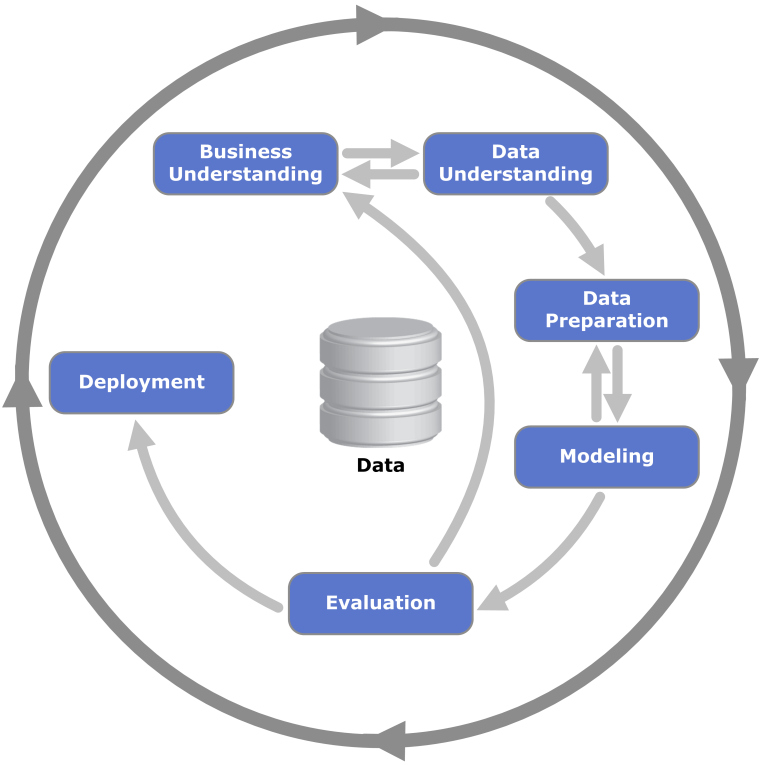

Figura 2.1: Esquema de CRISP-DM.

efectividad y correcta realización de todo el proceso.

2. *Comprensión de Datos*: el objetivo es extraer los datos y realizar una exploración detallada de estos, buscando asegurar su calidad y comprender su estructura e importancia. Examinamos la naturaleza de cada una de sus características e identificamos posibles problemas (valores atípicos, nulos o errores). Así es como determinaremos que los datos cumplen con los requisitos necesarios para abordar los objetivos establecidos.

3. *Preparación de Datos*: este proceso abarca la elección y depuración de los datos, así como también la posibilidad de crear nuevas características. También se consideran tareas de escalado y normalización para adecuar los datos a los requisitos específicos de ciertos modelos. En resumen, se busca optimizar los datos para obtener un mejor desempeño en las siguientes fases.

4. *Modelado*: en esta etapa, se seleccionan los modelos más adecuados y se ajustan sus parámetros para obtener los mejores resultados posibles. Luego se utilizará un conjunto de entrenamiento para poder evaluar el desempeño de los diferentes modelos según el tipo de problema que enfrentamos (clasificación, regresión...). El objetivo es identificar aquellos modelos que se adapten de manera óptima a la naturaleza de los datos y al

objetivo del análisis.

5. *Evaluación*: se procede a evaluar los algoritmos previamente seleccionados utilizando un conjunto de datos de prueba y analizando su rendimiento mediante métricas específicas. El propósito es identificar las ventajas y desventajas de cada modelo en base a su capacidad para lograr los objetivos de negocio establecidos. Este proceso será crucial para determinar qué modelo se ajusta mejor a los requerimientos y necesidades del proyecto.

6. *Despliegue*: se establece una distribución del modelo en base a las condiciones proporcionadas por escenarios reales, luego su rendimiento es supervisado a lo largo de su ejecución, siendo su resultado un informe final que evalúa la calidad del modelo. Como no disponemos de un escenario apropiado, hemos decidido prescindir de esta implementación.

Cada una de estas fases se desglosa en otras fases y subfases que siguen una secuencia lógica, componiendo así una estructura organizativa llamada jerarquía de actividades. Como se puede ver en la Figura 2.2, cada fase está compuesta de actividades que cada vez se van subdividiendo en instancias más detalladas.

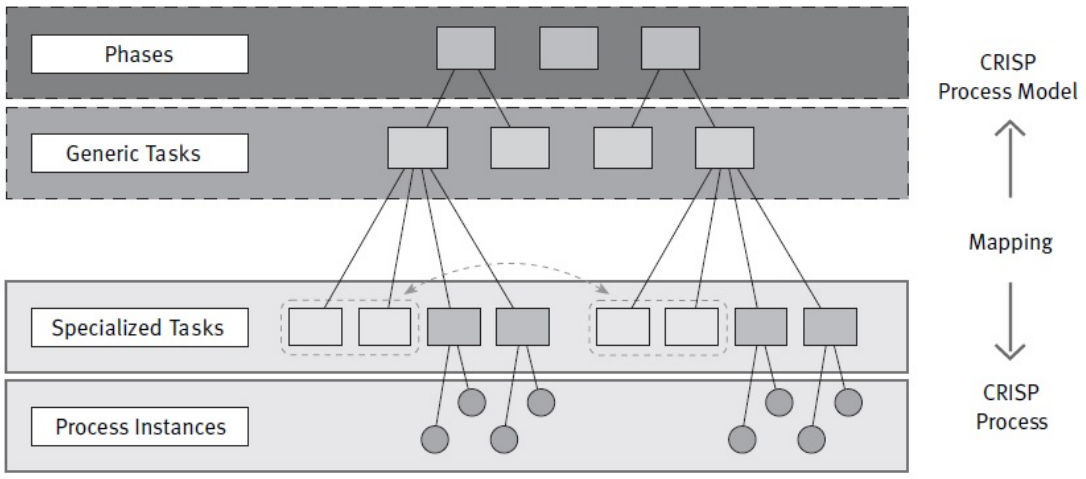

Figura 2.2: Modelo jerárquico de CRISP-DM.

Los próximos capítulos tratan cada una de las 6 fases principales de la metodología CRISP-DM. En cada capítulo, se proporciona una síntesis completa de las actividades realizadas de cada fase durante las diferentes iteraciones.

Capítulo 3

Comprensión del Negocio

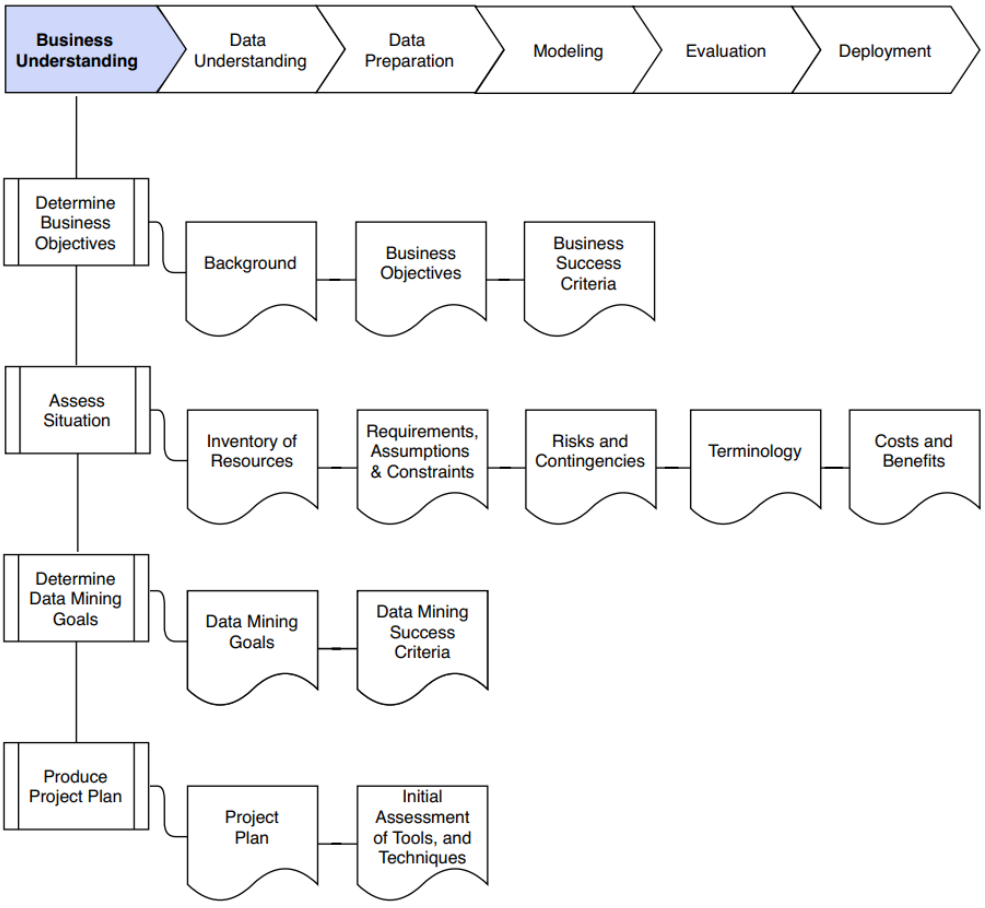

Figura 3.1: Fase CRISP-DM: Comprensión el negocio.

Comprensión del negocio es la primera fase del ciclo de vida CRISP-DM. En este capítulo veremos cómo se aplicaría dentro del marco específico de nuestro proyecto, lo cual implica entender todo aquello relacionado con él, incluyendo las tecnologías y herramientas

empleadas. Este conocimiento nos ayudará a especificar los objetivos, evaluarlos e identificar las estrategias y posibles soluciones que nos permitan alcanzar nuestras metas.

Durante esta fase determinaremos cuáles son los objetivos del negocio, evaluaremos la situación actual, determinaremos las metas que deberemos cumplir durante el proceso y finalmente generaremos el plan de proyecto.

3.1 Determinación de los objetivos del negocio

En esta parte se busca identificar y definir claramente los objetivos deseados. Esto implica comprender las necesidades y prioridades, así como establecer metas específicas y medibles que puedan orientar el proceso. Siguiendo la Figura 3.1, la primera actividad *background* consiste en entender el contexto empresarial y los antecedentes del problema a resolver. En la segunda actividad se definen de manera clara y precisa los objetivos del proyecto desde la perspectiva del negocio, mientras que en la tercera se determinan los criterios para evaluar el éxito del proyecto una vez se alcanzan los objetivos. En nuestro caso, considerando lo expuesto en capítulos anteriores, estas dos últimas fases ya están predefinidas.

El contexto requerido para este proyecto investiga cómo implementar un enfoque de Procesamiento del Lenguaje Natural ,y se recopila información sobre *Machine Learning*, ya que es fundamental comprender al detalle qué es y conocer los diferentes tipos de modelos, para luego hacer énfasis en los que son de mayor interés para nuestro proyecto.

También se llevó a cabo una revisión bibliográfica enfocada en investigaciones previas que emplean características textuales para la detección del ciberacoso. Esto se hizo con el propósito de situar nuestro trabajo en el contexto de la investigación existente y analizar las diversas aproximaciones y resultados obtenidos por diferentes investigadores en este ámbito específico [11] [12] [13] [14].

3.1.1 NLP: *Natural Language Processing*

Natural Language Processing o NLP es una rama que junta la inteligencia artificial y la lingüística computacional. Su objetivo principal es permitir a las máquinas comprender, interpretar y generar el lenguaje humano de manera que puedan realizar tareas relacionadas con el procesamiento del texto y el análisis lingüístico. Esto lo consigue haciendo uso de algoritmos o modelos que permiten reconocer patrones y estructuras del lenguaje natural. Este

campo se puede dividir en dos partes [15]:

- **NLU** (*Natural Language Understanding*): centrada en la capacidad de las máquinas para entender y procesar el lenguaje humano. Utiliza diversas técnicas y algoritmos de interpretación semántica y sintáctica del texto con el objetivo de extraer información relevante. Se emplea en tareas de clasificación del texto, análisis de sentimientos...
- **NLG** (*Natural Language Generation*): esta parte se ocupa de la capacidad de las máquinas para generar textos coherentes. Utiliza técnicas de generación de texto para crear respuesta, resúmenes, traducciones...

La cohesión de estas dos partes permiten a las máquinas interactuar de manera más natural y efectiva con el lenguaje humano. A continuación, nos enfocaremos en revisar algunas de las técnicas clave aplicadas en este campo y que están relacionadas con la generación de muchas de las características que conforman nuestro conjunto de datos.

Sentiment Analysis

Esta técnica tiene como objetivo identificar y clasificar las emociones, opiniones o actitudes expresadas en un texto. Su finalidad es determinar si estas emociones presentan un tono negativo o positivo. El análisis del sentimiento se desglosa típicamente en tres fases [16]:

1. Clasificación binaria: en esta tarea el texto es etiquetado como ”positivo” o ”negativo”. es una forma sencilla de análisis de sentimiento en la que se identifican las emociones binarias. Es común que esta clasificación se haga en base a la influencia que tienen ciertas palabras en el resultado de una opinión.
2. Clasificación polaridad múltiple: en esta fase el texto se etiqueta dentro del rango de polaridades, como ”muy positivo”, ”positivo”, ”neutral”, ”negativo” o ”muy negativo”. Está fuertemente relacionada con la fase anterior y permite una clasificación más detallada de las emociones expresadas.
3. Análisis de sentimiento de aspectos: es una tarea opcional que se centra en identificar la polaridad con aspectos específicos o entidades dentro del texto.

En el contexto de nuestro escenario, esta técnica nos permite analizar detalladamente las emociones expresadas en un comentario específico, lo que nos brinda una mayor comprensión de la situación y nos ayuda a abordar de una forma más efectiva su relación con el ámbito del *ciberbullying*. Hay dos enfoques que permiten la realización de esta técnica, uno basado en la semántica (*lexicons*, *stemming*...) y otro basado en el aprendizaje máquina (*tf-idf, bag of words*...).

TF-IDF

Term Frequency-Inverse Document Frequency se trata de una técnica utilizada en el campo del Procesamiento del Lenguaje Natural para evaluar la importancia de una palabra en un documento dentro de un conjunto más grande de documentos. Su cálculo implica los siguientes componentes:

- Frecuencia de Término: mide la frecuencia de una palabra en un documento específico. Cuantas más veces aparece, mayor será su *TF* para ese documento.
- Frecuencia Inversa de Documento: mide la rareza de la palabra dentro de un conjunto de documentos. Si aparece pocas veces en los textos, su *IDF* será alto

El puntaje final *TF-IDF* de una palabra en un documento se obtiene mediante la multiplicación de su frecuencia de término (TF) por su inverso de frecuencia de documento (IDF). Este valor resultante, desempeña un papel fundamental en tareas de recuperación de información y clasificación de texto, ya que ayuda a resaltar la importancia de ciertas palabras y contribuye significativamente a mejorar la precisión en diversas tareas relacionadas con el procesamiento de texto [17].

Lematización y Stemming

La lematización y el *stemming* son dos técnicas utilizadas para reducir las palabras a su forma base o raíz. Ambas buscan simplificar las palabras para normalizar el texto y facilitar su procesamiento y análisis.

La lematización consiste en encontrar el lema o forma base de una palabra en función de su significado y contexto, mientras que *stemming* se basa en técnicas heurísticas para recortar palabras y eliminar sufijos y prefijos, reduciéndolas a su forma raíz. Ambas técnicas permiten mejorar la eficiencia y precisión en diversas tareas de procesamiento de texto [18].

Modelado de Tópicos

El modelado de tópicos (*Topic Modeling*) es una técnica que tiene como objetivo identificar los temas principales que están presentes en un conjunto de documentos. Para ello se utilizan algoritmos de aprendizaje automático y técnicas estadísticas que asignan probabilidades a las palabras y documentos relacionados con cada tema. Se puede considerar que cada documento contiene una mezcla de varios tópicos, y que cada palabra en el documento se asigna a un tópico con cierta probabilidad. De esta forma se revela la estructura subyacente del documento. Dependiendo del número de tópicos diferentes que contenga un documento, y la variabilidad de los mismos, la precisión del algoritmo podría ser mayor o menor.

Un enfoque común de modelado de tópicos es el modelo de LDA (*Latent Direchlet Allocation*). En 2003 David Blei, Andrew Ng y Michael Jordan propusieron este algoritmo de aprendizaje automático no supervisado utilizado para el modelado de tópicos en un conjunto de documentos [19]. Su objetivo principal es descubrir tópicos latentes ocultos en los documentos y asignar probabilidades a las palabras asociadas con cada tópico. Esta técnica se ha convertido en una herramienta poderosa en el procesamiento y análisis de grandes volúmenes de texto. Su proceso básico de ejecución se podría resumir de la siguiente forma:

1. Inicialización: se seleccionan el número de tópicos y se le asigna aleatoriamente palabras a cada tópico en los documentos.

2. Muestreo de palabras: cada palabra es asignada a un tópico al que pertenece con cierta probabilidad, en base a las probabilidades de tópicos y las distribuciones de palabras en estos.

3. Muestreo de tópicos: se ajusta la distribución de cada tópico en base a las palabras que tenga asignadas.

4. Iteración: se repiten los pasos 2 y 3 varias veces con el fin de que la distribución converja a una más estable.

3.1.2 *Machine Learning*

El *Machine Learning*, o aprendizaje máquina, es una rama de la inteligencia artificial que ha transformado la forma en la que abordamos problemas complejos, permitiendo a las máquinas aprender y mejorar su rendimiento a través de la experiencia.

Históricamente, el término *"Machine Learning"* fue acuñado por Arthur Samuel en 1959, quien lo definió como el campo de estudio que permite a las máquinas aprender a realizar tareas sin ser programadas explícitamente [20]. En sus primeras etapas, estaba enfocado en algoritmos de regresión y clasificación lineal, y su aplicación estaba limitada por la capacidad computacional y la falta de datos masivos. Según fueron pasando las décadas, se fueron realizando numerosas investigaciones que podemos dividir en tres etapas temporales:

1. **Edad del Razonamiento**: durante esta temprana etapa comprendida entre 1950 y 1980, se realizaron importantes avances en el campo con la creación de algoritmos de regresión y clasificación lineal.

2. **Edad del Conocimiento**: en esta etapa situada entre 1980 y 2010, el *Machine Learning* experimentó un progreso significativo en términos de teoría y algoritmos. Se desarrollaron técnicas más avanzadas, como máquinas de soporte vectorial (*SVM*), árboles de

decisión, redes neuronales y métodos de *boosting* y *bagging*. También surgieron nuevos campos como el aprendizaje profundo (*Deep Learning*) y el aprendizaje por refuerzo (*Reinforcement Learning*). Durante este período, tanto el aumento de la capacidad de computación, como la disponibilidad de grandes conjuntos de datos, permitieron la aplicación práctica de dichos algoritmos.

3. ***Deep Learning* y *Big Data***: esta fase se ve marcada por un notable avance exponencial, comprendida entre 2010 y la actualidad. Se caracteriza por el impacto revolucionario que tuvieron en diversas áreas los algoritmos de aprendizaje profundo. El uso del *Big Data* junto con el poder de cómputo, han permitido la creación de modelos más profundos y complejos que superan ampliamente los enfoques tradicionales. También se caracteriza por una integración mayor de las técnicas de *Machine Learning* en aplicaciones y servicios cotidianos, ganándose así una gran relevancia en la sociedad actual.

Desafortunadamente, no todos los algoritmos sirven para resolver cualquier tipo de problema, sino que este tendrá que ser escogido según la distribución de los datos, el método de aprendizaje y el objetivo a cumplir. En base al tipo de aprendizaje que pueden seguir, podemos clasificarlos en las siguientes categorías [6]:

- **Aprendizaje Supervisado**: el algoritmo se entrena utilizando un conjunto de datos etiquetado. Por lo tanto, son datos sobre los que se conocen las respuestas o salidas deseadas. El objetivo del algoritmo consiste en aprender a seleccionar las salidas correctas en base a las entradas recibidas. Una vez entrenado el modelo, este podría empezar a realizar predicciones sobre nuevos datos. Algunos de los algoritmos incluidos en esta categoría serían regresión logística, SVM, árboles de decisión…

- **Aprendizaje No Supervisado**: en este caso, el algoritmo se entrena utilizando un conjunto de datos no etiquetado, por lo que no se conocen las salidas deseadas. El objetivo del algoritmo es encontrar patrones y estructuras ocultas en los datos. Una vez entrenado, podría realizar predicciones sobre nuevos datos en base a lo aprendido, retroalimentándose para predicciones futuras. Dentro de esta categoría destacan algoritmos como análisis de componentes principales (PCA), modelado de tópicos (LDA) o *K-Means Clustering*.

- **Aprendizaje Semisupervisado**: este enfoque utiliza una combinación de datos etiquetados y no etiquetados para entrenar al algoritmo. Es comúnmente utilizado cuando la combinación de ambos tipos de datos puede mejorar el rendimiento del modelo, especialmente cuando hay escasez de datos etiquetados. En esta grupo entrarían algoritmos como *Label Propagation*, *Co-Training*, aprendizaje semisupervisado basado en grafos (*Graph-bsed Semi-Supervised Learning*)…

A su vez, también los podemos clasificar en base al objetivo a cumplir, como podemos observar en la Figura 3.2:

- **Clasificación**: se tratan de algoritmos utilizados para asignar datos a categorías o clases discretas. La función objetivo es aprender una regla de decisión que permita clasificar los nuevos datos en las clases predefinidas. Son comúnmente utilizados SVM, regresión logística, *K-Nearest Neighbors*...
- **Agrupamiento** (*Clustering*): se enfocan en agrupar datos similares en grupos o *clusters* sin llegar a etiquetarlos. La función objetivo consiste en encontrar estructuras subyacentes y patrones en los datos, siendo estos agrupados en base a su similitud. Ejemplos de este tipo de algoritmos son *k-means*, *Mean Shift*...
- **Reducción de la dimensionalidad**: estos algoritmos son utilizados para reducir la cantidad de características en un conjunto de datos mientras se mantiene la mayor cantidad posible de información relevante. La función objetivo es simplificar los datos para mejorar la eficiencia del análisis. Entre los extensamente utilizados se encuentran el PCA, *Recursive Feature Elimination with Cross-Validation* (RFECV), *autoencoders*...
- **Regresión**: útiles cuando se busca predecir valores continuos o numéricos. La función objetivo es aprender una relación entre las características de entrada y la variable de salida continua. Destacan regresión lineal, regresión polinómica y regresión con árboles de decisión entre otros.

Como indicamos anteriormente, en este trabajo partimos de un dataset que contiene una gran cantidad de dimensiones y entradas, por lo que decidimos hacer diversas pruebas con los algoritmos RFECV y PCA para reducir su dimensionalidad.

En relación a los modelos, dado que estamos abordando una problemática de clasificación binaria, hemos empleado varios algoritmos de clasificación conocidos por su sólido rendimiento en conjuntos de datos similares. Además, algunos de estos modelos nos permiten comparar sus resultados con el conjunto de datos original, el cual utilizaremos como punto de referencia [21]. A continuación, entraremos más en detalle sobre los algoritmos empleados.

3.1.3 PCA

Análisis de Componentes Principales o PCA (por sus siglas en inglés, *Principal Component Analysis*) es una técnica ampliamente utilizada en el campo de la reducción de dimensionalidad. Esto lo consigue al transformar el conjunto de datos en un espacio de menor dimensión, a la vez que retiene la mayor cantidad posible de información relevante presente en los datos originales. Las nuevas dimensiones generadas se llaman componentes principales, y son

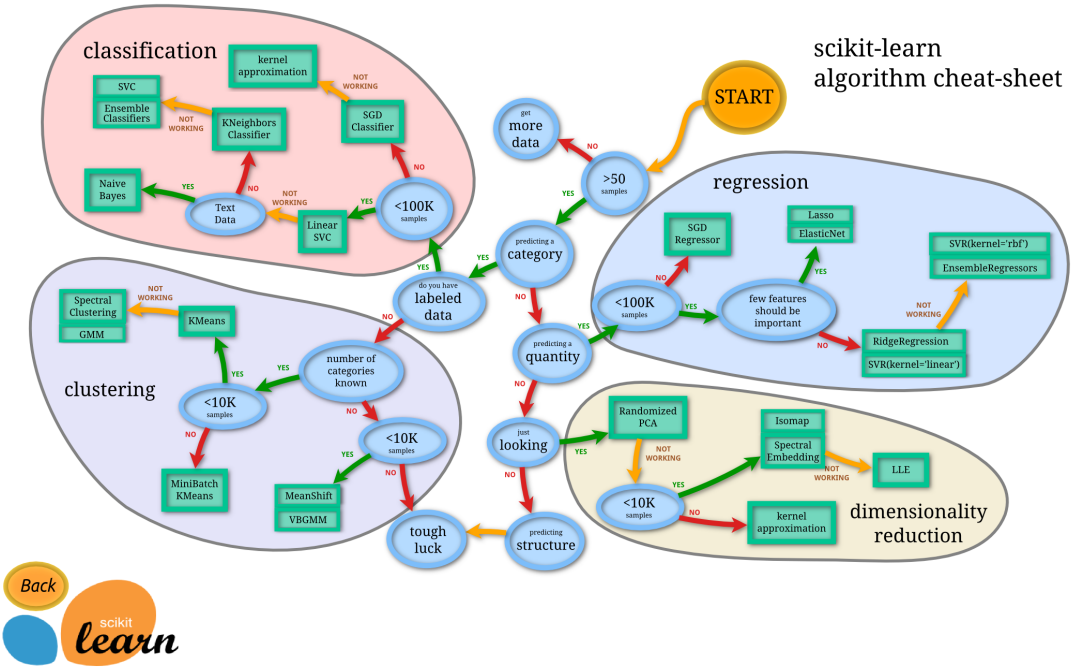

Figura 3.2: Diagrama de algoritmos de *Machine Learning*.

combinaciones lineales de las variables originales que explican la mayor parte de la varianza en los datos [22].

Resulta muy útil a la hora de trabajar con datos de alta dimensionalidad dada su eficiencia y versatilidad a la hora de proporcionar una representación más simple y significativa de estos. Además, también revela su estructura interna. Como ya se indicó en el apartado anterior, este algoritmo se utilizará reducir la dimensionalidad del *dataset* y seleccionar las características más relevantes para la fase de modelado.

3.1.4 Naïve Bayes

Naïve Bayes es un algoritmo de clasificación supervisada basado en el teorema de Bayes. Se trata de una herramienta matemática que calcula la probabilidad de que una instancia pertenezca a una clase determinada dado el conjunto de características observadas. Parte de la ”ingenua” suposición de que todas las características son independientes entre sí, ya que se conoce la clase objetivo. Aunque dicha presunción no sea cierta, se emplea debido a que simplifica el cálculo de las probabilidades, haciéndolo más sencillo y eficiente.

Este algoritmo es ampliamente empleado y ha demostrado ser efectivo en diversas tareas de clasificación. Es especialmente útil cuando se trabaja con conjuntos de datos de alta dimen-

sionalidad y texto dado su rendimiento. Además, como solo se requieren las varianzas de cada característica, no necesita un número de datos de entrenamiento elevado en comparación con otros algoritmos más complejos (SVM, Redes Neuronales Profundas...) [23].

3.1.5 Regresión Logística

La Regresión Logística es un método estadístico ampliamente utilizado, tanto en el campo del *Machine Learning*, como en la estadística. Se utiliza para modelar la relación entre una variable dependiente y una o más variables independientes. Su nombre proviene de su relación con la regresión lineal, ya que realmente se ha implementado en tareas de regresión en el sentido tradicional, si no que es comúnmente utilizado para la resolución de tareas de clasificación.

El objetivo de la Regresión Logística es predecir la probabilidad de que una observación pertenezca a una categoría específica. Usa la Función Sigmoide, que podemos apreciar en la Figura 3.3, para transformar una combinación lineal de las variables independientes en un valor entre 0 y 1, representando así su probabilidad estimada [24].

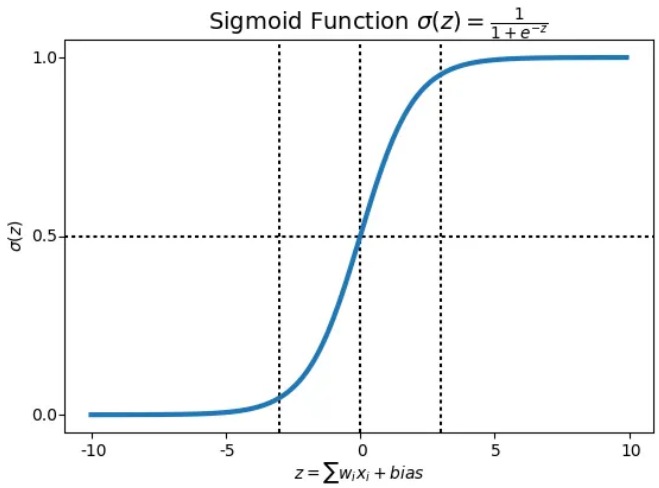

Figura 3.3: Función sigmoide.

Aunque se trate de una técnica lineal, su función de transformación permite modelar relaciones no lineales entre las variables independientes y la variable dependiente. Este algoritmo es comúnmente utilizado debido a su rápida convergencia y los buenos resultados que obtiene en cuanto a la resolución de problemas de clasificación binaria.

3.1.6 *Random Forest*

Random Forest es un algoritmo de aprendizaje supervisado utilizado tanto en tareas de clasificación como de regresión. Su concepto fundamental es la idea de *ensemble*, ya que crea una colección de árboles de decisión (bosque), siendo cada uno entrenado con una muestra y selección de características aleatorias. Cada árbol tendrá influencia en la decisión final, siendo en el caso de clasificación determinada mediante una ”votación”, como se puede observar en la Figura 3.4. Cabe destacar que el depender de un conjunto de árboles en vez de uno solo mejora notablemente la precisión y generalización del modelo.

La aleatoriedad a la que se enfrenta este algoritmo introduce la suficiente diversidad para que sea más sencillo capturar diferentes patrones y relaciones en los datos, evitando posibles sobreajustes (*overfitting*) y mejorando la robustez del modelo. No obstante, hay que tener especial cuidado durante el proceso de *hiperparametrización*, ya que dependiendo de los valores escogidos se puede caer igualmente en este sobreajuste, perdiendo así la capacidad de generalización del modelo [25].

Random Forest es ampliamente reconocido por su capacidad de manejar conjuntos de datos de alta dimensión y características correlacionadas. Su eficiencia, tolerancia a datos ruidosos o *outliers*, y su alta precisión lo hacen un algoritmo altamente recomendado para resolver tareas de clasificación.

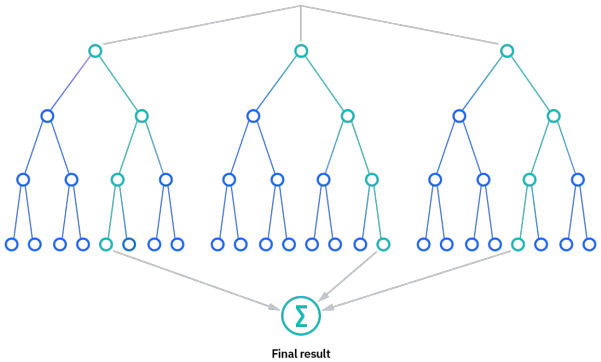

Figura 3.4: Random forest.

3.1.7 *AdaBoost*

Acrónimo de *Adaptative Boosting*, pertenece a la categoría de ensamblado de modelo. Fue diseñado para mejorar la precisión y el rendimiento de modelos de aprendizaje considerados como ”débiles”, los cuales presentan una precisión basada en la aleatoriedad. Fue presentado

en 1997 por Yoav Freund y Robert Schapire [26]. Actualmente es extensamente utilizado en diversas aplicaciones de clasificación y regresión.

Su principio fundamental es combinar varios modelos ”débiles” en un modelo más completo que obtiene una mayor precisión de predicción. Cada modelo ”débil” es entrenado secuencialmente y se le asigna un peso en base a su rendimiento. Durante el entrenamiento, se le otorga más peso a las instancias mal clasificadas para que los siguientes modelos intenten corregir los errores cometidos. Finalmente, los modelos ”débiles” que tengan un mayor rendimiento en las instancias mal clasificadas tendrán una mayor influencia en el modelo final, siendo este una combinación ponderada de estos [27].

3.1.8 *LinearSVC*

Linear Support Vector Classification pertenece a la familia de *Support Vector Machines* (SVMs). Es un algoritmo de aprendizaje automático utilizado para la clasificación de datos en problemas de aprendizaje supervisado.

Su objetivo principal es encontrar el hiperplano de separación que maximiza el margen entre las clases, separándolas de esta forma y permitiendo la clasificación de las nuevas instancias en función del lado del hiperplano en el que se encuentren.

Aunque su nombre sugiere que solo funciona en problemas lineales, en realidad, también es utilizado con funciones *kernel* que permiten mapear los datos a un espacio de características de mayor dimensión, haciendo que sean linealmente separables y permitiendo al algoritmo abordar problemas no lineales [28].

3.2 Evaluación la situación

Como ya se comentó anteriormente en el Capítulo 1, el *cyberbullying* es un campo relevante que afecta a nuestra sociedad valiéndose de diferentes herramientas, siendo las redes sociales una de las más utilizadas [4]. Afortunadamente, también es un campo relevante de investigación, lo que nos proporciona una gran cantidad de información al respecto.

El *dataset* escogido contiene muestras provenientes de comentarios de la conocida red social *Instagram*. Estas están compuestas por una selección de datos del *dataset* original y métricas generadas a partir de técnicas de Procesamiento del Lenguaje. Sobre el conjunto de muestras original ya se han realizado diferentes estudios de detección [21][29], los cuales nos servirán como puntos de referencia para contrastar los resultados obtenidos.

3.2.1 Tecnologías empleadas

En cuanto a las tecnologías empleadas, hemos sopesado distintas alternativas comúnmente utilizadas en el ámbito de la Ciencia de Datos y el *Machine Learning*. Finalmente, dentro de las escogidas, caben destacar Anaconda, *scikit-learn*, *Jupyter Notebook*, *Pandas*, *seaborn* y *matplotlib*, entre otras [6]. A continuación, profundizaremos un poco en las más relevantes.

Anaconda
Anaconda es una plataforma de distribución y administración de software diseñada para Python y enfocada a la ciencia de datos. Su capacidad de creación de entornos virtuales independientes y su administrador de paquetes la hacen especialmente útil para trabajar diversos entornos. En un contexto científico, su empleo garantiza una configuración estable y eficiente. Por ende, nos pareció una elección ideal para emplear como base de nuestro código.

scikit-learn
Scikit-learn, también conocido como *sklearn*, es una biblioteca de aprendizaje automático de código abierto desarrollada para Python. Ofrece una amplia gama de herramientas y algoritmos de aprendizaje automático para diversas tareas, entre las que se encuentran clasificación, regresión, agrupación, reducción de dimensionalidad, selección de características…. Además de sus incontables funciones y opciones, también destaca por su facilidad de uso e integración con el ecosistema de Python.

La documentación que ofrece es muy completa, y dada su popularidad, es sencillo encontrar multitud de ejemplos e implementaciones para usar como referencia a la hora de construir el código. En resumen, sus numerosas virtudes la convierten en una herramienta esencial para cualquier proyecto que involucre análisis y modelado de datos.

Jupyter Notebook
Es un entorno interactivo basado en la web que permite crear y compartir documentos que combinan código, visualizaciones y explicaciones en un único archivo. Permite integrar fragmentos en lenguajes como *Python*, R y Julia, junto con los resultados obtenidos, ya sea en forma de gráficos, tablas o texto. Los cuadernos también se pueden exportar en diferentes formatos (PDF, HTML…), facilitando así la creación de informes y su distribución. En nuestro caso, también nos ayudamos del IDE *Visual Studio Code* (*VS Code*) para conectarnos al servidor *Jupyter* y desarrollar el código.

3.3 Determinación de las metas a alcanzar

Para cumplir con los objetivos indicados anteriormente en la Sección 1.2, en este proyecto de minería de datos tenemos que completar las siguientes tareas:

- Realizar una tarea de clasificación binaria a partir del análisis de los datos obtenidos de la red social *Instagram*. Cada comentario será clasificado entre dos categorías, ciberacoso o normal.
- Seguir la metodología *CRISP-DM*, ya que engloba prácticamente la totalidad de los objetivos que debemos cumplir. Por lo tanto, su uso es objetivo y requisito para la correcta realización de este proyecto.

3.4 Producción de un plan de proyecto

El primer paso para producir un plan de proyecto es establecer la metodología a seguir. Como ya se ha indicado en el Capítulo 2, se trata de CRISP-DM. En este caso, sirvió más como una guía que como un conjunto estricto de instrucciones a seguir, ya que la fase de despliegue no se ha considerado en la ejecución del proyecto y debido a limitaciones temporales hemos omitido ciertos protocolos formales.

En cuanto a la asunción de roles, a los directores del proyecto Diego Fernández Iglesias y Patricia Martín Rodilla se les concedió el rol de supervisores, mientras que al autor Andrés Abal Aldao desempeña el rol de analista desarrollador. Seguidamente, detallaremos las fases del proceso de desarrollo y analizaremos cómo se han abordado a lo largo de la evolución del proyecto.

3.4.1 Comienzo

El primer paso fue tener una reunión introductoria donde se aclararon los objetivos marcados, las tecnologías que se utilizarían, cuáles serían los siguientes pasos a seguir y una breve presentación detallada del ciclo de vida de CRISP-DM. También se acordó programar las subsiguientes reuniones en intervalos regulares de aproximadamente una o dos semanas, dependiendo de la carga de trabajo establecida o de posibles problemas encontrados. El propósito de estas reuniones era examinar los progresos alcanzados y estudiar la realización de los siguientes pasos.

La primera tarea establecida fue coger confianza con el entorno de desarrollo y aprender lo básico sobre *Machine Learning*. Se estimó un plazo de 2 semanas para su realización.

3.4.2 Planificación y seguimiento

Con el objetivo de mejorar la comprensión del diseño del proyecto, hemos adoptado una nomenclatura coincidente entre cada una de las etapas del ciclo de vida de CRISP-DM y las fases de nuestro propio proyecto. Esto nos permite tener una referencia clara del estado del proyecto y abordar de manera efectiva cualquier dificultad que pueda surgir. Es importante destacar que cada fase está condicionada por la finalización de la fase anterior. Además, es relevante mencionar que la elaboración de la memoria se llevó a cabo de manera simultánea durante la segunda mitad del desarrollo del proyecto.

La línea base estimada se puede consultar en el *Diagrama de Gantt* mostrado en la figura 3.5. Mientras que en la tabla 3.1 se puede apreciar con más detalle la comparativa entre el tiempo estimado para cada tarea y el tiempo real que llevó su resolución. En ella podemos observar que hubo un destacable retraso en la fase de preparación de los datos, ya que no teníamos la suficiente potencia computacional para trabajar con el conjunto. La solución que elegimos fue abordar la reducción de la dimensionalidad mediante técnicas que serán detalladas en profundidad en el Capítulo 5, mientras incrementábamos la capacidad computacional para poder llevar a cabo los experimentos.

Afortunadamente, la anticipación en la planificación de todas las etapas del proyecto y la implementación de un seguimiento continuo, nos facilitó la comprensión de la duración de cada fase y no brindó una visión más clara de los obstáculos que podrían emerger durante su desarrollo.

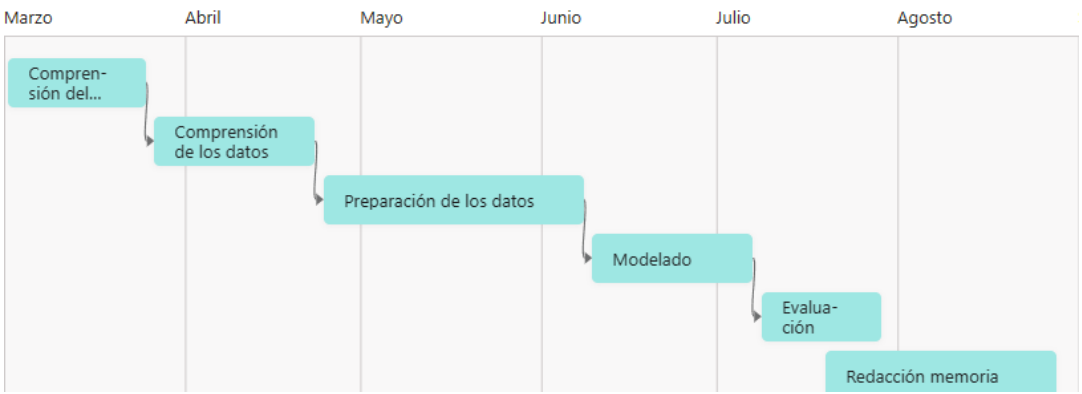

Figura 3.5: Diagrama de Gantt del seguimiento general estimado.

Tarea	Estimado (semanas)	real
Comprensión del negocio	2	2
Familiarizarse con los datos	1	1
Repasar conceptos de Machine Learning	1	1
Comprensión de los datos	2	2
Recolectar y explorar los datos	2	2
Preparación de los datos	10	12
Limpieza, detección anomalías y formateo	5	4
Reducción dimensionalidad	5	8
Modelado	4	4
Selección modelos	1	1
Diseño del entrenamiento	2	2
Hiperparametrización	1	1
Evaluación	3	2
Comparativa entre modelos	2	1
Revisión y líneas futuras	1	1
Redacción de la memoria	5	7
Revisiones (se estiman cada dos semanas)	12	9
Horas totales	272	299

Tabla 3.1: Seguimiento del proyecto.

3.4.3 Estimación de costes

Hemos calculado el costo estimado de la realización de este proyecto teniendo en cuenta el tipo de trabajador, las horas trabajadas y la correspondiente tarifa por hora.

En nuestro equipo, contamos con tres trabajadores: un analista desarrollador y dos supervisores. Para determinar la remuneración de cada tipo de trabajador, hemos tomado como referencia las tarifas establecidas en el BOE (26 de julio de 2023) [30], que fija los salarios a partir del 1 de julio de 2022 en base a un total de 1800 horas de trabajo anuales. Por lo tanto, siguiendo el salario promedio para cada puesto, la tarifa por hora para el analista desarrollador es de 15€/h, mientras que para los supervisores es de 16€/h. Al considerar la cantidad total

de esfuerzo invertido, podemos calcular los valores que se muestran en la tabla 3.2.

Puesto	Esfuerzo (h)	Coste (€)
Analista desarrollador	299	4.485
Supervisor 1	60	960
Supervisor 2	60	960
Total	419	6.405

Tabla 3.2: Coste estimado de los recursos.

En lo que respecta a los recursos materiales empleados, únicamente consideraremos el equipo utilizado para las pruebas computacionales. En nuestro caso, usamos una máquina proporcionada por la Universidad de A Coruña. Entonces, podemos tomar como referencia el coste del servicio de un proveedor como OVH, ya que su opción Élite es muy similar a la máquina empleada y sus precios se encuentran en línea con el promedio actual del mercado. Por lo tanto, partiríamos de un coste de 31,74€/mes , y dado que la duración del proyecto es de 6 meses, esto representaría un gasto de 190,44€ en total. Teniendo en cuenta todos los gastos, podemos visualizar el costo total estimado de la ejecución de este proyecto en la Tabla 3.3

Tipo	Coste (€)
Recurso	6.405
Material	190,4
Total	6.514,4

Tabla 3.3: Coste total estimado del proyecto.

Es importante señalar que en una situación práctica, sería necesario considerar costos adicionales relacionados con recursos indirectos como el consumo eléctrico, suministro de agua, y conectividad a Internet, entre otros.

Capítulo 4

Comprensión de los Datos

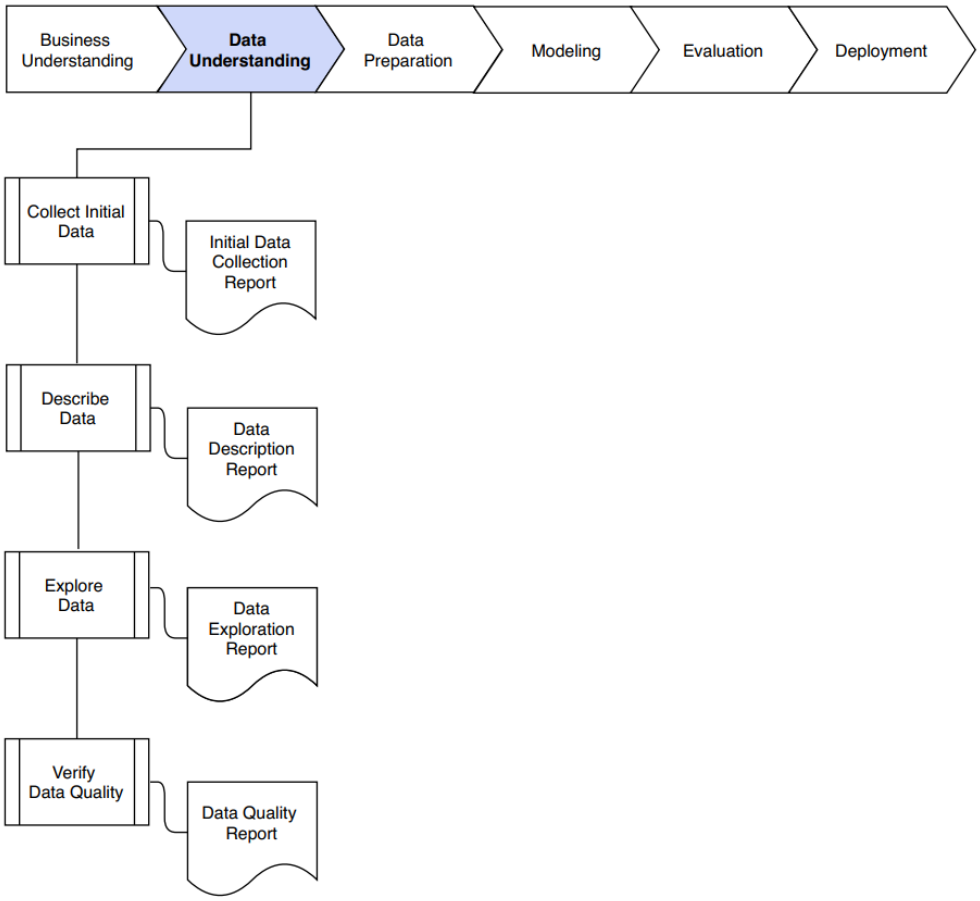

Figura 4.1: Fase CRISP-DM: Comprensión de los datos.

Durante este capítulo veremos la fase de comprensión de datos, la cual es crucial para explorar y comprender la naturaleza, calidad y posibles problemas de los datos disponibles. Empezaremos documentando los datos y realizando un análisis inicial ayudándonos

de técnicas estadísticas y visuales. Luego evaluaremos la calidad de los datos, haremos una limpieza de valores erróneos o faltantes, detectaremos posibles valores atípicos y verificaremos su integridad y consistencia. Gracias a esto, tendremos una base sólida que nos servirá de ayuda en los análisis posteriores y en futuras tomas de decisiones, consiguiendo así optimizar los resultados maximizando su valor y utilidad.

Como podemos apreciar en la Figura 4.5, esta fase se dividirá en distintas etapas: recolección inicial de los datos, descripción de los datos, exploración de los datos y verificación de la calidad.

4.1 Recolección inicial de los datos

La elaboración de un conjunto de datos haciendo técnicas de procesamiento del Lenguaje (LDP) quedó fuera de los objetivos del proyecto debido a la falta de tiempo. Por lo tanto, optamos por utilizar uno ya creado con anterioridad.

El *dataset* se ha obtenido de forma serializada en un archivo con formato *PKL*. Este fue importado como un *dataframe* haciendo uso de la librería *Pandas* con el objetivo de facilitar las siguientes fases de descripción y exploración.

4.2 Descripción de los datos

El dataset está compuesto por un total de 122234 entradas. Cada una se trata de un comentario dentro de una sesión en la red social Instagram. Los datos incorporan diversas propiedades organizadas en columnas, resultando 447 atributos en total. Todas son numéricas y la mayor parte son de punto flotante, mientras que el resto se presentan como valores enteros.

Cabe destacar que el conjunto de datos con el que trabajamos se trata de una simplificación de otro *dataset* más grande, el cual fue creado utilizando un método de muestreo en ”bola de nieve”. Este proceso de recolección involucró una selección inicial de 25 mil usuarios con perfiles públicos, lo que resultó en un conjunto que abarca más de 3 millones de *media_sessions*. Con el objetivo de lograr un conjunto de datos más equilibrado, se excluyeron sesiones con menos de 15 comentarios y solo se retuvieron aquellas sesiones que contenían al menos una palabra obscena en los comentarios. Este proceso culminó en un conjunto de datos compuesto por 2.218 *media_sesions*, las cuales fueron etiquetadas (etiquetado a nivel de sesión) por colaboradores humanos siguiendo una rigurosa definición de ciberacoso [31]. Esto se logró a través de una variante del método de votación mayoritaria, en la que a cada

colaborador se le asignó un nivel de confianza basado en diversas pruebas y cuestionarios, y se requirió un umbral mínimo del 60% para que una sesión se mantuviera en el conjunto de datos. Cada sesión en este conjunto de datos incluye su propio conjunto asociado de comentarios. Por último, también se incluyen una serie de atributos LDA que recopilan los diez temas principales extraídos de todos los comentarios [21].

Sobre esto, también se añadieron tres tipos de características adicionales con el fin de mejorar los resultados obtenidos: el análisis de cada comentario para calcular su *TF-IDF* respecto a otros comentarios, un conjunto de atributos *doc2vec* y otro de características que miden la diferencia de tiempo entre comentarios en estos casos:

- Diferencia de tiempo con el último comentario.
- Diferencia de tiempo desde todos los comentarios anteriores.

En ambas casuísticas, los valores se agregan calculando el promedio, mediana, máximo y mínimo, siendo incluidos como nuevas características para el comentario procesado.

Entre todos los atributos que han sido incorporados, merece la pena resaltar:

- **label**: se trata de la variable objetivo del dataset y determina si un comentario es ciberacoso o no.
- ***_profane_words**: es el número de palabras malsonantes destro de una sesión o comentario.
- ***_polarity y *_subjectivity**: indican la positividad y subjetividad obtenidas a través de un proceso de *Sentiment Analysis*.
- ***_tfidf_***: calcula la relevancia o importancia de cada palabra en un comentario o sesión en comparación con otros elementos similares. Para ello, se utiliza un enfoque de procesamiento de lenguaje natural conocido como *Bag of Words*.
- ***_lda_***: indica la relación de un comentario o sesión respecto a ciertos temas o tópicos.
- ***_doc2vec_***: consiste en una forma de representar los comentarios o sesiones en forma de vectores, lo que permite capturar las conexiones cercanas entre diversos documentos utilizando palabras clave.

También hemos hecho una agrupación con las que consideramos que tendrán una relevancia destacada:

- **Discretas con pocos valores**: *session_id*, *this_comment_n_words*.

- **Discretas con un amplio rango definido y/o que contengan valores negativos**: *this_comment_polarity*, *this_comment_subjectivity*, *this_session_polarity* y *this_session_subjectivity*.
- **Continuas**: relacionadas con *lda*, *tfidf* y *doc2vec*.

En las figuras 4.2, 4.3 y 4.4 se presentan ejemplos que ilustran las distribuciones de estas variables.

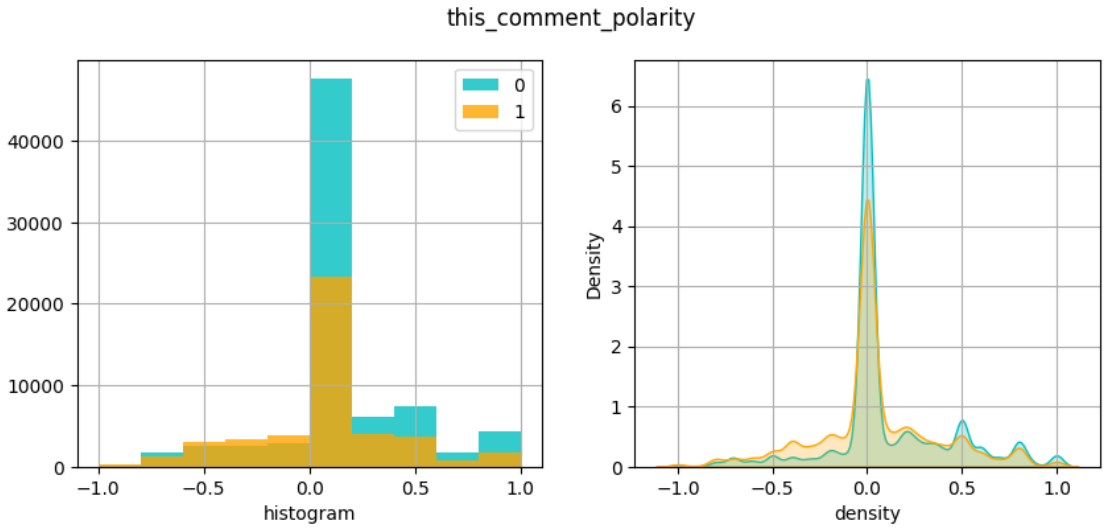

Figura 4.2: Distribución de *this_comment_polarity* respecto a *label*.

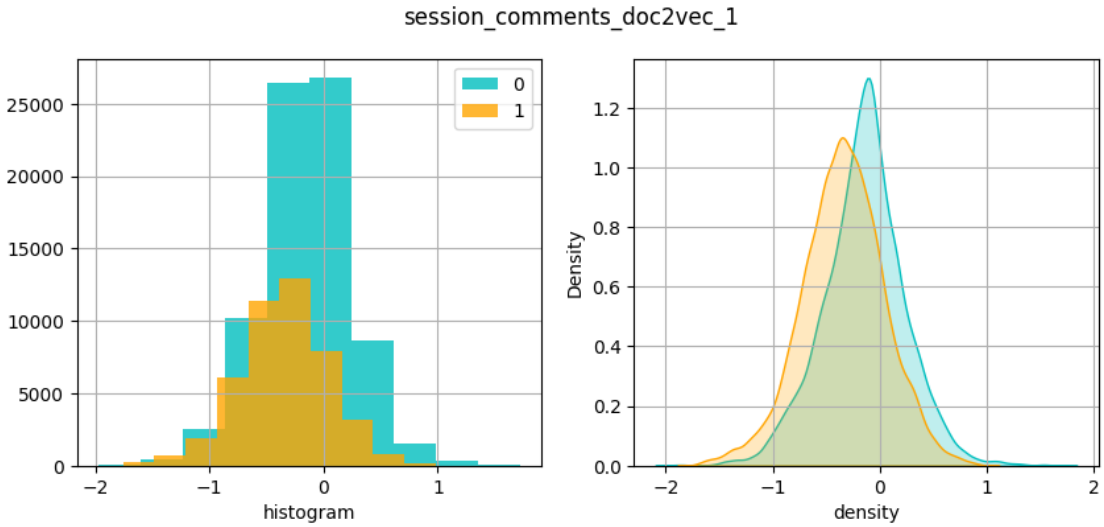

Figura 4.3: Distribución de *session_comments_doc2vec_1* respecto a *label*.

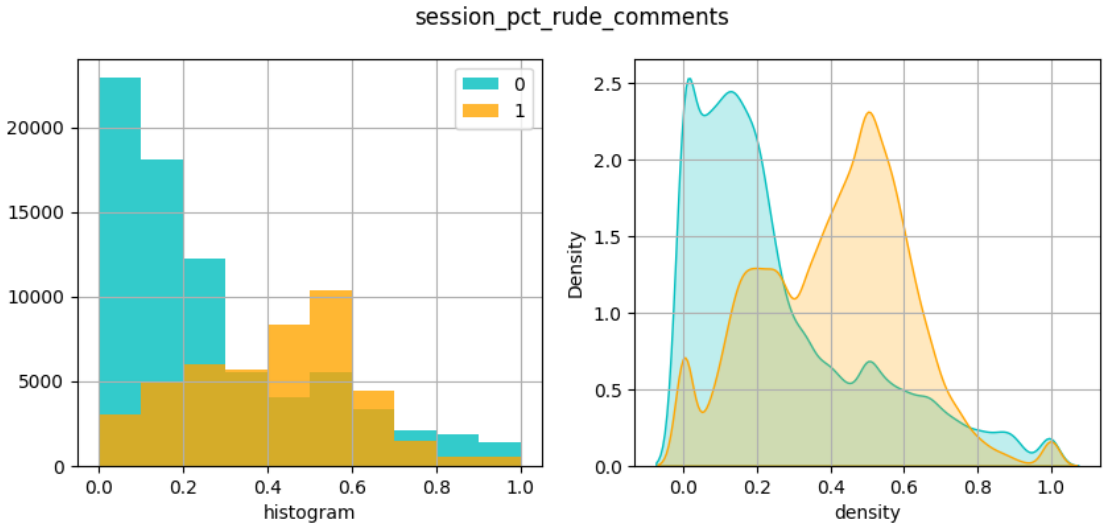

Figura 4.4: Distribución de *session_pct_rude_comments* respecto a *label*.

En cuanto a la variable objetivo *label*, esta es de naturaleza binaria, presentando valores 1 o 0 para denotar si un comentario está asociado al ciberacoso o no. Como se puede apreciar en la Tabla 4.1, de un total de 122.234 entradas, 76.862 están clasificadas como comentarios normales, mientras que las 45.372 restantes han sido identificadas como casos de ciberacoso. Por lo tanto, como podemos observar en la Figura 4.5, contamos con un 62,88% de entradas etiquetadas como 0 y un 37,12% etiquetadas como 1.

En lo que respecta al número de sesiones, disponemos de un total de 1954, de las cuales 1369 (70.06%) se encuentran etiquetadas como normales, mientras que las 585 (29.94%) restantes están clasificadas como ciberacoso.

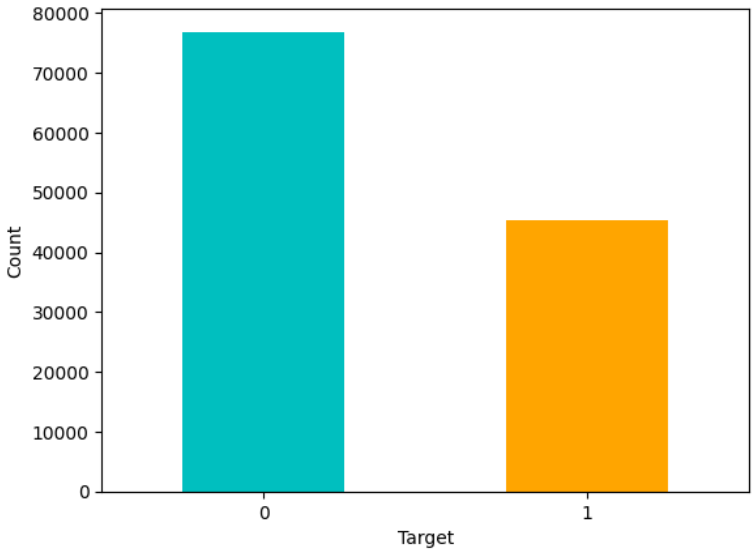

Figura 4.5: Comentarios normales vs comentarios de ciberacoso.

	Normal	Ciberacoso	Total
Comentarios	76862	45372	122234
	62,88%	37,12%	100%
Sesiones	1369	585	1954
	70,06%	29,94%	100%

Tabla 4.1: Distribución de la clase objetivo.

Debido al desequilibrio observado entre las clases y la alta dimensionalidad del conjunto de datos, hemos optado por abordar este desafío mediante técnicas de submuestreo, las cuales serán detalladas en el Capítulo 5.

A partir de este dataset al que a partir de ahora nos referiremos a él como *Baseline*, derivaremos otros tres con el objetivo de valorar el verdadero impacto de las características en las predicciones finales:

- *WithoutLDA*: se descartan las características relacionadas con LDA.
- *WithoutDoc2vec*: se eliminan los atributos relativos a *doc2vec*.
- *WithoutBoth*: en este caso probamos a descartar las características de LDA y *doc2vec*.

Estos datasets los contemplamos de forma independiente en lo que respecta a las fases

de preparación de los datos, modelado y evaluación. Mientras que los siguientes análisis que mostraremos a continuación, son hechos únicamente sobre el *baseline*.

4.2.1 *Correlaciones Lineales*

Explorar las diversas relaciones existentes entre los datos puede resultarnos beneficioso, ya que nos permite identificar posibles asociaciones de interés y, por consiguiente, realizar una primera evaluación de qué características pueden ser más relevantes a la hora de predecir el valor de la variable objetivo *label*.

Para poder realizar este análisis, utilizamos la matriz de correlaciones, ya que evalúa la linealidad entre cada par de características dentro de un conjunto de datos, ofreciendo valores en el rango [-1,1] que indican la presencia de correlación negativa o positiva respectivamente. Esto nos permite inferir la tendencia de una variable a partir de otra, sin tener en cuenta su dependencia.

Dado que no disponemos de un estándar de clasificación definido para los coeficientes de correlación, hemos adoptado los grupos habitualmente utilizados. A continuación, presentamos estos grupos junto con el número de características abarcado por cada uno:

- **Muy baja** (0,00 - 0,19): 323 características.
- **Baja** (0,20 - 0,39): 118 características.
- **Moderada** (0,40 - 0,59): 5 características.
- **Alta** (0,60 - 0,79): 0 características.
- **Muy Alta** (0,80 - 1,00): 1 característica.

Al observar los datos, notamos que la mayoría de las características muestran una correlación relativamente baja con respecto a la variable objetivo *target*. Si bien esto puede presentar un desafío al intentar predecir la variable objetivo, no es un factor determinante por sí solo. Los resultados finales dependerán en gran medida de los modelos específicos que utilicemos, los cuales detallaremos más adelante. En la figura 4.6 podemos ver cuáles son las 20 características que tienen un valor de correlación más alto respecto a *label*.

También llevamos a cabo un análisis de las correlaciones lineales entre todas las combinaciones de características. No obstante, nuestro enfoque principal se dirigió hacia las correlaciones que más involucración tienen con la variable objetivo. En la Figura 4.7 se muestra un mapa de calor donde se muestran las correlaciones entre las 10 características que mayor

relación tienen con *label*.

	Value
label	1.000000
session_comments_lda_topic_3	0.466862
session_comments_lda_topic_2	-0.459969
session_comments_doc2vec_38	-0.416939
session_pct_negative_comments	0.412476
this_comment_doc2vec_90	-0.410645
this_comment_doc2vec_71	-0.397261
this_comment_doc2vec_94	0.385296
this_comment_doc2vec_66	-0.382685
this_comment_doc2vec_86	-0.380114
session_comments_doc2vec_70	0.377353
session_n_negative_comments	0.377055
session_comments_tfidf_13	0.373313
session_comments_tfidf_31	0.366934
this_comment_doc2vec_89	0.365520
this_comment_doc2vec_33	0.365173
this_comment_doc2vec_72	-0.363357
session_comments_n_words	0.361256
session_comments_doc2vec_69	-0.361031
this_comment_doc2vec_92	-0.359613

Figura 4.6: Correlaciones más altas respecto a *label*.

Es importante destacar que para llevar a cabo la reducción de dimensionalidad implementamos el algoritmo PCA, tal como se detalla en el Capítulo 5. Este algoritmo tiene la capacidad de identificar y considerar estas correlaciones en su proceso de transformación.

Asimismo, también consideramos la posibilidad de abordar correlaciones no lineales. Sin embargo, debido a las restricciones de tiempo, decidimos no profundizar en este análisis y optamos por enfocarnos en el estudio de las correlaciones lineales.

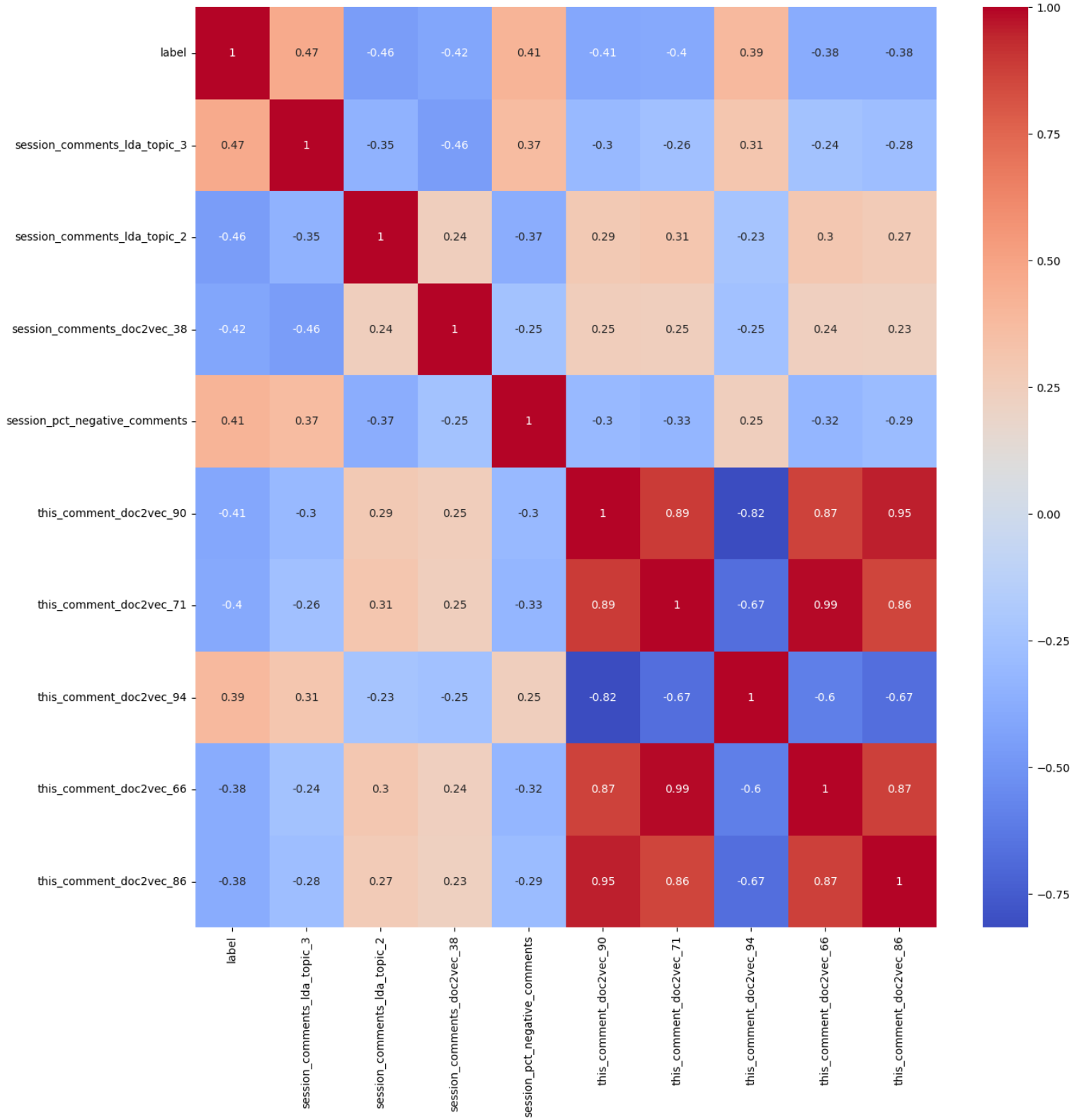

Figura 4.7: Mapa de calor de las 10 características que mayor correlación tienen con *label*.

4.2.2 *Feature Importance*

Para evaluar la contribución de cada característica en nuestro conjunto de datos al rendimiento global del modelo, hemos optado por utilizar el algoritmo *Random Forest*. Este algoritmo calcula automáticamente la importancia de las características durante su proceso de entrenamiento, lo que nos permite identificar de manera general cuáles variables son las más relevantes.

Concretamente, hemos utilizado la métrica conocida como Importancia de Gini. Esta métrica se emplea para medir cuánto mejora la pureza de las clasificaciones en un árbol de decisión cuando se usa una característica específica para dividir los nodos. Los atributos que más reducen la impureza Gini son considerados los más importantes [32].

En la Figura 4.8, se presentan las 25 características con mayor Importancia de Gini. Es evidente la notable diferencia entre las ocho características principales y las restantes. Cabe destacar que el atributo mejor valorado en términos de importancia es *session_id*. Sin embargo, un análisis más detenido de su distribución respecto a la variable objetivo reveló que esta característica destaca debido a una pecularidad particular. En el capítulo 5, profundizaremos en este aspecto y exploraremos en detalle las razones detrás de los resultados que esta característica arroja.

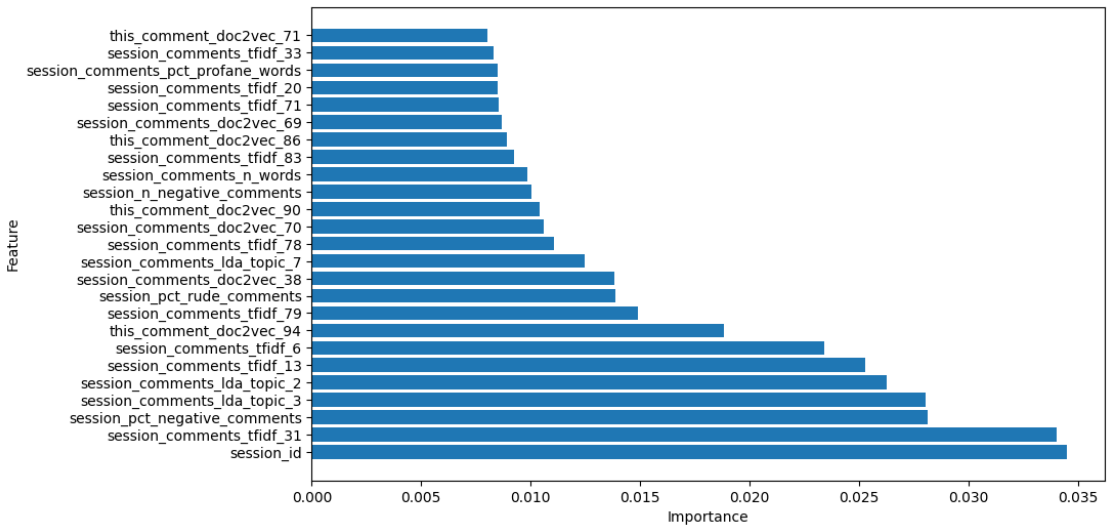

Figura 4.8: Top 25 *Feature importance*.

4.2.3 Verificación de la calidad

En esta fase se verifica la calidad de las entradas del conjunto de datos. El objetivo es detectar valores nulos o diferentes anomalías. Cabe destacar que el conjunto de datos ya ha sido procesado previamente y no debería presentar ni datos faltantes ni anomalías, pero de todas formas revisaremos las muestras por si se diera el caso.

Para la resolución de los valores nulos, utilizamos la función que nos ofrece *Pandas* llamada *dropna*, la cual elimina las filas o columnas que contienen valores nulos en un dataframe. Tras su uso pudimos verificar que no se eliminó ningún valor y que, por lo tanto, el dataset carecía de nulos.

Respecto a los valores anómalos, buscamos detectar la presencia de elementos negativos en columnas que no deberían presentarlos (*session_id*, *likes*...). En cuanto a las que su rango de medición se encuentra entre [-1, 1], confirmamos que efectivamente los valores presentados estaban dentro del rango definido. Por lo tanto, tampoco se tuvo que eliminar o modificar ningún elemento.

Capítulo 5

Preparación de los Datos

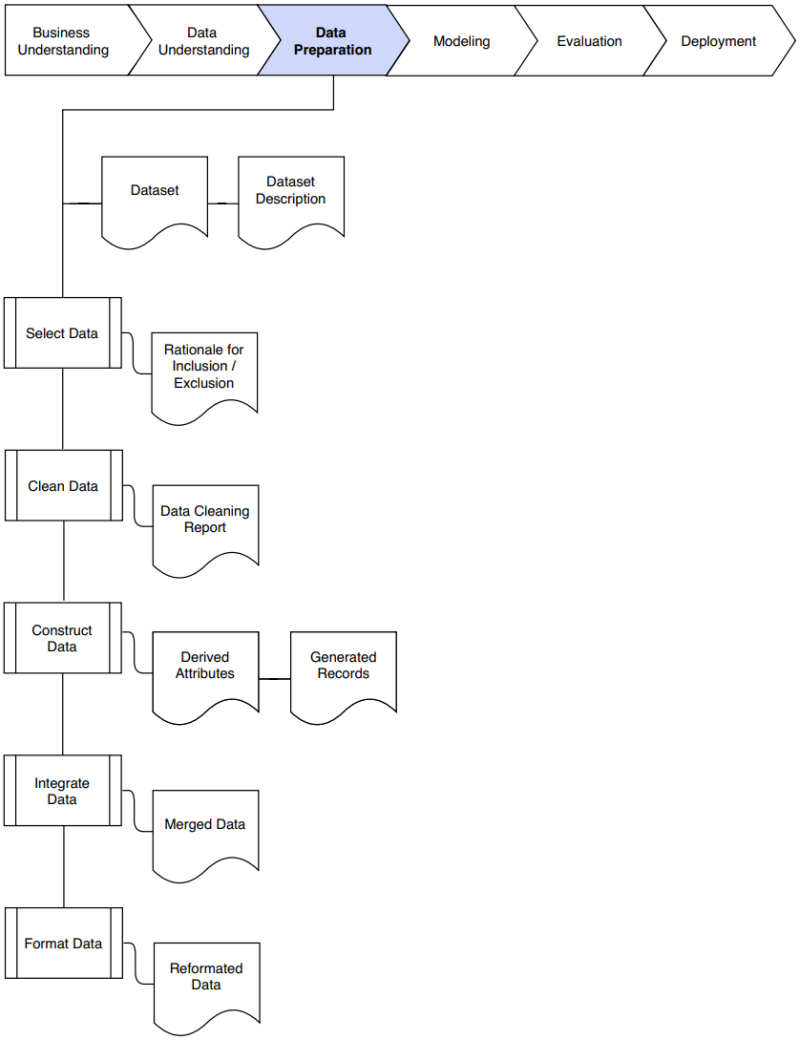

Figura 5.1: Fase CRISP-DM: Preparación de los datos.

La fase de preparación de los datos contiene diversas tareas que desempeñan un papel fundamental en el proceso de minería de datos. Entre ellas se encuentra la limpieza y mejora de la calidad de los datos, la selección de atributos relevantes, normalización de los datos, corrección de errores, división de los datos para entrenar/evaluar los modelos y quizás la generación de nuevas características derivadas a partir de otras.

5.1 Limpieza de los datos

La calidad de los datos influye directamente en el rendimiento de los modelos y en la confianza en los resultados. En esta etapa se corrigen errores, se eliminan valores inexactos y se manejan problemas como valores atípicos o datos faltantes.

5.1.1 Detección de anomalías

La detección de anomalías, también conocida como la detección de *outliers* o valores atípicos, es un proceso en el que se identifican observaciones en un conjunto de datos que se desvían significativamente del patrón esperado. Estos datos pueden ser valores inusuales, inesperados o atípicos que difieren de la mayoría de los datos.

Dado que la presencia de estos *outliers* disminuye la eficacia de los resultados presentados por los modelos de aprendizaje automático, es importante asegurarse de que estas observaciones no sean muy comunes para poder mantener la calidad de los datos. Sin embargo, esto no significa que no deba haber ninguno, ya que los modelos necesitan conocer un amplio rango de información para poder cubrir diferentes situaciones. En otras palabras, les permite hacer predicciones generalizadas en lugar de ajustarse demasiado a un conjunto particular de datos.

Hay varias formas o enfoques para descubrir anomalías en el conjunto de datos. Las más comunes son:

- **Métodos basados en la estadística**: se centran en el análisis de características estadísticas de los datos. Evalúan cómo los valores se desvían de las medidas típicas, como la media y la variabilidad. Algunos ejemplos son *Z-Score / Coeficiente de Variación*, Detección de Umbrales y el cálculo del Rango Intercuartílico (IQR).
- **Métodos basados en la densidad**: enfocados en cómo se distribuyen los puntos en el espacio. Examinan sus agrupaciones en términos de densidad para identificar cuáles están aislados o son poco frecuentes. Entre estos métodos se encuentran *Local Outlier Factor* (LOF) y *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN).

- **Métodos basados en el *Machine Learning***: este conjunto de métodos construye modelos para entender el patrón general de los datos. Fabrican una ”imagen” de lo que son los datos normales para poder detectar los que se desvían significativamente. Destacan *Isolation Forest*, *One-Class SVM* y los *Autoencoders* (Redes Neuronales).

En este trabajo decidimos utilizar las técnicas de IQR y *Isolation Forest*, probando así la detección de valores anómalos desde distintos enfoques.

IQR

El Rango Intercuartílico es una medida estadística que representa la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1) de un conjunto de datos. Se calcula como:

$$IQR = Q3 - Q1 \tag{5.1}$$

Resulta útil para identificar valores extremos que se encuentran por encima o por debajo de ciertos umbrales basados en el rango intercuartílico multiplicado por un factor k que determina la sensibilidad del método:

$$\begin{aligned} Lmite\_inferior &= Q1 - K * IQR \\ Lmite\_superior &= Q3 + K * IQRSA \end{aligned} \tag{5.2}$$

En nuestro caso, el k tendrá el valor 1.5, ya que en la distribución Gaussiana, aproximadamente el 99.7% de los datos se encuentran dentro de res desviaciones estándar (±3σ) del punto medio. Por lo tanto, al tomar este valor nos alineamos con esta regla empírica que nos permite detectar valores potencialmente atípicos [33].

Al aplicar esta técnica a nuestro conjunto de datos, la media de valores atípicos oscilaba entre el 6 y 10% del total. Estos valores pueden ser resultado de la naturaleza dispersa de nuestros datos, por lo que 1.5 quizás se trate de un rango demasiado agresivo. Con el fin de contrastar los resultados, escogimos la técnica de *Isolation Forest*.

Isolation Forest

Isolation Forest es un algoritmo de detección de anomalías basado en árboles de decisión que fue introducido por Liu et al. en el artículo ”*Isolation-based Anomaly Detection*” [34]. Se basa en la idea de que los valores atípicos son más fáciles de aislar que los valores normales en un conjunto de datos.

Funciona construyendo un conjunto de árboles de decisión de forma aleatoria, midiendo la cantidad de pasos necesarios para aislar cada punto de datos. Su ejecución consta de las siguientes fases:

1. **Selección de características aleatorias**: para cada árbol, el algoritmo selecciona una característica aleatoria de los datos y un valor de separación aleatorio dentro del rango de valores de ese atributo.
2. **Construcción del árbol**: se construye un árbol de decisión utilizando los elementos indicados anteriormente. Los datos son divididos en dos grupos en función de si son menores o mayores que el valor de separación.
3. **Repetición del proceso**: los dos primeros puntos se repiten hasta construir el número de árboles especificado. Cabe destacar que cada árbol opera de manera independiente.
4. **Medición del aislamiento**: para cada punto de datos se mide la profundidad necesaria para aislarlo. Cuanto menor sea dicha profundidad, más atípico se considera.
5. **Calculo e identificación de anomalías**: por último, se promedia el número de niveles necesarios para aislar un punto en todos los árboles, convirtiéndose este valor en una puntuación de anomalía. Los puntos con puntuaciones más bajas son considerados como *outliers*.

El algoritmo lo implementamos a partir de la librería de *scikit-learn* y utilizamos los siguientes parámetros:

- **random_state**: indica la semilla que se utilizará para la generación de números aleatorios en el proceso de construcción de los árboles. Establecimos un valor fijo con el fin de que esta construcción sea reproducible.
- **n_jobs**: este parámetro controla la cantidad de núcleos de CPU utilizados para paralelizar la construcción de los árboles. Aquí indicamos el valor ”-1” para que utilice todos los núcleos disponibles.
- **contamination**: determina la proporción esperada de valores atípicos en el conjunto de datos.

Esta técnica nos devolverá dos métricas a tener en cuenta, las *score samples* y la *función de decisión*. Las *score samples* son valores numéricos que se generan para cada punto de datos en base a la función de decisión, indicando la facilidad con la que dicho punto puede ser aislado y permitiéndonos de esta forma cuantificar su grado de anormalidad. En cuanto a la *función de decisión*, esta mide cuántos árboles de decisión se necesitan para aislar un punto específico (cuantos menos árboles sean necesarios, más anómalo se considerará la muestra).

Con el fin de comparar los resultados obtenidos al usar la técnica IQR, inicialmente probamos a introducir una contaminación del 10%. Basándonos en las *score samples*, se marcaron

10224 muestras como anómalas. No obstante, tras probar diferentes valores de contaminación, esta cifra se mantenía constante. Por lo tanto, decidimos aplicar una contaminación del 5% y clasificar los *outliers* mediante la *función de decisión*. Esto nos da como resultado un total de 6116 valores anómalos, siendo un valor más óptimo dada la naturaleza del dataset. En la Figura 5.2 se ve la distribución de los datos en base a su puntuación de anomalía y el punto exacto a partir del cual estamos marcando los *outliers*.

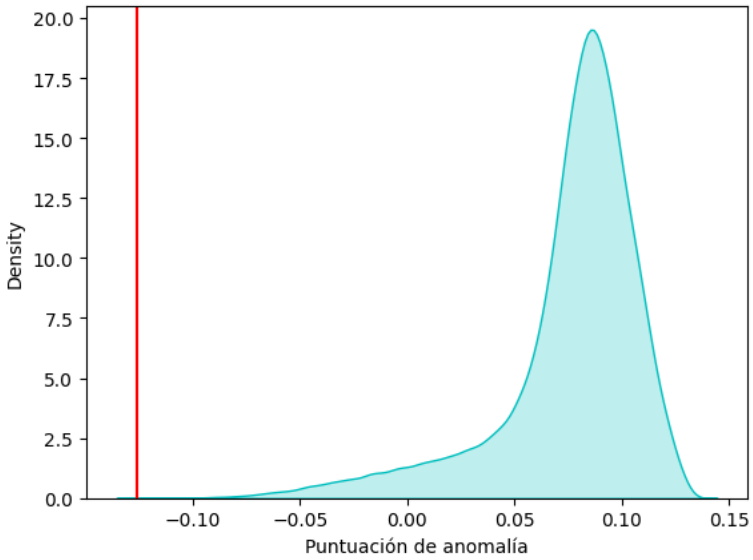

Figura 5.2: Distribución de las anomalías.

5.1.2 Corrección del *session_id*

La función de la característica *session_id* es relacionar los comentarios que pertenecen a una misma publicación, por lo que se trata de un atributo de etiquetado. Por otra parte, el conjunto de datos se encuentra etiquetado al nivel de este atributo, entonces presentará el mismo valor de la variable objetivo *label* para todas las entradas que pertenezcan a un *session_id*.

Analizando la distribución de esta característica, descubrimos que el valor que toma está relacionado directamente con la variable objetivo. Como podemos apreciar en la Figura 5.3, cuanto menor sea su valor, más probabilidad hay de que el comentario sea clasificado como acoso. También se observa que los valores tomados por *session_id* se dividen en tres rangos.

Como esta relación podría provocar un sesgo e influenciar las predicciones efectuadas por los modelos, decidimos aleatorizar los valores que toma la característica *session_id* para deshacer su relación numérica con la variable objetivo, ya que consideramos que en un escenario

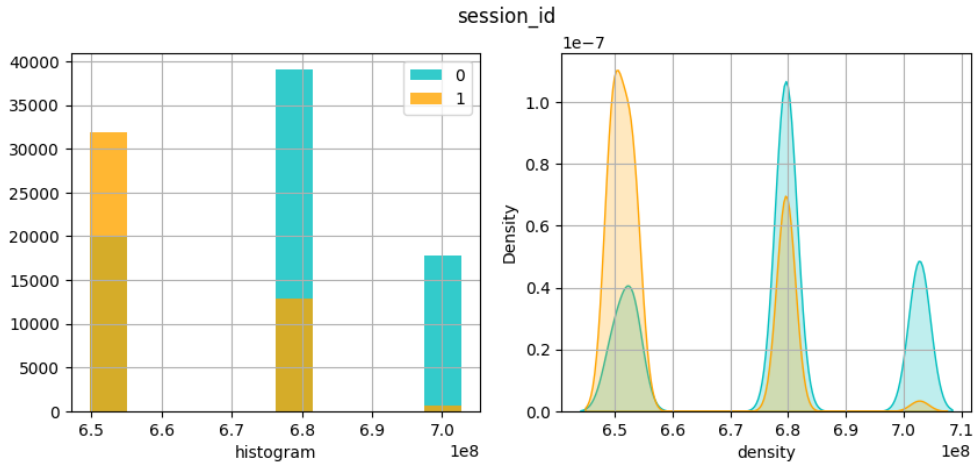

Figura 5.3: Distribución *session_id*.

real los modelos no contarían con esta ventaja para sus estimaciones. También modificamos los intervalos de valores que solía tener esa característica, transformándolos en una secuencia continua.

Esta aleatorización consiste en cambiar el valor que toma un *session_id* en diferentes entradas por otro valor aleatorio de una lista de valores secuenciales. En la Figura 5.4 podemos apreciar como el atributo perdió la relación numérica que tenía previamente con la variable objetivo *label*.

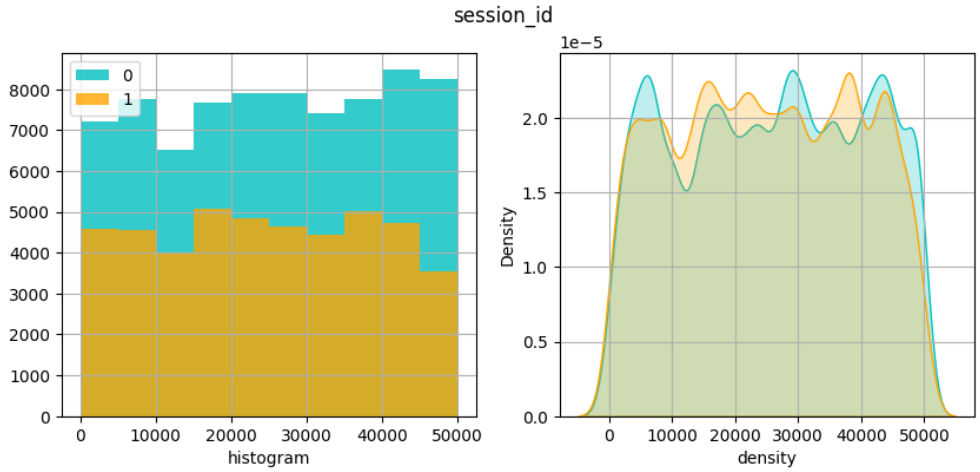

Figura 5.4: Distribución de *session_id* regenerado y aleatorizado.

5.2 Construcción de datos

La fase de Construcción de datos implica examinar las características del conjunto de datos con el objetivo de generar nuevos atributos adicionales que enriquezcan los ya presentes o mejoren su interpretación.

Dado que nuestro dataset dispone de un gran número de características, ya que antes de este trabajo ya se había realizado un procesamiento que incorporaba características consideradas significativas, como vendrían siendo *doc2vec*, *Tf-idf* o los intervalos temporales, hemos decidido no generar más atributos adicionales.

5.3 Formateo de los datos

Esta fase implica preparar los datos para el análisis y modelado. Para ello se normalizan las escalas transformando las variables y ajustando sus formatos, asegurándonos de que sean coherentes y adecuados para los algoritmos de procesamiento y análisis empleados. Destacan las siguientes aproximaciones para realizar esta fase:

- ***Label Encoding***: se trata de una técnica de codificación donde a cada valor único se le asigna un número entero, codificando de esta forma las categorías con valores numéricos ordenados. Resulta útil cuando hay un orden intrínseco en las categorías. Sin embargo, puede llevar a problemas si el algoritmo asume que hay una relación numérica significativa inexistente entre estas.
- ***One-hot Encoding***: esta otra técnica codifica las variables categóricas en un formato más adecuado para los algoritmos de *Machine Learning*. Cada valor único es convertido en una nueva columna binaria, donde se refleja la presencia o ausencia de dicho valor. De esta forma se evitan ordenamientos implícitos en las variables categóricas y las diferentes categorías se consideren por separado.

En nuestro caso, el dataset solo tiene características de tipo entero o flotante, por lo que no ha sido necesario aplicar ninguna técnica de formateo, ya que los modelos de aprendizaje máquina que vamos a utilizar aceptan este tipo de datos.

5.4 Selección inicial de los datos

Esta fase se enfoca en elegir los conjuntos de datos adecuados para el análisis y modelado. Esto implica identificar qué datos son relevantes y necesarios para resolver el objetivo del proyecto. Se pueden eliminar variables redundantes o irrelevantes y escoger las muestras que

serán utilizadas, consiguiendo así una mayor eficiencia y efectividad en las etapas posteriores al evitar la inclusión de información inútil y reduciendo la complejidad del análisis.

5.4.1 Submuestreo y Sobremuestreo

El Submuestreo y Sobremuestreo son técnicas utilizadas en el análisis de datos y el aprendizaje automático para abordar el desequilibrio en conjuntos de datos. En problemas de clasificación binaria como el nuestro resultan especialmente útiles cuando una clase es significativamente más frecuente que la otra.

- **Submuestreo (*Undersampling*)**: es una técnica utilizada para reducir el tamaño de la clase mayoritaria al eliminar aleatoriamente instancias de esa clase. Esto ayuda a equilibrar la proporción entre las clases, aunque hay que ser precavidos, ya que puede resultar en la pérdida de información valiosa.
- **Sobremuestreo (*Oversampling*)**: consiste en aumentar el tamaño de la clase minoritaria duplicando o generando instancias sintéticas para igualar la proporción entre ambas. Sin embargo, hay que tener en cuenta que esto puede aumentar el riesgo de sobreajuste.

En nuestro caso, dado el desbalanceo existente entre las dos clases, comentado en el Capítulo 4, y el tamaño del conjunto de datos, decidimos aplicar la técnica de submuestreo para balancear la proporción entre ambas clases y ceñirnos al uso de la métrica *accuracy*, ya que esta se ve especialmente beneficiada en estas situaciones [35].

Además, también implementamos el enfoque de validación cruzada estratificada (*Stratified Cross Validation*). Entonces, partir de un conjunto de datos balanceado garantiza que los subconjuntos de entrenamiento y prueba conserven una proporción estratificada, lo cual previene que el modelo obtenga un alto rendimiento debido al desequilibrio entre clases.

Para realizar el submuestreo, nos servimos de la librería *scikit-learn* que pone a nuestra disposición la función *RandomUnderSampler*.

5.5 Escalado de características

La normalización y el escalado de características son técnicas esenciales durante el preprocesamiento de los datos en el ámbito de la minería y el análisis de datos. Su objetivo es

asegurar que las variables tengan una escala y rango adecuados para garantizar un rendimiento óptimo en los algoritmos de *Machine Learning*.

Mediante la normalización, se ajustan los valores de las características a una escala uniforme, mientras que el escalado se centra en mantener las proporciones relativas entre las características. Las técnicas más comúnmente utilizadas son:

- **Estandarización *(Z-score normalization)***: es un método que transforma los valores de las características para que tengan una media de 0 y una desviación estándar de 1. La fórmula para la estandarización es:

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{std}(x)}$$

- **Normalización *(Min-Max Scaling)***: es un proceso donde se ajustan los valores de las características en un rango específico, generalmente este suele ser entre 0 y 1. Su objetivo es asegurar que todas las características tengan una magnitud similar y estén en la misma escala, evitando que una característica domine a las demás debido a que simplemente tenga valores más grandes.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

En este trabajo, implementamos la estandarización utilizando la biblioteca *Standard Scaler* proporcionada por *scikit-learn*. Es importante resaltar que las acciones de escalado se llevaron a cabo exclusivamente en el conjunto de entrenamiento, con el propósito de evitar filtrar información del conjunto de prueba, ya que esto influiría en los resultados finales.

5.6 Reducción de dimensionalidad

La reducción de dimensionalidad es una técnica esencial en el análisis de datos y el aprendizaje automático que nos permite abordar el desafío de lidiar con conjuntos de datos de alta dimensionalidad como el nuestro. Su objetivo es preservar la información relevante mientras disminuye el número de características. Entre sus diversas ventajas se encuentran la eliminación de ruido, la aceleración de los algoritmos y la mejora de la interpretación de los datos.

A continuación, exploraremos en detalle las técnicas que utilizamos durante esta etapa y elegiremos la que nos ayude a encontrar un equilibrio adecuado entre la cantidad de atributos eliminados y los recursos requeridos para llevarla a cabo.

5.6.1 RFECV

Recursive Feature Elimination with Cross-Validation es una técnica que combina la eliminación recursiva de características con la validación cruzada. Su objetivo principal es identificar y seleccionar las características más relevantes de un conjunto de datos. Opera mediante la eliminación progresiva de las características menos informativas, permitiéndole así al modelo enfocarse en aquellas que contribuyen significativamente a las predicciones. Su integración con la validación cruzada asegura que las decisiones de eliminación se tomen de manera robusta y generalizable.

Respecto a su implementación, utilizamos la funcionalidad de RFECV proporcionada por la biblioteca *Scikit-Learn* y fueron configurados los siguientes parámetros:

- ***estimator***: especifica el algoritmo de *Machine Learning* base que se utiliza para realizar la selección recursiva de características. Primero empezamos a realizar pruebas con *Random Forest*, pero dados los tiempos de ejecución elevados, la posibilidad de sobreajuste y los inconsistentes resultados obtenidos, decidimos utilizar *Logistic Regresion* como estimador.
- ***step***: indica la cantidad de características que se eliminarán en cada iteración del proceso de selección recursiva. Escogimos utilizar un *step* de 1.
- ***n_jobs***: este parámetro determina el número de núcleos de CPU que se utilizarán para ejecutar las tareas en paralelo durante la selección recursiva. Indicamos el valor -1 para utilizar todos los núcleos disponibles.
- ***random_state***: permite fijar una semilla con el fin de hacer las ejecuciones reproducibles, es decir, que diferentes ejecuciones arrojen los mismos resultados.
- ***scoring***: establece la métrica utilizada para evaluar el rendimiento de las características durante la selección. Como disponemos de un dataset perfectamente balanceado, utilizamos *accuracy* como métrica de evaluación.
- ***cv***: escoge la estrategia de validación cruzada utilizada durante la selección recursiva. La validación cruzada es una técnica que divide los datos en conjuntos de entrenamiento y prueba de manera repetida para evaluar el modelo de manera más robusta. Puede tomar dos valores:
  - Un número entero, se refiere al número de pliegues (*folds*) en la validación cruzada *StratifiedKFold*.

– Un objeto de validación cruzada predefinido, permite indicar un generador de *folds* de la librería *Scikit-Learn* para utilizarlo como estrategia.

En este caso, definimos el valor 5 con el fin de realizar una validación cruzada 5-fold, en la cual se dividen los datos en cinco partes, siendo cuatro utilizadas para entrenamiento y una para test en cada iteración.

En la Figura 5.5, se presenta la ejecución del algoritmo con los parámetros previamente mencionados. En esta gráfica, se observa cómo la media de las métricas aumenta de manera gradual hasta alcanzar aproximadamente un valor de 220 características. Sin embargo, una vez alcanzado este punto, los valores muestran una mayor irregularidad y comienzan a descender. Como la idea principal es seleccionar el número mínimo de atributos necesario sin perder demasiada precisión, el valor de 220 parece adecuado, dado el análisis de las métricas resultantes y la reducción de la dimensionalidad que proporciona.

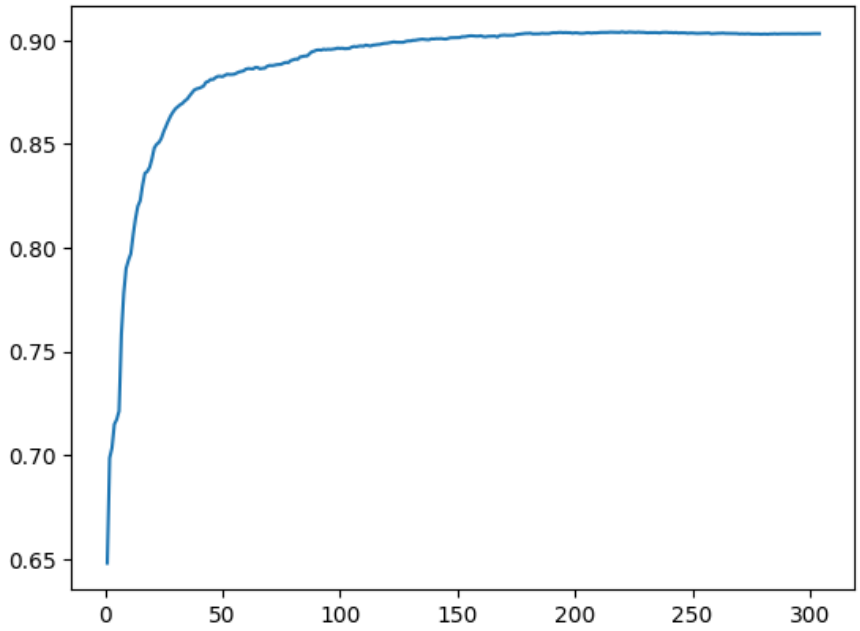

Figura 5.5: Gráfica de la ejecución de RFECV. Media accuracy vs Número de features escogidas.

Sin embargo, debido a los prolongados tiempos de ejecución que requerían esperas de varios días debido a las dimensiones de nuestro conjunto de datos, y teniendo en cuenta el plazo disponible para completar el proyecto, optamos por una alternativa computacionalmente más eficiente que, como explicaremos a continuación, también proporciona resultados satisfactorios.

5.6.2 PCA

Principal Component Analysis se trata de una técnica matemática utilizada en la estadística y el análisis de datos para simplificar datasets de alta dimensionalidad mientras mantiene la mayor parte de la variabilidad presente. Opera transformando las variables originales en un nuevo conjunto de variables no correlacionadas llamadas ”componentes principales”, que son calculados a partir de los vectores propios de la matriz de covarianza. Estos componentes representan las direcciones en las que los datos varían más y son ordenados en función de la cantidad de variabilidad que explican. Posteriormente son seleccionados los primeros k componentes para reducir la dimensionalidad a la par que se conserva la mayor parte de información relevante.

En cuanto a su implementación, utilizamos la funcionalidad PCA proporcionada por la biblioteca *Scikit-Learn*. Como configuración adicional, utilizamos la opción *n_components* que puede tomar dos tipos de valores diferentes. Si indicamos un valor entero, estaríamos seleccionando el número de componentes que representarán las características, mientras que, si indicamos un valor flotante, estaríamos indicando la varianza explicada total que nos interesa conservar (independientemente del número de características necesario para mantenerla). Decidimos mantener el 95% de esta varianza explicada, por lo que la opción de configuración tomó el valor de 0.95.

Tras aplicar esta técnica a los diferentes datasets utilizados en este trabajo, obtuvimos los siguientes resultados:

- *Baseline* conservó 283 características.
- *Without_lda* conservó 272 características.
- *Without_doc2vec* conservó 202 características.
- *Without_both* conservó 190 características.

5.6.3 Selección final de características

En nuestro caso, la técnica que elimina una mayor cantidad de características es RFECV. Sin embargo, esta eliminación no garantiza resultados superiores, ya que existe la posibilidad de perder información valiosa. Asimismo, también debemos tener en cuenta el alto costo computacional que conlleva su uso sobre las dimensiones de un conjunto de datos como el nuestro, el cual nos llevó a descartar esta opción.

Por otro lado, el algoritmo PCA no realiza una eliminación tan agresiva de características, lo que puede traducirse en mejores resultados al conservar más información relevante. Además, presenta ventajas adicionales, como la abstracción del cálculo de correlaciones entre características y, por supuesto, una mejora significativa en los tiempos de ejecución.

Capítulo 6

Modelado

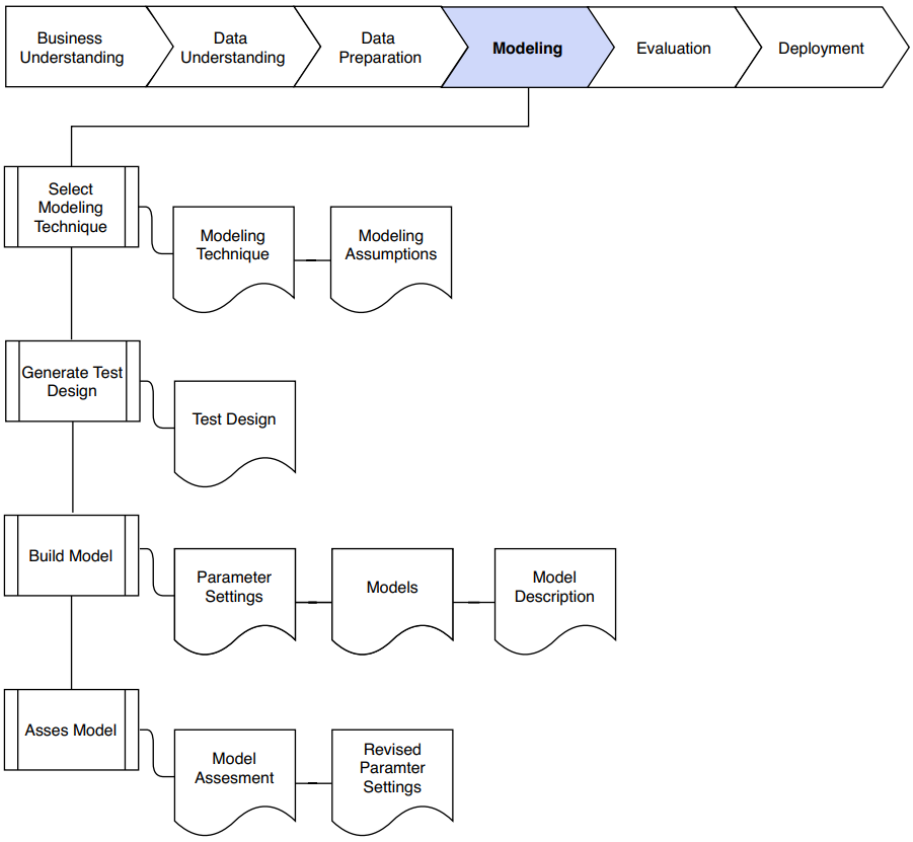

Figura 6.1: Fase CRISP-DM: Modelar.

En esta fase clave de la metodología CRISP-DM, nos sumergimos en el importante proceso de modelado. Una vez preparados los datos en fases previas, comenzaremos a aplicar diversas técnicas y algoritmos de aprendizaje automático con el fin de capturar patrones, tendencias y relaciones ocultas, las cuales nos serán de gran utilidad para hacer predicciones.

Esta etapa involucra la selección y configuración de modelos, la evaluación de su rendimiento y la iteración continua para lograr resultados precisos y robustos.

6.1 Criterios de selección

La selección de los modelos que utilizaremos fue hecha en base a los siguientes criterios:

- El período de tiempo requerido para completar el proceso de entrenamiento.
- Su nivel de eficacia al considerar la métrica *accuracy*, definida en la Sección 7.1.2.
- Aspectos relacionados con la naturaleza de los datos, como vendrían siendo sus dimensiones, correlaciones...

Para este trabajo, elegiremos diversos modelos, ya que uno de los objetivos es poder contrastar los resultados entre diversos algoritmos con el fin de determinar cuál será el más óptimo en cuanto a los resultados arrojados. Teniendo en cuenta estos factores, así como el buen rendimiento que han demostrado en conjuntos de datos de naturaleza similar [36], hemos optado por elegir las siguientes opciones:

- **Naïve-Bayes**: emplearemos este algoritmo cuyo enfoque se adapta bien a problemas en los que las características siguen una distribución gaussiana. Utilizaremos la implementación *GaussianNB* de *Scikit-Learn*, que nos permite aplicar esta técnica de manera efectiva en problemas de clasificación.
- **Logistic Regresion**: utiliza una función logística (*sigmoide*) para transformar la combinación lineal de las características en una probabilidad en el rango [0, 1], luego se establece un umbral para determinar la clase final de la observación. A pesar de su nombre, es perfectamente adecuada para problemas de clasificación binaria. Utilizaremos la implementación *LogisticRegression* de *Scikit-Learn*.
- **Random Forest**: algoritmo que opera mediante la combinación de múltiples árboles de decisión generados de forma aleatoria. Optaremos por la implementación *RandomForestClassifier* de *Scikit-Learn*.
- **AdaBoost**: algoritmo que combina varios modelos más débiles para obtener un modelo final más robusto. Utilizaremos la implementación *AdaBoostClassifier* de *Scikit-Learn*.
- **Linear SVC**: para problemas de clasificación lineal, el modelo de Máquinas de Soporte Vectorial (SVM) resulta eficaz. Optaremos por su variante lineal, *LinearSVC* en *Scikit-Learn*.

6.2 Proceso de entrenamiento

El dataset con el que trabajaremos una vez finalizada la fase de "Preparación de los Datos" se divide en dos conjuntos. Uno se trata del conjunto de entrenamiento, al que a partir de ahora nos referiremos como E, el cual nos permitirá determinar la configuración más óptima para cada modelo. El otro conjunto es el de test (T), que será utilizado por los algoritmos para evaluar su rendimiento en base a las predicciones hechas sobre dicho conjunto.

El primer paso a seguir es dividir cada uno de los conjuntos E y T haciendo uso del algoritmo *StratifiedKFold* en 5 grupos estratificados. Luego, en cada iteración contaremos con cuatro subconjuntos de entrenamiento (E_i, donde $_i$ indica el conjunto i-ésimo) y un fold de test (T'), cuyos componentes cambiarán en cada iteración. Esta metodología de división nos permite mantener las proporciones de cada categoría, previniendo posibles sobreajustes y reflejando las características particulares del conjunto de muestras.

Finalmente, vamos a someter los subconjuntos E_i a un proceso llamado *Grid Search* con validación cruzada, el cual nos lo proporciona la biblioteca *Scikit-Learn* con el nombre de *GridSearchCV*. Este proceso se desarrolla de la siguiente manera:

1. Cada conjunto E_i se divide en N subconjuntos más pequeños de entrenamiento a los que nos referiremos como E_ij (donde $_j$ representa el subconjunto j-ésimo dentro de cada E_i) y un conjunto de validación (V). Este último nos aportará una puntuación de métrica, que en nuestro caso será la precisión del modelo, pero se puede ajustar para utilizar otras métricas, como por ejemplo *recall*.

2. Como la naturaleza de la variable objetivo es binaria, se aplicará automáticamente una división en 5-fold estratificados.

3. Una vez terminadas todas las iteraciones, contaremos con un atributo llamado *best_params_*, el cual nos proporcionará los valores óptimos para los parámetros especificados en la variable *param_grid* para cada modelo en particular.

Este proceso itera para cada distribución de E_i y T'. Una vez recogidos la totalidad de los resultados, seleccionamos aquellos que obtengan las mejores puntuaciones, teniendo en cuenta la importancia de evitar el sobreajuste. Cabe destacar que en las situaciones donde las diferencias entre los resultados no sean significativas, se atenderá a la desviación típica de los resultados.

6.3 Hiperparametrización

El proceso de hiperparametrización se refiere a la práctica de ajustar y afinar los hiperparámetros de un modelo de aprendizaje automático con el objetivo de optimizar su rendimiento y generalización. Para ello se exploran diferentes combinaciones de valores.

Se trata de un proceso crucial para evitar el sobreajuste y lograr un equilibrio entre sesgo y varianza del modelo. Generalmente se realiza utilizando técnicas como *Grid Search* (mencionada en el apartado previo), *Random Search* o técnicas más avanzadas como la Optimización Bayesiana.

A continuación, especificaremos para cada modelo los parámetros escogidos y los valores que tomaron cada uno de ellos.

6.3.1 Naïve Bayes

Este enfoque se apoya en el análisis de cómo las muestras se distribuyen en forma gaussiana, lo que implica el cálculo de la variabilidad vinculada a cada característica del conjunto de datos, bajo la premisa de que cada atributo es independiente de los demás.

La implementación utilizada, *GaussianNB* (*Scikit-Llearn*), presenta un único parámetro relevante para su entrenamiento, conocido como *var_smoothing*. Este influencia la magnitud de la desviación típica y su ajuste impacta en la dispersión de la distribución. Cuanto mayor sea su valor, más probable es caer en el sobreentrenamiento. Los valores evaluados son los siguientes: 1^{e-0}, 1^{e-1}, 1^{e-2}, 1^{e-3}, 1^{e-4}, 1^{e-5}.

6.3.2 Regresión Logística

La regresión logística es un algoritmo de aprendizaje supervisado que a través de una función logística, convierte una combinación lineal de características en una probabilidad de pertenencia a una clase específica.

Implementamos la función *LogisticRegression* ofrecida por *Scikit-Learn*, la cual involucra el parámetro *C*. Este parámetro influye en la inversión de la regularización, afectando la intensidad de la penalización por complejidad del modelo. Cuanto menor sea el valor que toma, mayor regularización habrá, mientras que valores más altos permiten que el modelo se ajuste más a los datos de entrenamiento. Cabe destacar que la elección adecuada busca un equilibrio entre ajuste y generalización del modelo. En nuestro caso, los valores empleados son los siguientes: 0.1, 0.01, 0.001, 0.0001.

6.3.3 *Random Forest*

El modelo *Random Forest* emplea múltiples árboles de decisión entrenados con los subconjuntos de datos para luego combinar sus resultados en una media. En este caso tenemos una gran variedad de parámetros de configuración, pero decidimos enfocarnos en:

- ***n_estimators***: indica la cantidad de árboles de decisión que serán entrenados. Los valores empleados son 500, 1000, 1200
- ***max_depth***: determina la profundidad máxima de los árboles de decisión. Los valores empleados son 4, 7, *None*. Cabe destacar que el valor *None* indica que no se establecerá una profundidad máxima predefinida para los árboles de decisión, por lo que estos se podrán expandir libremente hasta que contengan un número mínimo de muestras por hoja o hasta que se alcance un nivel en el que las hojas sean completamente puras.

6.3.4 AdaBoost

AdaBoost es un algoritmo fundamentado en la combinación ponderada de salidas de algoritmos más "débiles". Utilizamos la implementación de *Scikit-Learn*, la cual nos ofrece dos opciones: SAMME (*StageWise Additive Modeling*) y SAMME.R. La distinción entre ambas radica en cómo generan las predicciones: la primera proporciona valores en el rango de 0 a 1, mientras que la segunda indica la probabilidad de que una muestra pertenezca a una clase o a otra. En nuestro caso utilizamos la opción por defecto, que es SAMME.R.

Respecto a los diversos parámetros de configuración que también tenemos en este caso, nos enfocamos en:

- ***n_estimators***: indica el número de modelos empleados en la combinación de las salidas. Los valores especificados son 1000, 1200, 1500.
- ***learning_rate***: especifica la contribución de cada algoritmo al resultado final. Existe una conexión inversa entre este atributo y la cantidad de estimadores utilizados en el modelo. Los valores empleados son 10, 1, 0.1, 0.01.

6.3.5 LinearSVC

Variante del modelo SVC con kernel lineal, es un algoritmo de clasificación supervisado implementado también en *Scikit-Learn*. En este caso decidimos enfocarnos en los siguientes parámetros:

- **tol**: indica el criterio de parada de la ejecución del algoritmo. Los valores empleados son 1^{e-1}, 1^{e-2}, 1^{e-3}, 1^{e-4}.

- **C**: configura la penalización al hacer las predicciones, cuanto menos sea su valor, más penalización habrá. Los valores empleados son 0.1, 0.01, 0.001.

6.4 Resultados del modelado

Una vez aplicado el entrenamiento descrito previamente en la sección Proceso de entrenamiento, recopilamos los resultados correspondientes a los mejores parámetros, la precisión (*accuracy*) y el tiempo de ejecución de cada modelo para cada conjunto de datos.

Durante la selección de las combinaciones óptimas, hemos considerado tanto los resultados más destacados en cada etapa del *Grid Search* como aquellos provenientes de los folds estratificados. También hemos tenido en cuenta la dispersión de los resultados, puesto que las combinaciones con una menor variabilidad reflejan una mayor estabilidad en los conjuntos de E y V.

Estos resultados se presentan en las Tablas 6.1, 6.2, 6.3, 6.4, y 6.5.

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	var_smoothing = 1e-1	0.7534	2.512
WithoutLDA	var_smoothing = 1e-1	0.7201	2.450
WithoutDoc2vec	var_smoothing = 1e-1	0.7134	1.833
WithoutBoth	var_smoothing = 1e-1	0.7155	1.726

Tabla 6.1: Gridsearch GaussianNB.

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	C = 0.01	0.8962	4.665
WithoutLDA	C = 0.01	0.8941	4.129
WithoutDoc2vec	C = 0.01	0.8337	2.011
WithoutBoth	C = 0.01	0.8279	1.859

Tabla 6.2: Gridsearch LogisticRegression.

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	n_estimators = 1000, max_depth = None	0.9249	3041.175
WithoutLDA	n_estimators = 1000, max_depth = None	0.9286	2990.372
WithoutDoc2vec	n_estimators = 1000, max_depth = None	0.8823	2604.521
WithoutBoth	n_estimators = 1000, max_depth = None	0.8879	2392.356

Tabla 6.3: Gridsearch RandomForest.

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	n_estimators = 1500, learning_rate = 1	0.9042	11894.731
WithoutLDA	n_estimators = 1500, learning_rate = 1	0.9054	11661.087
WithoutDoc2vec	n_estimators = 1500, learning_rate = 1	0.8421	8730.383
WithoutBoth	n_estimators = 1500, learning_rate = 1	0.8392	8074.622

Tabla 6.4: Gridsearch AdaBoost.

Dataset	Mejores parámetros	Accuracy	Tiempo (s)
Baseline	tol = 1e-2, C = 0.01	0.8962	208.298
WithoutLDA	tol = 1e-2, C = 0.001	0.8943	243.650
WithoutDoc2vec	tol = 1e-2, C = 0.01	0.8322	173.980
WithoutBoth	tol = 1e-2, C = 0.01	0.8282	134.288

Tabla 6.5: Gridsearch LinearSVC.

6.5 Pronóstico para la fase de evaluación

En base a los resultados obtenidos en la sección anterior, podemos formular las siguientes premisas en relación a la etapa de evaluación:

- El modelo GaussianNB será el que menos tarde en la fase de entrenamiento, pero también será el que peores métricas de precisión presente.
- El modelo AdaBoost será el que más tiempo tarde, pero arrojará valores elevados de precisión.
- El modelo LogisticRegression tardará muy poco tiempo respecto a otros algoritmos y arrojará buenos resultados en las métricas de precisión.
- El modelo Random Forest tardará un tiempo razonable y será que el presente los mejores valores de precisión.
- El modelo LinearSVC tardará mucho más respecto al modelo de LogisticRegression, pero arrojará métricas de precisión similares.
- Los conjuntos de datos *WithoutLDA*, *WithoutDoc2vec* y *WithoutBoth*, al tener un menor número de características en comparación con el dataset *Baseline*, se espera que exhiban métricas de precisión más pobres.
- El número de características que contengan los conjuntos de datos influyen directamente en el tiempo de ejecución, siendo estos menores cuantas menos características se presenten.

Capítulo 7

Evaluación

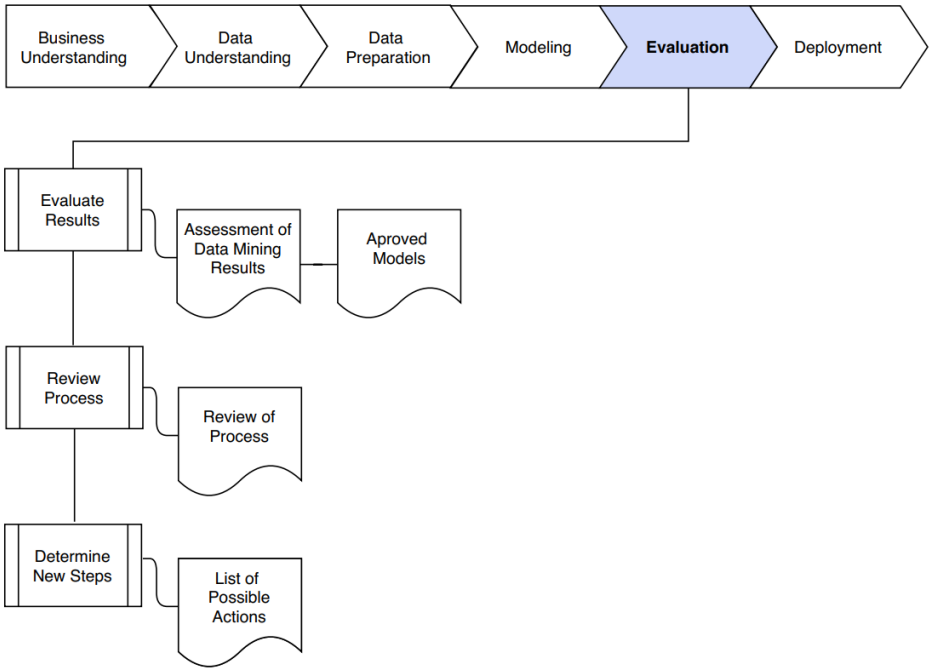

Figura 7.1: Fase CRISP-DM: Evaluación.

La fase de evaluación representa un paso crítico en la metodología CRISP-DM. En esta etapa ponemos a prueba los modelos construidos y cuidadosamente refinados mediante la optimización de sus hiperparámetros. Aquí, la atención se centra en medir y comprender el rendimiento de estos modelos bajo diversas circunstancias. Utilizamos métricas específicas con el fin de evaluar su capacidad para hacer predicciones precisas y comprender cómo interactúan con los datos. Además, examinamos detalladamente el proceso que respalda estas predicciones, lo que nos permite obtener una perspectiva holística de la efectividad y robustez de los algoritmos. En base en los resultados y análisis obtenidos, también tomamos decisiones

informadas para determinar la mejor dirección a seguir en nuestro análisis de datos [37].

7.1 Evaluación los resultados

Cada modelo se ha entrenado con los mejores parámetros obtenidos en la Sección 6.4 Resultados del modelado. Además, también se ha medido el tiempo que consumió dicho proceso. Finalmente se han realizado las predicciones sobre el conjunto T y se han obtenido una matriz de confusión junto con una serie de métricas para cada modelo.

7.1.1 Matriz de confusión

La matriz de confusión es una herramienta fundamental en la evaluación de modelos de clasificación. Se trata de una representación tabular de las predicciones realizadas por un modelo en relación a las clases del conjunto de datos. Esta matriz desglosa la cantidad de predicciones correctas e incorrectas para cada clase. Nuestro problema es de naturaleza binaria, entonces la matriz generada tendrá una dimensión 2x2 como se muestra en la Figura 7.2, donde contaremos con los siguientes campos :

Figura 7.2: Matriz de confusión.

- **Verdaderos negativos (TN)**: se encuentran las predicciones marcadas como negativas y que verdaderamente lo eran. Considera el valor 0 como negativo y el 1 como positivo.
- **Falsos positivos (FP)**: indica las predicciones marcadas como positivas pero que realmente eran negativas.
- **Falsos negativos (FN)**: abarca las predicciones marcadas como negativas pero que realmente eran positivas.

- **Verdaderos positivos (TP)**: engloba las predicciones que se marcaron como positivas y que verdaderamente lo eran.

7.1.2 Métricas de supervisión

Las métricas que vamos a describir a continuación se obtienen a partir de la matriz confusión. Estas nos proporcionan una manera cuantitativa de medir el desempeño de los modelos, facilitando así una comparación más eficaz entre ellos:

- ***Accuracy***: mide la proporción de predicciones correctas sobre el total de predicciones.

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

- ***Recall***: representa la proporción de instancias positivas que fueron correctamente identificadas.

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

- ***Precision***: indica la proporción de instancias identificadas como positivas que realmente son positivas.

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

- ***F-score***: combina las métricas *precision* y *recall* en una sola, siendo esta una media armónica de ambas.

$$\text{F-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{7.4}$$

- ***Índice Kappa de Cohen***: métrica cuyo objetivo es determinar la cantidad de acuerdo que va más allá de lo que podría esperarse por azar. Su cálculo implica la comparación de las clasificaciones observadas con las clasificaciones esperadas por azar, siendo *Observed Agreement* el resultado de sumar TP con TN y dividirlo entre el número de muestras, mientras que *Expected Agreement* es la probabilidad esperada de que los clasificadores coincidan por azar.

$$\text{Kappa} = \frac{\text{Observed Agreement} - \text{Expected Agreement}}{1 - \text{Expected Agreement}} \tag{7.5}$$

  Un valor de *Kappa* de 1 indica un acuerdo perfecto, mientras que el 0 representa el acuerdo esperado por azar. Un valor negativo indica un acuerdo peor que el esperado por azar.

Nuestro problema consiste en clasificar de forma correcta los comentarios en base a su contenido en ciberacoso, lo que nos lleva a obtener una proporción factible de predicciones acertadas (TP y TN). Esto implica que a la hora de escoger una métrica de referencia para contrastar la efectividad de los modelos, optemos por utilizar *accuracy* y *F-score* como principales, mientras que consideramos las demás como complementarias para reforzar nuestra evaluación.

7.1.3 Métricas de referencia

Dado que previamente se han llevado a cabo pruebas de predicción sobre el conjunto de datos original, hemos optado por emplear esas métricas como un punto de referencia establecido para la comparación de los resultados que logremos obtener [21]. En la tabla 7.1 podremos ver dichos resultados.

Modelo	Accuracy	Precision	Recall
Naïve Bayes	0.71	0.75	0.66
linearSVM	0.85	0.84	0.88

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

7.2 Resultados obtenidos

Una vez hechas todas las predicciones previstas para cada modelo y dataset, hemos obtenido las siguientes métricas sobre cada ejecución:

7.2.1 GaussianNB

Dataset	Accuracy	Precision	Recall	F-Score	Kappa	Tiempo (s)
Baseline	0.798	0.807	0.797	0.796	0.595	0.099
WithoutLDA	0.780	0.781	0.780	0.776	0.560	0.141
WithoutDoc2vec	0.749	0.801	0.749	0.738	0.498	0.137
WithoutBoth	0.716	0.789	0.716	0.697	0.431	0.103

Tabla 7.2: Métricas GaussianNB.

Como podemos comprobar en la figura 7.2, las métricas obtenidas cumplen con las premisas que establecimos anteriormente. Se trata del algoritmo más rápido, pero desafortunada-

mente, también es el que peores resultados obtiene en comparación con el resto de modelos. También cumple la premisa de que cuanto menor es el número de características en los conjuntos de datos derivados, peores serán los resultados conseguidos y menores sus tiempos de ejecución.

En cuanto al dataset de referencia 7.1, obtenemos unos valores superiores en las métricas de *accuracy*, *precision* y *recall*.

7.2.2 LogisticRegression

Dataset	Accuracy	Precision	Recall	F-Score	Kappa	Tiempo (s)
Baseline	0.897	0.896	0.896	0.895	0.794	0.624
WithoutLDA	0.896	0.895	0.896	0.896	0.697	0.431
WithoutDoc2vec	0.831	0.832	0.832	0.831	0.664	0.299
WithoutBoth	0.828	0.829	0.828	0.828	0.657	0.257

Tabla 7.3: Métricas Regresión Logística.

El modelo de Regresión Logística también cumple con la premisa establecida de que cuanto menor sea el número de características, peor será el valor de sus métricas, lo cual se puede verificar en la tabla 7.3. Aunque en este caso la diferencia entre *Baseline* y *WithoutLDA* en términos de *precision* y *recall* es ínfima. Sus valores compiten entre los mejores y es el segundo modelo más rápido de entre todos.

7.2.3 RandomForest

Dataset	Accuracy	Precision	Recall	F-Score	Kappa	Tiempo (s)
Baseline	0.935	0.935	0.934	0.935	0.869	1497.989
WithoutLDA	0.935	0.936	0.936	0.935	0.871	1777.219
WithoutDoc2vec	0.894	0.894	0.894	0.894	0.788	1532.982
WithoutBoth	0.897	0.897	0.897	0.897	0.795	1166.419

Tabla 7.4: Métricas Random Forest.

Este modelo presenta los mejores valores en cuanto métricas sin excederse en sus tiempos

de ejecución, aunque sea el segundo algoritmo con mayores tiempos. Sin embargo, también contradice algunas de las suposiciones que habíamos hecho previamente.

En primer lugar, las métricas del conjunto *WithoutLDA* son ligeramente superiores a las del conjunto *Baseline*. Además, este conjunto de datos también presenta un tiempo de ejecución más elevado. En cuanto a los datasets *WithoutDoc2vec* y *WithoutBoth*, se observa una dinámica similar, donde *WithoutBoth* muestra valores ligeramente superiores.

Los resultados indican que este algoritmo muestra un mejor rendimiento al omitir las características LDA. Esto sugiere que la selección de los temas utilizados en la creación de estas características es mejorable, ya que en lugar de aportar valor, introducen sesgo a la hora de realizar las predicciones.

7.2.4 AdaBoost

Dataset	Accuracy	Precision	Recall	F-Score	Kappa	Tiempo (s)
Baseline	0.909	0.909	0.909	0.909	0.819	4399.651
WithoutLDA	0.908	0.908	0.908	0.908	0.814	4341.942
WithoutDoc2vec	0.845	0.846	0.845	0.845	0.691	3302.771
WithoutBoth	0.843	0.844	0.844	0.843	0.688	1166.419

Tabla 7.5: Métricas AdaBoost.

Como se puede ver en la tabla 7.4, el desempeño del modelo AdaBoost sigue ajustándose a las suposiciones iniciales. Aunque este modelo requiere el tiempo del ejecución más largo, sus métricas son notables, aunque no lleguen al nivel del modelo Random Forest.

Otro aspecto relevante es que el modelo sigue la tendencia prevista: a medida que se reducen las características del conjunto *Baseline*, su rendimiento decae gradualmente (siendo esta degradación ínfima cuando las características de LDA son descartadas). Esta tendencia también se refleja en el tiempo de ejecución, el cual va disminuyendo a medida que se descartan características.

7.2.5 LinearSVC

Finalmente, en la tabla 7.6 se muestran los resultados obtenidos de los modelos LinearSVC. Aunque cumple la premisa de que cuanto menos características haya, peores resultados se

Dataset	Accuracy	Precision	Recall	F-Score	Kappa	Tiempo (s)
Baseline	0.897	0.898	0.897	0.897	0.794	8.337
WithoutLDA	0.896	0.896	0.896	0.896	0.792	16.154
WithoutDoc2vec	0.832	0.832	0.831	0.831	0.663	3.406
WithoutBoth	0.828	0.829	0.829	0.828	0.657	9.782

Tabla 7.6: Métricas LinearSVC.

obtendrán, podemos observar un incremento del tiempo de ejecución al eliminar las características de LDA.

También presenta unos valores muy similares a los arrojados por el modelo de Regresión Logística 7.3, aunque desafortunadamente tiene unos tiempos de ejecución considerablemente superiores. En cuanto al dataset de referencia 7.1, obtenemos unos valores superiores en las métricas de *accuracy*, *precision* y *recall*.

7.3 Revisión del procedimiento

En base a los resultados obtenidos en el apartado anterior, podemos confirmar que nuestro modelo GaussianNB ha arrojado resultados superiores en comparación con el modelo GaussianNB del artículo de referencia 7.1. Esta tendencia también se observa al comparar nuestros otros modelos con los resultados presentados en el caso del modelo de LinearSVC utilizado como referencia. Si examinamos minuciosamente las métricas obtenidas, podemos deducir las siguientes conclusiones:

- En general, el conjunto de datos *Baseline* es el que presenta mejores resultados, por lo que descartar características provoca métricas más débiles. Esto se debe a que cuanta más información descartemos, menos precisas serán las predicciones de los modelos.
- La exclusión de las características de *doc2vec* conlleva una notable reducción en los tiempos de ejecución, pero tiene un impacto adverso en las métricas de evaluación. No obstante, prescindir de las características LDA no resulta en una disminución significativa de los tiempos de ejecución, ni deteriora de manera notable los resultados reflejados en las métricas, excepto en el caso del modelo Random Forest, que parece verse beneficiado de dichas características. Lo que indica que los *topics* escogidos para su generación son mejorables.

- La influencia del descarte de características en las métricas generadas puede variar en función del tipo de modelo utilizado.
- Bajo la perspectiva de los resultados obtenidos, es razonable pensar en los modelos Random Forest y Regresión Logística como los mejores para ser utilizados en un escenario similar. Aunque todos muestran buenos resultados, estos dos se destacan entre los demás debido a la correlación entre la calidad de las métricas obtenidas y el tiempo y recursos necesarios para lograrlas.

Cabe destacar que a lo largo del desarrollo del proceso de *hiperparametrización* tratamos de abarcar una diversidad significativa de valores con el objetivo de optimizar los resultados arrojados por las ejecuciones. Esta estrategia nos permite contrastar más efectivamente el rendimiento de cada modelo y obtener resultados consistentes.

7.4 Comparativa adicional

Como comparación adicional, hemos decidido incluir los resultados obtenidos por nuestro compañero Maura García Barro [38]. Aunque su conjunto de datos difiere del nuestro, lo que podría implicar diferencias significativas en los resultados, ambos comparten el mismo dominio y características. Por lo tanto, consideramos que su inclusión podría proporcionar un valor significativo al contrastarlo con los resultados obtenidos.

En las Tablas 7.7, 7.8, 7.9 y 7.10, podemos observar que obtiene métricas ligeramente inferiores a las nuestras, pero notoriamente similares. Es importante destacar que el conjunto de datos que utilizó contiene 56.799 muestras, que en comparación con las 122.234 entradas de nuestro conjunto, implica un volumen de información significativamente menor sobre el que realizar las predicciones.

Dataset	Accuracy	Precision	Recall	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 7.7: Comparativa adicional GaussianNB.

Dataset	Accuracy	Precision	Recall	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 7.8: Comparativa adicional Random Forest.

Dataset	Accuracy	Precision	Recall	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 7.9: Comparativa adicional AdaBoost.

Dataset	Accuracy	Precision	Recall	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 7.10: Comparativa adicional LinearSVC.

Capítulo 8

Conclusiones

Para concluir, comenzaremos este último capítulo realizando una evaluación exhaustiva del nivel de cumplimiento de los objetivos que fueron establecidos al comienzo y que se detallan en la Sección 1.2. Luego, compartiremos las valiosas lecciones que hemos extraído durante la realización de este trabajo. Finalmente, delinearemos una serie de directrices a seguir en caso de que surjan proyectos futuros que opten por aprovechar como punto de partida el trabajo que hemos desarrollado.

8.1 Grado de compleción

Respecto al grado de compleción de este trabajo, hemos alcanzado un resultado beneficioso dentro del plazo establecido. Cabe destacar que el estudiante contaba con un nivel básico de conocimiento en el ámbito del *Machine Learning* y las diversas técnicas relacionadas en este campo, lo que implicó un esfuerzo adicional en términos de aprendizaje. Sin embargo, consideramos que hemos completado los objetivos previamente establecidos:

- En el Capítulo 3, hemos indicado de forma exhaustiva los fundamentos en los que se basa este trabajo. Además, hemos explorado y evaluado las diversas opciones que se presentan en cada disciplina.
- En los Capítulos 4 y 5, hemos realizado un análisis exploratorio exhaustivo del conjunto de datos con el objetivo de comprender la naturaleza de cada atributo. Posteriormente, aplicamos técnicas basadas en la Ingeniería de Características para identificar anomalías utilizando métodos como IQR y *Isolation Forest*. Además, también procesamos cada atributo mediante un proceso de estandarización. Luego, seleccionamos las características más relevantes para reducir la dimensionalidad del conjunto, minimizando al mismo tiempo la pérdida de información. Este proceso se llevó a cabo utilizando las técnicas RFECV y PCA.

- En el Capítulo 3 hemos escogido y analizado detalladamente los modelos que se utilizaron. Luego, en el Capítulo 6, llevamos a cabo un proceso de optimización de hiperparámetros mediante Grid Search con validación cruzada para afinar la configuración de cada algoritmo. Esta etapa también nos brindó una estimación inicial de los comportamientos de los modelos en función de los resultados obtenidos.
- En el Capítulo 7, llevamos a cabo una comparación de los resultados obtenidos por los modelos entre sí, además de contrastarlos con los trabajos previamente realizados que hemos utilizado como referencia.

8.2 Conocimientos adquiridos

Durante la realización de este proyecto se ha tenido la valiosa oportunidad de adquirir un extenso conjunto de conocimientos y habilidades en el ámbito del *Machine Learning*. Esto no solo ha implicado una profundización en los temas abordados previamente en la mención en *Tecnologías de la Información*, si no también una expansión de los conceptos aprendidos durante su programa de grado. Concretamente:

- Ampliación del conocimiento en diversas técnicas de Procesamiento del Lenguaje Natural (NLP), lo que permitió una mayor comprensión de los atributos en el conjunto de datos.
- Desarrollo de habilidades estadísticas y analíticas para comprender la distribución y el comportamiento de los datos.
- Expansión significativa del conjunto de conceptos relacionados con la minería de datos.
- Adquisición de conocimientos en el desarrollo e implementación de técnicas de *Machine Learning*.
- Formación en el lenguaje *Python* y el manejo de dataframes con *Pandas*.
- Mejora en la comprensión de la gestión de proyectos, abarcando desde la planificación inicial con estimaciones de esfuerzo y costos, hasta el seguimiento continuo. Esto proporcionó una valiosa experiencia que puede servir de base para futuros proyectos.

En líneas generales, este proyecto representa una valiosa contribución, tanto en términos personales como profesionales, abordando una temática de vital relevancia en la sociedad actual y haciendo uso de tecnologías cada vez más consolidadas. Además, he tenido la oportunidad de adquirir nuevos conocimientos en diversas áreas, al mismo tiempo pude reforzar lo aprendido durante mi formación académica.

8.3 Lineas Futuras

Por último, presentamos una lista de posibles propuestas a seguir que utilizan nuestro trabajo como punto de partida:

- Explorar a fondo la aplicación de algoritmos de *Deep Learning*, ya que, dada la gran cantidad de datos disponibles, podríamos obtener modelos más eficaces.
- Explorar con mayor profundidad la implementación de diversos algoritmos basados en *SVM* sería una opción valiosa para expandir el conocimiento en este campo.
- Generar conjuntos de datos adicionales similares para estudiar el comportamiento de los diferentes modelos a una escala más amplia.
- Generar un conjunto de datos personalizado mediante diversas técnicas de Procesamiento del Lenguaje Natural (NLP), ya sea ampliando las existentes o explorando nuevas, con el propósito de aprovechar al máximo el valor que estas métricas pueden proporcionar y tener datos adicionales para contrastar los resultados.
- Evaluar el rendimiento de los modelos en un entorno real para validar su comportamiento en casos prácticos.

Bibliografía

[1] D. Olweus, *Bullying at school: Long-term outcomes for the victims and an effective school-based intervention program. In Aggressive behavior: Current perspectives*, 1st ed. Boston, MA: Springer US., 1994.

[2] R. Slonke and P. K. Smith, *Cyberbullying: Another main type of bullying?*, 1st ed. Scandinavian journal of psychology., 2008.

[3] T. Ivarsson, A. G. Broberg, T. Arvidsson, and C. Gillberg, *Bullying in adolescence: Psychiatric problems in victims and bullies as measured by the Youth Self Report (YSR) and the Depression Self-Rating Scale (DSRS)*, 5th ed. Nordic journal of psychiatry., 2005.

[4] M. Garaigordobil, "Ciberbullying en adolescentes y jóvenes del país vasco: Cambios con la edad," 2015, consultado en septiembre de 2023. [En línea]. Disponible en: https://revistas.um.es/analesps/article/view/analesps.31.3.179151

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

[6] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*, 1st ed. O'Reilly Media, 2016.

[7] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," 1996, consultado en septiembre de 2023. [En línea]. Disponible en: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1230

[8] A. Azevedo and S. M. Filipe, "Kdd, semma and crisp-dm: a parallel overview," 2008, consultado en septiembre de 2023. [En línea]. Disponible en: http://hdl.handle.net/10400.22/136

[9] S. Studer, T. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K. Müller, "Towards crisp-ml (q): a machine learning process model with quality assurance methodology," 2021, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.mdpi.com/2504-4990/3/2/392

[10] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," 2000, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf

[11] M. Rybnicek, R. Poisel, and S. Tjoa, *Facebook Watchdog: A Research Agenda for Detecting Online Grooming and Bullying Activities.* IEEE, 2854–2859., 2013.

[12] K. Dinakar, B. Jones, C. Havasi, H. Lieberman, and R. Picard, *Common sense reasoning for detection, prevention, and mitigation of cyberbullying.* CM Transactions on Interactive Intelligent Systems (TiiS), 2012.

[13] K. Dinakar, R. Reichart, and H. Lieberman, *Modeling the detection of Textual Cyberbullying.* The Social Mobile Web, 2011.

[14] I. Huang, V. K. Singh, and P. K. Atrey, *Cyber bullying detection using social and textual analysis.* Proceedings of the 3rd International Workshop on Socially-Aware Multimedia. ACM, 3–6., 2014.

[15] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. Pearson, 2009.

[16] B. Pang and L. Lee, *Opinion mining and sentiment analysis. Foundations and Trends® in information retrieval*, 1st ed. PO Box, 2008.

[17] W. Zhang, T. Yoshida, and X. Tang, "A comparative study of tf* idf, lsi and multi-words for text classification," 2011, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S0957417410008626

[18] A. G. Jivani, "A comparative study of stemming algorithms," 2011, consultado en septiembre de 2023. [En línea]. Disponible en: https://kenbenoit.net/assets/courses/tcd2014qta/readings/Jivani_ijcta2011020632.pdf

[19] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," 2003, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf?ref=https://githubhelp.com

[20] A. L. Samuel, "Some studies in machine learning using the game of checkers. ibm journal of research and development," 1959, consultado en septiembre de 2023. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/5392560

[21] H. Hosseinmardi, S. Arredondo, R. Ibn, R. Han, Q. Lv, and S. Mishra, "Detection of cyberbullying incidents on the instagram social network," 2015, consultado en septiembre de 2023. [En línea]. Disponible en: https://arxiv.org/abs/1503.03909

[22] J. Shlens, "A tutorial on principal component analysis," 2014, consultado en septiembre de 2023. [En línea]. Disponible en: https://arxiv.org/abs/1404.1100

[23] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," 1998, consultado en septiembre de 2023. [En línea]. Disponible en: http://yangli-feasibility.com/home/classes/lfd2022fall/media/aaaiws98.pdf

[24] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, 1st ed. John Wiley Sons, 2013.

[25] L. Breiman, "Random forests," 2001, consultado en septiembre de 2023. [En línea]. Disponible en: https://link.springer.com/article/10.1023/a:1010933404324

[26] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," 1997, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S002200009791504X

[27] R. E. Schapire, "Explaining adaboost," 1997, consultado en septiembre de 2023. [En línea]. Disponible en: https://link.springer.com/chapter/10.1007/978-3-642-41136-6_5

[28] C. J. Hsieh, K. W. Chang, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm." 2008, consultado en septiembre de 2023. [En línea]. Disponible en: https://dl.acm.org/doi/abs/10.1145/1390156.1390208

[29] H. Hosseinmardi, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Prediction of cyberbullying incidents in a media-based social network," 2016, consultado en septiembre de 2023. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/7752233/authors#authors

[30] "Boe 2023," 2023, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.boe.es/boe/dias/2023/07/26/pdfs/BOE-A-2023-17238.pdf

[31] P. Yi and A. Zubiaga, "Session-based cyberbullying detection in social media: A survey," 2021, consultado en septiembre de 2023. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S2468696423000095

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

[33] H. P. Vinutha, B. Poornima, and B. M. Sagar, "Detection of outliers using interquartile range technique from intrusion dataset," 2018, consultado en septiembre de 2023. [En línea]. Disponible en: https://link.springer.com/chapter/10.1007/978-981-10-7563-6_53

[34] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-based anomaly detection," 2012, consultado en septiembre de 2023. [En línea]. Disponible en: https://dl.acm.org/doi/abs/10.1145/2133360.2133363

[35] S. J. Yen and Y. S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," 2006, consultado en septiembre de 2023. [En línea]. Disponible en: https://link.springer.com/chapter/10.1007/978-3-540-37256-1_89

[36] 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, consultado en septiembre de 2023. [En línea]. Disponible en: https://link.springer.com/article/10.1007/s13278-016-0398-x

[37] P. Chapman, J. Clintin, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, *CRISP-DM 1.0: Step-by-step data mining guide.* SPSS inc, 9(13), 1-73., 2000.

[38] M. García, D. Iglesias, and P. Martín, "Uso de algoritmos de aprendizaje máquina para la detección de ciberacoso en redes sociales," 2023, consultado en septiembre de 2023. [En línea]. Disponible en: https://ruc.udc.es/dspace/handle/2183/33398