Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería de Sistemas y Computación

Fecha: 09/02/24

Autores: Kevin Steven Ocampo Morales
Juan Sebastian Arango Salazar

Nombre del Proyecto de Grado: Identificación de lenguaje ofensivo en mensajes de texto, utilizando técnicas de aprendizaje automático.

Director: María Constanza Pabón Burbano.

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Firma de Director(a) del Proyecto de Grado

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y Computación.

Dr. Hernan Camilo Rocha Niño
Decano de la Facultad de Ingeniería

ING. GERARDO MAURICIO SARRIA
Director Carrera Ingeniería Sistemas y Computación.

ING. María Constanza Pabón Burbano
Director(a) Trabajo

ING. Mario Julián Mora
Jurado 1

ING. Julian Gil Gonzalez
Jurado 2

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias.
Ingeniería de Sistemas y Computación.
Trabajo de Grado.

Identificación de lenguaje ofensivo en mensajes de texto, utilizando técnicas de aprendizaje automático.

Kevin Steven Ocampo Morales
Juan Sebastian Arango Salazar

Director: Dra. María Constanza Pabón Burbano

14 de noviembre del 2023

Santiago de Cali, 14 de noviembre del 2023.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que los estudiantes de Ingeniería de Sistemas y Computación Kevin Steven Ocampo Morales (cod: 8943835) y Juan Sebastian Arango Salazar(cod: 8943011) trabajan bajo mi dirección en el proyecto de grado titulado "Identificación de lenguaje ofensivo en mensajes de texto, utilizando técnicas de aprendizaje automático.".

Atentamente,

Dra. María Constanza Pabón Burbano

Santiago de Cali, 14 de noviembre del 2023.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

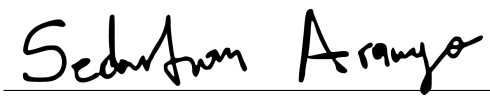
Cordial Saludo.

Nos permitimos presentar a su consideración el anteproyecto de grado titulado "Identificación de lenguaje ofensivo en mensajes de texto, utilizando técnicas de aprendizaje automático." con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, damos fe que entendemos y conocemos las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

Kevin Steven Ocampo Morales
Código: 8943835



Juan Sebastian Arango Salazar
Código: 8943011

Resumen

Las redes sociales son aplicaciones o plataformas digitales que permiten el intercambio de información entre individuos, la creación de comunidades, sitios de expresión, entre otros. Gracias a ellas, las personas tienen un medio para compartir sus ideas u opiniones sobre un tema en particular, ejerciendo así su derecho universal a la libre expresión. Entre ellas encontramos sitios web como Twitter, donde los usuarios pueden manifestarse mediante textos, contenido audiovisual, imágenes o emojis. Debido a la libertad que ofrecen, el anonimato y la polarización cada vez más marcada en la sociedad, se da lugar al uso del lenguaje ofensivo y contenido tóxico o negativo que algunos sujetos generan hacia otros con la intención de ofender, lastimar o discriminar.

Es por esto que el análisis de sentimientos, en conjunto con el aprendizaje automático, se presentan como un área de estudio y una herramienta muy útil para detectar y contrarrestar esta problemática. Se usan técnicas para determinar el tipo de intención que tiene el contenido que comparten las personas en esta red social, distinguiendo entre positivas o negativas. En particular, nuestro proyecto busca, mediante la utilización de este sistema previamente mencionado, desarrollar modelos que permitan clasificar mensajes de texto provenientes de Twitter (tweets) en ofensivos y no ofensivos, usando el análisis de sentimientos.

Palabras Clave: Redes sociales, libertad de expresión, lenguaje ofensivo, análisis de sentimientos, aprendizaje automático, clasificación.

Summary

Social networks are digital applications or platforms that allow the exchange of information between individuals, the creation of communities, sites of expression, among others. Thanks to them, people have a means to share their ideas or opinions on a particular topic, thus exercising their universal right to free expression. These include websites such as Twitter, where users can express themselves through text, audiovisual content, images or emojis. Due to the freedom they offer, the anonymity and the increasingly marked polarization in society, they give rise to the use of offensive language and toxic or negative content that some subjects generate towards others with the intention of offending, hurting or discriminating.

This is why sentiment analysis, in conjunction with machine learning, is presented as an area of study and a very useful tool to detect and counteract this problem. Techniques are used to determine the type of intention that has the content shared by people in this social network, distinguishing between positive or negative. In particular, our project seeks, through the use of this previously mentioned system, to develop models that allow classifying text messages coming from Twitter (tweets) into offensive and non-offensive, using sentiment analysis.

Key words: Social networks, freedom of speech, offensive language, sentiment analysis, machine learning, classification.

Índice general

Introducción

En los últimos años, el uso de foros en línea y sitios web ha venido en un incremento constante y potencialmente positivo tanto para las empresas como para la sociedad actual, esto debido a que la tecnología está cada vez más presente en la vida cotidiana. Twitter, una red social muy conocida y relevante a nivel mundial en la actualidad, cuenta con un tráfico de no menos de 350.000 tweets cada 60 segundos [1] y más de 320 millones de usuarios activos mensualmente [2]. Estos medios digitales permiten que las personas estén conectadas a través de internet, permitiéndoles expresar de manera abierta y pública sus opiniones e ideas sobre temas presentes o de gran controversia, a la vez que se informan o comunican a otros sobre lo que acontece en el día a nivel mundial o personal.

Es por este amplio grado de libertad que tiene la sociedad en estos medios que, en muchos casos, debido al gran volumen de datos y el contenido ofensivo que puede existir en este, ocasiona que la supervisión y control llegue a ser mínimo. Adicional a esto, la máscara del anonimato que los mismos ofrecen [1] incentiva a los individuos a llevar a cabo este tipo de comportamientos y comentarios con el fin de ofender, lastimar o discriminar, afectando la autoestima de las personas y su estado mental, así mismo generando un impacto negativo en la sociedad [1]. Debido a que los usuarios en las redes sociales producen grandes volúmenes de información, es imposible monitorear de forma manual este tipo de contenido; por esto es que se han desarrollado modelos de aprendizaje automático que, de manera autónoma, lo detecten [3].

En consecuencia, la identificación oportuna del lenguaje ofensivo se convierte en un reto importante por tratar. Es por esto que nuestro proyecto busca entrenar modelos de aprendizaje automático supervisado que identifiquen y clasifiquen de manera apropiada los tweets entre ofensivos y no ofensivos, además de evaluar el desempeño de estos y analizar los resultados obtenidos.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

Las redes sociales son plataformas digitales que permiten el intercambio de información entre usuarios. Únicamente se considera red social a la plataforma que es abierta para los usuarios, en donde cada persona posea un perfil propio en el que tiene la libertad de subir cualquier tipo de información con respecto a sí mismo, compartir contenidos tales como comentarios, fotos, videos, entre otros. También permiten promocionar marcas y acciones que se integran a un gran abanico de posibilidades que las mismas ofrecen [5]. Todo esto con el fin de que los demás individuos del sitio web tengan permitido visualizar e interactuar entre ellos.

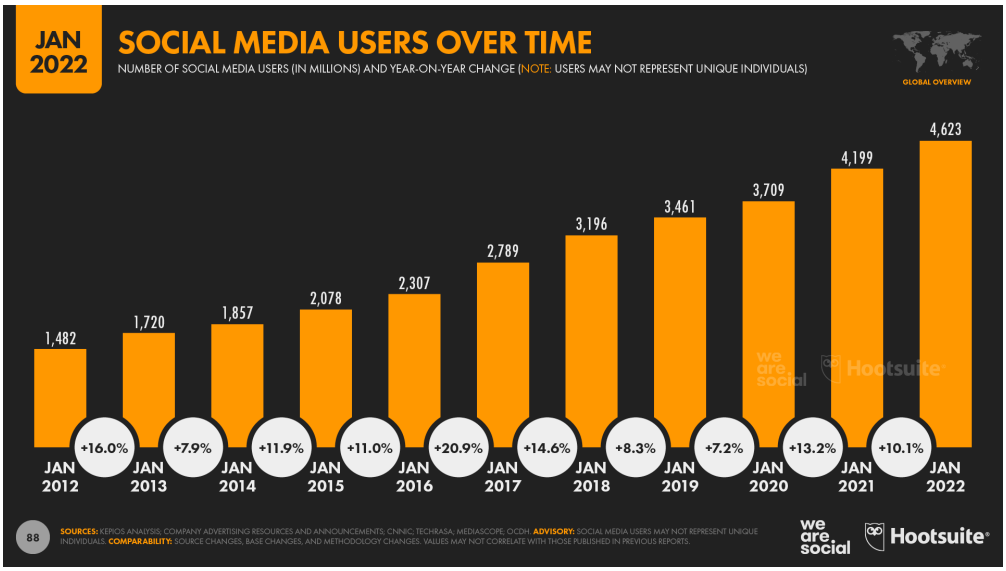


Figura 1.1: DataReportal (2022),"*DIGITAL 2022: GLOBAL OVERVIEW REPORT*", recuperado de https://datareportal.com/reports/digital-2022-global-overview-report

Como se evidencia en la Figura 1.1, el incremento de usuarios que hacen uso de las redes sociales a medida que pasa el tiempo se hace más notorio. Es por ello que estas plataformas digitales se han convertido en una parte esencial, incluso en algunos casos una necesidad, en nuestra vida diaria.

Internet y todo su entorno de comunicación en línea proveen a las personas una gran herramienta con propósitos o efectos personales, sociales, democráticos y/o políticos. Debido a la cantidad de redes sociales que tienen disponibles los usuarios en la actualidad, los individuos tienen una mayor facilidad de entrar en contacto con otros e interactuar entre uno o varios a la vez [6]. Ahora bien, estas plataformas digitales tienen una gran cantidad de problemas, como la privacidad de los consumidores, la posible suplantación de identidad, la dependencia a las redes, en algunos casos, la tergiversación del concepto del derecho de la libertad de expresión por parte de sujetos que hacen uso de esta, entre muchas otras.

La libertad de expresión como derecho fundamental permite, mediante la comunicación, manifestar sus ideas y opiniones, así como el libre actuar de las personas, sin distinción de género, etnia o religión. Lo cual indica que se encuentra desligado de cualquier limitante que pueda tener un individuo, esto quiere decir que los sujetos pueden hacer saber sus pensamientos sin importar el contenido de estos [7]. A pesar de que lo mencionado previamente sugiere que no existen límites definidos para la libre expresión de ideas o pensamientos por parte de una persona, es importante destacar que existen restricciones legales que pueden llevar a la reflexión antes de compartir ciertos comentarios en público. Un ejemplo de ello es la difamación de un individuo de manera pública sin pruebas sólidas o mediante acusaciones falsas, lo cual puede conllevar consecuencias legales. Es importante reconocer entonces que, gracias a la libre expresión, los usuarios de las redes sociales pueden opinar en gran medida de cualquier tema de la forma que ellos quieran. Plataformas como Twitter, buscando respetar este derecho universal, poseen muy pocas políticas de restricción sobre las publicaciones que se pueden realizar en este medio.

Estos espacios digitales han dado un golpe de revolución en la historia humana, pero una alta cantidad de los usuarios pertenecientes a estas plataformas están haciendo uso inadecuado de las mismas, realizando discursos de odio y lastimando psicológicamente a demás consumidores de estas. Debido a la forma irrespetuosa de expresarse de algunos, que intencionalmente basan sus comentarios en lastimar a otros, permite que se afecte “al mundo actual a nivel micro (individual), meso (grupal) y macro (social)”, generando un impacto negativo en la sociedad [8].

1.1.1. Formulación

¿Cómo utilizar las técnicas de aprendizaje automático supervisado para entrenar modelos que detecten contenido ofensivo en mensajes de texto (tweets)?

1.1.2. Sistematización

- ¿Cómo determinar y limpiar la información no relevante de los mensajes de texto (tweets)?
- ¿Cómo realizar el preprocesamiento de los datos?
- ¿Qué técnicas de aprendizaje supervisado serán utilizadas para detectar el contenido ofensivo en los mensajes de texto (tweets)?

- ¿Cómo estimar los hiperparámetros en las técnicas que lo requieran con el fin de mejorar el desempeño de los modelos?
- ¿Cómo realizar la evaluación y/o comparación de los modelos?

1.2. Objetivos

1.2.1. Objetivo General

Construir un modelo con técnicas de aprendizaje automático supervisado para la detección de contenido ofensivo en mensajes de texto (tweets).

1.2.2. Objetivos Específicos

- Identificar la información no relevante en los mensajes de texto (tweets) y realizar su limpieza, por ejemplo: caracteres especiales, emojis, signos de puntuación, símbolos, etc.
- Realizar las tareas de preprocesamiento de los mensajes.
- Analizar y escoger tres técnicas de aprendizaje automático supervisado que serán utilizadas para construir los modelos, con el fin de clasificar los mensajes de texto (tweets) en ofensivos y no ofensivos.
- Optimizar los hiperparámetros en las técnicas que lo requieran, de manera que mejore el desempeño de los modelos.
- Realizar la evaluación y comparación del desempeño de los modelos construidos.

1.3. Justificación

Dado el continuo crecimiento de la tecnología y las redes sociales, es cada vez más común el uso de estos medios para buscar un espacio de esparcimiento y compartir experiencias, ideas u opiniones de todo tipo. Desafortunadamente, hay quienes utilizan estos medios como herramienta para la expresión de comportamientos de odio y discriminación, a través de comentarios, publicaciones, videos, imágenes o demás. Es por esto por lo que Twitter, dentro de sus herramientas tecnológicas, tiene implementadas inteligencias artificiales con el fin de poder detectar y combatir el contenido inapropiado en su plataforma, todo esto gracias a la tecnología IBM Watson que pone a disposición de la compañía un conjunto de herramientas de procesamiento de lenguaje natural, inferencia de entonación y significado de imágenes para rastrear y eliminar mensajes con este tipo de contenido [2]. De esta manera, se busca generar un entorno social menos agresivo y más amigable entre los usuarios activos, pero esto no es suficiente, dado que muchos tweets consiguen superar este filtro y no logran ser identificados como contenido ofensivo debido a que no violan directamente el reglamento y política de la aplicación, la cual se basa en la no intervención de las publicaciones de sus usuarios a

menos que lleve consigo una intencionalidad de abuso y/o acoso. Esto da lugar a contenido cargado con una connotación ofensiva de manera implícita o referencial y que puede afectar a usuarios que se sientan aludidos en dicho contenido.

Ya se han realizado anteriormente varias aproximaciones y estudios en torno al aprendizaje automático aplicado en esta área, cada uno enfocado en detectar un sector en particular del contenido tóxico y abusivo. En particular, uno de estos trabajos es el realizado en 2018 titulado "Predicting the Type and Target of Offensive Posts in Social Media" [4], en donde haciendo uso de técnicas de aprendizaje automático detectaron, clasificaron y etiquetaron tweets de una forma más general, enfocándose en proveer la información sobre el objetivo (individuo o grupo) del contenido ofensivo, generando un nuevo esquema jerárquico de tres niveles de categorías [4]:

- **A. Detección del lenguaje ofensivo**
  - NOT - No ofensivo
  - OFF - Ofensivo
- **Categorización del lenguaje ofensivo**
  - TIN - Contenido ofensivo dirigido a un individuo, grupo u otro.
  - UNT - Contenido ofensivo sin un objetivo.
- **Identificación del objetivo del lenguaje ofensivo**
  - IND - Individuo
  - GRP - Grupo
  - OTH - No pertenece a ninguna de las dos categorías anteriores, sino a una organización, situación, acontecimiento o problema.

Nuestro proyecto busca explorar, estudiar, evaluar y analizar los modelos creados a partir de las técnicas de aprendizaje automático supervisado que permitan reconocer y clasificar mensajes de texto de la red social Twitter.

1.4. Delimitaciones y Alcances

- El lenguaje de programación donde se desarrollará todo el proceso será Python.
- Se construirán y compararán tres modelos usando técnicas distintas de aprendizaje automático supervisado.
- Se utilizará un data set previamente etiquetado. La información almacenada aquí, se encuentra en el idioma ingles y se realizará una debida separación de los datos para el entrenamiento y evaluación de los modelos.

Capítulo 2

Marco de referencia

2.1. Áreas Temáticas

Este proyecto abarca principalmente dos áreas temáticas, las cuales son el aprendizaje automático y el análisis de sentimientos. El aprendizaje automático es un área de las ciencias de la computación y en particular, buscamos aplicar técnicas de aprendizaje automático supervisado. Por otro lado, el análisis de sentimientos es una manera de procesar el lenguaje natural y sus representaciones, para determinar comportamientos o estados de ánimo de un individuo frente a un producto, servicio, publicación, comentario, etc.

2.2. Marco Teórico

2.2.1. Conceptos sobre aprendizaje automático y análisis de sentimientos

Aprendizaje automático: Es un área de las ciencias de la computación, la cual se basa en que los dispositivos aprendan gracias a los datos. Se basa en algoritmos genéricos, los cuales pueden extraer patrones de los datos en los que estén operando [10].

Aprendizaje supervisado: Es un método de análisis que se aplica a los datos que se encuentran previamente etiquetados. Las técnicas de este método se entrenan usando estos datos, permitiendo identificar y clasificar los nuevos datos desconocidos para el modelo[10].

Procesamiento de lenguaje natural: Conocido también por su nombre en inglés, Natural Language Processing (NLP) es una rama de la inteligencia artificial que se enfoca en la interacción entre las computadoras y el lenguaje humano. Esta disciplina se basa en una variedad de técnicas y herramientas que abarcan un amplio campo, adaptándose a diversas alternativas para abordar problemas específicos. Los métodos empleados en NLP varían según la naturaleza de los datos en cuestión y los objetivos del proyecto. Esto permite abordar tareas como el análisis de sentimientos, la generación de texto, la clasificación de texto y muchas otras aplicaciones relacionadas [14].

Análisis de sentimientos: Es el uso de técnicas, las cuales buscan determinar y clasificar el estado emocional o la opinión en un texto. Esto se realiza gracias al uso de procesamiento de lenguaje natural, lingüística y aprendizaje automático, los cuales son usados para analizar y clasificar [15].

Sub muestreo (undersampling): Es una técnica que se usa en el aprendizaje automático, esto se hace para trabajar el problema del desequilibrio de los datos, lo que hace es extraer información más detallada de los conjuntos de datos y hacer un balance [25].

Para este algoritmo existen dos métodos proveídos en la librería sklearn, el primero conocido por el nombre de "Método de selección de prototipos" y el otro "Método de generación de prototipos". El primero es un método que selecciona una submuestra de la clase mayoritaria que es lo más similar posible a la clase minoritaria, mientras que el segundo es un método que genera nuevos ejemplos a partir de la clase minoritaria. En particular, para este proyecto el método utilizado fue el de selección de prototipos, ya que permitía reducir el tamaño de la clase mayoritaria de un conjunto de datos, S, seleccionando de forma aleatoria muestras para eliminarlas, hasta que el número de muestras para la clase mayoritaria fuese igual al número de muestras de la clase minoritaria. Además, la complejidad del algoritmo es de $O(nlogn)$, donde n es el tamaño del conjunto de datos original, S. Esta complejidad se debe a que el algoritmo necesita ordenar el conjunto de datos por la clase mayoritaria, antes de poder seleccionar las muestras que se van a eliminar [39].

Sobre muestreo (oversampling): Es otra técnica que utilizada en el aprendizaje automático para trabajar el problema del desequilibrio de datos, permitiendo también hacer un balance del conjunto de datos [25].

Lo anterior lo realiza creando datos sintéticos o copiando datos de la clase minoritaria para aumentar el número de muestras con respecto a la clase mayoritaria, un elemento a considerar es que este algoritmo en su forma base lo que hace es crear datos duplicados de los ya existentes, si esto resulta ser un problema, se tienen dos caminos, usar el parámetro "shrinkage" para suavizar esta generación y evitar tantos duplicados del mismo registro o usar la creación de datos sintéticos con "SMOTE" o "ADASYN". Además, la complejidad del algoritmo base (sin generar datos sintéticos ni suavizar la generación) es de $O(n)$, donde n es el tamaño del conjunto de datos. Esta complejidad se debe a que el algoritmo necesita recorrer el conjunto de datos una sola vez, para seleccionar las muestras que se van a replicar [40].

Búsqueda de hiperparámetros: Es el proceso de encontrar los valores óptimos de los hiperparámetros de un modelo de aprendizaje automático. Los hiperparámetros son parámetros que controlan el comportamiento del modelo, pero que no se aprenden directamente a partir de los datos.[38]. Esta etapa es de suma importancia si lo que se desea es que el modelo mejore su rendimiento y trabaje de manera óptima para la tarea que se esté desarrollando.

Grid Search: También conocida como búsqueda en grilla, es una técnica de búsqueda de hiperparámetros que evalúa el rendimiento del modelo para una cuadrícula de valores predefinidos para los hiperparámetros. [38]. Es un método bastante sencillo de implementar o usar a partir de una librería y permite explorar todo un espacio de búsqueda, resultando efectiva a la hora de encontrar los valores óptimos para los hiperparámetros de manera exhaustiva.

2.2.2. Conceptos sobre el procesamiento de lenguaje natural

Expresiones regulares: Las expresiones regulares son patrones de búsqueda que son usadas para encontrar caracteres alfanuméricos, espacios, signos de puntuación y/o caracteres especiales dentro de un texto. Se utilizan comúnmente en la programación, el procesamiento de texto y manipulación de datos [16].

Tokenización: La tokenización es el proceso de descomponer un texto, una frase o una oración en unidades más pequeñas llamadas tokens. Estos tokens pueden ser palabras, caracteres, números, etc. La tokenización es una técnica fundamental en el procesamiento de texto, ya que permite convertirlo en un formato admisible que permita ser procesado y analizado por algoritmos de aprendizaje automático [17].

Palabras vacías (Stopwords): Son palabras comunes que son eliminadas del análisis de texto debido a que no contribuyen de manera significativa para el análisis del mismo. Es una técnica que se usa comúnmente en el procesamiento de lenguaje natural para mejorar eficiencia y precisión de los algoritmos utilizados [18].

Lematización: Proceso el cual permite reducir las palabras a su forma más básica para facilitar el análisis. Esta técnica tiene en cuenta el contexto de la palabra y usa lingüística para determinar la forma base. Se puede aplicar esta técnica sobre adjetivos, sustantivos y/o verbos [18].

Vocabulario: El vocabulario es un catálogo de palabras que se han extraído del conjunto de datos. Es utilizado para representar el contenido del documento como un vector de características [22].

Matriz de términos de documentos: Es una representación numérica del conjunto de datos que es utilizado. Es una matriz, donde cada las filas son el conjunto de datos y las columnas son las palabras únicas del conjunto de datos [22].

Normalización del texto: Es un proceso el cual modifica el texto para que todo el contenido que está en el este de una forma estándar. Esto se refiere a que debe estar en el mismo idioma, con todo el texto en mayúscula o minúscula, eliminación de signos de puntuación y tildes, etc. La normalización es una de las etapas importantes en el preprocesamiento de datos [23].

Regularización/normalización: Es un proceso el cual permite que las características tengan escalas similares y estén bajo un rango determinado. Esto se realiza debido a que algunos modelos son susceptibles a que las características estén repartidas bajo un rango amplio, haciendo que trabajen de una manera no adecuada [24].

2.2.3. Conceptos sobre la ingeniería de características

Ingeniería de características (Feature Engineering): Es el proceso de transformar los datos brutos en características que sean relevantes para el aprendizaje automático, estas características son variables que representan los atributos de los datos. Un buen conjunto de características permite que los modelos de aprendizaje automático puedan aprender patrones y relaciones en los datos que de otro modo serían difíciles de detectar [36].

Bolsa de palabras: Es una técnica usada en aprendizaje automático, el cual se encarga de representar el conjunto de documentos como un conjunto de palabras, además de contabilizar su frecuencia de aparición [21]. Al no requerir un orden en particular para el almacenamiento de las palabras, es común que su uso sea en tareas de procesamiento de lenguaje natural, tales como la clasificación de texto, la recuperación de información y el resumen de texto.

Tf-idf: Es una técnica de extracción de características que mide estadísticamente el valor de una palabra en un documento, a su vez, dicho documento se encuentra en una conjunto de documentos más grande (conjunto de datos) del mismo tipo. Esta técnica se aplica, ya que permite dar equilibrio a la frecuencia de aparición de las palabras [21]. Al combinar la frecuencia de una palabra en un documento con su frecuencia inversa en un corpus, permite identificar palabras que son importantes para un documento en particular, incluso si esas palabras no son muy frecuentes.

Word embedding: Es una técnica de extracción de características que representa palabras en forma de vectores numéricos, estos vectores buscan que los modelos comprendan de mejor forma los significados de las palabras y relacionen de una mejor manera los términos [31]. Esta técnica resulta muy eficaz a la hora de trabajar tareas de procesamiento de lenguaje natural y sobre todo, con modelos complejos como las redes neuronales, ya que son capaces de procesar información más compleja como la semántica de las palabras.

2.2.4. Modelos de aprendizaje automático y métricas de evaluación

En el siguiente módulo, se analizarán las tres técnicas de aprendizaje automático que se utilizan en el proyecto: Support Vector Machines (SVM), Redes Neuronales (NN) y Naive Bayes. Se proporcionará una introducción sobre cada técnica, así como una descripción de las implementaciones de Naive Bayes utilizadas en el proyecto: complementario y multinomial.

Support Vector Machines (SVM): Las máquinas de soporte vectorial (SVM) son un algoritmo de aprendizaje supervisado que se usa en gran medida para tareas de clasificación y regresión de datos, se caracteriza por ser altamente efectivo en la clasificación de datos y tiene bastante uso en el análisis de opiniones y sentimientos [12]. Este algoritmo se basa principalmente en encontrar un hiperplano óptimo que divide los datos en diferentes categorías, con el fin de realizar el proceso de clasificación, se altera el algoritmo buscando que el hiperplano maximice la distancia entre las muestras de clases [30].

Para este proyecto se utilizó la implementación "Classifier" (SVC) de sklearn con el parámetro de kernel. El kernel es una función que busca mapear los datos de entrada a un espacio dimensional mayor, esto permite que los datos puedan volverse linealmente separables, el kernel calcula el producto escalar entre los datos mapeados en el nuevo espacio. Debido a que la gran mayoría de los problemas son no lineales, el kernel permite que las máquinas de soporte vectorial sean flexibles y logren abordar estos problemas de una manera más óptima [33].

Es fundamental considerar la elección del kernel en el algoritmo, ya que esta elección puede afectar de manera importante en la complejidad temporal de las máquinas de soporte vectorial (SVM). Aunque existen diversas variantes de SVM, la complejidad promedio de estas variantes se encuentra en función del número de muestras de los datos y la cantidad de características resultantes de este conjunto. Esta complejidad, se expresa en notación de O grande y es del orden de $O(n \cdot d)$, donde n representa la cantidad de muestras y d es el número de características.

Esta formulación destaca la relación entre el tamaño del conjunto de datos y la complejidad computacional, evidenciando el impacto significativo en el rendimiento y la eficiencia del proceso de entrenamiento de las SVM, como también la importancia de la representación de las características.

Redes neuronales: Para hablar de las redes neuronales convolucionales, primero se debe saber que son las redes neuronales. Las redes neuronales son un campo de estudio de la inteligencia artificial y el aprendizaje automático que se basa en el funcionamiento y trabajo del cerebro humano [34]. Se encuentran compuestas con conjuntos de unidades de procesamiento interconectadas que simulan las neuronas, estas neuronas trabajan en conjunto para realizar tareas de procesamiento de datos. Estas tienen la capacidad para manejar y descubrir patrones complejos en los datos y es utilizado para el análisis de sentimientos [13].

Particularmente, las Redes Neuronales Convolucionales (CNN) demuestran su eficacia en el procesamiento de datos estructurados, como imágenes, señales o texto. Esto se logra mediante adaptaciones específicas y las hace altamente versátiles en la captura de características abstractas y complejas. Es precisamente esta capacidad la que llevó a la elección de las CNN para la tarea del proyecto. Se identificó que tanto los datos utilizados como la tarea en desarrollo se centraban en la captura de características locales. Por lo tanto, las CNN resultaron ser una elección adecuada debido a su habilidad para destacar patrones locales de relevancia en los textos.

Las CNN usan capas de convolución en lugar de las capas tradicionales totalmente conectadas, esto permite que se puedan aplicar filtros a ciertas regiones de las entradas y extraer de esta manera características locales relevantes que fluyen a través de la red y se refinan para tener una mejor precisión en la tarea que se realiza. Adicionalmente, se benefician de capas de agrupamiento con la finalidad de reducir la dimensionalidad de las características extraídas, permitiendo preservar la información relevante y haciendo que la red sea más robusta ante los cambios y variaciones de datos

[35].

Además de los aspectos anteriores, las redes neuronales convolucionales (CNN) tienen dos tipos de complejidad que deben tenerse en cuenta al trabajar con ellas: la complejidad computacional y la complejidad paramétrica [41].

La complejidad computacional se refiere a la cantidad de operaciones matemáticas que realiza una CNN para procesar una entrada. Estas operaciones incluyen convoluciones, activaciones, pooling y multiplicaciones de matrices. La complejidad computacional se puede denotar como $O(N^2 * M * K * F)$, donde:

- N es el tamaño de los datos de entrada
- M es el número de canales de la entrada
- K es el tamaño del kernel de convolución
- F es el número de filtros de convolución

Por su parte, la complejidad paramétrica, por otro lado, se refiere al número de parámetros que una CNN necesita aprender. Estos parámetros incluyen los pesos y sesgos de los filtros de convolución, así como los parámetros de las funciones de activación y pooling. La complejidad paramétrica se puede denotar como $O(F * M * K^2)$, donde:

- F es el número de filtros de convolución
- M es el número de canales de la entrada
- K es el tamaño del kernel de convolución

Naive Bayes: En aprendizaje automático, Naive Bayes es un algoritmo probabilístico conocido y muy utilizado a causa de su simplicidad y eficacia en tareas de clasificación. Existen variantes de Naive Bayes, estas variantes son adaptadas a diferentes tipos de datos y tareas específicas para mejorar los resultados. En particular, para este proyecto se usan dos de las variantes más comunes de Naive Bayes, que son Naive Bayes multinomial y Naive Bayes complementario [29].

Este algoritmo comparte similitudes en términos de complejidad temporal con las Máquinas de Soporte Vectorial (SVM). Su complejidad está determinada por la cantidad de datos y la cantidad de características presentes en el conjunto de entrenamiento, y se sitúa en $O(n \cdot d)$, donde n representa la cantidad de datos y d es el número de características. Se debe tener en cuenta que la aplicación de diferentes técnicas de ingeniería de características puede afectar tanto de manera positiva como negativa a esta complejidad y por ende el funcionamiento óptimo del algoritmo [42].

- **Naive Bayes multinomial:** Es un modelo de lenguaje uni grama basado en que las características son representadas como un conteo discreto, esto se refiere al conteo de aparición de una palabra o la frecuencia de aparición de un término en un documento [32]. Se utiliza este algoritmo debido a que posee un enfoque ampliamente usado en problemas de clasificación de texto y análisis de sentimientos.

  Utiliza la ley de probabilidad multinomial para hacer cálculos sobre probabilidades condicionales de pertenencia a clases, dada una clase sobre un conjunto de características. Luego de que se aplica el conteo discreto, el modelo estima las probabilidades de ocurrencia para finalizar el entrenamiento y poder realizar predicciones sobre nuevas instancias [22].

- **Naive Bayes complementario:** Es una variante del algoritmo de Naive Bayes Multinomial y adecuado para conjunto de datos desequilibrados, esto es importante a tener en cuenta, ya que el algoritmo estándar de Naive Bayes supone que las clases del conjunto de datos son independientes y bien distribuidos. Es por ello que este algoritmo deja a un lado esa suposición y se encarga de considerar esa dependencia y desequilibrio entre las clases y características del conjunto de datos para así, por medio de la probabilidad condicional complementaria, mejorar la precisión en los casos donde se presenten estos problemas [30].

  Utiliza la estimación de las probabilidades condicionales de los términos sobre la pertenencia a una clase dada la presencia o ausencia de características. El Naive Bayes complementario calcula la probabilidad de que el término pertenezca a todas las clases debido al desequilibrio que pueda existir en el conjunto de datos.

Además de las técnicas enunciadas anteriormente, también se incluyen métricas de evaluación para analizar su funcionamiento. Se introducirán, a manera de conceptualización, cuatro métricas de evaluación de modelos de aprendizaje automático: Precision, Recall, F1-score y Accuracy.

Precision: Es una métrica de evaluación que mide la proporción de los valores positivos predichos y que son realmente positivos. Se calcula como la proporción de verdaderos positivos (TP) sobre el número total de valores positivos predichos (TP + FP) [37]. Es decir, mide el grado de error causado por los falsos positivos.

Recall: Es una métrica de evaluación que mide la proporción de valores positivos reales que fueron correctamente clasificados como positivos. Se calcula como la proporción de verdaderos positivos (TP) sobre el número total de valores positivos reales (TP + FN)[37]. Es decir, mide el grado de error causado por los falsos negativos.

F1-Score: Es una métrica de evaluación que combina la precisión y el recall. Se calcula como la media armónica de la precisión y el recall. El F1-score es una métrica útil para problemas en los que es importante equilibrar la precisión y el recall [37]. Esta métrica es usada cuando falsos positivos y/o falsos negativos no son deseables y entre más cercano se encuentre su valor a 1, mejor será el modelo.

Accuracy: Es una métrica de evaluación que mide la proporción de observaciones correctamente clasificadas. Se calcula como la proporción de observaciones correctamente clasificadas sobre el total de observaciones [37]. Esta métrica es simple y fácil de entender, pero puede ser engañosa en problemas con clases desequilibradas.

2.3. Trabajos Relacionados

- El texto "Predicting the Type and Target of Offensive Posts in Social Media" [4] se centra en el desarrollo de un modelo para predecir el tipo y el objetivo de los mensajes ofensivos en las redes sociales. Se menciona el uso y entrenamiento de 3 modelos, SVM, BiLSTM y CNN, utilizando un conjunto de datos etiquetados manualmente con 14,100 mensajes ofensivos y un conjunto de datos no etiquetados con más de 290,000 mensajes, todos en el idioma inglés. Los resultados del estudio mostraron que el modelo CNN propuesto logró una precisión promedio del 78 % en la clasificación del tipo de mensaje ofensivo y una precisión promedio del 81 % en la clasificación del objetivo del mensaje ofensivo en comparación a los demás modelos.

- El texto "Sentimental Analysis of Twitter Data using Machine Learning Algorithms" [9] habla sobre la utilización de algoritmos de aprendizaje automático para el análisis de sentimientos en Twitter. El estudio explica lo importante que es realizar un preprocesamiento adecuado a los datos, esto con el motivo de eliminar la información irrelevante y que los algoritmos de clasificación trabajen de una manera más adecuada. En el estudio se hizo uso de 7 técnicas en las que se encuentran, regresión logística, bosques aleatorios, k-vecinos, entre otros. El resultado de la investigación concluyo que el algoritmo SVC lineal tuvo el mejor rendimiento con una precisión del 98,83 % en el entrenamiento y un 77,52 % en los de práctica, además se descubrió el buen rendimiento de clasificador de votación y la regresión logística.

- El texto "Evaluation of Machine Learning Methods in Sentimental Analysis" [11] es un estudio sobre diversas técnicas de aprendizaje automático utilizadas en el análisis de sentimientos de textos. Los datos utilizados para el estudio se encontraban previamente etiquetados como positivo, negativo o neutro en función de su polaridad y en el idioma inglés. Adicionalmente, se utilizaron múltiples técnicas de aprendizaje automático para entrenar y clasificar el conjunto de datos, entre ellos están: Naive Bayes, máquinas de soporte vectorial (SVM), regresión logística y redes neuronales, entre otros. En cuanto a los resultados, se evaluaron y compararon todos los modelos construidos por los distintos algoritmos, mostrando en términos de precisión y velocidad que SVM y Regresión Logística superaban a los demás. Sin embargo, también se mencionó que cada algoritmo tiene ventajas y desventajas, considerando clave el hecho de adaptarlos al problema que implique el análisis de sentimientos.

- El texto "Deep Learning for Automated Sentiment Analysis of Social Media" [20] habla sobre la exploración de técnicas de aprendizaje profundo para el análisis de sentimientos en las redes sociales. Empieza con el análisis profundo de los datos con los que se van a trabajar para luego

realizar el preprocesamiento de datos y la selección del modelo de aprendizaje automático. En el texto se discute sobre los diferentes enfoques que se pueden utilizar, como las redes neuronales convolucionales y las redes recurrentes, las redes de memoria a largo plazo y las unidades recurrentes controladas. Al finalizar se logra detallar que estos modelos son eficaces para el análisis de sentimientos, además de destacar lo importante que es la limpieza de datos como el uso de lematización y el stemming.

- El texto "Hybrid Model for Social Media Sentiment Analysis for Indonesian Text" [19] habla sobre el análisis de sentimientos de la opinión pública sobre el gobierno (positivo, negativo y neutro) en indonesia usando métodos basados en léxico y máxima entropía. Este estudio utiliza un conjunto de 6000 textos de redes sociales. Los métodos usados son el preprocesamiento, las clasificaciones basadas en léxico, la capacitación de conjuntos de datos y la clasificación. Esta investigación hace uso de un modelo híbrido que se compone en 2 etapas:

  - Pre procesar el texto basado en reglas y extraer características relevantes.
  - Enfoque basado en aprendizaje automático para clasificar el sentimiento de cada texto.

  Los resultados arrojados luego de aplicar el modelo fue una puntuación de precisión del 84,31 % en comparación con estudios pasados. Este resultado fue bueno y se plantea las importantes aplicaciones de este modelo.

- El texto "A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter" [26] habla sobre las técnicas de preprocesamiento que se usan para el conjunto de datos. Todo empieza con las operaciones básicas de limpieza, el cual consiste en eliminar elementos de los datos que son irrelevantes y que pueden causar ruido al conjunto de datos y luego transformando los datos a un estándar, posteriormente realiza el proceso de revisión de ortográfica y stemming para la reducción de la entropía. Por último, la eliminación de stopwords, ya que pueden dar una falsa clasificación de los datos. En el texto se logra concluir que el uso de las técnicas de preprocesamiento mejoraron significativamente el rendimiento de los modelos.

- El texto "Deep Learning for Hate Speech Detection in Tweets" [27] habla sobre el uso de deep learning para detectar y clasificar discursos de odio en tweets, un problema que engloba el fenómeno del lenguaje ofensivo y resulta sumamente importante es lo que se conoce como discurso de odio, debido a que la revisión de estos tweets no es escalable y debe ser automatizada. En el documento se explica que su trabajo se basa específicamente en la clasificación de tweets entre diferentes categorías o clases, tales como: racistas, sexistas o ninguno de los dos. También mencionan los modelos usados y las técnicas de feature engineering para extraer características, además muestran los resultados obtenidos de recall, precision y F1-score a partir de todas las combinaciones de los modelos entrenados y las técnicas aplicadas.

- El texto "Twitter Sentiment Analysis using Deep Learning" [28] habla sobre el análisis de sentimientos de tweets, a los cuales se les aplican varios algoritmos de aprendizaje automático, al final del proceso los modelos clasifican los tweets entre sí es un tweet positivo o negativo.

El trabajo toca temas que van desde el preprocesamiento y todas las funciones necesarias para que los datos sean entradas válidas para los modelos, como también la descripción de los modelos y su aplicación. Se logra concluir que las características y la extracción de las mismas, resulta muy importante en el proceso, ya que puede cambiar la manera de como el modelo clasifica la información.

Capítulo 3

Desarrollo del proyecto

3.1. Descripción del conjunto de datos

Para llevar a cabo el desarrollo del proyecto, se emplea un conjunto de datos adquirido de la competencia llamada "*OffensEval 2019: OffensEval: Identifying and Categorizing Offensive Language in Social Media (SemEval 2019 - Task 6)*", la cual se desarrolló en la plataforma *CodaLab* durante el período comprendido entre septiembre de 2018 y febrero de 2019. El conjunto de datos en cuestión consta de 13,240 tweets previamente etiquetados. Según la información proporcionada por la misma competencia, estos tweets fueron recolectados y catalogados en el mes de marzo de 2018. Es importante destacar que los tweets están redactados en el idioma inglés.

El propósito principal de esta competencia es la identificación y categorización del lenguaje ofensivo, la cual se organiza en tres niveles jerárquicos. En el primer nivel, conocido como "Subtarea A", se busca identificar el lenguaje ofensivo en su forma más básica. En el segundo nivel, denominado "Subtarea B", se procede a categorizar el tipo de ofensividad presente en los mensajes. Por último, en el tercer nivel, referido como "Subtarea C", tiene como propósito el identificar el objetivo o destinatario de la ofensa en los mensajes analizados.

id	tweet	subtask_a	subtask_b	subtask_c
86426	@USER She should ask a few native Americans what their take on this is.	OFF	UNT	NULL
90194	@USER @USER Go home you're drunk!!! @USER #MAGA #Trump2020 👊🇺🇸👊 URL	OFF	TIN	IND
43605	@USER @USER Obama wanted liberals & illegals to move into red states	NOT	NULL	NULL
97670	@USER Liberals are all Kookoo !!!	OFF	TIN	OTH
77444	@USER @USER Oh noes! Tough shit.	OFF	UNT	NULL
45157	@USER Buy more icecream!!!	NOT	NULL	NULL

Figura 2.1: Ejemplo del conjunto de datos

Como se observa en la imagen 2.1, el conjunto de datos se distribuye en 5 columnas y estas son:

- **id:** Columna que contiene los identificadores de los tweets, el tipo de dato es numérico.
- **tweet:** Columna que contiene tweets, el tipo de dato es texto. Una particularidad a destacar acerca de los datos contenidos en esta columna es que todos ellos inician con el texto "@USER". Esta característica se debe a que las menciones de otros usuarios en los tweets se han mantenido de forma anónima, por lo tanto, se han sustituido por la palabra anterior. De manera similar, los enlaces a otros sitios web o publicaciones también se han reemplazado por la palabra "URL".

Es importante subrayar que estos "textos basura" serán gestionados de manera apropiada durante la fase de limpieza de datos con el fin de llevar a cabo un procesamiento adecuado.

- **subtask_a:** Columna que identifica si un tweet es o no ofensivo, el tipo de dato es categórico.
- **subtask_b:** Columna que identifica a quien va dirigido el contenido ofensivo, el tipo de dato es categórico.
- **subtask_c:** Columna que identifica el objetivo del contenido ofensivo, el tipo de dato es categórico.

Teniendo en cuenta que el proyecto busca realizar una clasificación de los tweets en ofensivos y no ofensivos, se hace uso de solamente de las primeras 3 columnas (id, tweet, subtask_a), el subtask_a tiene como etiquetas de categorización las palabras "NOT" y "OFF", donde la categoría "NOT" significa que el contenido del tweet no es ofensivo, mientras que la categoría "OFF" significa que el contenido es ofensivo.

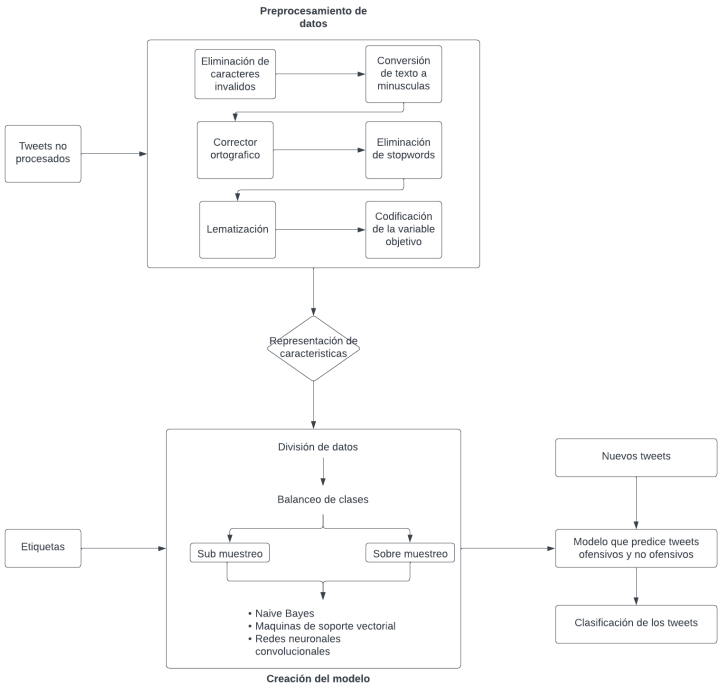


Figura 2.2: Ruta de diseño para los modelos

Como se ilustra en la Figura 2.2, se representa a gran escala el proceso de desarrollo de los modelos, que se inicia con la descomposición del conjunto de datos. A continuación, se lleva a cabo una fase de limpieza y preprocesamiento de los datos, en la cual se incluyen varios pasos, tales como

la eliminación de caracteres inválidos, la limpieza del texto y su normalización, con el objetivo de permitir que los algoritmos comprendan de manera más efectiva y la calidad del conjunto de datos mejore. Al abordar el desbalance de clases, se generan dos conjuntos adicionales debido al desequilibrio existente entre la clase de tweets no ofensivos y los tweets ofensivos. Finalmente, el modelo resultante se obtiene tras atravesar varias etapas.

En cuanto a los hiperparámetros, después de realizar un análisis de rendimiento sobre el modelo, se estudian los posibles hiperparámetros que podrían mejorar la clasificación del modelo. Mediante el uso de grid search y la constante iteración en búsqueda de mejora, se puede lograr obtener un modelo que clasifique de manera más efectiva. Estos pasos se abordarán de manera más detallada en las secciones siguientes, proporcionando así una comprensión más profunda del proceso de creación de los modelos finales.

3.2. Preprocesamiento

El preprocesamiento de datos es un paso fundamental para el desarrollo de modelos de aprendizaje automático. Este proceso nos permite garantizar la calidad y fiabilidad de los resultados obtenidos al entrenar y evaluar modelos. En esta sección se analizan los métodos, herramientas, procedimientos y/o técnicas que permiten manipular, seleccionar, transformar, reducir y limpiar el conjunto de datos con el que trabajaron los modelos.

Por lo tanto, para realizar un correcto y eficaz preprocesamiento para el proyecto, debemos considerar escenarios como la remoción de datos no deseados, normalización o transformación del texto, ortografía e idioma, tokenización, entre otros, permitiendo sentar bases sólidas con el propósito de desarrollar modelos precisos y fiables en el contexto del aprendizaje automático y de la finalidad del proyecto, buscando garantizar que los datos de entrada a los modelos sean coherentes y adecuados.

3.2.1. Librerías

Para este proyecto se hizo uso de variedad de librerías y/o herramientas de procesamiento de lenguaje natural y aprendizaje automático. Cada una suministra un conjunto de utilidades, funciones o métodos esenciales para la manipulación de los datos y el desarrollo de los modelos. A continuación, se mencionan y describen cada una de las librerías y/o herramientas utilizadas:

- **Pandas:** Es una herramienta muy útil para el procesamiento y análisis de datos en Python. Posee funcionalidades como el cargar, almacenar y manipular información sobre estructuras de datos. Además, de permitir el análisis estadístico y realizar transformaciones a los datos.
- **NLTK (Natural Language Toolkit):** Es una librería para procesamiento de lenguaje natural en Python. Posee funcionalidades como la tokenización, etiquetados, análisis de sentimientos, lematización, etc. Además, también tiene funcionalidades de recursos lingüísticos, diccionarios y desarrollo de aplicaciones de lenguaje natural.

- **Re (Regular expression):** Es una librería que funciona para el manejo y uso de expresiones regulares. Posee funcionalidades como búsqueda, extracción, manipulación, conversión de patrones en textos. Es una librería perteneciente a la biblioteca estándar de Python, por ende no necesita instalación aparte.

- **Spellchecker:** Es una librería para corregir ortografía en un texto. Esta tiene la funcionalidad de verificar la correcta ortografía de las palabras en los textos, brindar posibles candidatos a palabras mal escritas, indicar la frecuencia de palabras no reconocidas, etc.

- **Wordninja:** Es una librería que se utiliza para separar palabras que se encuentran como una sola palabra en un texto ("untexto" = "un", "texto"). Utiliza reglas para lograr separar las palabras y funciona solamente en el idioma inglés.

- **Sklearn (Scikit-learn):** Es una librería muy útil para el desarrollo de modelos de aprendizaje automático en Python. Permite construir instancias de modelos con los diferentes algoritmos existentes y entrenarlos. Provee herramientas para el preprocesamiento de los datos, modelado, evaluación y optimización de los modelos.

- **Numpy:** Es una librería muy reconocida y útil de Python. Posee una gran capacidad para manejar grandes volúmenes de datos, realizar operaciones matemáticas complejas sobre su particular estructura de datos y contener métodos/algoritmos para diversas tareas. Además de permitir su integración con otras herramientas como Pandas o Matplotlib.

- **Imblearn (Imbalanced-learn):** Es una librería muy útil cuando se requiere solucionar el problema del desbalance de clases en el aprendizaje automático, esto se refiere a cuando una clase de interés tiene pocos datos frente a las demás clases del conjunto de datos, permitiendo tener dificultados al entrenar de manera efectiva los modelos. Esta librería provee métodos para aplicar técnicas como el Undersampling (Sub muestreo) y Oversampling (Sobre muestreo) para mitigar este desbalance.

- **Gensim:** Es una librería de aprendizaje automático diseñada para ser de mucha utilidad a la hora de procesar y evaluar enormes colecciones de texto, como corpus de documentos. Entre sus herramientas se encuentran diversas técnicas de aprendizaje automático para desarrollar modelos de análisis de texto, como modelos vectoriales de palabras, modelos de modelado de temas y modelos de similitud de documentos.

- **Spacy:** Es una librería de procesamiento de lenguaje natural con herramientas muy útiles para el procesamiento de texto, tales como la tokenización, etiquetado de partes del discurso, análisis sintáctico, reconocimiento de entidades nombradas, análisis de sentimientos y aprendizaje de modelos. Esta librería es muy popular por dar prioridad a la eficiencia, escalabilidad y usabilidad a la hora de implementarla en tareas de aprendizaje automático.

- **Keras:** Es una librería de aprendizaje automático que sirve para construir y entrenar modelos de aprendizaje profundo como redes neuronales convolucionales (CNN), redes neuronales

recurrentes (RNN) y redes neuronales totalmente conectadas (DNN). Esta librería es muy reconocida en este apartado del aprendizaje automático debido a su simplicidad, modularidad y usabilidad. Posee distintas herramientas para el entrenamiento y validación de modelos de aprendizaje profundo, la visualización de la arquitectura de modelos e incluso contiene modelos preentrenados para tareas que lo requieran.

3.2.2. Limpieza y normalización del texto

Uno de los problemas principales para entrenar modelos que involucran procesamiento de texto y que funcionen de la manera esperada, es que los datos sean homogéneos y tengan el mismo estándar, es por esto que es importante efectuar la limpieza y normalización del texto. En este apartado se describen los procesos realizados.

1. **Eliminación de caracteres inválidos:** En muchas ocasiones, los datos que entran en el modelo contienen información que es irrelevante y no suman importancia para el entrenamiento de los mismos, es por esto que se hace la remoción de caracteres inválidos o de poco valor para el estudio. En particular, para el proyecto se usan las expresiones regulares para identificar y lograr eliminar de los tweets emojis, símbolos, pictogramas, caracteres especiales y numéricos. Para este paso se utilizó la librería "re", la cual permite aplicar una expresión regular a todo el conjunto de datos. Adicionalmente, en este proceso se utiliza la función nativa de Python llamada "strip" que permite eliminar el carácter especial de salto de línea o espacio en blanco al inicio o final de un texto.

2. **Conversión del texto a minúsculas:** Para entrenar los modelos es de suma importancia que los datos de entrada se encuentren en un mismo formato, en el caso en cuestión que sea en formato de letras minúsculas. La finalidad es la reducción de dimensiones del texto o simplificación de lo que procesara el modelo. Para este proceso se utiliza una función nativa de Python llamada "lower" que convierte todos los datos de texto a minúsculas.

3. **Corrector ortográfico:** Debido a la magnitud de los datos y a su naturaleza (tweets de una red social), no se puede asegurar que no existan palabras escritas incorrectamente o en otro idioma, este problema da cabida a un mal entrenamiento del modelo y por ende una mala evaluación y conclusión del mismo. Es por esto que se utiliza la librería spellchecker, la cual evalúa la construcción sintáctica de la palabra y la comprara con un diccionario predefinido en la librería misma, permitiendo identificar y corregir la ortografía de las palabras mal escritas o brindar posibles reescrituras de la misma, buscando mantener su significado en el idioma.

4. **Eliminación de palabras vacías (Stopwords):** Igual que con los caracteres especiales, números, etc. Existen palabras del idioma que no suman importancia en el significado del texto a la hora de entrenar el modelo. Es por ello que pueden ser eliminadas, por tanto, para este proyecto se decidió realizar dos aproximaciones con los datos disponibles: Una primera aproximación eliminando dichas palabras (denotada por el nombre **Data 1**) y otra en la que

no se eliminen (denotada por el nombre **Data 2**), esto con la finalidad de ver si agregaban o no valor al momento de que el modelo se entrene y se evalúe en la tarea de clasificación de tweets en lenguaje ofensivo y no ofensivo. Para este proceso se utiliza la librería NLTK, la cual establece que dichas palabras en un conjunto de palabras ya predefinido y que son denominadas palabras vacías, haciendo un mapeo de nuestro conjunto de datos respecto a las mismas y que de esa manera se descarten las que sean consideradas palabras vacías.

5. **Lematización:** Debido a que las palabras tienen conjugaciones y esto puede impactar en los resultados del modelo, se aplica la técnica llamada lematización, con el fin de normalizar o transformar las palabras del conjunto de datos a una forma más base que facilite su análisis, pero sobre todo, buscando mantener su significado semántico. Esto significa que tomando varias palabras que se encuentran conjugadas y transformándolas, permita la reducción de la densidad de los datos y se logren identificar de una manera más adecuada, para ello se utiliza NLTK, estableciendo sobre que categorías gramaticales se quiere aplicar la técnica: Adjetivos (ADJ_SAT), sustantivos (NOUN) y verbos (VERB). Es importante destacar que existe otra técnica ampliamente conocida como "Derivación" (o Stemming en inglés) que persigue una finalidad similar a la lematización, pero con diferencias fundamentales que permiten determinar cuál de ellas elegir según la tarea a desarrollar.

   El stemming, en particular, se caracteriza por ser una técnica considerablemente más simple que la lematización, dado que se enfoca en reducir las palabras a su forma raíz mediante la eliminación de sufijos o prefijos. Esto implica que el proceso es más rápido, pero conlleva ciertas imprecisiones en cuanto al significado de las palabras. En algunos casos particulares, incluso puede generar palabras que no son reales, lo que afecta la interpretación semántica de un texto y empeorando de esta manera el rendimiento de los modelos al analizar la información.

   Por esta razón, en el proyecto se ha optado por la técnica de lematización, ya que esta permite mantener una mayor precisión semántica y lógica en las palabras. Dado que la tarea a desarrollar requiere conservar los datos de manera concisa y precisa en términos de significado, la lematización se considera más adecuada para alcanzar este objetivo.

6. **Codificación variable objetivo:** Esta etapa resulta clave para poder llevar a cabo el entrenamiento de los modelos, ya que los algoritmos de aprendizaje automático entienden datos numéricos y no categóricos. Las etiquetas de los tweets del conjunto de datos son un tipo de dato categórico ("NOT", "OFF"), por lo que es necesario codificarlas a un tipo de dato numérico antes de poder entrenar un modelo. Se utiliza el método "LabelEncoder" de la librería Sklearn y su módulo de preprocesamiento con el fin de generar una codificación binaria simple para dichas categorías. Determinando entonces que el número 1, sería el valor asignado para la etiqueta "OFF" y el número 0, sería el valor asignado para la etiqueta "NOT".

3.2.3. Tokenización

La tokenización es una parte importante del preprocesamiento, los algoritmos que se utilizan en este proyecto requieren como entrada un vector de palabras, ya que se trata de procesamiento de lenguaje natural (NLP, por sus siglas en inglés). Para llevar a cabo este proceso, se debe dividir el texto en partes más pequeñas denominadas tokens, los cuales son cada una de las palabras del tweet que se está procesando. La técnica que se usa es la tokenización basada en palabras, la cual permite representar de forma vectorial los datos, haciendo que sea más fácil el análisis por parte del algoritmo. Se toma cada tweet como una cadena de texto y por medio de una función nativa de Python llamada split() hace que toda la cadena de texto se convierta en una lista/vector de palabras, además se usa la librería wordninja con su método split() para que palabras que se encontraban juntas se conviertan en palabras separadas e.g., (helloDog = hello Dog).

En el proceso previamente mencionado sobre la separación de palabras que estaban unidas y la corrección ortográfica, surgieron una serie de casos particulares en ciertos términos que presentaban problemas al realizar estas tareas. Se lograron identificar dos casos en particular. El primero se refiere a situaciones en las que, por ejemplo, se tenía el término "f**king". Al eliminar los caracteres especiales y realizar la correspondiente tokenización (con la ayuda de Word Ninja), el resultado que se obtiene es "[f, king]". Esto ocasiona que, al aplicar la corrección ortográfica, la palabra que se obtiene sea completamente diferente a la palabra original, lo que resulta en la pérdida del significado semántico del tweet.

El otro caso se presentaba cuando un tweet contenía el conocido apóstrofe (') del idioma inglés. Por ejemplo, si se tenía el término "should'veTaken", al eliminar los caracteres especiales y realizar la corrección ortográfica, se obtenía un valor "None", lo cual también conducía a una pérdida del significado semántico del tweet.

Por esta razón, se hizo necesario realizar un tratamiento específico de estos casos para evitar la pérdida de información valiosa para los modelos. Este proceso implicó identificar problemas particulares en los tweets y aplicar una secuencia de corrección y separación diferente para estos casos. La secuencia consistió principalmente en reconocer cuándo se trataba de uno de los casos mencionados y manejarlo en un orden específico. Por ejemplo, si un término contenía caracteres especiales, como en el primer ejemplo, se procedía primero a limpiar los caracteres especiales, luego se aplicaba la corrección ortográfica y, finalmente, se realizaba la tokenización. Todo esto con el propósito de preservar el significado semántico tanto del término como del tweet en su conjunto.

Este enfoque permitió reducir la pérdida de información y evitar un impacto negativo en el rendimiento de los modelos.

3.2.4. Ingeniería de características (Feature engineering)

La ingeniería de características desempeña un papel crucial en el preprocesamiento de datos en tareas relacionadas con el procesamiento de lenguaje natural. En este proceso, se busca seleccionar, transformar o generar nuevas características (variables) a partir de los datos disponibles con el objetivo de mejorar el rendimiento de los modelos. Esta práctica tiene como finalidad extraer información relevante y obtener representaciones adecuadas del texto a procesar, permitiendo a los modelos comprender de manera más efectiva el lenguaje humano.

El fundamento para aplicar la ingeniería de características radica en la necesidad de obtener representaciones apropiadas del texto. Esto implica la captura de la semántica y el contexto del contenido textual procesado, así como la posibilidad de reducir la dimensionalidad de las variables. La ingeniería de características juega un papel fundamental en la transformación del texto no estructurado en información procesable y estructurada en formato numérico para que los modelos puedan trabajar de manera óptima y lograr un mejor rendimiento en la comprensión del lenguaje humano.

Por esta razón, a pesar de las tareas de preprocesamiento previamente ejecutadas, resulta esencial llevar a cabo este procedimiento. La aplicación de este proceso en el proyecto contribuye a potenciar el desempeño de los modelos en diversas áreas, tales como la precisión, la capacidad predictiva, la mitigación del sobreajuste, y el proceso de aprendizaje, entre otros aspectos. En particular, las técnicas implementadas para los modelos trabajados son:

1. **Bolsa de palabras (BOW).**

   Para la implementación de esta técnica se hizo uso de la librería "sklearn" en su módulo "feature extraction text" y se utilizaron las implementaciones denominadas "CountVectorizer" para la creación y definición de la bolsa y "TfidfTransformer" para la normalización. Dicha técnica consta de las siguientes fases:

   - **Creación del vocabulario:** A partir de todos los tweets ya pre procesados, se crea un diccionario con todas las palabras y a cada palabra se le asigna un valor numérico único, esto con el fin de diferenciar una palabra de otra.

   - **Creación de la matriz de términos de documentos:** Con el diccionario ya creado, se construye la matriz de términos de documentos, en donde el tamaño está dado por el número de tweets (filas) y la cantidad de palabras del diccionario (columnas), además, la matriz es inicializada completamente con valores 0 en cada una de sus posiciones. Los valores en cada posición de la matriz indican la frecuencia de las palabras en los tweets.

   - **Normalización:** Para finalizar se usa la normalización con el propósito de igualar la frecuencia de los datos. En particular, se utiliza la normalización L2 en lugar de la L1, ya

que es frecuentemente usada en procesamiento de texto, permitiendo la reducción de la varianza de los datos, así como también trabajar de manera precisa los valores atípicos.

2. **TF-IDF (Term Frequency-Inverse Document Frequency).**

   En cuanto a esta técnica, también se empleó una librería específica para su implementación. Se hizo uso de la librería "sklearn" en su módulo "feature extraction text" y se utilizó la implementación denominada "TfidfVectorizer". Esta técnica se compone de las siguientes fases:

   - **TF**: En la primera fase, conocida como "TF" (Frecuencia del Término), se emprende la tarea de calcular la frecuencia de aparición de cada palabra en un documento específico. Este cálculo se realiza contando cuántas veces se repite cada palabra en el documento y relacionándolo con el número total de palabras presentes en dicho documento. Además, se lleva a cabo un proceso de normalización con el propósito de mitigar cualquier sesgo en la frecuencia del término. Este proceso se efectúa dividiendo el número de ocurrencias del término en el documento entre el número total de términos que componen el documento.

   - **IDF**: En esta segunda fase, denominada "IDF" (Frecuencia Inversa del Documento), se tiene como objetivo calcular la importancia relativa de un término en el conjunto de documentos. Para lograr esto, se evalúa cuán raro o común es un término en el conjunto de documentos. Este cálculo se realiza mediante la aplicación de un logaritmo al cociente entre el número total de documentos en el corpus y la cantidad de documentos que contienen el término en cuestión.

   - **TF-IDF**: En la fase final de este proceso, se realiza el cálculo que combina el valor "TF" obtenido en la primera etapa con el valor "IDF" calculado en la segunda. Esta operación permite determinar el valor o la importancia relativa "TF-IDF" de cada término en cada documento.

3. **Word Embedding.**

   Esta técnica avanzada implica la representación de las palabras mediante vectores numéricos, lo que posibilita la captura de relaciones tanto semánticas como sintácticas entre las palabras. En este contexto, se ha empleado la librería "gensim" y su módulo "downloader" y usando a función "api" para adquirir un modelo previamente entrenado, el cual contiene un conjunto de términos y contextos que resultan fundamentales para la identificación de patrones, contextos, semántica y relaciones entre los términos en proceso de análisis.

   Este enfoque es especialmente valioso en el contexto de las redes neuronales convolucionales (CNN), ya que contribuye a obtener una representación más precisa de los datos en forma de vectores numéricos densos. Esta técnica se compone de las siguientes fases

- **Tokenización**: Como se ha explicado previamente en una sección anterior, la tokenización se define como el proceso de descomponer las palabras de un término en unidades más elementales conocidas como tokens.

- **Creacion del vocabulario**: En esta etapa del proceso, se genera un vocabulario que engloba todas las palabras únicas que se encuentran en el corpus de texto. Cada una de estas palabras se vincula a un índice exclusivo.

- **Entrenamiento del modelo**: En esta etapa, se emplean modelos preentrenados para procesar el vocabulario y el corpus de texto con el fin de descubrir relaciones semánticas y contextuales. Estos modelos generan vectores numéricos densos para cada término, permitiendo que la red neuronal optimice la tarea de clasificación de manera más eficiente.

  Dos modelos preentrenados comúnmente utilizados en esta fase son "Word2Vec" y "GloVe". "Word2Vec" se basa en redes neuronales y se enfoca en la predicción de contexto, mientras que "GloVe" utiliza estadísticas de co-ocurrencia y técnicas matemáticas para aprender representaciones vectoriales de palabras. Luego de investigar sobre ambos, se llegó a la conclusión de que para este proyecto en particular, se optaría por utilizar "GloVe" debido a su simplicidad y a su eficacia probada en tareas relacionadas con la relación de palabras con su contexto en un corpus.

- **Generación de vectores de palabras**: En esta fase final, una vez que el modelo "GloVe" ha sido entrenado, es posible generar los vectores correspondientes a cada término del vocabulario. Estos vectores proporcionan una representación de cada término en un espacio vectorial multidimensional, donde sus ubicaciones posibilitan la identificación de similitudes semánticas y relaciones contextuales.

3.2.5. División de datos y balanceo de clases

La división de datos es un proceso fundamental en el aprendizaje automático, este nos permite evaluar el rendimiento del modelo de una manera objetiva y no caer en el sobreajuste o la memorización de los datos al entrenar el modelo. Para realizar esto se divide el conjunto de datos en dos o más subconjuntos y así reutilizarlo en diferentes fases del proceso. En general, los subconjuntos se denotan como: entrenamiento, validación y prueba (o testeo). Cada uno de los subconjuntos tiene su propósito, el de entrenamiento permite ajustar los parámetros del modelo, por su parte, los de validación ayudan a identificar los ajustes a realizar de los hiperparámetros del modelo y los de prueba/testeo proporcionan la evaluación final del rendimiento del modelo utilizando datos que no son conocidos para el mismo y que, por tanto, no fueron utilizados ni para entrenar ni validar.

En particular, para el desarrollo del proyecto, se procedió a dividir el conjunto de datos en dos partes: el 80 % de los datos se destinaron al entrenamiento de cada uno de los modelos, y el 20 %

restante se reservó para llevar a cabo el proceso de validación y optimización de los hiperparámetros. Esta distribución sigue una convención común en el área y se aplica tanto a los conjuntos de datos "Data 1" como "Data 2", los cuales ya fueron mencionados previamente en la sección de preprocesamiento sobre las stopwords.

Adicionalmente, existe un conjunto de datos diferente al utilizado para generar "Data 1" y "Data 2", por tanto, no hace parte del proceso de entrenamiento y validación. Este conjunto de datos es obtenido de la competencia mencionada anteriormente en el apartado "Descripción del conjunto de datos" y está compuesto por 860 registros, dichos registros se dividen en 620 tweets no ofensivos y 240 tweets ofensivos. Su propósito es llevar a cabo la fase final de prueba y evaluación de los modelos que han arrojado mejores resultados. Buscando observar cómo se desempeñan los modelos frente a datos completamente desconocidos.

Por otro lado, el balanceo de clases es una técnica cuya finalidad es el equiparar el número de muestras de una o más clases con respecto a las demás en el conjunto de datos de entrenamiento cuando estos se encuentran desequilibrados. El desequilibrio de datos puede repercutir negativamente en los modelos, ya que los mismos están diseñados para lograr una alta precisión global y si no se soluciona este problema (en caso de estar presente), puede conducir a una mala clasificación de las clases con menos datos, disminuyendo el rendimiento del modelo.

Existen varias técnicas para abordar esta problemática, siendo las más comunes el sobremuestreo y el submuestreo. En el contexto de este proyecto, se observó un desequilibrio en las clases al dividir los datos, como se mencionó previamente. En el conjunto de entrenamiento, se registraron 7107 tweets no ofensivos y 3485 tweets ofensivos, mientras que en el conjunto de validación se encontraron 1733 tweets no ofensivos y 915 tweets ofensivos.

Para mitigar este desequilibrio y evitar sesgar la capacidad de clasificación del modelo en la tarea, se optó por aplicar tanto sobremuestreo, como submuestreo a los dos conjuntos de datos de esta fase (Data1 y Data2) y para cada modelo implementado. Se utilizó la librería "imblearn" y su implementación correspondiente para cada técnica con el fin de equilibrar los datos de entrenamiento y validación. Por un lado, se aplicó la técnica de submuestreo para reducir el tamaño de las clases mayoritarias. Por otro lado, y de manera independiente, se implementó el sobremuestreo para aumentar el tamaño de las clases minoritarias. Esta estrategia permitió obtener, para cada modelo, conjunto de datos y técnica aplicada, un conjunto de datos más balanceado y, en consecuencia, evaluar el rendimiento de los modelos en condiciones más óptimas.

3.3. Desarrollo de los modelos

El entrenamiento de modelos de aprendizaje automático con un conjunto de datos sobre un contexto en particular, permite que algoritmos aprendan a partir de estos datos y puedan realizar tareas específicas sin ser programados de forma explícita. En particular, para este proyecto, se toma el enfoque del aprendizaje automático supervisado, el cual implica que se tiene dicho conjunto de datos etiquetado, es decir, se conoce cuál es su clasificación o característica dado el contexto que se estudia. Durante el proceso de entrenamiento, un modelo de aprendizaje automático analiza el conjunto de datos de entrada y ajusta parámetros para encontrar patrones, clasificar o predecir sobre el conjunto de datos, esto permite que el modelo realice la misma tarea con un conjunto diferente de datos.

Para evaluar el rendimiento de los modelos y principalmente identificar en qué aspectos enfocar los esfuerzos de optimización, a fin de mejorar su eficacia en la realización de la tarea específica, se recurre al uso de métricas de evaluación. Entre estas métricas, destacan las siguientes: Precision, Recall, F1-Score y Accuracy. La utilización de estas métricas posibilita la comparación de los distintos modelos y sus experimentos en función de su desempeño en los conjuntos de datos específicos (Data 1 y Data 2) y las técnicas aplicadas.

En este proyecto, se entrenan y estudian los resultados de tres algoritmos en el enfoque del aprendizaje automático supervisado, los cuales son: Naive Bayes, Máquinas de Soporte Vectorial (SVM) y Redes Neuronales Convolucionales (CNN), a continuación se presentarán los resultados iniciales obtenidos para cada algoritmo y los experimentos realizados en cada uno, analizando y evaluando a partir de las métricas anteriormente mencionadas.

3.3.1. Naive Bayes

Para la creación de los modelos “Multinomial” y “Complementario”, se hizo uso de las implementaciones proporcionadas por “sklearn” con sus parámetros por defecto, buscando mantener el modelo lo más simple y básico posible para optimizarlo posteriormente en la etapa de búsqueda de hiperparámetros. Los parámetros por defecto que utilizan los modelos son los siguientes: “alpha” con un valor por defecto de 1.

En particular, para cada implementación, los modelos base se refieren a aquellas a las que no se les aplica ninguna técnica de balanceo de clases. Por otro lado, los modelos con submuestreo o sobremuestreo se refieren a las implementaciones, las cuales se les aplican alguna de las técnicas de balanceo de clase en cuestión. Todo lo anterior se realiza para el conjunto de datos “Data 1” y “Data 2” y con la técnica de bolsa de palabras para la representación del texto.

Durante esta fase, se obtuvieron los siguientes resultados:

3.3.1.1. Naive Bayes Multinomial

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.86	0.20	0.32	0.71
Base - Data 2	0.87	0.14	0.24	0.70
Submuestreo - Data 1	0.66	0.75	0.70	0.68
Submuestreo - Data 2	0.66	0.78	0.72	0.69
Sobremuestreo - Data 1	0.69	0.70	0.70	0.70
Sobremuestreo - Data 2	0.68	0.71	0.69	0.69

Cuadro 3.1: Tabla de resultados para cada experimento de esta implementación con respecto a las métricas de evaluación

3.3.1.2. Naive Bayes Complementario

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.61	0.53	0.57	0.72
Base - Data 2	0.71	0.27	0.39	0.72
Submuestreo - Data 1	0.68	0.77	0.72	0.70
Submuestreo - Data 2	0.66	0.77	0.71	0.68
Sobremuestreo - Data 1	0.69	0.68	0.68	0.69
Sobremuestreo - Data 2	0.69	0.72	0.70	0.70

Cuadro 3.2: Tabla de resultados para cada experimento de esta implementación con respecto a las métricas de evaluación

Se observa de manera general un rendimiento similar en los modelos tanto de la variante de naive bayes multinomial como en el complementario, a excepciones de los modelos bases en sus variantes con el conjunto de datos “Data 1” y el conjunto de datos “Data 2”, aunque estos 4 modelos tienen “Precision” alta, su “Recall” es bastante bajo, esto significa que identifica correctamente los tweets ofensivos que clasifica, pero deja muchos tweets ofensivos sin clasificar correctamente, provocando falsos negativos. Por otra parte, se observa que, por parte de los modelos en los que se utilizaron submuestreo y sobremuestreo, existe una mejora general en todas las métricas en comparación con los modelos base. Esto sugiere que el balanceo de los datos, ayudó a mejorar el rendimiento de los modelos. Para terminar, los modelos de “Submuestreo - Data 2” tienen buenos resultados en las métricas en general, lo que indica un equilibrio entre la identificación y clasificación de tweets ofensivos y no ofensivos, así como su capacidad para capturar la mayoría de los tweets ofensivos.

En general, es importante considerar el equilibrio entre “Precision” y “Recall”; es decir, el “F1-Score”, al evaluar el rendimiento de un modelo de clasificación, especialmente cuando se trata de datos desbalanceados, como en este caso.

3.3.2. Máquinas de soporte vectorial (SVM)

Para la creación de los modelos "Classifier", se hizo uso de la implementación proporcionada por "sklearn", con los siguientes parámetros por defecto:"C" con un valor por defecto de 1, "kernel" con un valor por defecto de "rbf" y "gamma" con un valor por defecto de "scale". El objetivo era mantener el modelo lo más simple y básico posible para optimizarlo posteriormente en la etapa de búsqueda de hiperparámetros.

En particular, para cada implementación, los modelos base se refieren a aquellas a las que no se les aplica ninguna técnica de balanceo de clases. Por otro lado, los modelos con submuestreo o sobremuestreo se refieren a las implementaciones, las cuales se les aplican alguna de las técnicas de balanceo de clase en cuestión. Todo lo anterior se realiza para el conjunto de datos "Data 1" y "Data 2" y con la técnica de bolsa de palabras para la representación del texto.

Durante esta fase, se obtuvieron los siguientes resultados:

3.3.2.1. SVC con bolsa de palabras

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.78	0.46	0.58	0.77
Base - Data 2	0.79	0.46	0.58	0.77
Submuestreo - Data 1	0.77	0.67	0.71	0.73
Submuestreo - Data 2	0.73	0.67	0.70	0.71
Sobremuestreo - Data 1	0.78	0.63	0.70	0.73
Sobremuestreo - Data 2	0.77	0.65	0.71	0.72

Cuadro 3.3: Tabla de resultados para cada experimento de esta implementación con respecto a las métricas de evaluación

3.3.2.2. SVC con TF-IDF

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.78	0.46	0.58	0.77
Base - Data 2	0.80	0.46	0.58	0.77
Submuestreo - Data 1	0.76	0.65	0.70	0.72
Submuestreo - Data 2	0.75	0.66	0.70	0.72
Sobremuestreo - Data 1	0.78	0.62	0.69	0.72
Sobremuestreo - Data 2	0.77	0.63	0.69	0.72

Cuadro 3.4: Tabla de resultados para cada experimento de esta implementación con respecto a las métricas de evaluación

Se observa de manera general un rendimiento similar en los modelos tanto con bolsa de palabras como en TF-IDF, a excepción de los modelos base con los conjuntos de datos “Data 1” y “Data 2”, aunque estos 4 modelos tienen “Precision” alta, su “Recall” es bastante bajo, esto significa que identifica correctamente los tweets ofensivos que clasifica, pero deja muchos tweets ofensivos sin clasificar correctamente, provocando falsos negativos, teniendo resultados similares a los obtenidos en Naive Bayes. Aunque en los modelos de submuestreo y sobremuestreo mejoraron significativamente frente a los modelos base, el “Recall” aún tiene un posible rango de mejora. Por otra parte, los modelos con bolsa de palabras tienen un desempeño ligeramente mejor que los modelos con TF-IDF en términos de “Precision”, “Recall” y “F1-Score”. Esto sugiere que la representación de los textos utilizando bolsa de palabras puede estar capturando un poco mejor las características relevantes para la clasificación de tweets.

3.3.3. Redes neuronales convolucionales (CNN)

Para la creación del modelo CNN, se empleó la implementación de “red neuronal convolucional unidimensional (1D CNN)” proporcionada por Keras. La red opera por capas, por lo que se definieron inicialmente dos capas de convolución unidimensional. La primera capa consta de 64 filtros convolucionales con una ventana de convolución de tamaño 3, utilizando la función de acolchado “same” para preservar las dimensiones en los resultados. La función de activación utilizada es “relu”, y el tamaño de entrada se estableció en (39, 100), donde 39 representa el número de palabras en el tweet y 100 es el tamaño del vector para cada palabra. La segunda capa se diseñó con valores similares con el fin de capturar patrones más complejos basados en las características extraídas por la primera capa.

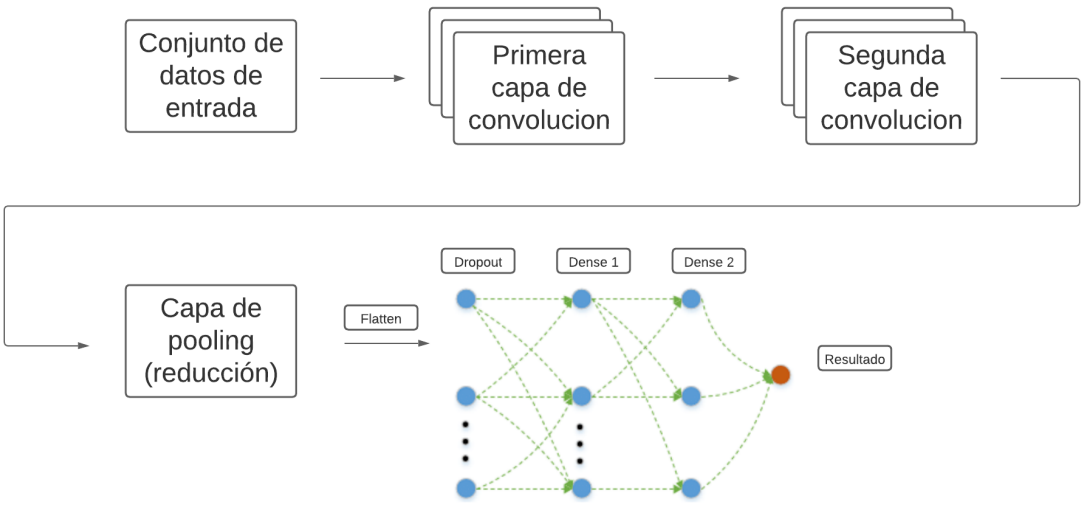


Figura 3.4.1: Arquitectura de la red nueronal

La Figura 3.4.1 ilustra la arquitectura de una red neuronal convolucional (CNN). La CNN comienza con un conjunto de datos de entrada que ya han sido preprocesados mediante técnicas de ingeniería de características, como word embedding. Luego, los datos pasan por las capas convolucionales, donde se extraen las características y relaciones relevantes para la tarea en cuestión. Posteriormente, la capa de reducción disminuye la dimensionalidad espacial de la información, lo que ayuda a evitar el sobreajuste.

La etapa de Flatten convierte la entrada en un formato adecuado para las capas densas. A continuación, se encuentran las capas de Dropout y dos capas densas. La capa de Dropout busca reducir la dependencia de las neuronas específicas, mientras que las capas densas implementan una transformación lineal y una función de activación para aprender relaciones más complejas. Esto permite obtener mejores resultados en la tarea que se desea realizar.

Por tanto, La capa "Dense" fue configurada con dos profundidades: la primera con 128 neuronas y una función de activación "relu", y la segunda con una neurona de salida y una función de activación "sigmoid". Respecto a la capa "Dropout", se asignaron dos tasas de abandono: 0.25 y 0.5 para las dos instancias respectivamente.

Estos valores de parámetros iniciales se seleccionaron debido a su idoneidad para la tarea, teniendo en cuenta la cantidad de datos disponibles, la representación escogida (word embedding) y el enfoque del proyecto. El uso de 64 filtros en la primera capa de convolución permite capturar una amplia gama de características locales, mientras que las capas "Dense" y "Dropout" ayudan a mejorar la capacidad de generalización del modelo y a evitar el sobreajuste en el proceso de entrenamiento.

En particular, para cada implementación, los modelos base se refieren a aquellas a las que no se les aplica ninguna técnica de balanceo de clases, en los demás casos, se aplica la técnica de sub muestreo y sobre muestreo para ambos conjuntos de datos. Para la representación de los datos se utiliza la técnica de word embedding.

Durante esta fase, se obtuvieron los siguientes resultados:

3.3.3.1. CNN con Word Embedding

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.60	0.64	0.62	0.73
Base - Data 2	0.62	0.53	0.57	0.72
Submuestreo - Data 1	0.56	0.73	0.63	0.71
Submuestreo - Data 2	0.59	0.67	0.63	0.73
Sobremuestreo - Data 1	0.57	0.69	0.62	0.71
Sobremuestreo - Data 2	0.55	0.67	0.60	0.70

Cuadro 3.5: Tabla de resultados para cada experimento de esta implementación con respecto a las métricas de evaluación

Se observa que el modelo entrenado con el conjunto de datos "Base - Data 2" tiene una "Precision" ligeramente superior en comparación con los demás conjuntos de datos. Aunque la diferencia entre los valores no es demasiada, esto sugiere que el conjunto de datos "Base - Data 2" permite obtener una correcta y mejor clasificación de las clases. Sin embargo, el modelo con el conjunto de datos "Submuestreo - Data 1" tiene un "Recall" más alto en comparación con los demás conjuntos de datos. Esto indica que el modelo con el conjunto de datos "Submuestreo - Data 1" logra identificar correctamente una mayor proporción de tweets ofensivos, a pesar de tener una precisión ligeramente inferior. Para finalizar, el rendimiento de los modelos de la CNN es moderado, esto sugiere que hay margen para mejorar la capacidad del modelo para encontrar un equilibrio entre "Precision" y "Recall", obteniendo así un mejor valor para el "F1-Score".

3.4. Búsqueda de hiperparámetros

En esta sección se exponen los resultados derivados del empleo de la técnica de búsqueda y optimización de hiperparámetros para cada uno de los modelos construidos y sus respectivas implementaciones o enfoques. Específicamente, se aplicó la técnica "Grid Search" con el propósito de obtener los valores más óptimos para dichos hiperparámetros, basándose en un rango predefinido y en la búsqueda exhaustiva facilitada por esta técnica.

Tras el análisis efectuado en la sección de "Desarrollo de los modelos", se pudo identificar la principal debilidad de los distintos modelos y sus respectivos experimentos en relación con el objetivo propuesto. Esta carencia se reflejaba en el bajo índice de la métrica "Recall", que indica la capacidad del modelo para identificar la mayor cantidad posible de tweets ofensivos en el conjunto de datos. Esta limitación impactaba considerablemente el rendimiento general de los modelos, ya que los tweets no ofensivos ejercían una influencia significativa, tal como se evidenciaba en los valores obtenidos de la métrica "F1-Score". Por este motivo, en esta sección se centró la optimización en torno a la métrica "Recall", considerando prioritario clasificar la mayor cantidad de tweets ofensivos y de esta manera reducir la cantidad de falsos negativos, aun a riesgo de generar una mayor cantidad de falsos positivos. La justificación radica en el contexto específico del proyecto, donde se valora más la identificación precisa de tweets ofensivos para evitar pasar por alto aquellos que, aunque no sean explícitamente ofensivos, podrían llegar a serlo. No obstante, se debe plantear un equilibrio entre la clasificación de las clases y así tener una buena métrica de "F1-Score". Además, a pesar de estar presente la métrica "Accuracy", que proporciona una evaluación general del desempeño del modelo, la misma no se considera como único indicador principal, debido a que puede reflejar un alto valor, incluso si el modelo no está clasificando adecuadamente una de las clases.

Finalmente, es importante mencionar que la determinación de los parámetros para cada modelo y la definición del rango de valores a evaluar se llevó a cabo mediante un proceso de investigación exhaustivo. Este proceso incluyó la exploración de información detallada en la documentación de la librería empleada para la implementación de los modelos, así como un proceso de prueba y error de diferentes valores en amplios rangos de evaluación. A medida que se realizaban los experimentos, se fue acotando progresivamente el intervalo de valores para identificar aquellos que producían los resultados más óptimos. Gracias a esta metodología, fue posible determinar con precisión los parámetros a optimizar y definir el rango de valores más adecuado para mejorar los resultados de la métrica de "Recall".

3.4.1. Naive Bayes

Para este modelo y cada una de sus implementaciones, se buscó optimizar el siguiente parámetro: **alpha**. El parámetro "alpha" controla la aplicación del método de suavizado de Laplace, el cual busca evitar que el modelo se vea afectado por valores atípicos.

Al aplicar la técnica de "Grid Search" en relación con los parámetros mencionados, se lograron obtener los siguientes resultados:

3.4.1.1. Naive Bayes Multinomial

Parámetro	Valores de prueba
Alpha	0.001, 0.1, 0.5, 1, 1.5, 5, 10, 15, 20, 25

Cuadro 3.6: Parámetros y rangos de evaluación

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.48	0.63	0.54	0.64
Base - Data 2	0.49	0.63	0.55	0.65
Submuestreo - Data 1	0.69	0.74	0.71	0.70
Submuestreo - Data 2	0.69	0.76	0.72	0.71
Sobremuestreo - Data 1	0.66	0.79	0.72	0.69
Sobremuestreo - Data 2	0.63	0.86	0.73	0.68

Cuadro 3.7: Tabla de resultados para cada experimento de esta implementación luego de la optimización

3.4.1.2. Naive Bayes Complementario

Parámetro	Valores de prueba
Alpha	0.001, 0.1, 0.5, 1, 1.5, 5, 10, 15, 20, 25

Cuadro 3.8: Parámetros y rangos de evaluación

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.44	0.80	0.57	0.58
Base - Data 2	0.42	0.85	0.56	0.55
Submuestreo - Data 1	0.66	0.80	0.72	0.69
Submuestreo - Data 2	0.62	0.89	0.73	0.68
Sobremuestreo - Data 1	0.70	0.74	0.72	0.71
Sobremuestreo - Data 2	0.68	0.78	0.73	0.71

Cuadro 3.9: Tabla de resultados para cada experimento de esta implementación luego de la optimización

Con el ajuste de hiperparámetros en los modelos de Naive Bayes Multinomial y Naive Bayes Complementario, se ha logrado una mejora significativa en el "Recall" para todos los experimentos.

Esta mejora permite obtener una mejor capacidad de identificación y clasificación de tweets ofensivos en comparación con sus versiones del apartado anterior "Desarrollo de los modelos".

En términos generales, la mayoría de los modelos exhiben una tendencia de incremento en el valor del "Recall", manteniendo simultáneamente un equilibrio adecuado entre las demás métricas de evaluación. Sin embargo, se observa un desbalance en las métricas de los modelos base tanto con el conjunto de datos "Data 1" como el "Data 2". Este desbalance sugiere que estos modelos no son idóneos para llevar a cabo la tarea de clasificación, ya que no logran capturar de manera efectiva tanto la precisión como la exhaustividad necesaria para una buena clasificación.

3.4.2. Máquinas de soporte vectorial (SVM)

Los hiperparámetros que se procuraron optimizar en este modelo abarcan los siguientes: **C**, **kernel**, **gamma**. El parámetro "C" constituye un hiperparámetro que regula la intensidad del regularizador en el modelo, evitando que este se ajuste demasiado a los datos de entrenamiento mediante la aplicación de una penalización adecuada. Respecto al parámetro "kernel", este define el tipo de función que opera en el modelo, pudiendo ser lineal, polinómico o gaussiano. Por último, el parámetro "gamma" se refiere al ancho de banda del kernel gaussiano.

Al aplicar la técnica de "Grid Search" en relación con los parámetros mencionados, se lograron obtener los siguientes resultados:

3.4.2.1. SVC con bolsa de palabras

Parámetro	Valores de prueba
C	0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5
Kernel	linear, poly, rbf
Gamma	scale, auto

Cuadro 3.10: Parámetros y rangos de evaluación

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.72	0.53	0.61	0.77
Base - Data 2	0.73	0.52	0.61	0.77
Submuestreo - Data 1	0.75	0.70	0.72	0.73
Submuestreo - Data 2	0.73	0.67	0.70	0.71
Sobremuestreo - Data 1	0.84	0.47	0.60	0.69
Sobremuestreo - Data 2	0.86	0.49	0.62	0.70

Cuadro 3.11: Tabla de resultados para cada experimento de esta implementación luego de la optimización

3.4.2.2. SVC con TF-IDF

Parámetro	Valores de prueba
C	0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5
Kernel	linear, poly, rbf
Gamma	scale, auto

Cuadro 3.12: Parámetros y rangos de evaluación

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.66	0.57	0.61	0.75
Base - Data 2	0.58	0.55	0.61	0.76
Submuestreo - Data 1	0.71	0.68	0.70	0.70
Submuestreo - Data 2	0.70	0.67	0.69	0.69
Sobremuestreo - Data 1	0.85	0.47	0.60	0.69
Sobremuestreo - Data 2	0.74	0.68	0.71	0.72

Cuadro 3.13: Tabla de resultados para cada experimento de esta implementación luego de la optimización

Después de ajustar los hiperparámetros, los modelos de SVM con bolsa de palabras y TF-IDF han mostrado un rendimiento variado en comparación con los modelos desarrollados en la sección anterior. Aunque se observaron mejoras notables en aspectos como la "Precision", los resultados de "Recall", que era la métrica a mejorar, mostraron cierta disminución en varios casos.

En particular, los modelos con bolsa de palabras mantuvieron un rendimiento superior en términos de "Recall" en comparación con los modelos de TF-IDF después del ajuste de hiperparámetros. Además, los modelos con submuestreo continúan mostrando un equilibrio entre "Precision" y "Recall", lo que confirma la efectividad del submuestreo para mantener métricas balanceadas. Sin embargo, los modelos con sobremuestreo presentaron un bajo "Recall", indicando cierta dificultad en capturar adecuadamente las instancias de tweets ofensivos.

3.4.3. Redes neuronales convolucionales (CNN)

Los hiperparámetros que se procuraron optimizar en este modelo comprenden: **dropout_rate**, **neurons**, **batch_size** y **epochs**. El parámetro "dropout_rate", este representa una técnica de regularización que ayuda a prevenir el sobreajuste al eliminar de manera aleatoria algunas neuronas de cada capa durante el entrenamiento. Esto garantiza que el modelo no se adapte excesivamente a los detalles específicos de los datos de entrenamiento.

En relación con el parámetro "neurons", este indica el número de neuronas presentes en cada capa del modelo. En cuanto al parámetro "batch_size", se refiere al número de muestras que se

emplearán para actualizar los parámetros del modelo en cada iteración del algoritmo de optimización. Por último, el parámetro "epochs" determina la cantidad de veces que se entrenará el modelo utilizando los datos de entrenamiento.

Al aplicar la técnica de "Grid Search" en relación con los parámetros mencionados, se lograron obtener los siguientes resultados:

Parámetro	Valores de prueba
Dropout rate	0.2, 0.25, 0.3
Neurons	128, 256, 512
Batch size	32, 64, 128
Epochs	10, 20, 30

Cuadro 3.14: Parámetros y rangos de evaluación

Modelo	Precision	Recall	F1-score	Accuracy
Base - Data 1	0.62	0.61	0.62	0.74
Base - Data 2	0.64	0.57	0.61	0.74
Submuestreo - Data 1	0.55	0.72	0.63	0.70
Submuestreo - Data 2	0.54	0.68	0.60	0.69
Sobremuestreo - Data 1	0.62	0.59	0.60	0.73
Sobremuestreo - Data 2	0.58	0.54	0.56	0.71

Cuadro 3.15: Tabla de resultados para cada experimento de esta implementación luego de la optimización

Para este último modelo, luego de ajustar los hiperparámetros, se evidencia una disminución en la métrica "Recall" para el modelo con los conjuntos de datos en comparación con los resultados anteriores. Esta disminución puede atribuirse a múltiples factores, entre los cuales destaca el volumen relativamente insuficiente de registros en el conjunto de datos o posibles deficiencias en la preparación y representación de los mismos. Además, la complejidad en la interacción de los hiperparámetros podría haber desencadenado una mala estimación de los valores a evaluar en busca de la mejora, pero también de mantener un equilibrio entre "Precision" y "Recall", o sea, el "F1-Score".

Es importante destacar que aunque se espera una mejora en el rendimiento de los modelos al ajustar los hiperparámetros, esto no siempre es el caso, y en algunos escenarios, como el presentado, la mejora no se manifiesta, manteniendo un desempeño similar o incluso puede haber una disminución en el rendimiento.

Por último, es relevante mencionar que el modelo con el conjunto de datos "Data 2" mostró una leve mejora en sus métricas de evaluación en comparación con la sección anterior, logrando un mejor "Recall". Sin embargo, su desempeño aún se mantiene inferior en comparación con el modelo que

utiliza el conjunto de datos "Data 1", incluso con respecto a la sección anterior.

3.5. Evaluación y análisis final

En esta sección se plasman los resultados finales sobre la evaluación y análisis definitivo de los tres modelos que arrojaron los mejores resultados después de la etapa de búsqueda de hiperparámetros. En este sentido, se seleccionó un modelo por cada algoritmo y se evaluaron con el conjunto de datos que es desconocido para los modelos y que fue descrito en la sección de "División de datos y equilibrio de clases" del proceso de preprocesamiento. La finalidad de esta evaluación fue observar el rendimiento de los modelos frente a datos completamente desconocidos. Es importante señalar que el conjunto de datos desconocido también atravesó cada una de las etapas de preprocesamiento del proyecto, incluida la etapa de ingeniería de características para la representación de los datos, de modo que fueran comprensibles para los modelos y poder así evaluar su desempeño.

En el caso del algoritmo de Naive Bayes, el modelo seleccionado para esta última fase fue el "Complementario con submuestreo data 2", el cual para nuestro enfoque principal, que fue la métrica "Recall", obtuvo un valor de 0.89 y un desempeño general en su métrica "F1-Score" de 0.73. Por otro lado, para el algoritmo de Máquinas de Vectores de Soporte (SVM), se seleccionó el modelo "SVC con bolsa de palabras, modelo submuestreo data 1", obteniendo un valor de 0.70 en su métrica "Recall" y un desempeño general en su métrica "F1-Score" de 0.72. Por último, para la red neuronal convolucional (CNN), se eligió el "Modelo base data 1" pero de la sección "Desarrollo de los modelos", es decir, sin el proceso de optimización de hiperparámetros. Presentando un valor de 0.64 en su métrica "Recall" y un desempeño general en su métrica "F1-Score" de 0.62.

3.5.1. Naive Bayes

3.5.1.1. Complementario, modelo submuestreo - data 2

	Predicción Negativa	Predicción Positiva
Actual Negativo	151	89
Actual Positivo	124	116

Cuadro 3.16: Matriz de confusión final con datos desconocidos

Modelo	Precision	Recall	F1-score	Accuracy
Submuestreo - Data 2	0.57	0.48	0.52	0.56

Cuadro 3.17: Tabla de resultados finales con datos desconocidos

3.5.2. Máquinas de soporte vectorial (SVM)

3.5.2.1. SVC con bolsa de palabras, modelo submuestreo - data 1

	Predicción Negativa	Predicción Positiva
Actual Negativo	508	112
Actual Positivo	185	55

Cuadro 3.18: Matriz de confusión final con datos desconocidos

Modelo	Precision	Recall	F1-score	Accuracy
Submuestreo - Data 1	0.33	0.23	0.27	0.65

Cuadro 3.19: Tabla de resultados finales con datos desconocidos

3.5.3. Redes neuronales convolucionales (CNN)

3.5.3.1. CNN, modelo base - data 1

	Predicción Negativa	Predicción Positiva
Actual Negativo	489	131
Actual Positivo	85	155

Cuadro 3.20: Matriz de confusión final con datos desconocidos

Modelo	Precision	Recall	F1-score	Accuracy
Submuestreo - Data 1	0.54	0.65	0.59	0.75

Cuadro 3.21: Tabla de resultados finales con datos desconocidos

En esta etapa final de evaluación de los mejores modelos con datos desconocidos, se logra determinar cuál de los modelos exhibe los resultados más destacados en su conjunto para este escenario. Específicamente, el modelo de la red neuronal convolucional (CNN) sin optimizar sus hiperparámetros, entrenado con el conjunto de datos denominado “Submuestreo - Data 1” y validando su rendimiento con el conjunto de datos desconocidos, logró alcanzar un valor de 0.65 en la métrica de “Recall”. Además, este modelo estableció un “F1-Score” de 0.59, denotando un desempeño general positivo, pero regular, aunque posea un “Accuracy” de 0.75.

En contra parte, los modelos de Naive Bayes y SVM no tuvieron un desempeño realmente destacable, puesto que en su métrica “Recall” obtuvieron un valor bastante bajo, ya que no lograron siquiera un valor mínimo de 0.50 o por encima de este y en cuanto a su desempeño general, reflejado en la métrica “F1-Score”, solo Naive Bayes logro estar por encima de 0.50, aunque por muy poco,

dando a entender de que posiblemente los valores obtenidos en las etapas anteriores fueron más óptimos porque se tuvieron más datos o en su defecto, los modelos pudieron haber comenzado a caer en un posible sobreajuste.

En resumen, lo anterior nos da a entender que la CNN, al ser capaz de capturar características y patrones más complejos del conjunto de datos y que teniendo en consideración el proceso empleado para la manipulación de los datos y su representación (a pesar de no ser posiblemente la más óptima para la etapa de hiperparámetros), logró obtener un rendimiento realmente positivo con estos datos completamente desconocidos, aunque no llegue a ser sobresaliente o muy destacable para el contexto y la tarea en cuestión.

3.6. Conclusiones

En el desarrollo de este proyecto de grado, se abordó la problemática de clasificar tweets como ofensivos o no ofensivos, lo cual está vinculado al campo del análisis de sentimientos. Se exploró como resolver este problema de clasificación usando el aprendizaje automático. Durante el proceso, se implementaron diversas técnicas y métodos para mejorar la precisión y confiabilidad de los modelos de clasificación.

La clasificación de tweets es una problemática crucial que requiere análisis debido a sus profundas implicaciones tanto en el mundo real como en el digital. En un contexto global cada vez más conectado, las redes sociales han amplificado la importancia de entender las opiniones públicas. Considerando lo anterior, el propósito central de este proyecto fue llevar a cabo una investigación exploratoria en área del aprendizaje automático. Durante este estudio, se investigaron y aplicaron diversos métodos y técnicas para el desarrollo de los modelos.

Se abordó la preparación de los modelos mediante un proceso de preprocesamiento del conjunto de los tweets, diseñado para que los modelos pudieran reconocer los datos de entrada de manera efectiva. En una primera fase, se realizó una limpieza con el uso de expresiones regulares, la cual involucró la eliminación de números, caracteres especiales, emojis y palabras que no aportaban valor al conjunto de datos. Posteriormente, se llevó a cabo la tokenización de los datos, que básicamente constaba en la separación o división de los tweets en unidades más pequeñas, que pueden ser palabras individuales, frases, símbolos o cualquier otra entidad significativa en los tweets. Luego de realizar este proceso, se llevó a cabo la revisión de corrección ortográfica y la separación de palabras unidas. En este punto, surgieron algunas dificultades debido a conflictos entre las dos librerías utilizadas. La corrección ortográfica resultaba esencial para asegurar la calidad lingüística, mientras que, por otro lado, la separación adecuada de las palabras era crucial para la comprensión semántica de los tweets. En la búsqueda de superar esta problemática, se implementó una función que aprovechaba ambas librerías. Esta función se diseñó con el fin de identificar en que casos se generaban dichos conflictos y de esta manera poder darles un manejo oportuno, evitando la perdida de información.

Para concluir la etapa de preprocesamiento, se implementaron dos pasos adicionales para mejorar la calidad y relevancia semántica de los datos. Primeramente, la eliminación de palabras vacías (stopwords) en un dataset, en esta fase resultaron dos conjuntos de datos diferentes: Data 1 y Data 2. El conjunto de datos Data 1 corresponde a los tweets procesados y a los cuales se les eliminaron estas palabras vacias, mientras que, por otro lado, el conjunto de datos Data 2, corresponde a los tweets procesados, pero que mantuvieron estas palabras vacias. Lo anterior se realizó debido a que se consideró interesante el hecho de analizar y evaluar el desempeño de los modelos con un conjunto de datos que sí las tuviese en cuenta y otro conjunto de datos que no. Por último, se aplicó la lematización, que desempeñó el papel de simplificar las palabras a su forma base, unificando variaciones gramaticales.

Una vez finalizada la etapa de preprocesamiento, se continúa hacia la fase de ingeniería de características, en esta fase aplicamos técnicas de ingeniería de características con el fin de que los modelos utilizados tuvieran una representación adecuada del conjunto de datos, estas técnicas fueron: Bolsa de palabras, tf-idf y word embedding. Es importante destacar que debido al orden de los pasos, puede verse alterado el conjunto de datos y su representación por cualquiera de las técnicas de la ingeniería de características, esto ocurre debido a que hay muchas variaciones para hacer preprocesamiento y es un aspecto valioso e importante a tener en cuenta cuando se desarrolla un proyecto en este campo y sobre todo, teniendo presente el objetivo que se quiere lograr.

En este proyecto, se emplearon tres modelos de aprendizaje automático para la clasificación de tweets: Máquinas de soporte vectorial, redes neuronales convolucionales y las variantes de Naive Bayes (multinomial y complementario). Al entrenar estos modelos con los conjuntos de datos de entrenamiento (Data 1 y Data 2). Luego de evaluar sus desempeños con los datos de prueba, se obtuvieron resultados satisfactorios, con valores de recall promedio entre 0.7 y 0.8. Sin embargo, los resultados variaron significativamente en los datos de validación. Tanto los modelos de Naive Bayes como las máquinas de soporte vectorial mostraron un rendimiento deficiente en todas las métricas, indicando un bajo desempeño de estos modelos y que, posiblemente para los resultados obtenidos en etapas anteriores, se pudo haber llegado a tener un sobreajuste en los mismos. Por otro lado, la red neuronal convolucional pese a no tener un desempeño excepcional, fue la única que mantuvo un comportamiento consistente con los resultados obtenidos en las pruebas finales con el conjunto de datos desconocido.

Los resultados obtenidos, aunque fueron positivos, no estuvieron a la par con nuestras expectativas respecto al comportamiento de los modelos. Esta diferencia nos llevó a revisar posibles factores que podrían estar afectando su desempeño. En las primeras etapas del proyecto, identificamos una problemática crucial: el desbalance significativo entre las clases de tweets ofensivos y no ofensivos. Esta proporción desigual tenía un fuerte impacto en el rendimiento de los modelos y que a pesar de emplear técnicas y métodos para contrarrestarlo, no se consiguieron obtener resultados favorables o que entrasen dentro de nuestras expectativas. Por otra parte, otro aspecto importante que consideramos fue la calidad de los datos y la configuración de los pasos en la etapa de preproce-

samiento. Observamos que ciertos pasos, como la lematización o corrección ortográfica, reducían palabras ofensivas a palabras no ofensivas, debido al conjunto de palabras que contenían dentro de su librería, lo que afectaba el conjunto de datos resultante aplicado posteriormente a los modelos.

A pesar de estos desafíos durante el proyecto, es importante resaltar una vez más el estable rendimiento de la red neuronal convolucional que implementamos. Este notable rendimiento se debe en gran parte a la ingeniería de características que aplicamos. Específicamente, queremos destacar el papel fundamental del word embedding y su funcionamiento en el campo del procesamiento de lenguaje natural. Esta técnica, debido a la manera en que funciona, que es transformando las palabras en vectores de alta dimensionalidad, logro capturar las complejidades lingüísticas de los datos y permitió que mantuvieran sus relaciones semánticas. Esta coherencia semántica puede ser la razón para el desempeño óptimo y destacable del modelo de red neuronal convolucional, proporcionando una base sobre la cual se construyeron relaciones significativas. La técnica de word embedding permitió presuntamente obtener un análisis más profundo y preciso en nuestro modelo. La capacidad innata de los word embeddings para preservar las relaciones lingüísticas entre palabras con contextos similares se presentó como un posible elemento clave en la comprensión semántica de los tweets y su posterior clasificación.

3.7. Trabajos futuros

Recomendamos considerar los siguientes puntos a tocar para futuras investigaciones en el campo de la clasificación de tweets, los cuales podrían resultar de gran utilidad.

- En primer lugar, se sugiere un enfoque exhaustivo en el desbalance de las clases a analizar. Se recomienda explorar técnicas adicionales que vayan más allá del submuestreo o sobremuestreo, y experimentar con la creación de datos sintéticos para reducir la brecha entre las clases.
- Otro punto importante y que se ha enfatizado anteriormente, es que puede resultar crucial revisar y ajustar la configuración de los pasos o técnicas durante la etapa de preprocesamiento. Se pueden explorar nuevos pasos, técnicas o aplicar diferentes configuraciones para evaluar cómo impactan en los resultados de los modelos.
- Por último, se sugiere la implementación de modelos más avanzados y complejos que permitan extraer de mejor manera las características y coherencia de los tweets, y que hayan demostrado buen rendimiento en tareas de procesamiento de lenguaje natural. Estos enfoques podrían contribuir significativamente al avance en la precisión y efectividad de los modelos para la clasificación de tweets en futuros proyectos de investigación.

Bibliografía

[1] A. Gaydhani, *et al.*, "*Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach.*" arXiv preprint arXiv:1809.08651, 2018.

[2] F. Garcia, "*Cómo Twitter usa Big Data e inteligencia artificial (IA)*", Artículo de Linkedin, 2021.

[3] M. Wich, *et al.*, "*Are Your Friends Also Haters? Identification of Hater Networks on Social Media*", Association for Computing Machinery, pp. 481–485, 2021.

[4] M. Zampieri, *et al.*, "*Predicting the Type and Target of Offensive Posts in Social Media*", Proceedings of NAACL, 2019.

[5] M. pertegal, *et al.*, "*Revisión sistemática del panorama de la investigación sobre redes sociales:Taxonomía sobre experiencias de uso*", Revista científica de educomunicación, vol. 27, 2019.

[6] B. Palop, "*La construcción de los límites a la libertad de expresión en las redes sociales.*", Revista de estudios políticos, pp. 55-112, 2016.

[7] G. Gunatilleke, "*Justifying Limitations on the Freedom of Expression*", Hum Rights Rev 22, pp. 91–108, 2021.

[8] N. Alkiviadou, "*Hate speech on social media networks: towards a regulatory framework?*", Information & Communications Technology Law, 2018.

[9] A. Prema, *et al.*, "*Sentimental Analysis of Twitter Data using Machine Learning Algorithms*", International Confernece on Forensics, Analytics, Big Data and Security, 2021.

[10] J. Bobadilla, "*Machine Learning y Deep Learning usando python, scikit y Keras*", RA-MA Editorial, 2020.

[11] S. Chand, *et al.*, "*Evaluation of Machine Learning Techniques in Sentimental Analysis*", 5th International Conference on Information Systems and Computer Networks, 2021.

[12] Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[13] Kim, Y, "*Convolutional neural networks for sentence classification*", arXiv preprint arXiv:1408.5882, 2014.

[14] Liddy E, "*Natural Language Processing*", In Encyclopedia of Library and Information Science, 2001.

[15] B. López, R. Rutilio, "*Análisis de sentimientos en textos de opinión: una evaluación práctica*", Plaza y Valdés, 2019.

[16] Python Software Foundation. "*re — Regular expression operations*", Python 3.10.6 documentation, 2021.

[17] TensorFlow. "*Tokenization and Text Data Preparation with TensorFlow*", TensorFlow documentation, 2021.

[18] S. Bird, E. Klein, and E. Loper, "*Natural Language Processing with Python*", O'Reilly Media, Inc., 2009.

[19] S. Putra, *et al.*, "*A Hybrid Model for Social Media Sentiment Analysis for Indonesian Text*", 20th International Conference on Information Integration and Web-based Applications and Services, 2018.

[20] L.Cheng, S. Tsai, "*Deep Learning for Automated Sentiment Analysis of Social Media*", International Conference on Advances in Social Networks Analysis and Mining, 2019.

[21] D. Jurafsky and J. H. Martin, "*Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*", 3rd ed., Prentice Hall, 2020.

[22] C. D. Manning, *et al.*, "*Introduction to Information Retrieval*", Cambridge University Press, 2008.

[23] A. Bracco, "*Normalización de Texto en Español de Argentina*", Universidad Nacional de Córdoba.

[24] J. Martínez, "*Regularización Lasso L1, Ridge L2 y ElasticNet*", IArtificial.net, 2020.

[25] N. Japkowicz, S. Shaju,"*The class imbalance problem: A systematic study*". Intelligent Data Analysis, 2002.

[26] G. Angiani,*et al.*,"*A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter*". conferencia, 2016.

[27] P. Badjatiya,*et al.*,"*Deep Learning for Hate Speech Detection in Tweets*". Proceedings of the 26th International Conference on World Wide Web Companion, 2017.

[28] V. Priyanka,"*witter Sentiment Analysis using Deep Learning*". Sridevi Women's Engineering College, 2021.

[29] G. Webb,"*Naïve Bayes*". Encyclopedia of Machine Learning and Data Mining, 2017.

[30] Pedregosa, *et al.*,"*Scikit-learn: Machine Learning in Python*", JMLR 12, pp. 2825-2830, 2011.

[31] F. Almeida, G. Xexeo,"*Word Embeddings: A Survey*", Federal University of Rio de Janeiro, 2019.

[32] A. McCallum, K. Nigam, "*A Comparison of Event Models for Naive Bayes Text Classification*", AAAI/ICML-98 Workshop on Learning for Text Categorization, 1998.

[33] The MathWorks, "*Hiperplanos óptimos como límites de decisión*", Version 9.13.0, 2022.

[34] J. Schmidhuber, "*Deep Learning in Neural Networks: An Overview*", Arxiv. Cornell University, 2014.

[35] Y. Lecun, *et al.*, "*Deep learning*", Nature Vol 521, Pag(436-44), 2015.

[36] E. Alpaydin, "*Machine learning: a probabilistic perspective (4th ed.)*", MIT Press, 2020.

[37] T. Fawcett, "*An introduction to ROC analysis*", Pattern Recognition Letters, 27(8), 861-874, 2006.

[38] J. Bergstra, Y. Bengio and O. Chapelle, "*A Survey on Hyperparameter Optimization for Machine Learning*", Journal of Machine Learning Research, 13(1), 281-320, 2012.

[39] Obtenido de: `https://imbalanced-learn.org/stable/under_sampling.html`

[40] Obtenido de: `https://imbalanced-learn.org/stable/over_sampling.html`

[41] Goodfellow, Ian J. and Bengio, Yoshua and Courville Aaron, "*Deep Learning*", Book published by MIT Press, 2016.

[42] C. Fleizach, S. Fukushima, "*A naive Bayes classifier on 1998 KDD Cup*", University of California San Diego.