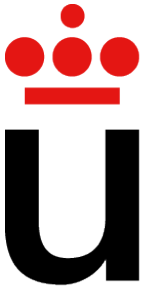

Universidad
Rey Juan Carlos

MÁSTER EN INGENIERÍA DE SISTEMAS DE DECISIÓN

Curso Académico 2021/2022

Trabajo Fin de Máster

ANÁLISIS DE LOS DESEQUILIBRIOS DE UNA BASE DE DATOS DE NOTICIAS FALSAS Y VERDADERAS EN MODELOS DE APRENDIZAJE AUTOMÁTICO SUPERVISADOS

Autor : Teilor Spencer Castro Montoya

Tutor : Dr. José Felipe Ortega Soto

Trabajo Fin de Máster

Análisis de Desequilibrios de una base de datos de noticias falsas y verdaderas en Modelos de Aprendizaje Automáticos Supervisados

Autor : Teilor Spencer Castro Montoya

Tutor : Dr. José Felipe Ortega Soto

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2022, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de Julio de 2022

Dedicado a

mis padres en los cielos y a mis hermanos en la tierra

II

Agradecimientos

Agradezco a Álex, por la paciencia infinita. El viaje está siendo divertido.

A mis padres, en cualquier parte del espacio que estén. El tiempo fue corto, pero intenso, y lo aprendido fue para siempre.

A mis hermanos y sobrinos, por vivir con mi ausencia. Siempre los llevo conmigo.

A mis amigos, por compartir y acompañarme en esta aventura. Y a todas esas personas que me hacen ser.

A Felipe, por ser mi guía en esta parte del camino.

Al universo, por permitirme estar.

Resumen

Las noticias falsas hacen cuestionarnos sobre la estabilidad de los modelos de comunicación. La crisis generada por la pérdida de credibilidad, la manipulación y el sesgo de los medios convencionales ha generado que como individuos nos cuestionemos la veracidad de la información que recibimos.

El ecosistema de medios de comunicación se ha ampliado con la incorporación de las redes sociales y la difusión de las noticias ha aumentado su velocidad, creciendo también su impacto. Los medios sociales se han dado cuenta de su necesidad de verificar la información y clasificarla e incorporar sistemas de verificación de noticias falsas para evitar así su propagación y prolongar su existencia.

Por medio de este estudio pretendo analizar el desequilibrio de una base de datos de noticias falsas y noticias verdaderas mediante los algoritmos de clasificación de aprendizaje supervisado como son los bosques aleatorios (*Random Forest*), las máquinas de vectores de soporte (*Support Vector Machine*) y potenciación del gradiente (*Gradient Boosting*).

Este análisis se ha realizado con la base de datos de Kaggle llamada "Fakenews" y con las librerías de Spacy y Scikit-learn de Python.

Este estudio se ha realizado con tres experimentos. El primer experimento consiste en el análisis de la base de datos mediante la técnica de clasificación de bosques aleatorios (*Random Forest*), la técnica de análisis de máquinas de vectores de soporte (*Support Vector Machine*), usando una función *kernel* o núcleo lineal, y la técnica de potenciación del gradiente (*Gradient Boosting*).

El segundo experimento es el análisis de la base de datos mediante la técnica de clasificación (*Random Forest*), la técnica de análisis de máquina de vector soporte (*Support Vector Machine*) en su función del núcleo lineal y la técnica de potenciación del gradiente (*Gradient Boosting*), habiendo degradado la base de datos hasta que no sea aceptable.

El tercer experimento ha sido analizar los resultados ofrecidos por un dataset desequilibrado, utilizando la técnica *SMOTE (Synthetic Minority Over-sampling Technique*) para reequilibrar los datos adecuadamente.

Summary

Fake news raises questions about the stability of communication models. The crisis generated by the loss of credibility, manipulation and the bias of the conventional media has generated that as individuals we question the veracity of the information we receive.

The media ecosystem has expanded with the incorporation of social networks and the dissemination of news has increased its speed, also growing its impact. Social media have realized their need to verify information and classify it and incorporate fake news verification systems to prevent its spread and prolong its existence.

By means of this study I intend to analyze the imbalance of a fake news and true news database using supervised learning classification algorithms such as Random Forests, Support Vector Machines and Gradient Boosting.

This analysis has been carried out with the Kaggle database called “Fakenews”, along with the Spacy and Scikit-learn Python libraries.

This study has been carried out through three experiments. The first experiment is the analysis of the database using the Random Forest classification technique, the Support Vector Machine algorithm, using a linear kernel function, and the Gradient Boosting technique,

The second experiment consists on the analysis of the database using the Random Forest classification technique, the Support Vector Machine algorithm with a linear kernel function, and the Gradient Boosting technique, after the database has been degraded to the extent that it is not acceptable.

The third experiment has been to analyze the results of an unbalanced dataset, using the SMOTE technique to rebalance the dataset adequately.

Índice general

Índice de figuras

Capítulo 1

Introducción

En la última década se ha visto la proliferación de las noticias falsas con la intención de desinformar, persuadir, manipular, desprestigiar instituciones y personas para obtener beneficios políticos y económicos. Este fenómeno ha aumentado masivamente desde la campaña presidencial en 2016 en Estados Unidos de América. Según Wikipedia sobre Fake news: "La campaña de Trump había utilizado las redes sociales como instrumento de divulgación, a diferencia de la de Clinton, que se había basado en anuncios televisivos y en otros medios tradicionales. Los mensajes políticos distribuidos mediante Facebook, Instagram o Twitter llegaron a unos 126 millones de personas en Estados Unidos. Un procedimiento de minería de datos aplicado a los perfiles de los usuarios, permitió la difusión de mensajes diferentes, diseñados según los intereses y preocupaciones de los destinatarios, dirigidos de modo específico a las personas preocupadas por la inmigración —especialmente de personas provenientes del mundo islámico—, el control de las fronteras, la legislación sobre las armas, etc." [1] Al mismo tiempo, la palabra *post-verdad* se ha convertido en el término del año según el diccionario de Oxford.

El mensaje es el objeto de la comunicación por medio del cual enviamos o recibimos información. Es importante que este mensaje pueda ser descifrado para que la comunicación sea efectiva. Entre las funciones que cumple la comunicación están la de informar, formar, persuadir, entretener. Estas funciones se han visto alteradas a consecuencia del auge de las noticias falsas.

Nuestra participación en el proceso de comunicación de masas, como el reconocimiento de nosotros mismos como audiencia, nos permite ser críticos con el sistema y con el mensaje que

[1] https://es.wikipedia.org/wiki/Fake_news

recibimos. Además, la democratización de Internet y la accesibilidad de teléfonos inteligentes ha permitido que durante la interacción nos cuestionemos toda la información que recibimos y que dudemos de su veracidad antes de compartirla. Sin embargo, la realidad como los mensajes son interpretados según lo que se conoce y se entiende, esta idea está fundamentada por la *teoría de equilibrio de la disonancia/consonancia cognoscitiva* la cual sostiene que la sicología del individuo predispone a la recepción del mensaje [8].

Los poderes políticos y económicos han implantado una nueva manera de narrar. Los discursos han dejado de contener evidencias y pruebas objetivas para contener relatos que llaman la atención del público y cambian el foco de cualquier idea que les perjudique. Como ejemplo de esto Michiko Kakutani introduce estos hechos recientes [5]:

> "En todo el mundo se han producido oleadas de populismo y fundamentalismo que están provocando reacciones de miedo y de terror, anteponiendo estos al debate razonado, erosionando las instituciones democráticas y sustituyendo la experiencia y el conocimiento por la sabiduría de la turba. Las afirmaciones falsas sobre la relación financiera del Reino Unido con la Unión Europea –bien resaltadas en un autobús de la campaña «Vote Leave» («Vota Salir»)– contribuyeron a desviar la intención de voto y orientarla hacia el Brexit, y Rusia se lanzó a la siembra de *dezinformatsiya* en las campañas electorales de Francia, Alemania, Holanda y otros países, como parte de un proyecto propagandístico organizado y encaminado a desestabilizar los sistemas democráticos."

Las noticias falsas, para que sean efectivas, deben tener dos elementos fundamentales que comparten con las obras de ficción y son: la verosimilitud y la suspensión de la incredulidad, que cada vez son más utilizadas en la manipulación y la comunicación con las masas.

Existe variedad de teorías fundamentales y modelos para el estudio de la comunicación en masas. Se ha elegido para este estudio, dentro de las teorías de equilibrio, el modelo de coorientación de la cometa para entender mejor este fenómeno. Como explican Denis McQuail y Sven Windahl en su libro "Modelos para el estudio de la comunicación colectiva" [8], el modelo de Newcomb basado en las investigaciones de Heider sobre los procesos cognoscitivos internos de los participantes en una conversación (A y B), planteó que la base afectiva (aprecio o rechazo) de una relación conversacional se da sobre el objeto de referencia (X), de modo que cuando

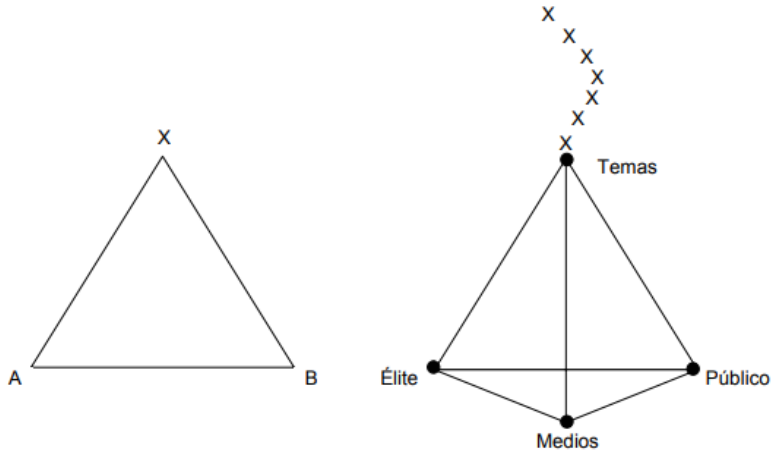

Figura 1.1: Modelo Newcomb (ABX) y modelo de co-orientación tipo cometa.

se establece ese equilibrio entre A, B y X, se establece una relación resistente al cambio. Sin embargo, Newcomb concluyó que la comunicación tiende a establecer relaciones de simetría entre los participantes y es especialmente útil en situaciones de tensión, incertidumbre o desequilibrio: cuando existen discrepancias entre A y B respecto de X, la comunicación acerca de X tenderá a equilibrar la relación. La comunicación sólo se activa si existe una relación entre las partes y si al menos uno de ellos está interesado en el tema. Este principio psicológico se aplica mucho en la comunicación persuasiva (publicitaria y política).

Como podemos observar en la gráfica 1.1, el modelo de Newcomb (ABX) y el modelo de co-orientación tipo cometa, que muestra las relaciones entre las élites, los medios de difusión, el público y los temas, las líneas representan las relaciones, actitudes y percepciones y la unilateralidad o bilateralidad de las relaciones. Los modelos son muy parecidos entre sí, y vemos que el modelo de co-orientación, incorpora los medios como un ente más, que es más o menos independiente en la relación entre las élites y el público.

Los medios de comunicación, que son utilizados por las élites no sólo para la difusión del mensaje sino también para la producción de contenidos, terminan definiendo una línea editorial y compartiendo los valores y políticas de sus anunciantes y proveedores. Sin embargo, la función en el proceso comunicativo de los medios de comunicación es la de intermediación y equilibrio entre las élites (poderes económicos y políticos) y el ciudadano. Este equilibrio se ha visto afectado no solamente por la pérdida de credibilidad de los medios de comunicación

frente a su labor informativa y periodística, sino también por la incorporación de un nuevo ente en el ecosistema de los medios de comunicación como son los medios sociales digitales y el acceso que tiene el público a la fuente. Así mismo, este público tiene la posibilidad de ser el origen de noticias falsas, de modificar o de agregar contenido a noticias según sus motivaciones psicológicas personales.

En el modelo Newcomb la proposición más importante es que la discrepancia en el mensaje estimula la comunicación y el efecto restablece el equilibrio. Sin embargo, en el modelo de co-orientación es que el resultado de la situación dinámica depende de las relaciones entre el público y las élites, la actitud del público frente a los medios y de las élites frente a los medios de difusión. Las discrepancias que se generan entre las élites y el público tienden a llevar a las élites a manipular las percepciones, actuando sobre los acontecimientos o inten tando controlar los medios de difusión. No obstante, el público ahora también se cuestiona y tiene el poder de generar nuevamente ruido y acrecentar o disminuir el conflicto.

La percepción selectiva es uno de los supuestos en los que se basa las teorías de equilibrio, sostiene que es el público el que busca información con la cual está de acuerdo. El otro supuesto es el de reforzar esa idea y este es el papel del mensaje. En el momento de divergencia o de discrepancia del público con las élites, se tiende a utilizar más los medios de difusión digitales como son las redes sociales para buscar y difundir este mensaje. Marián Alonso González en su artículo “Fake News: desinformación en la era de la sociedad de la información” [2] indica lo siguiente:

> “Gracias al uso de las redes sociales como fuentes de información estamos contribuyendo a la deconstrucción informativa ya que se atenta contra los conceptos de verdad y objetividad”.

Además, por la proliferación de las noticias falsas y para mantener el equilibrio entre las élites y el público, se han incorporado sistemas de verificación de noticias falsas en redes sociales para evitar el daño y el efecto negativo de tiene la difusión de información engañosa y sesgada que han supuesto una amenaza para su existencia [7].

La verificación basada en el conocimiento utiliza un proceso llamado verificación de datos. En este proceso se da la verificación manual de los hechos fundamentada por la verificación basada en expertos o por múltiples fuentes. También se da la verificación automática de los

hechos que no se adapta al volumen de información recién creada, especialmente en las redes sociales y para resolver este problema de escalabilidad se entrenan máquinas con técnicas como el *Machine learning* (ML) o el procesamiento natural del lenguaje (NLP) [12].

En el proceso de verificación de los datos tanto manual, como automáticos es necesario una manipulación previa de los datos. Tanto como si la verificación se da por medio de la verificación manual de una población, como si es dada por un sistema de verificación automática. El proceso de verificación de datos automática se puede dividir en dos etapas: extracción de hechos (también conocida como construcción basada en conocimientos y verificación de hechos (también conocida como comparación de conocimientos).

Este proceso de verificación basado en el conocimiento evalúa la autenticidad de la noticia dada. Sin embargo, existe también un método similar que se basa en el estudio del estilo, en el estudio del contenido y en la intención de la noticia.

Todo estudio, experimento o proceso requiere una manipulación y por eso esta investigación se basa en el estudio de las noticias falsas en base al estilo, donde se aplican técnicas de manipulación de una base de datos con diferentes modelos de clasificación. Se procesa una base de datos desequilibrada de noticias falsas y noticias verdaderas, y se evalúa los métodos que se utilizan en la categorización del texto en los diferentes métodos de clasificación de algoritmos de aprendizaje supervisado como son la técnica de bosques aleatorios *Random Forest*, la máquina de vector soporte *Support Vector Machine* (SVM) y la técnica de potenciación del gradiente *Gradient Boosting*. Se observa los resultados de la manipulación de la base de datos con diferentes desequilibrios y cómo se comportan los modelos, y además se analiza las técnicas de balanceo *SMOTE*.

1.1. Estructura de la memoria

Esta memoria de Trabajo Fin de Master está estructurada de la siguiente manera:

- En este primer capítulo se presenta una introducción al proyecto, sus principales objetivos, junto con la planificación temporal para su ejecución.

- En el capítulo 2 se describen las tecnologías utilizadas para la realización del proyecto.

- En el capítulo 3 se resume el estado del arte sobre la detección de noticias falsas en análisis de datos textuales.

- En el capítulo 4 se presenta el diseño e implementación de la comparativa propuesta en este proyecto.

- El capítulo 5 describe los experimentos realizados, junto al método de validación propuesto y se examinan los resultados obtenidos.

- Por último, el capítulo 6 se analizan las principales conclusiones a las que se llega tras realizar el proyecto.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo principal de este trabajo es analizar los desequilibrios de una base de datos construida con noticias falsas y noticias reales, mediante algoritmos de aprendizaje supervisados como Bosques Aleatorios (*Random Forests*), Máquinas de Vector Soporte (*Support Vector Machines*) y Potenciación del Gradiente (*Gradient Boosting*), observando su comportamineto en combinación con diferentes técnicas de rebalanceo de datos.

1.2.2. Objetivos específicos

- Procesar la base de datos para extraer las características de cada clase, su distribución y palabras más comunes para luego procesarla.

- Construir una modelo supervisado de *Machine learning* de clasificación binaria o categorica para procesar, extraer y transformar las caracteristicas de la base de datos que se entrenarán en el modelo para conocer las técnicas utilizadas en el procesamiento natural del lenguaje.

- Entrenar el modelo de clasificación con tres algoritmos supervisados como son los Bosques Aleatorios (*Random Forests*), Máquinas de Vector Soporte con núcleo lineal (*Support Vector Machines with lineal kernel*) y Potenciación del Gradiente (*Gradient Boosting*).

- Desequilibrar la base de datos penalizando la clase minoritaria y observar el comportamiento en los tres algoritmos supervisados.

- Analizar las técnicas de balanceo de una base de datos desequilibrada con los tres algoritmos y la técnica de balanceo *SMOTE (Synthetic Minority Over-sampling Technique*).

- Evaluar los resultados de los procesos anteriores.

1.3. Planificación temporal

Para la planificación del proyecto se ha utilizado la herramienta de planificación Trello, en la cual existe un tablero con las fases en las que he dividido el proceso de investigación y desarrollo del del proyecto.

El plan de trabajo inicial tiene una duración de 10 semanas.

- En las cuatro primeras semanas se ha buscado información general acerca del procesamiento natural del lenguaje y de las *Fakenews*.

- Una vez definido los experimentos en las dos semanas siguientes se ha buscado información técnica relacionada con ellos y con la base de datos.

- En las dos semanas siguientes se ha entrenado los modelos del primer experimento y segundo experimento.

- En las dos semanas posteriores se ha entrenado los modelos del tercer experimento.

- En las dos últimas semanas se unifica la información y se evalúan los resultados obtenidos y se hacen las tareas pendientes.

Capítulo 2

Tecnologías utilizadas

2.1. Lenguaje de programación: Python

Python es un lenguaje de programación de alto nivel interpretado, cuya filosofía hace hincapié en la legibilidad de su código, con semántica dinámica y orientado a objetos. Fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde —& Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba. El 20 de febrero de 1991, van Rossum publicó el código por primera vez en alt.sources, con el número de versión 0.9.0.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License.

Es un lenguaje interpretado, dinámico y multiplataforma con una curva de aprendizaje suave. Cuenta con una variedad de entornos de desarrollo y librerías de herramientas científicas, numéricas, de herramientas de análisis, de estructuras de datos y de algoritmos machine learning que permiten que todas las fases del proceso de ciencia de datos se hagan con él.

Según la clasificación del índice de la comunidad de programación TIOBE[1] (en inglés: TIOBE programming community index) que mide la popularidad de los lenguajes de programación. En mayo del 2021, Python ha superado a Java en la 2ª posición y es más probable que supere a C mejor clasificado porque C está perdiendo popularidad igual que Java como vemos en la figura 2.1.

[1] https://www.tiobe.com/tiobe-index/

TIOBE Index

May 2021	May 2020	Change	Programming language	Ratings	Change
1	1		C	13.38%	-3.68%
2	3	↑	Python	11.87%	+2.75%
3	2	↓	Java	11.74%	-4.54%
4	4		C++	7.81%	+1.69%
5	5		C#	4.41%	+0.12%

Figura 2.1: Índice TIOBE (mayo, 2021).

PYPL Index (Worldwide)

May 2021	Change	Programming language	Share	Trends
1		Python	29.9 %	-1.2 %
2		Java	17.72 %	-0.0 %
3		JavaScript	8.31 %	+0.4 %
4		C#	6.9 %	-0.1 %
5	↑	C/C++	6.62 %	+0.9 %

Figura 2.2: Índice PYPL (mayo, 2021).

Como podemos observar en la figura 2.2 el PYPL[2] (en inglés: PopularitY of Programming Language Index) que se crea mediante el análisis de la frecuencia con la que se buscan tutoriales de lenguaje en Google. Python es el idioma principal en los cinco países (EE. UU., India, Alemania, Reino Unido, Francia) en mayo de 2021. En la figura 2.3 vemos ambos índices.

2.2. Entorno de desarrollo: Google Colaboratory

Es un entorno de cuadernos Jupyter gratuito que se ejecuta en la nube y almacena sus cuadernos en Google Drive, permite escribir y ejecutar código, compartirlo, comentarlo y editarlo. También es llamado Colab y fue originalmente un proyecto interno de Google; Se hizo un intento de abrir todo el código como fuente abierta y trabajar más directamente en sentido ascendente, lo que llevó al desarrollo de la extensión «Open in Colab» de Google Chrome, pero esto finalmente terminó y el desarrollo de Colab continuó internamente. A partir de octubre de 2019, la interfaz de usuario solo permite la creación de cuadernos con *kernels* o núcleos de

[2] https://pypl.github.io/PYPL.html

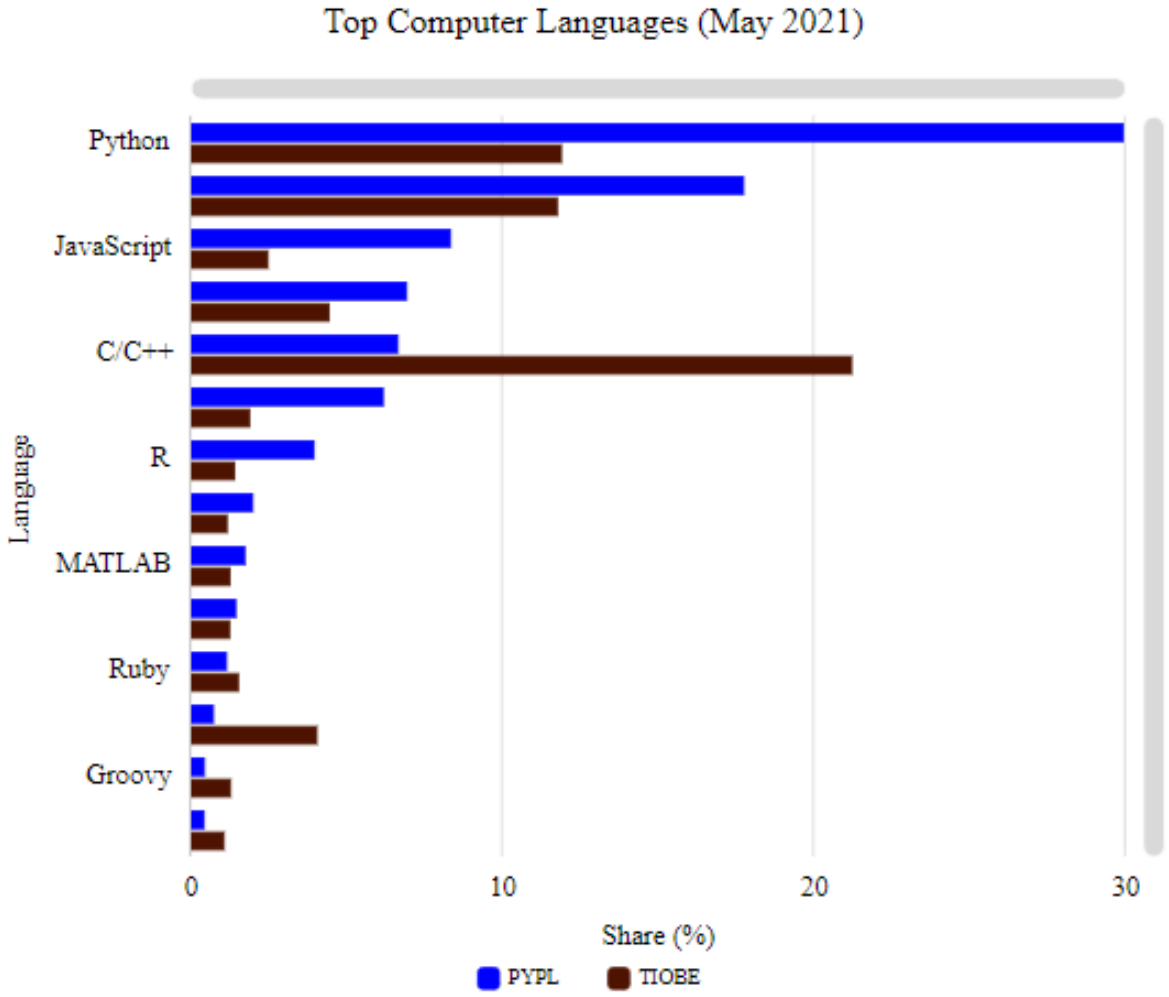

Figura 2.3: Comparativa índices TIOBI y PYPL para diferentes lenguajes de programación (mayo, 2021).

Python 2 y Python 3. Los cuadernos de Colab ejecutan código en los servidores en la nube de Google, lo que permite aprovechar la potencia del hardware de Google, incluidas las GPU y TPU, independientemente de la potencia de tu equipo. Con Colab, se puede aprovechar toda la potencia de las bibliotecas más populares de Python para analizar y visualizar datos. No requiere configuración y los cuadernos de Colab te permiten combinar código ejecutable y texto enriquecido en un mismo documento, además de imágenes, HTML, LaTeX y mucho más. [3]

2.3. Alojamineto del código: Git Hub

Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git (que es un software de control de versiones para la creación de código fuente de programas de ordenador diseñado por Linus Torvalds). El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena típicamente de forma pública. Github fue desarrollado por Chris Wanstrath, P. J. Hyett, Tom Preston-Werner y Scott Chacon. Aunque el 4 de junio de 2018 Microsoft compró GitHub, esta sigue siendo la plataforma más importante de colaboración para proyectos Open Source.

El código fuente de este proyecto se aloja en el siguiente link: https://github.com/taylorcastro/tfm_fake_news.

2.4. Redacción de la memoria: LaTeX y Overleaf

Overleaf [4] es un editor colaborativo de LaTeX basado en la nube que se utiliza para escribir, editar y publicar documentos científicos. Se asocia con una amplia gama de editoriales científicas para proporcionar plantillas oficiales de LaTeX para revistas y enlaces de envío directo.

Se lanzó en 2012 como WriteLaTeX por la compañía WriteLaTeX Limited, cofundada por John Hammersley y John Lees-Miller. El 20 de julio de 2017, Overleaf adquirió ShareLaTeX

[3] https://colab.research.google.com
[4] https://www.overleaf.com

para crear una comunidad combinada de más de dos millones de usuarios. Esto llevó a la creación de Overleaf v2, que combina características originales de ambos en una única plataforma basada en la nube alojada donde se puede visualizar paralelamente, en todo momento, el código fuente junto con el resultado de la compilación de este.

2.5. Librería: Spacy

SpaCy es una librería de software para procesamiento de lenguajes naturales desarrollado por Matt Honnibal e Ines Montani y programado en lenguaje Python Y Cython. Fue lanzado en febrero de 2015 estando su desarrollo activo y siendo utilizado en distintos entornos.

Es software libre con Licencia MIT su repositorio se encuentra disponible en Github. La librería SpaCy se enfoca en proporcionar software para uso en producción, a diferencia de NLTK, que se usa ampliamente para la enseñanza y la investigación. SpaCy también admite flujos de trabajo de aprendizaje profundo que permiten conectar modelos estadísticos entrenados por bibliotecas de aprendizaje automático populares como TensorFlow, PyTorch o MXNet a través de su propia biblioteca de aprendizaje automático Thinc. SpaCy presenta modelos de redes neuronales convolucionales para etiquetado de parte de la voz, análisis de dependencia, categorización de texto y reconocimiento de entidad con nombre (NER).

Sus principales características son la tokenización no destructiva en más de 65 idiomas. Soporte integrado para componentes de canalización entrenables, como reconocimiento de entidades nombradas, etiquetado de parte de la voz, análisis de dependencias, clasificación de texto, vinculación de entidades y más. Modelos estadísticos para 17 idiomas. Aprendizaje multitarea con transformadores previamente entrenados como BERT. Compatibilidad con modelos personalizados en PyTorch, TensorFlow y otros marcos. Velocidad y precisión de última generación. Sistema de formación listo para la producción y visualizadores integrados para sintaxis y entidades con nombre. Fácil gestión del flujo de trabajo, implementación y empaquetado de modelos. En su página principal [5] se puede encontrar toda la información de la librería, la usabilidad, modelos y cursos.

[5] https://spacy.io/

2.6. Librería: Scikit Learn

Scikit-learn (anteriormente scikits.learn) es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python. Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte (*Support Vector Machine*), bosques aleatorios (*Random Forests*), Gradient boosting , K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.

Un proyecto de Google Summer of Code por David Cournapeau más adelante Matthieu Brucher se unió al proyecto y lo utilizó como parte de su trabajo de tesis. Su nombre viene de la idea que se trata de un una extensión auxiliar desarrollada y distribuida independientemente de SciPy. El código de base original fue reescrito más adelante por otros desarrolladores. En 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort y Vincent Michel, todos de INRIA, el Instituto francés para la Investigación en Informática y Automatización, se involucró en el proyecto y la primera distribución pública (v0.1 beta) se realizó a finales de enero de 2010.

A fecha de junio de 2022, Scikit-learn sigue bajo desarrollo activo, actualizando versiones y aumentado su comunidad. Se puede encontrar toda la información de la librería en su página principal.[6]

[6] https://scikit-learn.org/stable/

Capítulo 3

Estado del arte

Una de las técnicas de verificación de información automática de hechos dentro de la inteligencia artificial es el procesamiento del lenguaje natural (PLN), que se encarga de la comprensión del lenguaje del ser humano, reuniendo modelos capaces de entender el lenguaje.

El objetivo del procesamiento del lenguaje naturas es proporcionar nuevas capacidades computacionales en torno al lenguaje humano: por ejemplo, extrayendo información de textos, traduciendo entre idiomas, respondiendo preguntas, manteniendo una conversación, recibiendo instrucciones, etc. Los conocimientos lingüísticos fundamentales pueden ser cruciales para llevar a cabo estas tareas, pero el éxito se mide en últimas instancias por si se hace el trabajo y qué tan bien se hace [3].

El procesamiento del lenguaje natural basa su aprendizaje automático analizando el estilo y el contenido de las noticias. Primero se define el estilo de las noticias falsas y verdaderas para que el modelo aprenda de forma automática. Luego se formula como un problema de clasificación binario o de clasificación de múltiples etiquetas. También se representa el vector de características y luego, en base a estas características identificadas, verifica las características del nuevo artículo y lo clasifica como verdadero o falso [12].

En este estudio se pretende analizar este método de verificación de noticias falsas basadas en el estilo, sus características, así como evaluar el modelo o clasificador. Las características en la clasificación de texto se pueden dividir como generales y latentes. Las características generales describen el estilo del contenido de al menos cuatro niveles de idioma: léxico, sintaxis, discurso y semántica. El léxico es evaluado por las estadísticas de frecuencia que se hace utilizando la

técnica de bolsa de palabras, en inglés *Bag-Of-Words* (BOW). La sintaxis es hecha por etiquetadores de parte de la oración, en inglés *Part-Of-Speech* (POS) y la Gramática libre de contexto probabilístico (GLCP), en inglés *Probabilistic Context-Free Grammar* (PCFG) utilizado en la generación de texto. El discurso está explicado por la teoría de la estructura retórica (RST) y la semántica por léxicos o frases de frecuencias en cada categoría psicolingüística, en inglés *Linguistic Inquiry and Word Count*(LIWC). [10]

El procesamiento del conjunto de datos se hace en base al enfoque de n-gramas y tokens. Los tokens son agrupaciones de caracteres en unidades contiguas y casi siempre son inútiles. Por eso es mejor hacer referencia a los tokens como agrupación de palabras y secuencias numéricas, separadas por caracteres de signos de puntuación o por espacios. Después, estos datos pasan por un clasificador de tokens donde después de ser ordenados, procesados y contados, tiene como resultado una representación vectorial de estos datos o documentos. Este espacio vectorial surge de las combinaciones posibles que puedan tener las palabras representadas, además a este modelo de palabras, declaraciones, documentos representados vectorialmente denominado "modelo de espacio vectorial" se le puede aplicar álgebra lineal para estudiar estos vectores, sus estadísticas, dimensiones e identificar patrones para observar su comportamiento con una técnica como la de análisis de componentes principales para luego continuar con su análisis con algunos algoritmo de aprendizaje supervisado.

Alternativamente a la canalización inicialmente propuesta con el análisis de componentes principales se pretende analizar esta canalización con capas de una red neuronal para el aprendizaje profundo, añadiendo a la arquitectura capas de procesamiento de extracción de entidades seguida de modelado.

Antes de hacer el procesamiento de los datos es necesario hacer el preprocesamiento.

3.1. Preprocesamiento

En el preprocesamiento de los datos se utilizan varias técnicas necesarias para poder trabajar con los datos de forma óptima, logrando que no se perturbe el proceso de análisis con ruido que pueda degenerar los resultados o convertirlos en irrelevantes. Durante este proceso se hace una limpieza de palabras vacías, eliminación de signos de puntuación y la transformación de

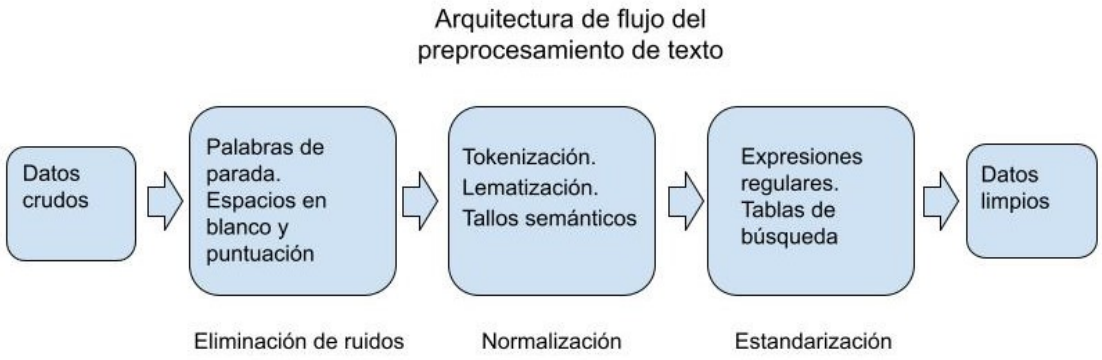

Figura 3.1: Flujo de preprocesamiento.

mayúsculas a minúsculas, para también hacer una segmentación y una posterior tokenización con datos refinados y relevantes.

En esta fase de preprocesamiento de datos es donde se hace limpieza de palabras irrelevantes con respecto a las palabras de contenido. Estas palabras son preposiciones, artículos, algunos pronombres y conjunciones cuya función es conectar el pensamiento y darle sentido gramatical a las oraciones. Además, se eliminan también los signos de puntuación y los espacios en blanco, la tabulación, los hiperenlaces y se normaliza todo el contenido para evitar la duplicidad de términos. La lematización y los tallos semánticos (*Stemming*) es otra de las técnicas que se aplican en esta parte de preprocesamiento.

En la gráfica 3.1 flujo de preprocesamiento podemos observar varias de las técnicas que utilizan en esta fase.

3.1.1. Tokenización

El método más sencillo de tokenizar una oración es usar como delimitador de palabras los espacios en blancos. Además, un buen tokenizador debe separar los signos de puntuación para que un segmentador de oraciones o el delimitador de oraciones pueda encontrar el final de la oración.

El tokenizador hace primero una representación vectorial matricial 1xN denominada *one-hot* vector utilizada para distinguir cada palabra de otra palabra del vocabulario. El vector está compuesto por «0s», excepto «1» en una celda utilizada únicamente para identificar la palabra.

La secuencia de estos vectores captura el texto original en una tabla de números, representando todo el documento de palabras en números. En esta representación cada fila es un vector para una sola palabra. Al realizar un seguimiento de etiquetas para cada columna marca cada columna con el token o la palabra que representa. Así que, un «1» en una columna indica una palabra de vocabulario que estaba presente en esa posición en el documento y la representación vectorial de palabras y tabular de documentos como una de sus características, no pierde información. a excepción de los espacios en blanco dividido por el tokenizador. Y estos vectores únicos quedan organizados léxicamente para que primero estén los números antes de las letras, y luego las mayúsculas antes de las minúsculas.

Como el procesamiento de todos los vectores de palabras en un documento necesita mucho procesamiento y es el origen del problema de la dimensionalidad, se busca reducir estos documentos en frases en vez de palabras, suponiendo que el significado de una frase está en las palabras.

El vector de bolsa de palabras se utiliza para comprimir el contenido de la estructura de datos, resumiendo la esencia del documento. Siendo el resultado de la suma de todos los vectores *one-hot* a la vez, en vez de reproducirlos uno a uno. También se denomina vector de frecuencia de palabras porque cuenta la frecuencia de las palabras, no su orden. Se puede utilizar también este vector para representar el documento o la oración en un único vector de longitud tan largo como el tamaño del vocabulario o el número de tokens únicos de los cuales se quiere hacer seguimiento.

Python pone los números decimales antes de los caracteres, las mayúsculas antes que las minúsculas. Este es el orden de los caracteres en los conjuntos de caracteres ASCII y Unicode. El orden de los caracteres no es importante, si es coherente en todos los documentos que tokeniza.

Cuando las palabras aparecen en más de una oración, se calcula la superposición para comparar o buscar documentos similares. Para verificar la similitudes entre las oraciones se cuenta el número de tokens superpuestos utilizando el producto de puntos.

El producto interno, producto de punto o producto escalar es una operación algebraica en la cual la dimensión interna de los dos vectores debe ser la misma, o sea, el mismo número de filas de la primera matriz y el número de columnas de la segunda y tiene como resultado un único número. Geométricamente, es el producto de dos magnitudes euclidianas de los dos

vectores y el coseno del ángulo entre ellos. Una medida de los vectores de bolsa de palabras es la superposición de palabras, su similitud y uso en ambas palabras de su vocabulario en ambos vectores. Siendo este el primer modelo de espacio vectorial (Space Vector Model).

El tokenizado separa las palabras en una oración con espacios, además con comas, puntos, comillas, punto y coma, guiones, etc. Dependiendo del caso, se trata como palabras o se ignoran. También se utiliza el rango de caracteres que es un tipo especial de clase de caracteres indicado entre corchetes y un guión y que coincidirá la expresión regular con cualquier carácter de subrayado $(_)$ o letra del alfabeto inglés (mayúscula o minúscula). $r'[a-z]'r'[0-9]'r'[0123456789]'r'[_a-zA-Z]'$.

Existen módulos como *regex* o el *matcher* de Spacy con variedad de reglas comunes para la tokenización de palabras en inglés, separar las puntuaciones que terminan la frase (?!.;,) de los tokens adyacentes y conservar los números decimales que contienen un punto como un solo token. Además, contienen reglas para las contracciones en inglés.

El tokenizador de n-gramas es una secuencia que contiene n elementos que se han extraído de una cadena. Estos elementos pueden ser caracteres, sílabas, palabras o símbolos. Sin embargo, se tratarán como palabras, por eso bi-gramas serán dos palabras y tri-gramas serán tres palabras que adicionalmente en la tokenización de vectores de bolsa de palabras, pueden tener significados importantes inherentes al orden de las palabras y que son lo suficientemente frecuentes para tener un puesto en el contador de tokens. Los n-gramas son una forma de mantener la información de contexto a través de su canalización. Sin embargo, si no son lo suficientemente útiles para correlacionar palabras, identificar temas o conectar documentos, se incurre en un problema de dimensionalidad del vector de entidades porque excede la longitud de todos los documentos, sus vectores tiene más dimensiones que documentos en su corpus y estaría sobre ajustado.

3.1.2. Las palabras de paradas o *Stopwords*

Las *stopwords* o palabras de parada, son palabras comunes que existen en cualquier idioma y con frecuencia alta pero que carecen de significado en una frase. Aunque se excluyen para reducir el esfuerzo computacional a veces proporcionan información importante en los n-gramas. Algunas son artículos, preposiciones y formas del verbo “to be”, tokens de una sola letra que aparecen cuando se dividen las contracciones y se deriva utilizando el tokenizador y el *stemmer*,

como son: a, an, the, this, and, or, of, on. Incluirlas en el vocabulario hace que la iteración de la lista de *tokens* sea más eficiente porque ahorra memoria y ayuda a reducir la dimensionalidad en grandes conjuntos de entrenamiento.

La normalización del vocabulario es un proceso importante para la reducción de *tokens* que tienen significados similares en uno normalizado para reducir el sobreajuste. Una técnica de normalización es el plegado de caja o plegado de casos que es cuando en el vocabulario se encuentra la misma palabra escrita en minúscula, con mayúscula al comienzo de la palabra o toda en mayúscula para enfatizar, se normaliza para que solo exista un token, ya que la palabra tiene el mismo significado. Si se quiere hacer un reconocimiento de entidades en la canalización, no sería bueno hacer esta normalización porque a menudo la mayúscula se usa para indicar un sustantivo propio, el nombre de una persona, lugar o cosa. Sin embargo, sin la normalización, el vocabulario será el doble y se necesitarán más recursos para ser más preciso y general, y no un conjunto de datos representativos para el entrenamiento del modelo.

3.1.3. Tallos semánticos o *Stemming*

Otra técnica de normalización es el *stemming*, que es la eliminación de los derivados, como son plurales, terminaciones posesivas y algunas formas verbales. El *stemming* identifica los tallos semánticos comunes entre las palabras y elimina el sufijo o su terminación. No es necesario que un tallo sea una palabra, puede ser un símbolo o una etiqueta, que representa varias ortografías de una palabra. Esta técnica reduce la dimensionalidad, el tiempo de procesado y limita la pérdida de información y significado, según la agresividad del *stemmer* que se elija en la búsqueda de palabra clave o recuperar información. Dos de los algoritmos de stemming más populares son los *stemmers* de Porter y Snowball. El *stemmer* de Porte trata las terminaciones “s” y “es”, “ed”, “ing”, “at”, “y”, terminaciones de sustantivos como “ational”, “tional”, “ence” y “able”, terminaciones de adjetivos como “icate”, “ful” y “alize”. adjetivos y sustantivos que terminan en “ive”, “ible”, “ent” y “ism”. Consonantes dobles finales cuyos tallos terminan en una sola “l”.

3.1.4. Lematización

La lematización es el procedimiento por el cual se mantiene baja la dimensionalidad de la representación vectorial reduciendo los *tokens*. La librería Spacy utiliza el diccionario predefinido WordNet para extraer los lemas y utiliza reglas para despojar las terminaciones de las palabras, reduciéndolas a tallos [11]. Este procedimiento de reducción y disminución de tipos y clases de palabras hacen la clasificación más rápida y eficiente.

Las palabras pueden dividirse en partes pequeñas como grafemas y letras, y como las sílabas, prefijos y sufijos, tienen un significado intrínseco y sentimiento. Los n-gramas se construyen mediante un algoritmo sencillo, que separa una cadena en palabras para conocer su tallo semántico. El resultado serán los pares de palabras (bigrams), tripletes de palabras (trigrams) y cuádruples que se incluirán en el vocabulario para su representación del documento en el espacio vectorial.

Durante la recuperación de un token se requiere de la separación de la puntuación de las palabras, las comillas y eliminar las contracciones de expresiones para mantener sólo las palabras que se incluirán en el vocabulario y volverá a ser evaluada con las combinaciones de palabras con significado similares con el proceso de tallos semánticos. Se unirá después a la representación vectorial denominada bolsa de palabras.

Por último la lematización es la forma más precisa de normalización porque tiene en cuenta el significado de la palabra. Usa una base de conocimiento de sinónimos y terminaciones de palabras para asegurar que sólo las palabras que tienen significado similares se conviertan en tokens y evitar así el *spoofing* o suplantación de identidad que invierte el significado de las palabras para obtener respuestas erróneas. Algunos lematizadores usan las etiquetas de palabras de *Part of Speech* (POS), además de su ortografía para mejorar su precisión, ya que su etiqueta indica su papel en la gramática de una frase u oración para que su contexto sea conocido e identificado y no tratadas como palabras de forma aislada. En la imagen 3.2 podemos ver las diferencias entre el método de lemantización y tallos semánticos.

Y como dicen Hapke, Lane, Howard [6]:

> "Tanto los stemmers como los lematizadores reducirán el tamaño de su vocabulario y aumentarán la ambigüedad del texto. Pero los lematizadores hacen un mejor

[1] https://elitedatascience.com/imbalanced-classes

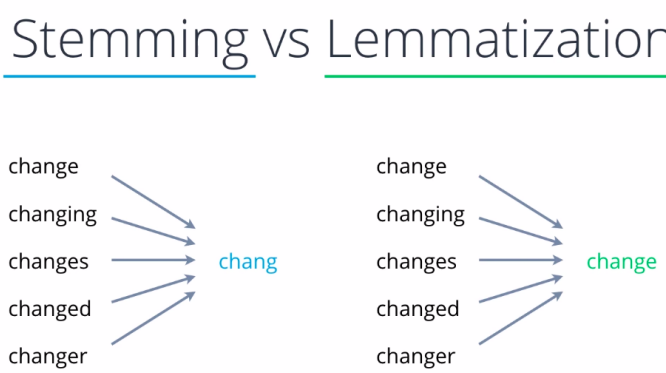

Figura 3.2: Diferencia entre Stemming y Lematización [1].

trabajo reteniendo la mayor cantidad posible del contenido de la información en función de cómo se usó la palabra dentro del texto y su significado previsto. Por lo tanto, algunos paquetes de PLN, como spaCy, no proporcionan funciones de derivación y solo ofrecen métodos de lematización".

3.2. Extracción de características

En la fase de análisis donde se aprende de los datos y se extraen las características se hace una reducción de términos y palabras y frases para que la carga computacional no sea tan elevada y sus características sean precisas y con alta precisión en los clasificadores. Para la reducción de la dimensionalidad de estos vectores se convierten en entidades, se evalúan estos tres métodos de selección de características: frecuencia de término (TF), frecuencia de términos- frecuencia inversa de documentos (TF-IDF) y Word2Vec.

El método de frecuencia de términos (TF) utiliza el recuento de palabras para hacer un vector, lo compara con los otros vectores de documentos para analizar la similitud. Se normaliza para que el resultado de sus elementos sea uno y luego su probabilidad de aparecer en otro documento es representada por grupos de palabras.

Los vectores de frecuencia de términos- frecuencia inversa de documentos (TF-IDF) ayudan a estimar la importancia de las palabras en un fragmento de texto en un corpus o conjunto de documentos. Utiliza métricas y estadísticas para extraer términos descriptivos y recuperar

información y cada vez que el término aparece, aumenta su importancia en el corpus, reduciendo el impacto que puedan tener términos que se repitan y carezcan de significado. [1]

3.2.1. Incrustaciones de palabras o *Embedding*

Word2Vec es un algoritmo de incrustaciones de palabras que aprende asociaciones de un gran corpus de texto. Representa cada palabra como un vector basado en el contexto circundante de cada palabra. Utiliza red neuronal de la bolsa continua de palabras (CBOW) para predecir qué palabra es más probable por su contexto, y su contrario *skip-gram* que predice las palabras circundantes dada una palabra de destino. [9]

Una vez que se ha recolectado y contado palabras dentro de un documento en tallos o lemmas con las cuales se pueden hacer estadísticas sobre el uso de la palabra o búsquedas de palabras clave, se pretende conocer cuáles son las palabras más importantes para un documento en particular o en todo el corpus en su conjunto. A su vez esta puntuación de importancia puede usarse para encontrar documentos relevantes dentro de un corpus según la importancia de las palabras claves dentro de cada documento. Después, convertimos las palabras en números continuos en vez de solo números enteros que representan el recuento de palabras o vectores de bits binarios que representan la presencia o ausencia de palabras para más adelante encontrar una representación numérica de las palabras que de alguna manera capturen la importancia o el contenido de la información de las palabras que representan.

Las tres formas más utilizadas de representar las palabras y la importancia en un documento son: Las bolsas de palabras que son vectores de recuento de palabras y frecuencia, las bolsas de n-gramas que recuentan pares (bigramas) y trillizos de palabras (tri-gramas) y vectores TF-IDF que representan la importancia de palabras por su puntuación. TF-IDF divide cada uno de los recuentos de palabras por el número de documentos en los que aparece la palabra.

Estas técnicas que se basan en la frecuencia se pueden aplicar por separado o como parte de una canalización, en este caso, se aplicarán como parte de la canalización.

3.2.2. Bolsa de palabras o *Bag Of World*

El vector de bolsas de palabras ofrece una representación vectorial que cuenta el número de frecuencia o ocurrencia de cada palabra en el texto. Los diccionarios de palabras son útiles

para hacer un conteo de palabras únicas y sus ocurrencias. Como el diccionario está optimizado para su almacenamiento, actualización y recuperación, es necesario ordenarlas desde las más comunes hasta la menos.

La frecuencia de términos (TF) se calcula con el número de veces que aparece la palabra en el documento o la frecuencia, dividido por el número de términos en el documento. Aunque esto es una probabilidad, se llama frecuencia.

El tokenizador de Treebank asume que el documento ya se ha segmentado en oraciones separadas e ignorará la puntuación final de la cadena y el tokenizador de Spacy hace segmentación y tokenización en una sola pasada lo que lo hace mucho más rápido y preciso.

El siguiente paso después de hacer el recuento de palabras es vectorizar este diccionario, que será una lista o una matriz ordenada para luego poder hacer operaciones con ellas. Es necesario que existan varios documentos los cuales se puedan vectorizar y que tengan alguna relación para que compartan un espacio vectorial en el cuál tengan el mismo origen, la misma escala o unidades en sus dimensiones. Primero hay que normalizar los recuentos calculando la frecuencia en términos normalizados y luego hacer los documentos de dimensión y longitud estándar.

3.2.3. Vectorización

Los vectores son una lista ordenada de números o coordenadas en un espacio vectorial. describen una posición o ubicación en ese espacio. Describen e identifican una distancia o dirección particular y magnitud en ese espacio. En un plano bidimensional, las coordenadas son x e y son perpendiculares y los vectores en este espacio son rectilíneos, este espacio es también llamado espacio euclidiano. Los vectores pueden tener cualquier dimensión. Sin embargo, esto conlleva un problema de dimensionalidad en el procesamiento y en la potencia del cálculo porque a medida que aumentan el número de vectores, se alejan exponencialmente unos de otros, a distancia euclidiana.

El tamaño del espacio vectorial es el recuento de palabras distintas que aparecen en todo el corpus. Una vez que se tiene la representación vectorial de todos los documentos que comparten el mismo espacio vectorial se puede medir la distancia entre ellos.

Existen varias distancias para medir el espacio entre vectores, algunas de ellas son la distancia euclidiana o cartesiana o raíz del error cuadrático medio (2-norma o L2), distancia euclidiana

al cuadrado que es la suma de la distancia de cuadrados (SSD) $L2^2$, coseno o distancia angular o proyectada, distancia de Minkowski, distancia fraccionaria, distancia de manhattan o distancia del taxi, distancia de jaccard, distancia de Mahalonobis y distancia de Levenshtein. Siendo las tres más comunes la distancia euclidiana, la distancia del coseno o la distancia de Manhattan.

La distancia euclidiana es el resultado de restar los dos vectores y calcular la longitud de la distancia entre ellos. La distancia euclidiana entre los vectores TF-IDF tiende a ser una buena medida de la distancia (similitud inversa) de los documentos.

3.2.4. Análisis semántico

Una vez que encuentra la longitud del documento en los vectores de recuento de palabras o frecuencia de término, se pueden estimar la similitud de documentos y encontrar que el uso de la misma palabra aproximadamente el mismo número de veces en proporciones similares. Dando la confianza de que los documentos están hablando de cosas similares su rango conveniente para la mayoría de los problemas de aprendizaje automático es entre -1 y 1.

Se toma el producto de puntos de dos de sus vectores en cuestión, se multiplican los elementos de cada vector por pares, y luego de resumir esos productos se divide por la norma (magnitud o longitud) de cada vector. La norma vectorial es la raíz cuadrada de la suma de los cuadrados de sus elementos. Es la misma que su distancia euclidiana desde la cabeza hasta la cola del vector. Es el coseno del ángulo entre estos dos vectores. Este valor es el mismo que la porción del vector más largo que está cubierta por la proyección perpendicular del vector más corto sobre el más largo. Le da un valor de cuánto apuntan los vectores en la misma dirección. Siendo la similitud del coseno 1 la representación de vectores normalizados idénticos que apuntan a la misma dirección a los largo de todas las dimensiones.

La representación vectorial de dos vectores teniendo una similitud de 0 no comparten componentes, son ortogonales y perpendiculares en todas las dimensiones. En los vectores de TF quiere decir que no comparten palabras en común, aunque pueden estar hablando de lo mismo. Una similitud de coseno de -1 representa dos vectores que son opuesto y apuntan en direcciones opuestas. Sin embargo, en términos de frecuencia o en términos de frecuencia normalizados no pueden ser negativos.

$$\text{tf}(t,\ d) = \frac{\text{count}(t)}{\text{count}(d)}$$

$$\text{idf}(t,\ D) = \log \frac{\text{number of documents}}{\text{number of documents containing } t}$$

$$\text{tfidf}(t,\ d,\ D) = \text{tf}(t,\ d) * \text{idf}(t,\ D)$$

Figura 3.3: Normalización de vectores.

3.2.5. TF-IDF

La frecuencia inversa de un documento es la relación que existe entre el número total de documentos y el número de documentos en los que aparece el término. Este índice se puede almacenar en diccionarios como se hace en términos de frecuencia. Cuando se comparan las frecuencias de dos palabras comunes similares, la palabra más frecuente tendrá una frecuencia exponencial más alta que la menos frecuente. Por esto, la Ley de Zipf sugiere escalar todas las frecuencias de palabras y de documentos con la función inversa para que estas palabras que tienen recuento similares, no sean exponencialmente diferentes en frecuencia. Asegurando que los puntajes de frecuencia de términos- frecuencia inversa de documentos se distribuyan uniformemente, definiendo el inversa de frecuencia de documento para que sea el registro de la probabilidad original de que esa palabra ocurra en uno de sus documentos.

Para normalizar los valores TF-IDF dado un término t, un documento d, en un corpus D, se tiene las siguientes fórmulas de la figura 3.3.

Cuantas más veces aparece una palabra en el documento, el TF y TF-IDF subirá. Al mismo tiempo, si aumenta el número de documentos que contiene esa palabra, las IDF y TF-IDF para esa palabra disminuirán. TF-IDF relaciona una palabra o token específico con un documento dado en un corpus y le asigna un valor numérico a la importancia de esa palabra en el documento dado, dado su uso en todo el corpus.

Una vez que se ha estimado la importancia de las palabras en el texto y se ha utilizado vectores y matrices TF-IDF para ver la importancia de cada palabra en el significado general de un texto o de un corpus, se hace un análisis semántico latente (LSA) para ver el significado de las combinaciones de palabras y los vectores que las representa. Con este algoritmo se pueden identificar las palabras y n-gramas que mejor representan el tema de una texto, documento o

corpus y comparar las declaraciones o documentos y ver qué tan cercanos están en significado entre sí.

Los vectores TF-IDF lematizados representan el significado de textos o palabras correlacionadas o co-ocurrencias. Sin embargo, es necesario extraer información adicional, compacta y significativa. Estos vectores son denominados vectores palabra-tema y vectores documento-tema o vectores temáticos pueden ser tan compactos como expansivos. Además, son muy útiles para la agrupación de documentos o la búsqueda semántica utilizada en buscadores y una vez que se tiene un vector documento-tema para cada documento del corpus, no es necesario que se reprocese todo el corpus, cada vez que se agrega un documento nuevo, como en el algoritmo de la asignación latente de Dirichlet.

Con el análisis semántico latente se obtiene un vector palabra-tema por cada palabra del léxico o vocabulario y se resuelven problemas como la polisemia, los homónimos y el zeugma. Se usa la ponderación para producir estos vectores. Sin embargo, se utiliza la norma L^2, que es la distancia euclidiana que calcula la longitud de la hipotenusa de un triángulo rectángulo del teorema de pitágoras.

LDA y LSA

LSA reduce la dimensión de los vectores para captar su significado, tiene la misma matemática que la técnica de componentes principales (PCA) y existen dos algoritmos similares con aplicaciones en PLN que son el análisis discriminante lineal (LDA), que divide un documento en un solo tema, y la asignación latente de Dirichlet (LDiA), que es más parecido a la LSA porque puede dividir el documentos en tantos temas como desee. Para reducir las dimensiones el análisis semántico latente utiliza la descomposición de valores singulares, descomponiendo la matriz en tres matrices cuadradas, una de ellas es diagonal. Otra de las aplicaciones es la inversión matricial, la cual la descompone en tres matrices cuadradas más simples, transponiendolas y volviendo a multiplicarlas de nuevo. Esta técnica es utilizada en motores de recomendación. En la reducción de la dimensionalidad mediante la descomposición matricial se puede ignorar algunas filas y columnas, lo que ayuda al problema de dimensionalidad del espacio vectorial, manteniendo la esencia y semántica de los documentos.

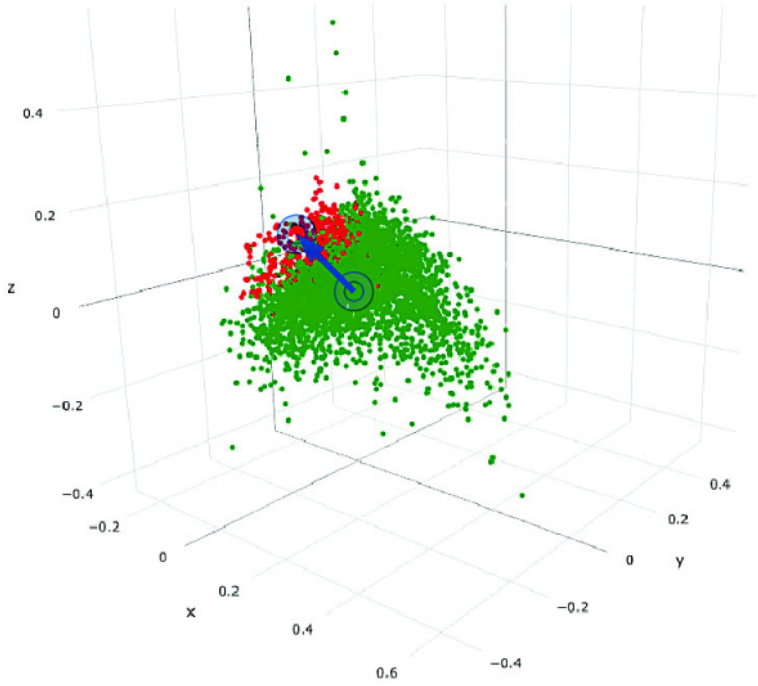

Figura 3.4: Diagrama de dispersión vectores TF-IDF [6].

> "LSA es un algoritmo para analizar su matriz TF-IDF para reunir palabras en temas. También funciona en vectores de bolsa de palabras, pero los vectores TF-IDF dan resultados ligeramente mejores" [6].

LDA y LSA funcionan de manera similar diferenciando que requieren etiquetas de clasificación o puntuaciones para encontrar las mejores combinaciones lineales de las dimensiones en el espacio de alta dimensión con vectores BOW o TF-IDF. En vez de maximizar la varianza entre todos los vectores del nuevo espacio, es la distancia entre los centroides dentro de cada clase la que se maximiza con LDA. Esto quiere decir que hay que entrenar el algoritmo LDA con los temas que nos gustaría modelar dándole vectores etiquetados, entonces el algoritmo calculará la transformación óptima de su espacio de alta dimensión al espacio de menor dimensión. El número de etiquetas o puntuaciones determinarán mínimamente el número de dimensiones que puede tener el vector.

Luego, se evalúa el modelo haciendo un análisis de validación cruzada para evaluar el modelo en su conjunto de entrenamiento.Se reserva un porcentaje de su conjunto de datos para las pruebas, para evitar que el modelo sea pobre y sobreajustado. Además, se puede hacer una combinación del modelo LSA con LDA para crear un modelo más preciso después de incluir

el análisis semántico con la ventaja de que el nuevo modelo crea vectores que representan la semántica de una declaración en más de una sola dimensión.

3.3. Aprendizaje profundo o *Deep learning*

Se llama aprendizaje profundo al uso de las redes neuronales multicapa para el aprendizaje automático. Gracias a la cultura de código abierto y el aumento de los recursos computacionales ha aumentado la comprensión más profunda del lenguaje.

La tecnología novedosa propuesta por Frank Rosenblatt en 1950 ofreció un algoritmo para encontrar patrones en los datos. La idea es una imitación aproximada del funcionamiento de una célula neuronal viva. El núcleo de una neurona recibe diferentes niveles de señales eléctricas por las dendritas y cuando esta tiene la suficiente carga envía una señal a través del axón. El proyecto original se realizó primero con reconocimiento de imágenes. El proyecto es un clasificador y primero se necesita determinar las características de la muestra y después un conjunto de pesos para asignar a cada una de las características. Los pesos son equivalentes a las pendientes en una regresión lineal multivariante o logística y el umbral es representado como una función de activación la cual incluye el sesgo y es el que determina la clasificación binaria. Los pesos de una neurona no entrenada son valores aleatorios cercanos a cero que se eligen de una distribución normal y el perceptrón aprende alterando los valores de peso hacia arriba o hacia abajo en base al resultado si es el esperado o no, más el sesgo. Y con muchas muestras, el error debe de tender a cero.

La convergencia se da cuando en un modelo su función de error se encuentra en un mínimo o en un valor consistente. Para que el perceptrón converge y prediga la variable objetivo con precisión, los datos tienen que ser separables linealmente o la relación tiene que estar descrita por una relación lineal. El mínimo local se da cuando el mejor mínimo o el costo más pequeño donde comenzaron los pesos o el mínimo global para múltiples mínimos. Si no se da tal convergencia el modelo no tendrá un poder predictivo útil. Como la mayoría de los datos no son separables con líneas ni planos, era necesario resolver este problema y gracias a la retropropagación a través de varias capas de neuronas surge la red neuronal.

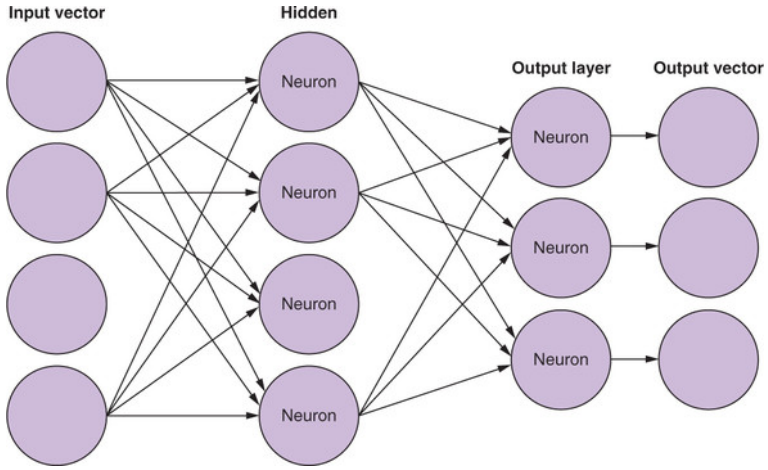

Figura 3.5: Ejemplo de red neuronal [6].

La función de costo o de pérdida cuantifica el desajuste entre respuestas correctas que la red debe emitir y la salida real, con qué frecuencia y qué tan equivocadas fueron las respuesta y el objetivo sigue siendo minimizar esta función de costo.

3.3.1. Redes Neuronales

Una red neuronal es una colección de perceptrones o neuronas con conexiones entre ellas. Las redes totalmente conectadas tienen una conexión con cada neurona de la siguiente capa, también capas ocultas entre la capa de entrada y salida. Las neuronas de la segunda capa tienen un peso asignado de cada una de las salidas de la primera capa, siendo el peso de la primera capa el que tiene un efecto que se pasa no solo a través de un único peso, sino a través de un peso en cada una de las neuronas de la siguiente capa com vemos en la 3.6.

La propagación es una entrada que fluye hacia adelante a través de la red y calcula la salida para esa entrada y esta función de activación tiene que ser cambiada. La función de activación en la retropropagación necesita ser no lineal y continuamente diferenciable como lo es la función sigmoide. Esta función permite que el error se propague suavemente a través de múltiples capas de neuronas acelerando su proceso de entrenamiento. Se puede calcular el error cuadrático medio, la derivada de la función de activación de cada neurona que la alimentaba y ajustarlo.

Una vez que se entrena una red, se pasan todas las entradas, se obtiene el error asociado, se retropropaga esos errores a cada uno de los pasos y luego se actualizan esos pesos con el cambio total del error y todos los datos de entrenamiento han pasado una vez por la red, y los errores se

hayan corregido, se llama a esto una época del ciclo de entrenamiento. Si se pasan mucho los datos de entrenamiento los pesos se pueden sobre ajustar y el modelo y este no podrá predecir bien.

Como el objetivo del entrenamiento es minimizar el costo de todos los diversos errores tomados en conjunto. El punto más bajo cercano a cero es el mínimo donde el modelo tiene el menor error que se encuentra con el error cuadrático medio y ese conjunto de pesos será el que mejor se adapte a todo el conjunto de entrenamiento. Este enfoque es el denominado aprendizaje por lotes en el cual en cada época el algoritmo está realizando un descenso del gradiente al tratar de minimizar el error. Sin embargo, la superficie de error puede presentar una curva de error no convexa y se pueden generar confusiones con el mínimo local. Para evitar este problema se utiliza el descenso de gradiente estocástico que actualiza los pesos después de cada ejemplo de entrenamiento en vez de analizar todo el conjunto de entrenamiento y los reorganiza. Otra de las opciones y la más utilizada es el entrenamiento por minilotes donde se entrena un subconjunto de pequeño lote con los errores asociados al minilote y se corrigen y luego se continúa con el siguiente mini-lote hasta terminar con el conjunto completo y esto constituye una época.

Thomas Mikolov codificó el significado de palabras en dimensiones vectoriales y entrenó una red neuronal para predecir ocurrencias cerca de cada palabra de destino en 2012. Word2vec es el software que crea estos vectores de palabras. Word2vec procesa un gran corpus de texto sin etiquetar, sin categorizar y sin estructurar y aprende el significado de las palabras. La naturaleza no supervisada de Word2vec es lo que lo hace tan poderoso, teniendo estos dos enfoques tan radicalmente diferentes uno del otro. En el aprendizaje supervisado los datos de entrenamiento se etiquetan previamente y solo puede mejorar si puede medir la diferencia entre la salida esperada o etiqueta y sus predicciones. Las incrustaciones de palabras o embedding de Word2vec son cuatro veces más precisas en los modelos del LSA. Otra opción a las incrustaciones de palabras que da una representación vectorial, tiene en cuenta el contexto y puede representar una oración completa son los codificadores bidireccionales de transformadores más conocidos como BERT.

La estructura de la red neuronal para predecir palabras circundantes consta de dos capas de pesos, capa oculta y un determinado número de dimensiones vectorizadas utilizadas para representar una palabra. La función de activación de la capa de salida es una función softmax que se utiliza habitualmente para problemas de clasificación y cuyo resultado puede considerarse una probabilidad porque la salida estará entre 0 y 1, siendo 1 la suma de todas las salidas. Para cada

uno de los nodos de salida, el valor de salida softmax se puede calcular utilizando la función exponencial normalizada. El uso de las incrustaciones de palabras son el resultado de la red neuronal que puntea la similitud de las palabras para identificar las que ocurren en contextos similares. Con estos vectores de palabras se puede identificar múltiples categorías como se puede hacer con el análisis semántico latente, n-gramos además de capturar la similitud semántica mediante la similitud del coseno o identificar significados implícitos.

Existen dos formas posibles de entrenar un modelo y son el enfoque skip-gram que son n-gramas que predicen palabras circundantes y funciona bien para corpus pequeños y el enfoque bolsa continua de palabras que predice la palabra de salida a partir de las palabras cercanas y que tiene mayor precisión en las palabras frecuentes y es más rápido de entrenar. Esta técnica también se extiende a oraciones, párrafos y documentos y ver su similitud y hacer una predicción, basándose en los vectores de párrafos o documentos anteriores.

La complejidad de la comunicación y los desafíos que contiene para que el objetivo de la comunicación se cumpla lo más aproximado posible ha contribuído a que se continúe investigando sobre las redes neuronales y gracias a herramientas de código abierto se han podido encontrar patrones y convertir el perceptrón en un perceptrón multicapa que ha desarrollado herramientas más eficientes y precisas para identificar patrones en grandes conjuntos como son las redes neuronales convolucionales y redes neuronales recurrentes.

3.3.2. Redes Neuronales Convolucionales

El orden y la proximidad son formas de expresar la correlación entre las palabras. Una red neuronal convolucional, o *convnet* se basa en el Neocognitron que es una red neuronal jerárquica de varias capas y que se utilizó originalmente en el reconocimiento de imágenes y patrones. Fue propuesta por Kunihiko Fukushima en 1979. Esta red no asigna pesos a cada elemento sino que define un conjunto de filtros o núcleos que se mueven a través de la imagen. Este filtro se mueve como una ventana por los datos y se compone de dos parámetros que son una función de activación y un conjunto de pesos. Una vez que se tiene una salida final se puede calcular el error y volver a propagar ese error a través de la red.

Las redes neuronales convolucionales también se pueden usar para el procesamiento del lenguaje natural por medio de los vectores de palabras o incrustaciones de palabras. Esto es posible porque no se hace un tratamiento del espacio bidimensional como en las imágenes,

sino que se centra en el filtro unidimensional y las relaciones relativas horizontales haciendo un reconocimiento de los patrones [6].

La arquitectura de una red neuronal convolucional es igual que una red neuronal, pero la primera pieza es una capa convolucional. La salida es menor que la entrada y se establece el relleno como válido. En cada turno en la convolución será un token y el ancho de la ventana o *Kernel* se establece en tres tokens. Se utiliza la unidad lineal rectificada o ReLU en inglés *(*Rectified Linear Units) como función de activación para obtener como salida 0's o valores positivos.

Mediante la agrupación o *Pooling*, se pretende reducir el problema de dimensionalidad y se acelera el proceso al permitir la paralelización de los cálculos. Hay tres opciones para agrupar, uno es el promedio que toma mayor cantidad de datos, otra es el máximo que toma el valor de activación más alto o característica prominente de la región dada, y la tercera es el máximo global.

La última capa de la red neuronal es una función de activación que se basa en la función sigmoide entonces da resultados entre 0 y 1. Siendo el clasificador real de la red, cualquier resultado entre 0 y 0.5 lo clasifica como 0 y por encima de 0.5 lo clasifica como 1. Se suele utilizar un optimizador para minimizar la función de pérdida de una red neuronal y cada uno de los optimizadores que se pueden utilizar tiene una gran cantidad de hiper parámetros que se pueden modificar, si no se aceptan los que vienen por defecto.

Luego se entrena el modelo mediante el fit y se activan todas las entradas con los pesos, las funciones de activación y la retropropagación. El modelo pre entrenado se puede guardar y luego cargarlo en una tubería.

3.3.3. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) son redes neuronales que se retroalimentan, de sucesos cercanos y de más alejados en el tiempo. En una red neuronal recurrente, la neurona de entrada no solo envía información a la neurona siguiente sino que también esta información es envíada a sí misma. Este suceso es lo que le permite que haya cierto grado de memoria en épocas anteriores, aprendiendo patrones de secuencias temporales.

Uno de los problemas que tienen las redes neuronales recurrentes es el descenso del gradiente o la derivada parcial que se hace cuando se busca el mínimo local tratado anteriormente,

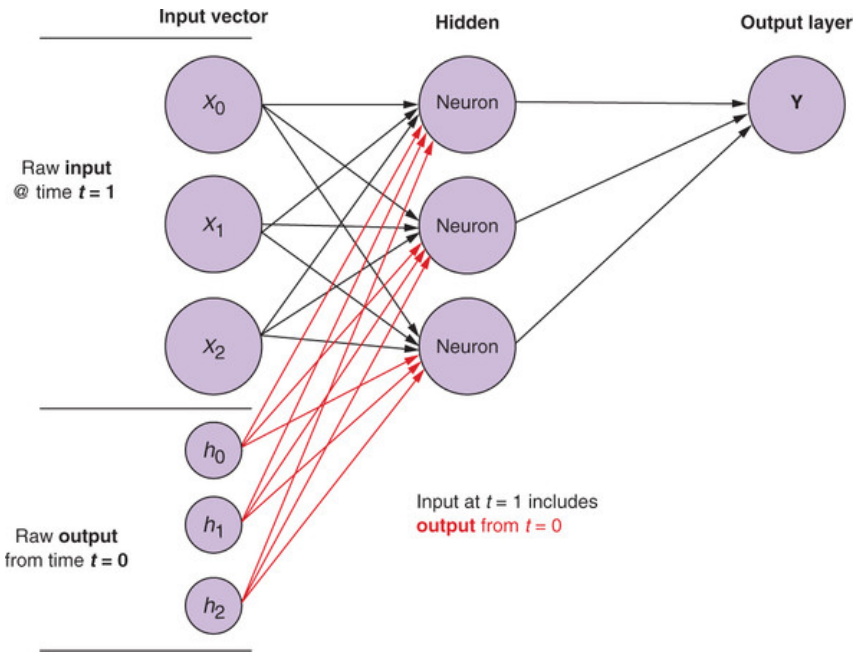

Figura 3.6: Redes Neuronales Recurrentes en el paso de tiempo 1.

porque en la retropropagación también se envía el error sin mejorarlo. Sólo se puede hacer una mejora calculando el error en el penúltimo paso y corregirlo. Sin embargo, esto puede generar los gradientes explosivos en inglés *"Exploding gradients"* o los gradientes desaparecidos *"Vanishing gradients"*, en uno la asignación del error son exagerados y en el otro tienden a desaparecer.

Las redes neuronales recurrentes extienden su memoria para aprender de sucesos que han pasado en largos periodos de tiempo por medio de la Memoria larga a corto plazo (LSTM) *(Long short-term memory)*, ayudando a solucionar el problema de desvanecimiento del gradiente.

La LSTM permite memorizar y almacenar intervalos aleatorios. Esta memoria persiste en todos los pasos temporales de las series temporales. Una neurona LSTM tiene tres puertas, una de entrada, una de olvido y otra de salida, regulando el flujo de información. Cada una de estas puertas son una capa de red compuesta por pesos y con una función de activación sigmoide que la incluye en la propagación y actualiza a su vez el estado de la memoria. manteniendo el error cerca de la neurona en cada paso, esto último es más conocido como el carrusel de errores.

3.4. Técnicas de balanceo

En el entrenamiento de un modelo de inteligencia artificial, como es este caso, es necesario que los datos que se van a entrenar estén equilibrados para que el modelo sea lo más exacto posible en su predicción. Si esto no sucede, el modelo entrenado tendrá una tendencia a predecir con mayor exactitud la clase con mayor número de datos de entrenamiento. Sin embargo, muchos algoritmos están diseñados para maximizar la precisión general de forma predeterminada.[2]

El problema radica en que surge un desequilibrio de clases porque existen diferencias significativas en las probabilidades previas de las dos clases, dando como resultado un algoritmo de clasificación sesgado hacia la clase mayoritaria y arrojando un mayor número erróneo de clasificación hacia la clase minoritaria.

Existen varias técnicas para tratar los datos desbalanceados como son:

1. El ajuste de parámetros del modelo de forma predeterminada como se mencionaba anteriormente. Este algoritmo penaliza a la clase mayoritaria durante el entrenamiento.

2. Métodos de conjunto equilibrado entrenando diversos modelos y entre todos obtienen el resultado final.

3. Modificar la base de datos eliminando muestras de la clase mayoritaria o agregando nuevas filas con los mismos valores.

4. Muestras artificiales utilizando diversos algoritmos para crear muestras sintéticas y no idénticas que intentan seguir la tendencia del grupo.[3]

En este estudio se analizan algunas de estas técnicas antes descritas de sobremuestreo, submuestreo y sobremuestreo con *SMOTE (Synthetic Minority Over-sampling Technique)*.

3.4.1. Submuestreo

El submuestreo o *Undersampling* es la técnica por la cual se equilibra la base de datos recortando la clase mayoritaria para favorecer a la clase minoritaria y así equilibrar la base de datos. La principal desventaja es que se puede perder mucha información valiosa, cuando el desequilibrio es alto.

[2] https://elitedatascience.com/imbalanced-classes

[3] https://aprendemachinelearning.com/clasificacion-con-datos-desbalanceados

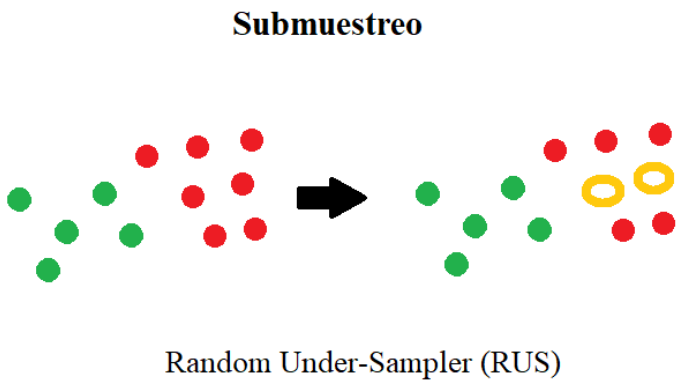

Figura 3.7: Ejemplo gráfico de las modificaciones introducidas en un *dataset* por el algoritmo RUS.

3.4.2. Sobremuestreo

El sobremuestreo o *Oversampling* es la técnica en la cual se duplican los datos de la clase minoritaria y se agregan a la base de datos. La desventaja es que no proporciona información adicional al modelo entrenando este con datos falsos o sintéticos. También se utiliza la técnica de sobremuestreo aleatorio, la cual también es conocida como validación cruzada de Monte Carlo, donde se puede elegir el tamaño de los subconjuntos en la división aleatoria de los datos.

3.4.3. SMOTE

La técnica *SMOTE (Synthetic Minority Over-sampling Technique)* hace que la copia no sea exacta y se explica a continuación. En el gráfico como se ha explicado antes y la podemos ver en la gráfica3.9 La técnica de sobremuestreo de minoría sintética parte de que hay muy pocos ejemplos de la clase minoritaria. Esta técnica descrita por Yunqian et al. hace una descripción del sobre muestreo sintético . ”. . . SMOTE primero selecciona una instancia de clase minoritaria al azar y encuentra sus k vecinos de clase minoritaria más cercanos. Luego, la instancia sintética se crea eligiendo uno de los k vecinos b más cercanos al azar y conectando a y b para formar un segmento de línea en el espacio de características. Las instancias sintéticas se generan como una combinación convexa de las dos instancias elegidas a y b.”[4]

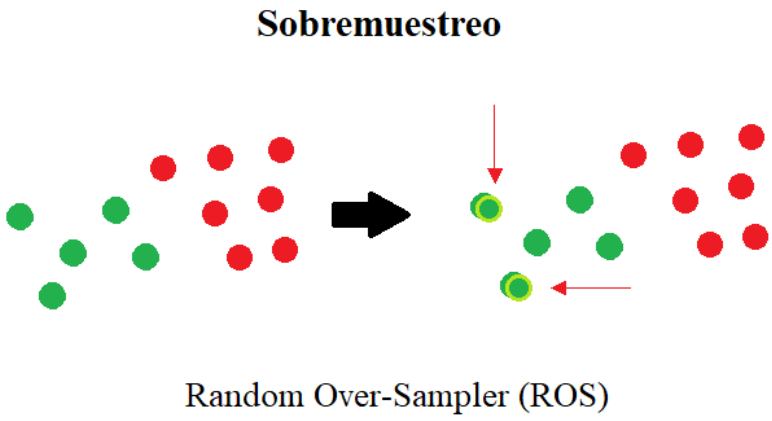

Figura 3.8: Ejemplo gráfico de las modificaciones introducidas en un *dataset* por el algoritmo ROS.

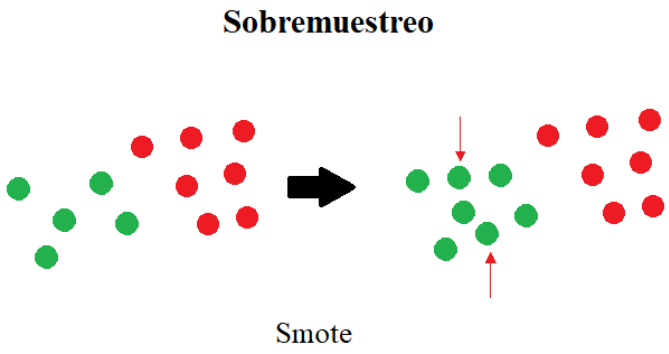

Figura 3.9: Ejemplo gráfico de las modificaciones introducidas en un *dataset* por el algoritmo SMOTE.

Capítulo 4

Diseño e implementación

En este caso se diseña una tubería de código en el que se aplican varias técnicas descritas en el estado del arte para el funcionamiento del modelo y su evaluación. El diseño se hace mayoritariamente con las librerías SpaCy y SciKit-Learn las cuales cubren los requisitos necesarios para el funcionamiento del modelo. La librería Spacy se encarga de la primera parte del procesamiento del texto y la librería Scikit-Learn se encarga de la clasificación. Adicionalmente se utilizan varias librerías como re, string y unicode para el apoyo de la extracción de características y Seaborn, Matplotlib y WordCloud para visualización y apoyar los resultados obtenidos.

El proceso empieza con la preparación de los datos y la carga de la base de datos, asignando a las noticias verdaderas la categoría 1 y a las noticias falsas la etiqueta 0. Se continúa con el preprocesamiento de los datos normalizando el data frame, removiendo los espacios en blanco, la puntuación, los caracteres especiales y las contracciones más comunes. Además, se crea el analizador semántico y se incorpora el eliminador de las palabras vacías de SpaCy.

En la siguiente etapa se carga la librería Scikit-learn para la extracción de características y tokenización. Se continúa con un clasificador para finalmente crear la tubería, la cual se entrenará con los algoritmos supervisados de *Random Forests*, el clasificador *Support Vector Machine* en su función del *kernel* o núcleo lineal y el *Gradient Boosting*.

Finalmente, se hace un modelo de evaluación que incluye la media precisión y la matriz de confusión para cada uno de los tres algoritmos implementados.

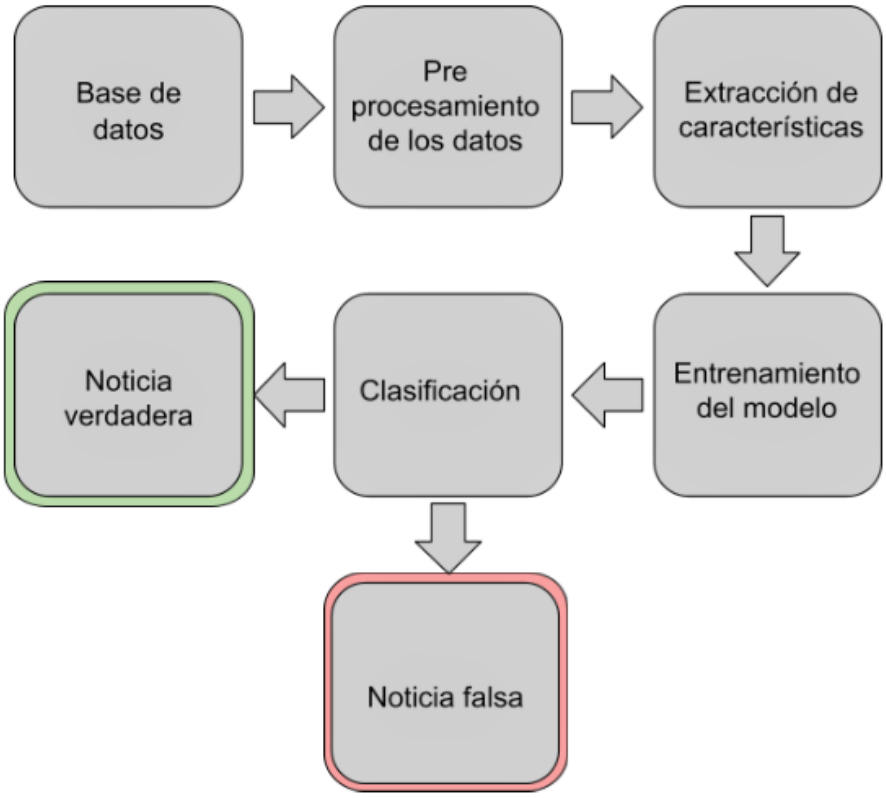

Figura 4.1: Flujo de trabajo.

4.1. Arquitectura general

La arquitectura general de la tubería de clasificación está constituida por cinco bloques como se puede observar en la figura 4.1. Inicialmente se carga la base de datos al modelo y posteriormente se hace una preprocesamiento de los datos con el objetivo de normalizar y limpiar la base de datos de ruidos que hacen que el modelo sea menos eficiente o no aportan información valiosa para su óptimo funcionamiento. Después del preprocesamiento de los datos, se continúa con la extracción de características y el flujo de la tubería con la extracción de características, y luego con el entrenamiento del modelo, que posteriormente clasificará las noticias en verdaderas o falsas.

Capítulo 5

Experimentos y validación

5.1. Experimento 1: Análisis de la base de datos original

El objetivo del primer experimento es conocer y hacer una descripción de la base de datos, analizar el modelo de clasificación mediante los algoritmos de bosques aleatorios (*Random forest*), máquina de vector soporte (*Support Vector Machine*) y potenciación del gradiente (*Gradient boosting*).

La base de datos se ha extraído de Kaggle [1] la cual tiene agradecimientos a Ahmed H, Traore I, Saad S. El conjunto de los datos está en su versión primera y ocupa un total de 116.37 MiB, los cuales están distribuídos en dos archivos csv (*Comma Separated Values*). Las noticias falsas *FakeNews* son 23481 y ocupan 62.79 MiB y las noticias verdaderas *Real News* ocupan 53.58 MiB, alcanzando el número de 21417 noticias.

En el gráfico de 5.1 se observa la distribución en las dos categorías clasificatorias.

Después de hacer una descripción de la base de datos se procede al preprocesamiento de esta, se eliminan los caracteres especiales y la puntuación. Adicionalmente, se agrega una lista de las contracciones más comunes para que su procesamiento sea más óptimo.

También se construye una lista de palabras de parada (*Stopwords*) para filtrar las palabras más comunes y que no aportan información al estudio. Con el gráfico de nube de palabras 5.2 se observa en mayor tamaño las palabras más frecuentes del conjunto de la base de datos. Por otra lado se observarán las palabras más comunes en mayor tamaño en cada uno de las características.

[1] https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

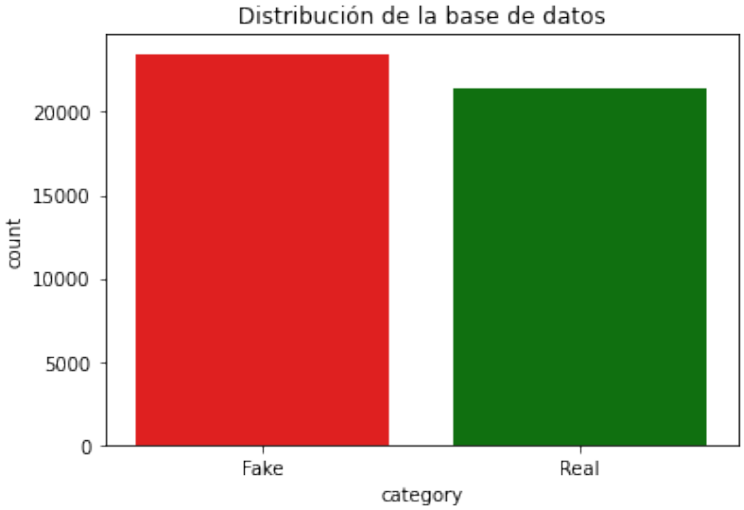

Figura 5.1: Categorías en la base de datos.

Figura 5.2: Nube de palabras de la BBDD.

Luego se procede a la extracción de características, que es donde se ensambla la primera parte de la tubería en la cual se tokeniza, vectoriza y clasifica la base de datos. La tokenización es posible gracias a la lemmatización y el parser (código 5.1). Además, se incorpora un transformador donde también se ajustan los parámetros y se limpia el texto (códigos 5.2 y 5.3).

```
def spacy_tokenizer(sentence):
    mytokens = parser(sentence)
    mytokens = [ word.lemma_.lower().strip() if word.lemma_
    != "-PRON-" else word.lower_ for word in mytokens ]
    mytokens = [ word for word in mytokens if word not in
    stopwords and word not in punctuations ]
    return mytokens
```

Código 5.1: Extracción de características.

```
class predictors(TransformerMixin):
    def transform(self, X, **transform_params):
        return [clean_text(text) for text in X]
    def fit(self, X, y=None, **fit_params):
        return self
    def get_params(self, deep=True):
        return {}
```

Código 5.2: Transformador personalizado usando spaCy.

Posteriormente se vectoriza en una matriz de recuentos de palabras donde se extraerán sólo los unigramas y se hará una reducción de las características y del tamaño del vocabulario como vemos en el código 5.4. El estimador que se utiliza es el algoritmo de aprendizaje automático y clasificador de vectores de soporte *Support Vector Classifier* en su núcleo por defecto que es el radial.

Se continúa con la siguiente parte de la tubería en la cual se etiqueta el conjunto de datos en características y categorías para entrenar el modelo, extrayendo un 20 % de los datos de prueba,

```
def clean_text(text):
    return text.strip().lower()
```

Código 5.3: Función básica para limpiar el texto.

```
vectorizer = CountVectorizer(tokenizer = spacy_tokenizer,
ngram_range=(1,1)) #unigramas
classifier = SVC() #por defecto el rbf= radial
```

Código 5.4: Función de vectorización de tokens textuales.

se siembra una semilla para que la muestra extraída sea siempre la misma. Esto lo podemos ver en el código 5.5.

```
x = df['text']
ylabels = df['category']
x_train, x_test, y_train, y_test = train_test_split
(x, ylabels, test_size=0.2, random_state=42)
##
x_train_vector = vectorizer.fit_transform(x_train)
x_test_vector = vectorizer.transform(x_test)

## División de conjuntos de datos
# Características y etiquetas
x = df['text']
ylabels = df['category']
x_train, x_test, y_train, y_test = train_test_split
(x, ylabels, test_size=0.2, random_state=42)
##
x_train_vector = vectorizer.fit_transform(x_train)
x_test_vector = vectorizer.transform(x_test)
```

Código 5.5: División de conjuntos de datos, características y etiquetas.

Seguidamente se continúa con otra parte de la tubería donde el modelo se transforma y se prueba su comportamiento con los tres algoritmos objeto de estudio, que son los bosques aleatorios (*Random Forests*), el de máquina de vector soporte en el núcleo lineal (*Support Vector Machine*) y con potenciación del gradiente (*Gradient Boosting*), como muestra el código 5.6.

```
# Random Forest
rf = RandomForestClassifier(random_state = 42)
rf.fit(x_train_vector, y_train)

# Support Vector Machines (SVM)
svc = SVC(kernel='linear', random_state = 42)
svc.fit(x_train_vector, y_train)

# Gradiente Boosting
gbrt = GradientBoostingClassifier(random_state = 42)
gbrt.fit(x_train_vector, y_train)
```

Código 5.6: Ajuste de algoritmos de aprendizaje máquina utilizados.

Finalmente se organizan y notifican los resultados numéricos de este experimento los cuales son resaltados en la tabla 5.3.

Además, se estructuran los resultados obtenidos y se utilizan algunas métricas de rendimiento para analizarlos. Las métricas utilizadas son: el puntaje de precisión, el porcentaje de recuperación, la media F que es la armónica entre las dos anteriores. Adicionalmente se agrega la matriz de confusión 5.4 para examinar el número de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos y la curva de ROC *(Receiver Operating Characteristic*) para observar la sensibilidad de clasificación del sistema y ver cuál es la predicción más óptima.5.5

5.2. Experimento 2: Análisis de la base de datos con calidad degradada

Con el segundo experimento se evalúa cómo es el comportamiento de los algoritmos degradando la base de datos manteniendo constantes las noticias falsas y penalizando la noticias verdaderas que es la clase minoritaria y en proporción 70 %-30 %, luego una distribución del 75 %-25 %, también otra proporción 80 %-20 % y finalmente una distribución 85 %-15 % como se observa en el gráfico 5.6.

Puntaje precisión de clasificación	
Algoritmo	**Precisión**
Random Forest	0,99310
SVC	0,99710
Gradient Boosting	0,99510

Informe de clasificación			
Algoritmo	**Métrica**	**Falsas**	**Verdaderas**
Random Forest	**Precisión**	1,00	0,99
Random Forest	**Recuperación**	0,99	1,00
Random Forest	**Media F**	0,99	0,99
Random Forest	**Soporte**	4650	4330
SVC	**Precisión**	1,00	1,00
SVC	**Recuperación**	1,00	1,00
SVC	**Media F**	1,00	1,00
SVC	**Soporte**	4650	4330
Gradient Boosting	**Precisión**	1,00	0,99
Gradient Boosting	**Recuperación**	0,99	1,00
Gradient Boosting	**Media F**	1,00	0,99
Gradient Boosting	**Soporte**	4650	4330

Figura 5.3: Resultados primer experimento.

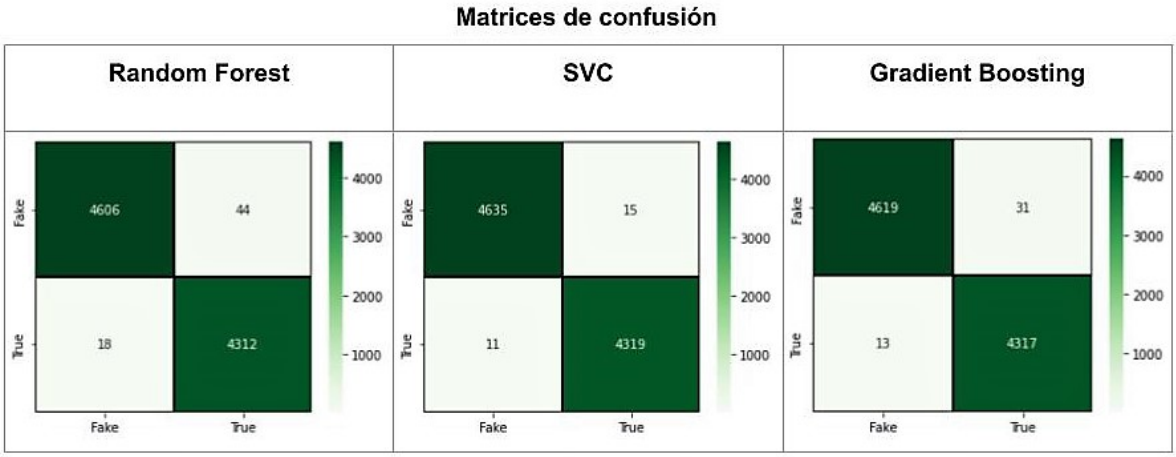

Figura 5.4: Matrices de confusión primer experimento.

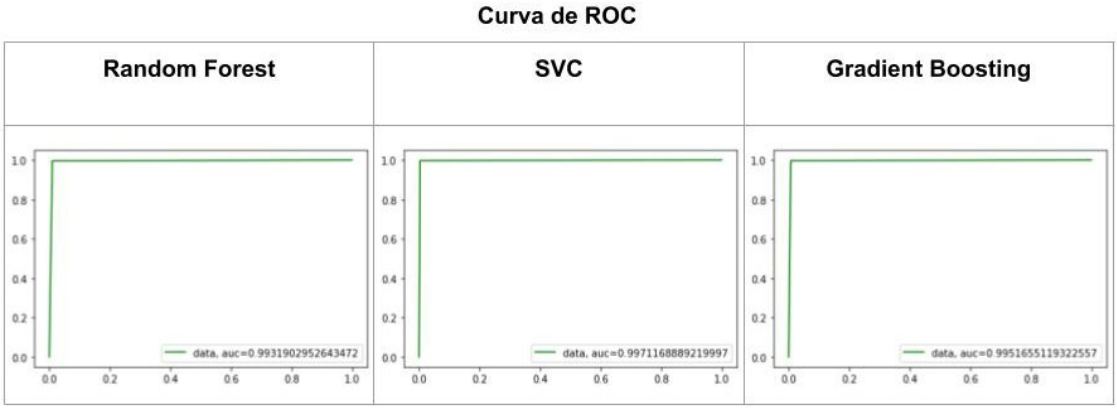

Figura 5.5: Curvas de ROC primer experimento.

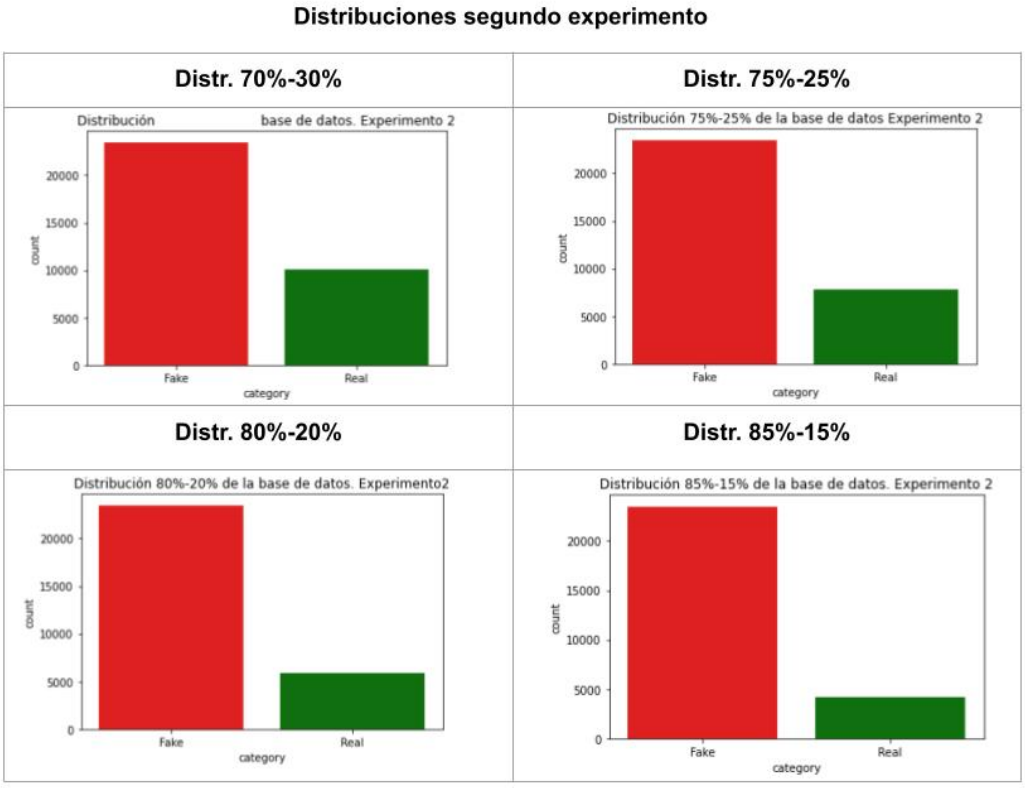

Figura 5.6: Distribuciones segundo experimento.

Puntaje de precisión de clasificación por distribución				
Algoritmo	Precisión dist. 70%-30%	Precisión dist. 75%-25%	Precisión dist. 80%-20%	Precisión dist. 85%-15%
Random Forest	0,98986	0,98994	0,98774	0,98208
SVC	0,99627	0,99601	0,99659	0,99729
Boosting Gradient	0,99493	0,99345	0,99336	0,99511

		INFORME DE CLASIFICACIÓN SEGUNDO EXPERIMENTO							
		Distribución 70%-30%		Distribución 75%-25%		Distribución 80%-20%		Distribución 85%-15%	
Algoritmo	Métrica	Falsas	Verdaderas	Falsas	Verdaderas	Falsas	Verdaderas	Falsas	Verdaderas
Random Forest	Precisión	0,99	0,99	0,99	0,99	0,99	1,00	0,98	1,00
Random Forest	Recuperación	1,00	0,97	1,00	0,97	1,00	0,94	1,00	0,88
Random Forest	Media F	0,99	0,98	0,99	0,98	0,99	0,97	0,99	0,94
Random Forest	Soporte	4698	2011	4704	1558	4743	1128	4704	821
SVC	Precisión	1,00	0,99	1,00	0,99	1,00	0,99	1,00	0,99
SVC	Recuperación	1,00	0,99	1,00	0,99	1,00	0,99	1,00	0,99
SVC	Media F	1,00	0,99	1,00	0,99	1,00	0,99	1,00	0,99
SVC	Soporte	4698	2011	4704	1558	4743	1128	4704	821
Boosting Gradient	Precisión	1,00	0,99	1,00	0,98	1,00	0,97	1,00	0,97
Boosting Gradient	Recuperación	0,99	1,00	0,99	1,00	0,99	1,00	0,99	1,00
Boosting Gradient	Media F	1,00	0,99	1,00	0,99	1,00	0,98	1,00	0,98
Boosting Gradient	Soporte	4698	2011	4704	1558	4743	1128	4704	821

Figura 5.7: Resultados segundo experimento.

El proceso que se sigue es el mismo realizado en el primer experimento para cada una de las distribuciones que se evalúan, cargando en sesiones separadas y cargando la base de datos en cada una de las proporciones y tuberías, para que los resultados sean limpios.

Los resultados obtenidos se pueden observar en la tabla 5.7 donde se testifican los datos de precisión de cada distribución y el comportamiento de los algoritmos supervisados que se están analizando. Se han marcado en verde los valores representativos.

En la gráfica 5.8 vemos el resultado de la predicción en la matriz de confusión.

Finalmente observamos los resultados de validación de los modelos con la curva de ROC 5.9.

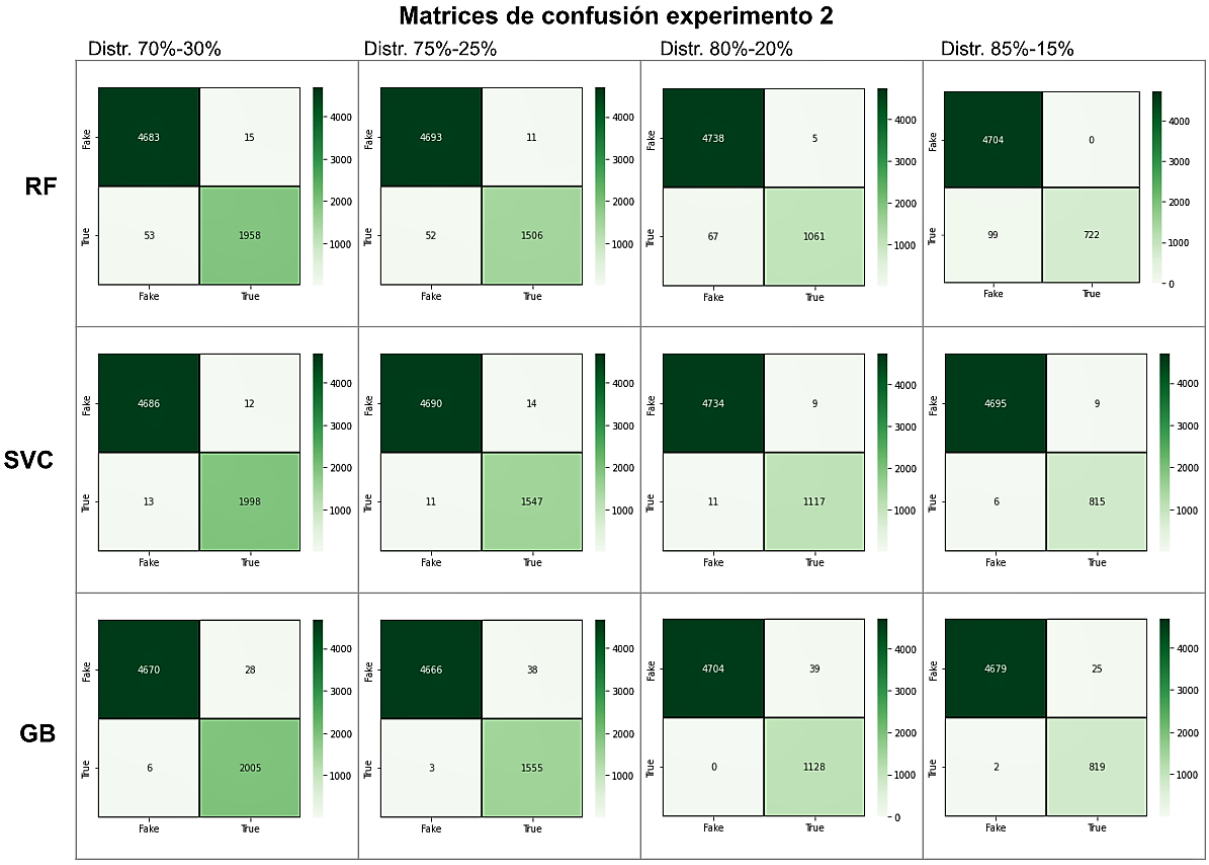

Figura 5.8: Matrices de confusión segundo experimento.

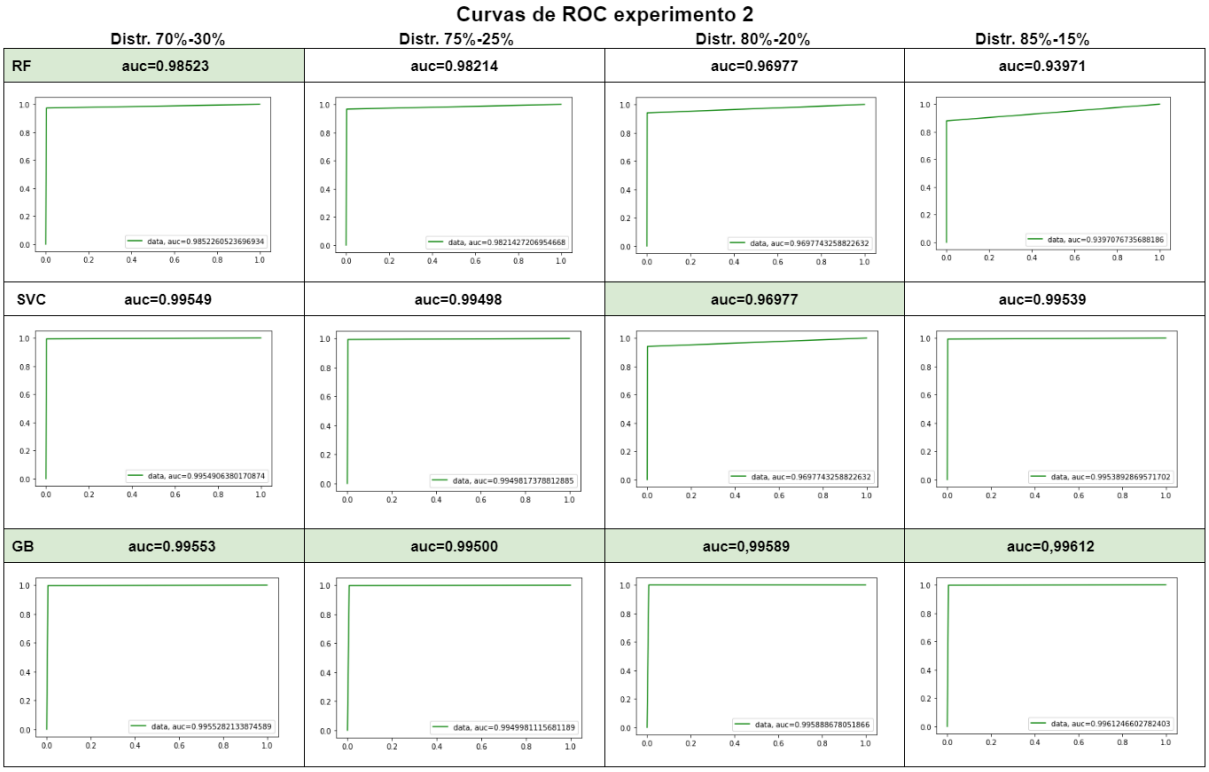

Figura 5.9: Curvas de ROC segundo experimento.

5.3. Experimento 3: Técnicas de balanceo y la técnica *SMOTE*

Para el tercer experimento se utiliza la distribución 80 %-20 % del experimento anterior para analizar las tres técnicas de ajuste de una base de datos desequilibrada. Las tres técnicas son el submuestreo, sobremuestreo y sobremuestreo con *SMOTE*.

Para analizar estas técnicas se utiliza la librería *Imblearn* que es de código abierto y basada en Scikit-learn. Para la técnica de submuestreo se emplea el algoritmo *RandomUnderSampler*. Como se puede observar en la figura 5.10, en el submuestreo la clase minoritaria es la que hace que la clase mayoritaria sea penalizada y la muestra aleatoria sea más pequeña.

```
rus = RandomUnderSampler(random_state=42)
x_res, y_res = rus.fit_resample(x_train_vector, y_train)
```

Código 5.7: Inicio del método de submuestreo.

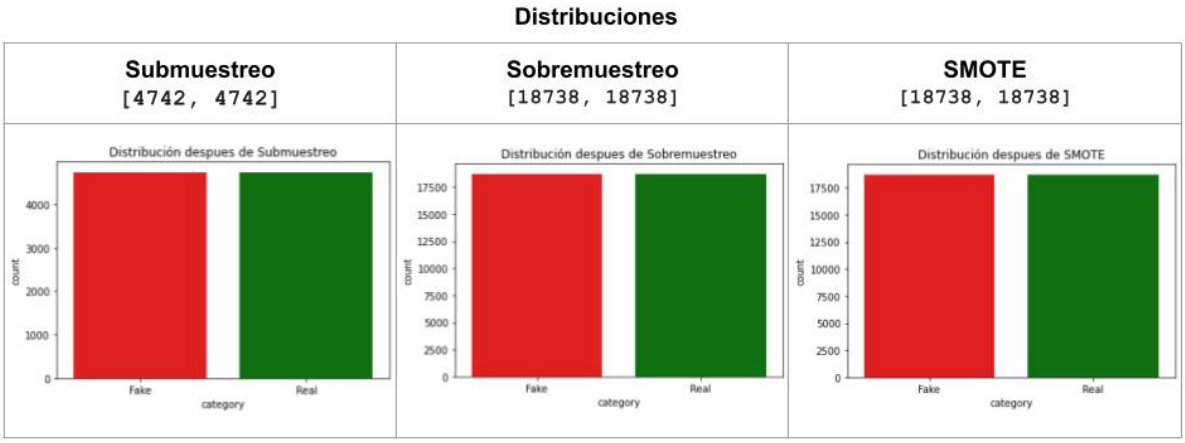

Figura 5.10: Distribuciones tercer experimento.

Para la técnica de sobremuestreo se utiliza *RandomOverSampler* para duplicar las entradas en la etiqueta de clase minoritaria y equilibrar ambas muestras.

```
ros = RandomOverSampler(random_state=42)
x_res, y_res = ros.fit_resample(x_train_vector, y_train)
```

Código 5.8: Técnica de sobremuestreo.

La técnica de sobremuestreo *RandomOverSampler* duplica los datos de la clase minoritaria para equilibrarlos con la clase *Fake*

Posteriormente se llama la técnica *SMOTE* la cuál equilibra la distribución en igual tamaño de muestra que en la técnica de sobremuestreo.

```
sm = SMOTE(random_state=42)
x_res, y_res = sm.fit_resample(x_train_vector, y_train)
```

Código 5.9: Invocación del algoritmo SMOTE.

En la figura 5.11 se examinan los resultados fruto del estudio de este trabajo. Se resalta en verde los valores representativos tanto para las distribuciones como para los algoritmos que se evalúan.

Por último se presenta los resultados de evalución de la curva de ROC para las diferentes técnicas 5.13.

Puntaje de precisión de clasificación por distribución				
Algoritmo	Precisión dist. 80%-20%	Precisión dist. Submuestreo	Precisión dist. Sobremuestreo	Precisión dist. SMOTE
Random Forest	0,98986	0,97701	0,99114	0,97990
SVC	0,99627	0,99353	0,99659	0,98433
Gradient Boosting	0,99493	0,99216	0,99370	0,99285

		INFORME DE CLASIFICACIÓN TERCER EXPERIMENTO							
		Precisión dist. 80%-20%		Precisión dist. Subremu		Precisión dist. Sobremu		Precisión dist. SMOTE	
Algoritmo	Métrica	Falsas	Verdaderas	Falsas	Verdaderas	Falsas	Verdaderas	Falsas	Verdaderas
Random Forest	Precisión	0,99	0,99	1,00	0,90	0,99	0,99	0,98	0,97
Random Forest	Recuperación	1,00	0,97	0,97	0,99	1,00	0,96	0,99	0,92
Random Forest	Media F	0,99	0,98	0,99	0,94	0,99	0,98	0,99	0,95
Random Forest	Soporte	4698	2011	4743	1128	4743	1128	4743	1128
SVC	Precisión	1,00	0,99	1,00	0,97	1,00	0,99	0,99	0,96
SVC	Recuperación	1,00	0,99	0,99	1,00	1,00	0,99	0,99	0,96
SVC	Media F	1,00	0,99	1,00	0,98	1,00	0,99	0,99	0,96
SVC	Soporte	4698	2011	4743	1128	4743	1128	4743	1128
Gradient Boosting	Precisión	1,00	0,99	1,00	0,96	1,00	0,97	1,00	0,96
Gradient Boosting	Recuperación	0,99	1,00	0,99	1,00	0.99	1,00	0,99	1,00
Gradient Boosting	Media F	1,00	0,99	1,00	0,98	1,00	0,98	1,00	0,98
Gradient Boosting	Soporte	4698	2011	4743	1128	4743	1128	4743	1128

Figura 5.11: Resultados tercer experimento.

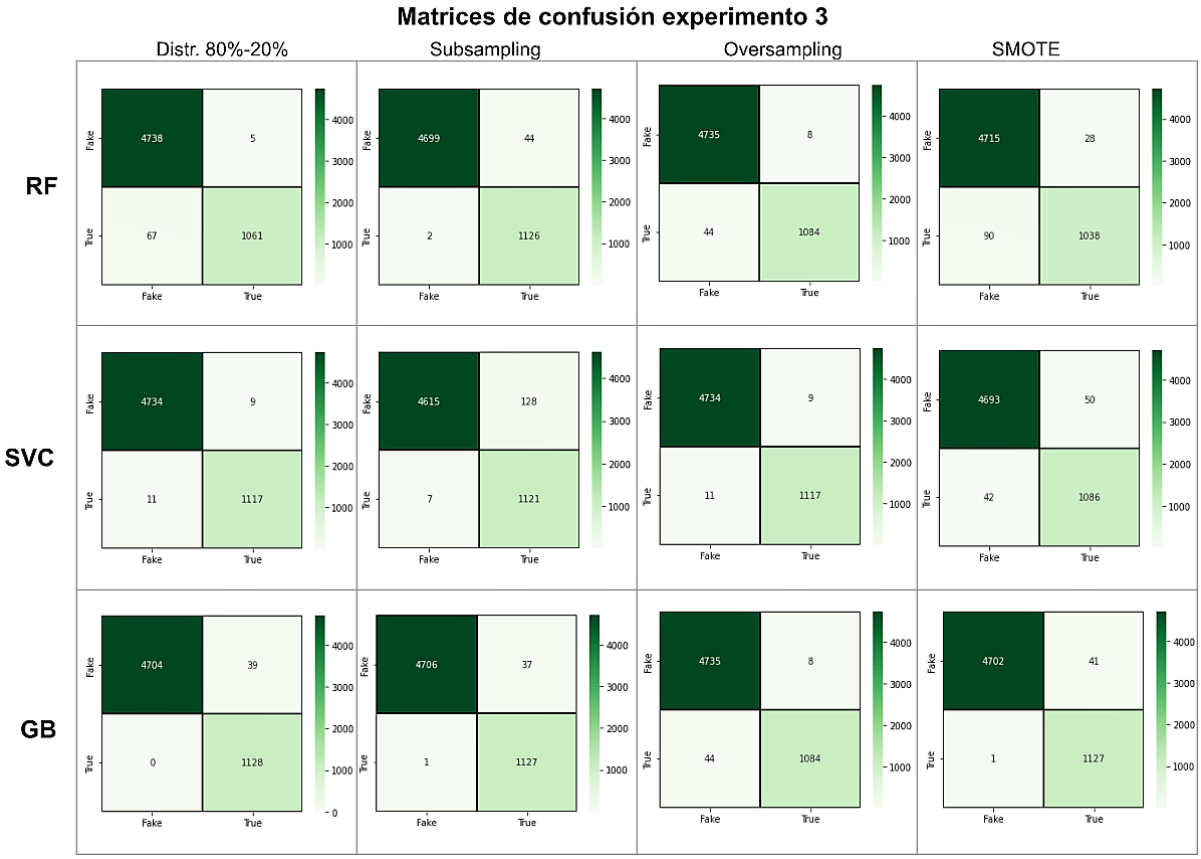

Figura 5.12: Matrices de confusión tercer experimento.

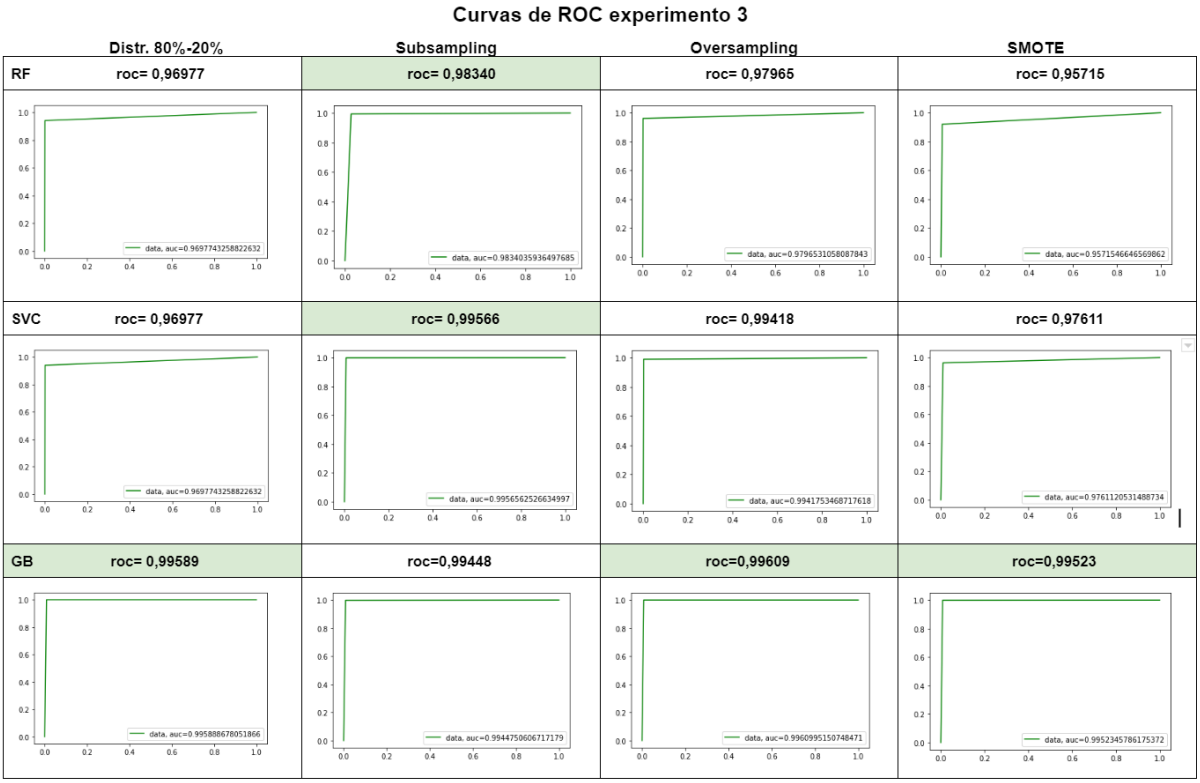

Figura 5.13: Curvas de ROC tercer experimento.

5.4. Resumen de resultados

Como se puede observar en los resultados del primer experimento el algoritmo clasificador de *Support Vector Machine* en su núcleo lineal ha obtenido mejores datos y menos errores de clasificación tanto para las noticias falsas como con las verdaderas. Seguido de el *Gradient Boosting* y por último *Random Forest*.

Por medio de las matrices de confusión, reflejadas en la figura 5.8, podemos confirmar que el algoritmo con menor falsos negativos y falsos positivos es *Support Vector Machine*. De la misma manera, la curva de ROC en este algoritmo es la más precisa con un 0,9971.

En el segundo experimento en la primera tabla de la figura 5.7 podemos observar que el algoritmo (*Support Vector Machine*) es el que mejor clásica de media en todas las proporciones según el puntaje de precisión, pese a la descomposición de la clase minoritaria.

También se puede ver que el indicador F1 o Media F, que es útil para las distribuciones desiguales, tiene en *Support Vector Machine* iguales resultados en las etiquetas falsas que el más modesto y exhaustivo en sus predicciones como es *Gradient Boosting* en todas las proporciones. A su vez, este último tiene mejores datos de recuperación o exhaustividad con las etiquetas verdaderas para todas las distribuciones.

De la misma manera se puede observar que se mantienen diferencias significativas en el número de ocurrencias entre las clases *Support* y varían muy poco entre las distintas distribuciones y algoritmos que se evalúan.

En la gráfica 5.8 vemos el resultado de la predicción en la matriz de confusión. Como se puede observar en el conjunto, los resultados son similares. Sin embargo, la que mejor resultado presenta es la *Gradient Boosting* en la distibución 80 %-20 % en calidad de precisión con falsos negativos y el *Random Forest* en la distribución 85 %-15 % en exhaustividad con los falsos positivos.

Para validar el segundo experimento también se utiliza la curva de ROC. Como se puede observa en la gráfica 5.9 el algoritmo *Random Forest* presenta mejores resultados en la distribucion 70 %-30 %, el algoritmo *Support Vector Machine* da mejores resultados en la distribución 80 %-20 % y el *Gradient Boosting* en la distribución 85 %-15 %.

Después de analizar los tres algoritmos de aprendizaje supervisados como son (*Random Forest*), *Support Vector Machine* y *Gradient Boosting* con disferentes distribuciones, el algoritmo que mejor presenta resultados de media después de su entrenamiento es el *Gradient Boosting*.

Analizando los el tercer experimento, la técnica de sobremuestreo *RandomOverSampler* es la que mejor precisión de clasificación genera con el algoritmo *Support Vector Machine* y por su parte ha hecho que el algoritmo *Random Forest* tenga ligeramente mejores resultados de precisión que la muestra objeto de comparación. La técnica *SMOTE* sin embargo, presenta resultados ligeramente inferiores a la muestra.

Al mismo tiempo la técnica *SMOTE* ha sido más eficiente dando resultados de precisión a el algoritmo *Gradient Boosting*, aún así siendo inferior a la técnica de submuestreo.

También el indicador F1 o Media F en las tres técnicas de balanceo. en las etiquetas falsas y verdaderas. tiene mejores resultados con el *Gradient Boosting* y además, su recuperación o exhaustividad con respecto a las etiquetas verdaderas es mucho mejor que los otros algoritmos.

En el tercer experimento igual que en el segundo experimento también se puede observar que se mantienen diferencias significativas en el número de ocurrencias entre las clases y varían muy poco entre los distintos algoritmos que estudian y entre las técnicas de balanceo.

Sin embargo, observado los resultados de las matrices de confusión se puede observa que la técnica que es más precisa es la *Oversampling* y la más exhaustiva es *Subsampling* para los tres algoritmos.

Finalmente observando las curvas de ROC, vemos que la técnica que mejor rendimiento tiene con los tres algoritmos es el *Subsampling* y que la técnica *SMOTE* da mejores resultados con el algoritmo *Gradient Boosting*.

Capítulo 6

Conclusiones

Se puede concluir que la técnica de balanceo *Oversampling* es la que mejor presenta resultados en el algoritmo *Gradient Boosting*. La técnica *Subsampling* da mejores resultados en los algoritmos *Random Forest* y *Support Vector Machine* y la técnica de balanceo *SMOTE* funciona mejor con *Gradient Boosting*.

El algoritmo *Gradient Boosting* es el más exhaustivo para las etiquetas de la clase minoritaria y se mantiene en las diversas distribuciones y la técnica *SMOTE* le ayuda a obtener mejores resultados de clasificación.

Un algoritmo que tenga un alto grado de precisión pero no sea exhaustivo en el etiquetado de las clases, puede generar dudas y marcar tendencias determinadas hacia la clase favorecida. Las palabras crean, crean frases, ideas y realidades y cuando están marcadas por la mentira pueden generar caos, incertidumbre y pérdida de credibilidad. Aunque se omita información o se manipule la realidad, siempre existirán factores externos que regularán el equilibrio o que surjan y se desarrollen sistemas contraculturales con un sentido crítico frente al propio sistema o modelo, susceptible al error y a aprender para mejorarlo, modificarlo o reestructurarlo. Esto permite que en tiempos convulsos se cuestione lo individual y lo colectivo y nuestra participación en como ente social.

Cuando se busca la verdad en medio de un mar de mentiras, se identifican patrones, estilos, estructuras, modelos y tendencias y a veces es fácil identificarlas. El problema radica cuando surgen nuevos modelos o sistemas que abanderan discursos radicales o con alta precisión, sin dar la oportunidad de ser cuestionados o estudiados. Así mismo, se apropian de la libertad, de la realidad, de los conceptos de valores, de verdad, del bien y del mal y de moralidad que tienden

a juzgar y no permiten otras formas de pensar. La verdad se construye con múltiples realidades y puntos de vista siendo relativa y absoluta.

Durante la manipulación de los datos es inevitable la intervención con instrumentos donde se distorciona la realidad o exactitud, generando un sesgo que puede ser natural o marcado por una tendencia de poder.

Aunque el método *SMOTE* no presenta mejores resultados, debido al conocimiento de la técnica que utiliza con respecto a las dos estudiadas, parece más verosímil su funcionamiento. Ya que los otros dos métodos unidos a los algoritmos de estudio pueden tener tendencia a sobreajustarse y hacer predicciones con alta precisión, pero con dudas respecto a los resultados.

Finalmente se puede concluir los buenos resultados en los modelos se pueden deber al ajuste de los parámetros durante el entrenamiento, a las librerías utilizadas, o al ensamblaje de las tuberías con múltiples modelos que cada vez son más eficientes.

6.1. Consecución de objetivos

El objetivo general de analizar la desequilibrios de una base de datos construida con noticias falsas y verdaderas mediante los algoritmos de aprendizaje supervisados como son bosques aleatorios (*Random Forests*), las máquinas de vectores soporte (*Support Vector Machine*) y la potenciación del gradiente (*Gradient Boosting*), se ha conseguido. Así como el análisis de los resultados de las técnicas de balanceo.

El objetivo específico de procesar la base de datos para extraer las características de cada clase, su distribución y palabras más comunes, ha permitido conocer la base de datos de origen y cómo ha sido su distribución.

Al construir un modelo supervisado de *Machine learning* de clasificación para procesar, extraer y transformar las características de la base de datos que se entrenó en el modelo, se pudo profundizar en las etapas principales y en las técnicas del procesamiento natural del lenguaje. Además, se conocieron otras aplicaciones y librerías que permiten ensamblar una canalización o tubería.

Se ha entrenado el modelo de clasificación con tres algoritmos supervisados como son los bosques aleatorios (*Random Forest*), máquinas de vectores soporte en su núcleo lineal (*Support Vector Machines with lineal kernel*) y potenciación del gradiente (*Gradient Boosting*), notando

que el algoritmo que mejor clasifica y obtiene mejores resultados es las máquinas de vector soporte (*Support Vector Machine*) usando un núcleo lineal, seguido de *Gradient Boosting* y de *Random Forests*.

Tras analizar la técnica de balanceo de la base de datos desequilibrada y la técnica *SMOTE*, se puede concluir que no es la técnica que obtenga mejores resultados. Sin embargo, su método hace que sea más fiable y más creíble, aceptando la posibilidad de errores y disminuyendo la tendencia o sesgo del algoritmo que aprende.

La técnica de clasificación de *SMOTE*, generando muestras sintéticas y disminuyendo el sesgo, presenta resultados levemente peores. Su mayor precisión es obtenida con el algoritmo de clasificación *Gradient Boosting*.

6.2. Aplicación de lo aprendido

Durante la formación académica he podido profundizar en modelos de análisis avanzado y aplicar los conocimientos matemáticos y numéricos que me han permitido conocer los algoritmos de *Machine Learning* aplicados en la programación, así como las siguientes aplicaciones prácticas.

1. Trabajar en entornos de desarrollo en la nube.
2. Conocer los diferentes lenguajes de programación utilizados en el mundo académico, científico y comercial.
3. Conocer los algoritmos utilizados y técnicas de análisis.
4. Distinguir las diferentes librerías más populares en el mundo.

6.3. Lecciones aprendidas

La lecciones aprendidas en la elaboración del estudio son:

1. Meditar sobre los temas que me interesan y me motivan, la problemática y las soluciones.
2. Seleccionar y estructurar la información de acuerdo al objeto de estudio.

3. Relacionar la información de diferentes fuentes y disciplinas.

4. Enfocarme en el tema y afrontar la dispersión.

5. Organizar la información y combatir el síndrome de diógenes digital.

6. Limpiar el ruido y sintetizar la información.

6.4. Trabajos futuros

Para futuros trabajos se puede continuar con la línea de investigación sobre el balanceo de datos. Cómo se comportan los algoritmos con técnicas mixtas utilizando diferentes métodos y librerías, y aumentando los diversos temas a evaluar dentro del algoritmo de clasificación.

Apéndice A

Siglas

ASCII	American Standard Code for Information Interchange. 18.
BBDD	Bases de Datos.XI. 52.
BERT	Bidirectional Encoder Representations from Transformers.(Representación de Codificador Bidireccional de Transformadores). 13. 31.
BOW	Bag Of Words.(Bolsa de palabras). 16. 28. 39.
CBOW	Continius Bag Of Words.(Bolsa continua de palabras) 23.
CWI	Centrum Wiskunde and Informatic. 9.
DBSCAN	Density-based spatial clustering of applications with noise.(Agrupamiento espacial basado en densidad de aplicaciones con ruido). 14.
HTML	HyperText Markup Language (Lenguaje de Marcas de Hipertexto). 12.
LDA	Linear Discriminant Analysis.(Análsis discriminante lineal). 27. 28.
LDiA	Latent Dirichlet Allocation.(Asignación Latente de Dirichlet). 27.
LIWC	Linguistic Inquiry and Word Count.(Investigación lingüística y recuento de palabras). 16.
LSA	Latent Semantic Analysis.(Análisis semántico latente). 26. 27. 28. 31.
LSTM	Long short-term memory.(Memoria corta a largo plazo). 34.
MiB	Mebibyte. 41.
MIT	Massachusetts Institute of Technology. 13.
ML	Machine Learning.(Aprendizaje automático). 5.
NER	Named Entity Recognition.(Reconocimiento de entidad nombrada). 13.

NLP	Natural Lenguaje Processing.(Procesamiento natural del lenguaje). 5.
NLTK	Natural Lenguaje Tool Kit.13.
PCA	Principal Component Analysis.(Análisis de componentes principales). 27.
PCFG	Probabilistic context free grammars.(Gramáticas probabilísticas libres de contexto). 16.
PLN	Procesamiento de Lenguaje Natural.15. 22. 27.
POS	Part Of Speech.(Parte del habla). 16. 21. 33.
PYPL	PopularitY of Programming Language Index. (Índice de popularidad del lenguaje de programación). 10. 11.
ReLU	Rectified Linear Units. (Unidad lineal rectificada). 33.
RNN	Recurrent Neural Network.(Red neuronal recurrente). 33.
ROC	Receiver Operating Characteristic.(Característica Operativa del Receptor). 47. 55.
ROS	Random Over Sampling.(Sobremuestreo aleatorio). 37.
SMOTE	Synthetic Minority Over-sampling Technique.(Técnica de sobremuestreo minoritario sintético). 5. 7. 35.
SSD	Sum of Square Differences.(Suma de diferencias cuadradas). 25.
SVM	Support Vector Machines.(Máquinas de Vector Soporte). 5.
TF	Term Frecuency.(Frecuencia del término). 22.
TF-IDF	Inverse Document Frecuency- Term Frecuency.(Frecuencia inversa del documento-frecuencia del término) 22. 23. 25. 26. 27. 28

Referencias

[1] Hadeer Ahmed, Issa Traoré y Sherif Saad. «Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques». En: *Proceedings of the First International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments - ISDDC 2017, Vancouver, BC, Canada, October 26-28, 2017*. Ed. por Issa Traoré, Isaac Woungang y Ahmed Awad. Vol. 10618. Lecture Notes in Computer Science. Springer, 2017, págs. 127-138. DOI: 10.1007/978-3-319-69155-8_9.

[2] Marián Alonso González. «Fake News: desinformación en la era de la sociedad de la información». En: *Ámbitos. Revista Internacional de Comunicación,* 45 (2019), págs. 29-52. DOI: 10.12795/Ambitos.2019.i45.03.

[3] Jacob Eisenstein. *Introduction to Natural Language Processing*. Adaptive Computation and Machine Learning Series. MIT Press, 2019. ISBN: 978-0262042840.

[4] Haibo He y Yunqian Ma, eds. *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley y Sons, 2013. ISBN: 978-1118074626.

[5] Michiko Kakutani. *La muerte de la verdad: Notas sobre la falsedad en la era Trump*. Barcelona: Galaxia Gutenberg, 2019. ISBN: 978-8417747046.

[6] Hobson Lane, Cole Howard y Hannes Max Hapke. *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Manning Publications, 2019. ISBN: 978-1617294631.

[7] Javier Mayoral, Sonia Parratt y Monserrat Morata. «Desinformación, manipulación y credibilidad periodísticas: una perspectiva histórica». En: *Historia y comunicación social* 24.2 (2019), pág. 395.

[8] Denis McQuail y col. *Modelos para el estudio de la comunicación colectiva*. Ediciones Universidad de Navarra (EUNSA), 1997.

[9] Ankur A. Patel y Ajay Uppili Arasanipalai. *Applied Natural Language Processing in the Enterprise*. O'Reilly Media, Inc., 2021.

[10] Verónica Pérez-Rosas y col. «Automatic detection of fake news». En: *arXiv preprint arXiv:1708.07104* (2017).

[11] Delip Rao y Brian McMahan. *Natural language processing with PyTorch: build intelligent language applications using deep learning*. O'Reilly Media, Inc., 2019.

[12] Xinyi Zhou y Reza Zafarani. «A survey of fake news: Fundamental theories, detection methods, and opportunities». En: *ACM Computing Surveys (CSUR)* 53.5 (2020), págs. 1-40.