Trabajo de Fin de Grado

Análisis de sentimiento en redes sociales: Bahía del Confital

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Leonardo José Gonçalves Ponte

TUTORIZADO POR:
Javier Jesús Sánchez Medina

Mayo/2024

Agradecimientos

A mis padres, tutor y amigos por la constante motivación.

Resumen

Actualmente, las redes sociales son el medio por las que la gran mayoría de personas interactúan, ofreciéndonos una amplia cantidad de recursos para el estudio del análisis de sentimientos y la aplicación de sus metodologías con fin de entender el sentimiento general. En este trabajo se ha realizado un análisis exploratorio de los *tweets* relativos a la Bahía del Confital; después se han ensayado diversas metodologías de cálculo automático de sentimiento, de esos *tweets*, las primeras basadas en la combinación con diccionarios de sentimientos (lexicones) existente y las segundas, entrenando un *ensemble* de aprendizaje automático (*random forest*) en función de la frecuencia relativa de las palabras en los *tweets*. Se han probado y comparado dichas metodologías, incluyendo la aumentación de datos.

Abstract

Nowadays, social media is how a vast majority of people interact with each other, offering an ample number of resources for the field of sentiment analysis and the application of its methodologies with the goal of having a general understanding of public sentiment. In this project an exploratory sentiment analysis of the tweets related to the Bahía del Confital has been carried out; the former is based in the combination with existing sentiment lexicons, and the latter being based in the training of an automatic learning ensemble (random forest) in function of the relative frequency of the words in the tweets. These methodologies have been tested and compared, with inclusion of data augmentation.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

> "El internet se está convirtiendo en la plaza del pueblo de la aldea global del mañana"
>
> Bill Gates

Como sociedad, las redes sociales se han convertido en una parte integral de la experiencia humana, donde en el mundo de las redes sociales *X* o antiguamente conocida como *Twitter* ha sido una de las redes sociales más influyentes de la última década. Siendo *Twitter* a la vez una de las herramientas principales para la organización de distintas protestas y movimientos tanto económicos como sociales. Esto da lugar a interpretar por medios como el análisis de sentimientos de las intenciones y el estado emocional, no solo de un individuo, sino la de una comunidad entera con más facilidad debido a su formato de texto corto.

El objetivo de este trabajo realizar un análisis de sentimiento y desarrollar un modelo de análisis de sentimiento sobre unos datos descargados de *Twitter*, alusivos a la Bahía del Confital, que han sido etiquetados en virtud de un Convenio de Colaboración con el Ayuntamiento de Las Palmas de Gran Canaria.

Partiendo de un *dataset* etiquetado de *tweets*, se efectuarán varios análisis exploratorios primero, con *R* y *Python*. Luego con *R*, *Python* u otros lenguajes que convengan al caso, se elaborarán modelos predictivos que serán evaluados utilizando métricas actuales. Se utilizarán librerías para procesamiento de datos textuales, así como para la elaboración de visualizaciones y para el entrenamiento y evaluación de modelos basados en *machine learning*.

1.1. Plan de trabajo

En esta sección se expondrán las distintas fases planificadas 1.1 de este trabajo junto con las horas estimadas y las horas reales. La estimación de horas ha sido en acorde con las horas reales en todas las fases excepto en las tareas 1.1 "Estudio de métodos de análisis de

Fases	**Duración Estimada** (horas)	**Tareas** (nombre y descripción, obligatorio al menos una por fase)
Estudio previo / Análisis	100	Tarea 1.1: Estudio de métodos de análisis de sentimientos para palabras en un léxico castellano.
		Tarea 1.2: Estudio de métodos de análisis de sentimientos de tweets.
		Tarea 1.3: Recopilación de Tweets para la creación de un conjunto de datos para entrenamiento y validación.
		Tarea 1.4: Búsqueda de *lexicón* castellano, y adaptación al problema.
Diseño / Desarrollo / Implementación	80	Tarea 2.1: Análisis exploratorio del *dataset* de *twits* facilitado.
		Tarea 2.2: Implementación de diferentes modelos de análisis de sentimiento convencionales, para textos en castellano.
		Tarea 2.3: Implementación de nueva metodología para el modelado de análisis de sentimiento para este caso.
Evaluación / Validación / Prueba	60	Tarea 3.1: Evaluación de los modelos desarrollados.
		Tarea 3.2: Desarrollo de las conclusiones y líneas futuras en función de los resultados obtenidos
Documentación / Presentación	60	Tarea 4.1: Redacción de la memoria del TFT
		Tarea 4.2: Realización y ensayos de la presentación del TFT

Ilustración 1.1: Estimación de plan de trabajo

sentimiento para palabras en un léxico castellano" y tarea 2.2 "Implementación de diferentes modelos de análisis de sentimiento convencionales, para textos en castellano". En la tarea 1.1 se terminó extendiendo el tiempo de investigación dado a la poca cantidad de recursos que hay disponibles para el análisis de sentimiento en castellano, donde la mayoría de los *lexicons* disponibles son traducciones directas al inglés. Luego, en la tarea 1.2 fue debido a la cantidad de tiempo que se requisó en el entrenamiento del modelo.

A pesar de un ligero incremento en el tiempo previsto, se ha cumplido con los tiempos establecidos por la ULPGC de 300 horas.

1.2. Organización del documento

El resto de este Trabajo de fin de Titulo está integrado por los capítulos correspondientes al estado actual del análisis de sentimientos en redes sociales y los objetivos iniciales, el capítulo dedicado a las Competencias específicas y aportaciones del trabajo, el capítulo de normativa y legislación, el capítulo de desarrollo, y el capítulo de Conclusiones y trabajos futuros.

Capítulo 2

Estado actual y objetivos iniciales

2.1. El análisis de sentimientos

"El análisis de sentimientos, también llamado minería de opiniones, es el campo de estudio que analiza las opiniones, sentimientos, evaluaciones, valoraciones, actitudes y emociones hacia entidades como productos, servicios, organizaciones, individuos, asuntos, eventos, temas y sus atributos."[33]

"El análisis de sentimiento se ha investigado en distintos niveles, nivel del documento, nivel de oración, nivel de frase y nivel de aspecto."[51] Estos niveles de investigación 2.1 van progresivamente más a detalle. Empezando con el nivel del documento en el que el análisis de sentimiento sería de todo el documento, hasta el nivel de aspectos, donde el análisis toma en cuenta el contexto que aportan los otros textos en conjunto para establecer una polaridad.

"A pesar de que la lingüística y el procesamiento del lenguaje natural (PLN) tienen una larga historia, poca investigación se realizó sobre las opiniones y sentimientos de la gente antes del año 2000. Desde entonces, el campo se ha convertido en un área de investigación muy activo. Esto se debe a varias razones. En primer lugar, cubre un amplio abanico de aplicaciones, en casi todos los ámbitos. La industria acerca el análisis de sentimientos también ha florecido debido a la proliferación de aplicaciones comerciales. Esto supone una fuerte motivación para la investigación. En segundo lugar, ofrece muchos problemas que nunca se habían estudiado antes."[33] El análisis de sentimiento es un proceso del análisis de texto, que, gracias a la evolución tecnológica en las aplicaciones comerciales y la cantidad de datos disponibles han aumentado a niveles inimaginables, permitiendo una amplia fuente de recursos para ello. A parte de ello, la reducción del coste en obtener dicha cantidad de opiniones se ha reducido a la milésima de su coste, dado a que si antes uno quería procurar de un *dataset* grande necesitaría organizar un censo en el que implicaría una gran labor. Con esta accesibilidad que se ofrece, el interés en este campo de estudio ha crecido notablemente.

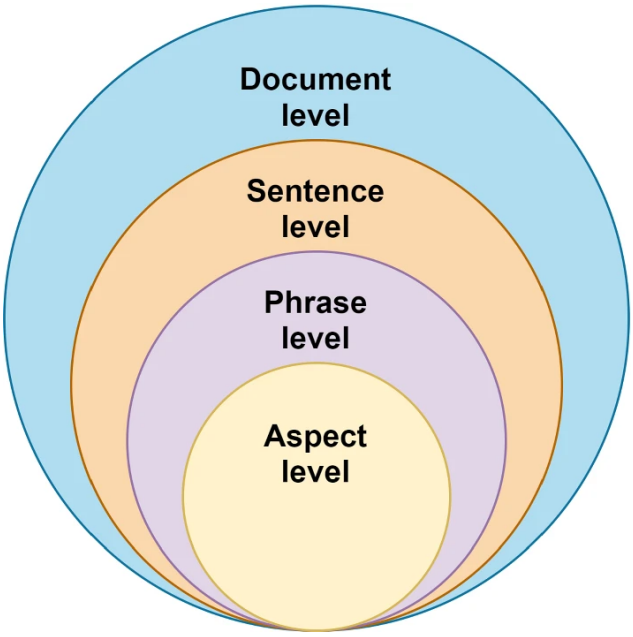

Ilustración 2.1: Level of sentiment analysis [51]

2.2. Interés en área de estudio

En España por ejemplo existen grupos como la Sociedad Española para el Procesamiento del Lenguaje Natural o **SEPLN**, que organizan talleres como el Taller de Análisis Semántico en la **SEPLN** o **TASS**. Donde, .[El] principal objetivo es promover el diseño de nuevas técnicas y algoritmos y la aplicación de los ya existentes para la implementación de complejos sistemas capaces de realizar un análisis de sentimientos basados en opiniones de textos cortos extraídos de medios sociales concretamente Twitter)"[53]. El **TASS** suele ofrecer unas series de "tareas" en las que están "Abierto a quien quiera participar: grupos de investigación académicos y de la industria."[53] en las que ofrece un *dataset* común de la que pueden trabajar. Estas sociedades demuestran un interés en cuanto a la resolución de una serie de desafíos. Siendo las distintas maneras en las que uno puede afrontar entre ellas:

- ✔ El preprocesamiento de los datos.
- ✔ El consenso en clasificación de sentimiento y dificultad de analizar un texto corto.
- ✔ La tendencia del bajo rendimiento promedio de los modelos.

"La ubicuidad de los medios sociales y el uso cada vez más extendido de mensajes de texto cortos se convierten en una potencial de extracción de sabiduría colectiva, especialmente en cuanto al sentimiento. Por lo tanto, la clasificación y el análisis de los sentimientos es una tarea importante en el ámbito de la investigación actual. El principal reto en este campo es controlar los datos en términos de ruido, relevancia, emoticonos, folcsonomías y jergas."[44] En el caso del artículo anterior, por ejemplo, su enfoque es el efecto del preprocesamiento donde se presta atención en las jergas usadas.

2.3. *Tweets* como medio de entrenamiento de modelos inteligentes

La red Social X o anteriormente conocida como *Twitter* ha sido y sigue siendo una de las redes sociales más influyentes de la web 2.0. Antes de ella, la web 1.0 estaba limitada a dinámicas de *host* y *consumidor* dado a la naturaleza estática de las páginas web en su momento. Con la web 2.0, las redes sociales como *Twitter* establecieron una dinámica donde el usuario ahora no solo es un consumidor sino un contribuyente junto con todos los otros usuarios. Actualmente *Twitter* ronda los 550 millones de usuarios con una media de 100 a 200 millones de *tweets* al día([28]). Este nivel de tráfico y su impacto social no puede ser ignorado, siendo ello una herramienta clave para la organización de múltiples movimientos sociales y promulgador de ideas.

"Twitter se ha convertido en una de las principales plataformas de redes sociales y ha suscitado un gran interés entre los investigadores del análisis de sentimientos. A pesar de esta atención, los enfoques de análisis de sentimientos de Twitter más avanzados tienen un rendimiento relativamente pobre, con precisiones de clasificación a menudo por debajo del 70 %, lo que afecta negativamente a las aplicaciones de la información de sentimientos obtenida." [54] Por esa misma dificultad que conlleva los mensajes cortos y la subjetividad que acompaña la clasificación se han visto rendimientos relativamente bajos en cuanto a las precisiones.

Además, la clasificación manual de un texto puede causar discrepancias debido a lo subjetivo que puede ser el sentimiento percibido. A parte de esto la gran dificultad que conlleva el análisis de sentimientos, viene dado a lo corto que son los *tweets*, ya que su tamaño máximo actual es de 280 caracteres por *tweet* (desde el 2017) y además de los *tweets* más antiguos que estaban limitados a 140 caracteres ([53]).

2.4. Emojis

Los *Emojis* tienen el potencial de funcionar como ancla a la intención de un *tweet*, aportando un contexto que anteriormente hubiera requerido múltiples palabras. Para ello se ha pillado un *lexicon* de *emojis* en los que podemos traducir los *emojis* a texto y con ello podremos identificar su polaridad. Una dificultad que se ha visto en cuanto a los *emojis*, es la diferencia en su formato dependiendo de la plataforma en la que se usa. La imagen de una cara feliz en *Twitter* puede ser distinta a una usada en *Facebook*. y en otros casos puede que ese *emoji* ni esté disponible en alguna plataforma.

2.5. *Stemming & Lemming*

"El *stemming* consiste en quitar y reemplazar sufijos de la raíz de la palabra. La lematización es un poco más compleja e implica hacer un análisis del vocabulario y su morfología para retornar la forma básica de la palabra (sin conjugar, en singular, etc.)."[36]. Luego tenemos la lematización que sería el proceso en extraer la "Palabra que encabeza un artículo de un diccionario o de una enciclopedia."[11]

El *stemming* y *lemming* permiten reducir el número de palabras distintas en una colección de textos, en casos como el entrenamiento de modelos inteligentes esto ofrece una reducción de términos proporcionados en la fase de entrenamiento ofreciendo una mejora de rendimiento en cuanto al tiempo de procesados. La posible consecuencia negativa de estas implementaciones es la posible pérdida de información, por ejemplo, uno de los efectos más vistos es la eliminación de géneros y pluralidad en las palabras.

2.6. *Lexicons* utilizados en el procesamiento de lenguaje natural

2.6.1. Emoji Sentiment Lexicon

El *lexicon* de *emojis* que se usará, es el **Emoji Sentiment Ranking 1.0**. Es un *lexicon* compuesto de 751 *emojis* con un sentimiento asignado automáticamente. El sentimiento fue computado de 70,000 *tweets*, etiquetados por 83 anotadores humanos en 13 lenguas europeas([29]). Luego, para la asignación del **Sentiment score** y la polaridad de cada *emoji* se calculó en base de su apariencia en *tweets* positivos y negativos. Esto facilitará el uso de este *lexicon* de *emojis* más adelante.

2.6.2. Stopwords en castellano

"Para un análisis de texto adecuado, es necesario despojar al texto de tokens "no informativos". Palabras como (en español): de, la, que no son muy informativas para las tareas comunes."[27] Para la lista de *stopwords* se ha usado la lista disponible en el repositorio **stopwords-es**([10]) y el propio del paquete de **tm**. Es importante tomar en cuenta que, para evitar el descarte de las locuciones, el descarte de las *stopwords* se debe aplicar después de aplicar el *lexicon*.

2.6.3. NRC

Luego podemos mirar el *lexicon* **NRC**, este *lexicon*, cuyas 14.182 palabras se asocian a 8 emociones básicas: ira, miedo, anticipación, confianza, sorpresa, tristeza, alegría y disgusto

([35]). Este *lexicon* también que es de origen inglés dispone de traducciones de obra comunitaria al castellano. La dificultad que trajo este *lexicon* fue el hecho de que, debido a su asociación a 8 emociones, resulta difícil clasificar cuales son adaptaciones adecuadas a un modelo de 3 estados, positivo, negativo y neutral.

2.6.4. AFINN

El *lexicon* **Afinn** dispone de 2.477 palabras en las que también incluyen unas 15 frases. Este *lexicon* tiene asignado las palabras con un puntaje en el rango de -5 a 5, siendo el numero negativo sentimientos desfavorables y el numero positivo como sentimientos favorables. Este *lexicon* tiene el inglés como idioma base teniendo disponible, por traducciones supervisadas una versión en español. Un problema con el que se encontró en aplicar este *lexicon* fue de que dado a que la cantidad de palabras disponible son unas 2477 se vio que el tamaño de los *tweets* filtrados terminaba teniendo una longitud notablemente reducida llegando a terminar con solo una palabra y en otros casos ninguna palabra restante en *tweet*.

2.6.5. ML-Senticon

Finalmente, el *lexicon* **ML-Senticon**. Este dispone de unas 12.078 siendo 3.080 de ellas locuciones, esto trae una gran ventaja ya que normalmente se suelen confundir por *stopwords* resultando en su descarte. Este *lexicon* está segmentado en palabras positivas y negativas.

"Es un conjunto de lexicones de polaridades semánticas a nivel de lemas para inglés, español, catalán, gallego y euskera. Estos lexicones están estructurados en capas, lo que permite seleccionar distintos compromisos entre la cantidad de estimaciones de positividad y negatividad y la precisión de dichas estimaciones. Los *lexicons* se han generado automáticamente a partir de una mejora del método utilizado para generar **SentiWordNet**, un recurso ampliamente utilizado que recoge estimaciones de positividad y negatividad a nivel de synsets."[8]

2.7. El caso con los *lexicons* traducidos del inglés

Como es común en la gran mayoría de las disciplinas, el uso del inglés se figura en un gran porcentaje de los recursos disponibles. En el caso de los lexicones de polaridad de palabras, se observa que la mayoría de ellas que disponen de un *lexicon* en español, son realmente traducciones directas del inglés. Esto puede impactar el análisis de sentimiento por el hecho de que la mayoría de las palabras no comparten la misma polaridad en distintos idiomas.

2.8. Objetivos del trabajo

En este trabajo se establecen como objetivos, el análisis exploratorio de los datos, el desarrollo de metodologías diversas para el análisis de sentimiento automático, para este caso, partiendo de un *dataset* de *tweets* etiquetado y una evaluación de las metodologías, para la selección de mejor de ellas. Estos objetivos ofrecen distintos puntos de análisis para el análisis de sentimiento de un texto.

2.8.1. Análisis exploratorio de los datos

Se realiza una investigación estadística de análisis de sentimientos en textos tipo *tweets* para poder evaluar y comprender los datos. Con objetivo de explorar que tipo de tendencias se pueden discernir de los datos etiquetados para tener una idea clara de que comparten en común según el sentimiento extraído.

2.8.2. Desarrollo de metodologías diversas para el análisis de sentimiento automático, para este caso, partiendo de un *dataset* de *tweets* etiquetado

Se aplicará una serie de filtros en la fase del preprocesado de datos en el que se evaluará cuál de ellos es el óptimo para el entrenamiento de un modelo de *machine learning*. Con un fuerte enfoque en el impacto de los *lexicons* y otras herramientas de enfatización según la herramienta utilizada.

2.8.3. Evaluación de las metodologías, para la selección del mejor

Se evaluará la precisión de los distintos modelos generados. Con ellos podremos ver que modelo de *machine learning* muestra ser el óptimo para el análisis de sentimientos de *tweets*. Además, se observará si por medio de una ampliación de datos habrá un efecto positivo o negativo a la precisión de dichos modelos.

Capítulo 3

Competencias específicas y aportaciones del trabajo

3.1. Competencias específicas

3.1.1. CII015

"Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica."

Para realizar un análisis de sentimiento sobre una colección de frases cortas como son los *Tweets* hay que tener la capacidad de reconocer cuales serían los algoritmos de aprendizaje más adecuados como el **Random Forest** y **Naive Bayes** entre otros. También es fundamental para un análisis de sentimientos tener un conocimiento amplio en cuanto al *Procesamiento de lenguaje natural* entre otras áreas de la inteligencia artificial.

3.1.2. CP04

"Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación."

Esta competencia requiere que uno sea capaz de no solo teorizar sobre los conocimientos fundamentales de los sistemas inteligentes sino también ser capaz de analizar y aplicar dichos conocimientos en la creación de un modelo que sea capaz de tener niveles de predicción realistas.

	Free	Basic	Pro	Enterprise
Getting access	Get Started	Get Started	Get Started	Get Started
Price	Free	$100/month	$5000/month	
Access to X API v2	✓ (Only Post creation)	✓	✓	
Access to standard v1.1	✓ (Only Media Upload, Help, Rate Limit, and Login with X)	✓ (Only Media Upload, Help, Rate Limit, and Login with X)	✓ (Only Media Upload, Help, Rate Limit, and Login with X)	
Project limits	1 Project	1 Project	1 Project	
App limits	1 App per Project	2 Apps per Project	3 Apps per Project	
Post caps - Post	1,500	3,000	300,000	
Post caps - Pull	✗	10,000	1,000,000	
Filtered stream API	✗	✗	✓	
Access to full-archive search	✗	✗	✓	
Access to Ads API	✓	✓	✓	

Ilustración 3.1: X niveles de acceso API y versiones [4]

3.2. Aportaciones

3.2.1. Entorno técnico y científico

Dado a la naturaleza de los *tweets* este trabajo principalmente puede aportar en el análisis de sentimientos de textos cortos, demostrando los efectos que pueden tener el preprocesado de datos al presentarlos a distintos modelos. Enfocándonos en las diferencias que pueden ofrecer los *lexicons* **NRC**, **AFINN** y **ML-SENTICON** en el análisis de sentimiento. A parte de ello también se mostrará la diferencia que aporta la inclusión de *emojis* en el análisis.

Actualmente, *twitter* ofrece acceso a su *API* mediante distintos niveles de pago3.1. Esto puede limitar el acceso a las instituciones académicas que no disponen de los fondos necesarios para el servicio. Debido a ello se llevó a cabo la Ley de Servicios Digitales que permite a "La investigación cualificada según el artículo 40 de la Ley de Servicios Digitales puede solicitar el acceso a la API X. Nota: esta solicitud es sólo para un subconjunto limitado de investigación de la UE relacionada con la DSA. Para la investigación académica en general, inscríbase en uno de nuestros niveles de acceso a la API de X."[5] A pesar de ello hay declaraciones por múltiples investigadores de que *twitter* está denegando sus solicitudes para acceder a la API [45]. Los datos que se han obtenido ofrecen un gran valor por causa de la escasez de datos que han generado estos cambios.

3.3. Aspectos económicos

Debido a la gran variedad de posibilidades que ofrece el análisis de sentimientos, las organizaciones gubernamentales y empresas están activamente dedicando recursos en la minería de opiniones([38]). En cuanto a las entidades empresariales, ellas tienen una idea clara de cómo se inclina el sentimiento general de los comentarios hechos en las redes sociales sobre sus productos o la misma empresa en sí. Un ejemplo de comentarios cortos, son las reseñas de productos, aparte de obtener la polaridad general de las reseñas, se puede luego extraer cuál de las palabras fue más usada según la polaridad dada.

En los aspectos políticos existe la posibilidad de tener una idea de la opinión pública sobre ciertas decisiones que han implementado el gobierno, sean legislativas, propuestas de regulación, asignación de fondos públicos o respuestas a problemas sociales. Por ejemplo, en este trabajo usamos datos compuestos de *tweets* publicados durante la pandemia del COVID-19. Con esto se puede contemplar el nivel de apoyo no solo del público, sino también existe la posibilidad se ver la opinión de los militantes de partidos específicos.

Capítulo 4

Normativa y legislación

4.1. La ley de inteligencia artificial de la unión europea

"En el parlamento europeo se aprobó la **Ley de Inteligencia Artificial** con objetivo de establecer una series de obligaciones para la IA en función de sus riesgos potenciales y su nivel de impacto"([12]). Estas leyes afectarán en mayor parte a las empresas debido a los requisitos de transparencia de datos. Reforzando a la vez los derechos que disfrutan los ciudadanos europeos.

Estas normas no afectarán notablemente al análisis de sentimiento en cuanto a las prohibiciones aplicadas dado a que se enfocan más en sistemas de categorización de características personales de los individuos. En cuanto a las posibles obligaciones, estará limitado a sistemas de "alto riesgo", en las que se requiere transparencia para la toma de la decisión del sistema de IA en cuanto a una percibida infracción de los derechos. Además, deberán "respetar la legislación de la UE sobre derechos de autor y publicar resúmenes detallados del contenido usado para entrenar sus modelos"[13][12]. Para este trabajo ya tenemos claramente detallado el origen de los datos que se usarán en el entrenamiento de los modelos creados. Se pedirá incluso en casos de modelos más potentes "realizar evaluaciones de los modelos, analizar y mitigar los riesgos sistémicos e informar sobre los incidentes"[12]. Se espera que esta ley sea la primera de una larga serie de controles hacia la aplicación de estas tecnologías.[13]

4.2. Ley de protección de datos

"Su objetivo es el de establecer un marco de garantías que haga posible el ejercicio de los derechos fundamentales y las libertades del ser humano y con el fin de impedir que el uso de la información personal pueda utilizarse indiscriminadamente en contra de dichos derechos y libertades inherentes al ser humano."[42] En el caso de las redes sociales la Ley de Protección de datos se enfoca en la protección de la información personal de un usuario. En el caso de este TFG, los datos que se han usado no contienen datos personales de los usuarios dado

que solo se usan los *tweets*, y debido a la política de privacidad de *twitter*, todo usuario de *twitter* tiene la libertad de elegir si el contenido que publica es destinado al público. Además, que en el preprocesado de datos se omite todo menos el contenido del *tweet*, la clasificación proporcionada, y se elimina referencias a otros usuarios[2][32].

Capítulo 5

Herramientas utilizadas

5.1. Herramientas de entorno y control de versión

RStudio

RStudio[41] es un entorno de desarrollo integrado orientado para el lenguaje de programación R. A pesar de ser orientado para R, gracias al paquete **reticulate** en este trabajo tenemos la capacidad de ejecutar código en *Python*.

Visual Studio Code

VS Code[34] es un editor de texto desarrollado por *Microsoft*, que por medio de un amplio catálogo de *plugins*, permite al usuario personalizar su propio entorno de desarrollo. En este trabajo se usó para el código escrito en *Python*.

Overleaf

Overleaf[37] es un editor LaTeX colaborativo basado en la nube que se utiliza para escribir, editar y publicar documentos[3]. La motivación al elegir Overleaf fue dado a su facilidad de uso y capacidad de mostrar en tiempo real el estado actual del documento. También dispone de una gran variedad de *Templates* que ofrecen estructuras estandarizadas por instituciones para orientar en cuanto a la estructura deseada de un trabajo.

GitHub

GitHub[19] es una plataforma que permite por medio del sistema de control de versiones Git, el fácil desarrollo, mantenimiento y divulgación de proyectos. GitHub ha sido una de las

plataformas más utilizadas por la gran mayoría de desarrolladores por todo el mundo. En el caso de este trabajo se ha usado el *plugin* interno de RStudio para la subida y resolución de conflictos en el código.

5.2. Lenguajes de programación y Frameworks

5.2.1. Lenguajes

R

R[15] Es un lenguaje y entorno *Open Source* orientado a la computación y generación de gráficas estadísticas([16]). R a pesar de ser más orientado a la manipulación de datos numéricos, dispone de una amplia cantidad de paquetes que nos permite poder trabajar con textos.

Python

Python[14] es un lenguaje de programación multidisciplinario cuya prevalencia en el desarrollo de modelos de inteligencia artificial ha creado una de las comunidades más grandes en el campo de estudio.

5.2.2. Bibliotecas y frameworks

TidyVerse/TidyModels

El tidyverse es una colección de paquetes de R con opiniones diseñada para la ciencia de datos. Todos los paquetes comparten una filosofía de diseño subyacente, gramática y estructuras de datos.([48])

Caret

"El paquete caret (abreviatura de Classification And REgression Training) es un conjunto de funciones que intentan agilizar el proceso de creación de modelos predictivos."[31]

Scikit-learn

Scikit-learn[39] es un módulo de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas supervisados y no supervisados

de escala mediana. Este paquete se centra en llevar el aprendizaje automático a no especialistas mediante un lenguaje de alto nivel de propósito general. Se hace hincapié en la facilidad de uso, el rendimiento, la documentación y la consistencia de la API.([40])

cuML

"cuML es un conjunto de algoritmos de aprendizaje automático rápidos y acelerados en la GPU diseñados para tareas analíticas y de ciencia de datos."[6] Gracias a este conjunto creado por el equipo de **RAPIDS**, se ha podido entrenar los modelos en menos tiempo.

5.3. Hardware

Para la fase de entrenamiento de modelos se utilizó un ordenador de 32GB de RAM con una CPU **AMD Ryzen 5 5600X 6 Core Processor** 4.50GHz, y una GPU GeForce RTX 3070ti.

Capítulo 6

Desarrollo

El desarrollo de este trabajo ha pasado por una larga cadena de estudio previo y análisis por cada fase del trabajo. En esta sección se dispondrá de múltiples tablas e ilustraciones del análisis de sentimiento y fragmentos de código principalmente enfocados en la fase de preprocesado de los datos y construcción del modelo. Todo el código utilizado estará disponible adjunto a la memoria.

6.1. Metodología

En este trabajo nos enfocaremos en la metodología CRISP-DM. En la que partiendo de un *dataset* etiquetado de *tweets* se efectuarán varios análisis exploratorios con R y Python.

"CRISP-DM, que son las siglas de *Cross-Industry Standard Process for Data Mining*, es un método probado para orientar sus trabajos de minería de datos."[22] Esta metodología se categoriza en 6 fases:

1. Comprensión comercial
2. Comprensión de los datos
3. Preparación de los datos
4. Modelado
5. Evaluación
6. Despliegue

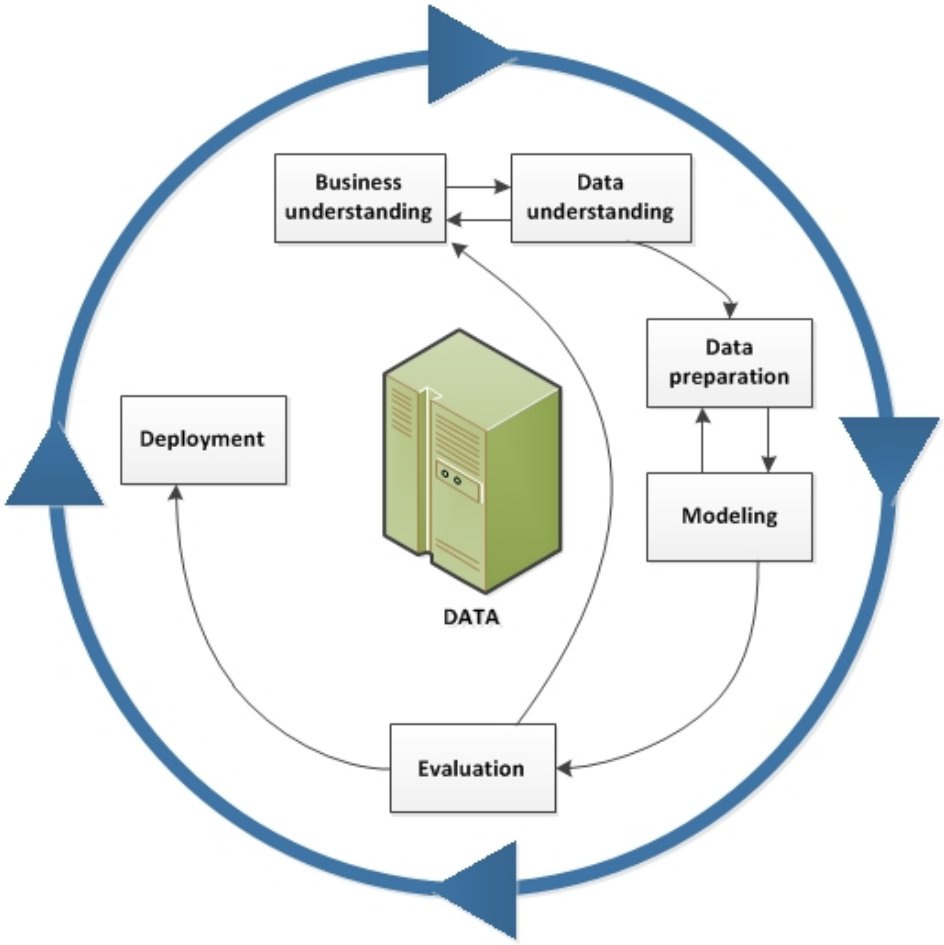

Ilustración 6.1: The data mining life cycle [22]

"El ciclo vital del modelo contiene seis fases con flechas que indican las dependencias más importantes y frecuentes entre fases. La secuencia de las fases no es estricta. De hecho, la mayoría de los proyectos avanzan y retroceden entre fases si es necesario."[22] En este trabajo se ha observado que al aplicar esta metodología, el transcurso de avance y retroceso entre fases se aplica de manera natural, donde se manifiesta más como una observación del proceso cíclico que una estructura rígida a seguir.

6.2. Comprensión comercial

En esta fase, su enfoque es "la determinación de los objetivos comerciales, evaluación de la situación, determinación de los objetivos de minería de datos y producción de un plan de proyecto"[23].

6.2.1. Determinación de los objetivos comerciales

Ya que esto es un trabajo orientado al ambiente académico, en vez de determinar los objetivos comerciales, se basará en identificar la pregunta que se quiere responder en este trabajo. En este caso, el objetivo es desarrollar un modelo de análisis de sentimiento sobre unos datos descargados de *Twitter*, alusivos a la Bahía del Confital, que han sido etiquetados en virtud de un Convenio de Colaboración con el Ayuntamiento de Las Palmas de Gran Canaria.

6.2.2. Evaluación de la situación

Esta fase se enfoca en la investigación de todo lo que tenga relación con el objetivo. Para este trabajo, en el capítulo de estado actual, uno puede visualizar el contexto en el que se encuentra el trabajo. En dicho capitulo se estudió el análisis de sentimiento como medio exploratorio de un *dataset*, el área de estudio y las organizaciones que uno puede hacer referencia. Se investigó también el rol que pueden formar los *tweets* en el entrenamiento de modelos inteligentes, los *emojis* y su potencia como variable clave en identificar la polaridad de un *tweet*, lo que puede aportar la aplicación del *stemming* y *lemming* a la identificación y reducción de términos, los distintos tipos de *lexicons* explorados y sus características y finalmente el caso de los *lexicons* que proviene su vocabulario de una traducción directa del inglés.

6.2.3. Determinación de los objetivos de minería de datos

En esta fase, en base de los objetivos establecidos anteriormente se usan de referencia de cuales son nuestros objetivos en cuanto a la minería de datos. En el caso de este trabajo el enfoque de minería de datos se basa en la obtención de *tweets* que comparten palabras claves en contexto de la Bahía del Confital y la playa de las canteras.

6.2.4. Producción de un plan de proyecto

Se formula un plan de proyecto para abarcar el flujo de pasos que se deberán tomar para cumplir con los objetivos establecidos. En este caso, el trabajo siguió un proceso evolutivo basado en una constante experimentación de distintas herramientas y bibliotecas de desarrollo. Manteniendo en perspectiva el ciclo de vida de la minería de datos, hasta llegar a un punto en el que se encontrase el trayecto más adecuado.

6.3. Comprensión de los datos

En esta fase, su enfoque es "la recopilación de datos iniciales, descripción de los datos, exploración de datos y verificación de calidad de datos"[23].

6.3.1. Recopilación de datos iniciales

En nuestro caso ya tenemos nuestros datos gracias a la colaboración con el **CICEI**. De ello se ha extraído la columna de **"text"** y **LABEL**.

En esta investigación tenemos disponible un *dataset* de *tweets* publicados durante la pandemia, originalmente se intentó filtrar por ubicación, pero dado a que pocos disponían de esa información se decidió descartarlas. Lo útil de este *dataset* en comparación a una simple extracción de *tweets* mediante la API de *Twitter* es que estos datos ya han sido etiquetados manualmente por humanos permitiéndonos un entrenamiento supervisado.

Para poder trabajar con estos datos será necesario filtrar un poco el *dataset* original. Primero vamos a pillar solo las columnas de texto y *LABEL* dado a que el nombre de usuario, aunque posiblemente útil para un análisis de sentimientos ya que se pudiera tomar en cuenta la costumbre del usuario que publica estos *tweets*. Por ejemplo, si un usuario suele publicar exclusivamente mensajes negativos podemos tomar eso como referencia al evaluar su *tweet*. Teniendo los tweets en la columna de texto podemos mirar que debido a la naturaleza y funcionalidad de Twitter los textos van a tener un cierto nivel de estructura.

Un *tweet* está compuesto de 280 caracteres como máximo. En la composición de un *tweet* es bastante común el uso de *hashtags*, los *hashtags* en Twitter asumen un rol bastante importante en la organización, descubrimiento y la participación en las conversaciones en Twitter. La utilidad de los *hashtags* es similar a los que pudiera proporcionar el nombre de usuario, pero en un ambiente más amplio que puede conectar un sentimiento colectivo sobre el tema remarcado en el *hashtag*.

6.3.2. Origen de los datos

En el marco de la colaboración con la concejalía de Ciudad de Mar, del Ayto. de Las Palmas de G.C. el "Desarrollo de acciones y estudios en el litoral Bahía del Confital-Las Canteras" durante el año 2020, se está desarrollando un modelo de detección automática de sentimiento en redes sociales. Para ello, se necesitó una base amplia de textos vertidos en redes sociales etiquetados, es decir, con valores de polaridad (positiva o negativa) asignada. Como fruto de esta colaboración se han etiquetado casi 64.000 *tweets* los cuales una vez filtrados se quedan en casi 20.000, con lo cual poder avanzar en la construcción de dicho modelo.

Ambas tarea, etiquetado y modelado, se le encargaron al **CICEI**, miembro del Instituto de Cibernética, Empresa y Sociedad de la Universidad de Las Palmas de Gran Canaria.

Los datos fueron *tweets* obtenidos entre el 27 de febrero del 2020 y el 30 de junio de 2022. Estos datos se han descargado utilizando palabras claves relacionadas con la bahía el confital y con un posterior filtrado de *tweets* no relacionados. En el etiquetado se etiquetaron con categoría 0 por no estar relacionados con el caso del estudio y esos *tweets* fueron descartados del análisis.

6.3.3. Objetivos del convenio "Desarrollo de acciones y estudios en el litoral Bahía del Confital-Las Canteras" con el que se relaciona

- ✔ **Fijar mecanismos para incentivar, entre los y las estudiantes de Grado y Posgrado e investigadores de la ULPGC:** La realización de trabajos aplicados de investigación en el área de la bahía del Confital, relacionados especialmente con las temáticas señaladas en las líneas de trabajo descritas en esta cláusula 2 del convenio.
- ✔ **Facilitar el desarrollo de nuevos trabajos aplicados de investigación:** Que estén directamente relacionados con las necesidades de información y gestión del Ayuntamiento de Las Palmas de Gran Canaria, y de las demás administraciones públicas que intervienen en esta Zona de Especial Conservación, así como con el desarrollo de herramientas y modelos centrados en la mejora de su sostenibilidad.
- ✔ **Fomentar la orientación de recursos universitarios existentes:** Hacia la formación académica de estudiantes e investigadores de la ULPGC en el ámbito de la ZEC del Confital.
- ✔ **Promover la producción científica** Aplicada en el ámbito de la ZEC, que aúne intereses y recursos, y que procure el desarrollo de información de relevancia para la mejora en la gestión de la sostenibilidad del espacio.

6.3.4. Etiquetas

En los datos obtenidos disponemos de 3 distintas etiquetas denominadas **LABEL** que están declaradas con los números 1, 2 y 3 representando positivo, negativo, y neutro respectivamente; Éstas etiquetas se usarán como clases predictoras en el entrenamientos de modelos y en la evaluación de la exploración de los datos.

6.4. Análisis de sentimiento exploratorio

Empezamos a mirar el *dataset*, extrayendo solo las columnas que nos resultan relevantes. El ID por utilidad de agrupación, el texto en su estado original, y el **LABEL** siendo esta columna la clasificación del texto que se realizó manualmente. En este primer análisis no vamos a descartar ningún *hashtag* ni cualquier referencia a otros usuarios y mantendremos

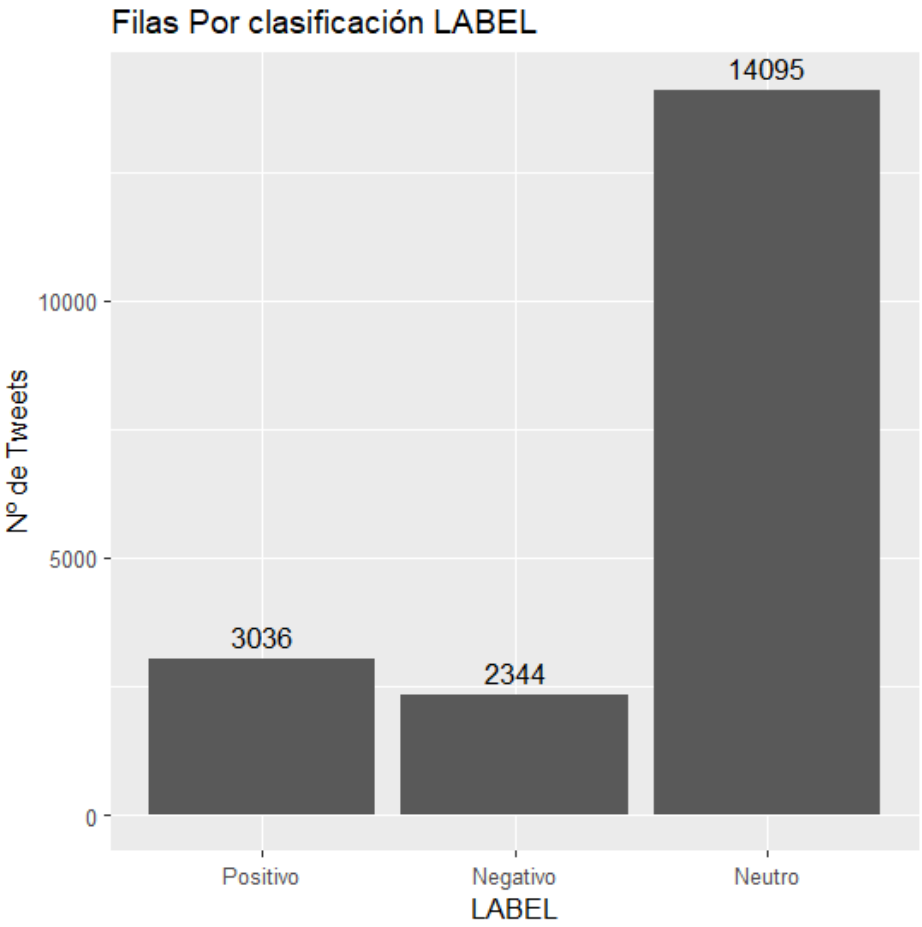

Ilustración 6.2: Distribución por Clasificación

cualquier posible locución. En este *dataset* como podemos ver en la figura 6.2 nos muestra que en la distribución de *tweets* hay una inclinación notable hacia los *tweets* neutros.

En esta primera extracción de las palabras más usadas, por ejemplo en los *tweets* positivos 6.3 las palabras [de, las, en, la, las, y a, que, el] son los que podemos considerar *stopwords*. Así que se necesitará filtrar el *dataset* por *stopwords* para tener una representación menos ruidosa de los datos. Además, dado a que como proceso de la obtención de datos se tuvo que filtrar por palabras como "Canteras" "Confital", las cuales vamos a añadirlas a la lista de *stopwords*. De esta manera tendremos palabras de contenido que nos podrá dar una idea más clara de sentimiento general de cada clasificación de los *tweets*.

En las figuras 6.3 podemos observar que hay muchas palabras en común entre todas las polaridades. Con esto podemos inferir que son palabras relevantes según el filtro que se utilizó en la obtención de datos. Se pudiera mirar las palabras que únicamente salen en cada polaridad pero se puede ver palabras como "gente" que salen con mayor frecuencia en *tweets* negativos que en *tweets* neutros o positivos. Con el contexto que ofrecen las palabras más frecuentes se puede interpretar que los *tweets* con polaridad positiva hablan de temas haciendo referencia a la playa como "bonito", "precioso" y "maravilloso". En *tweets* negativos

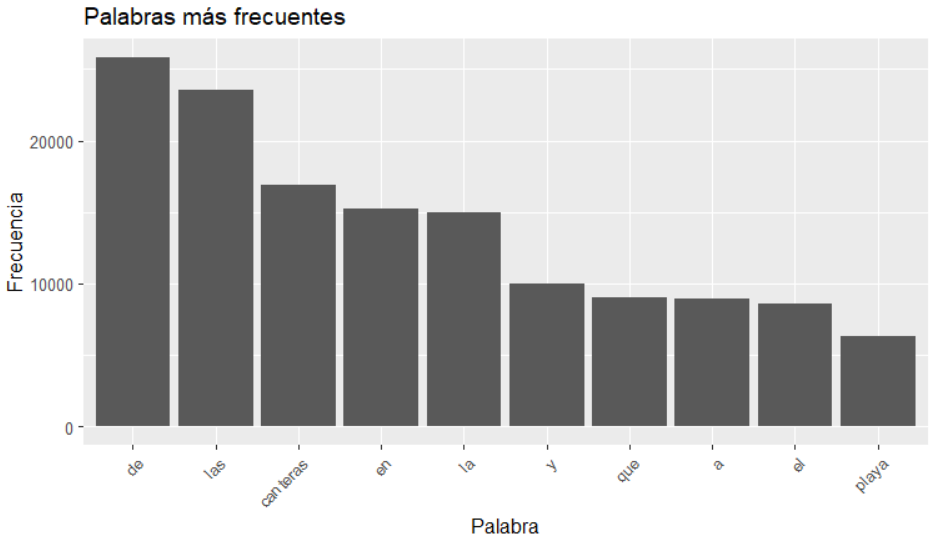

Ilustración 6.3: Palabras más frecuentes sin descartar *stopwords*

se habla con más frecuencia de la pandemia y las mascarillas. En *tweets* neutros se habla de temas más informativos de la ciudad.

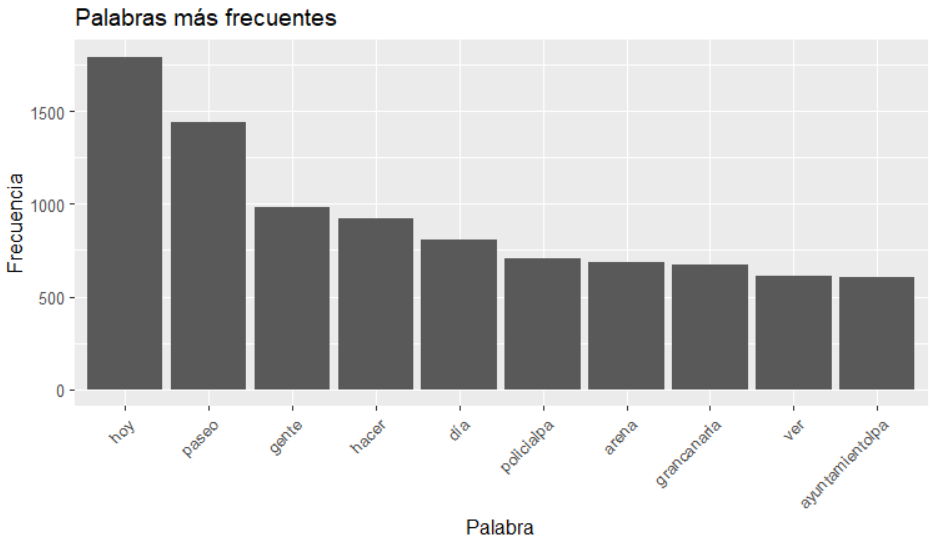

Ilustración 6.4: Palabras más frecuentes descartando *stopwords*

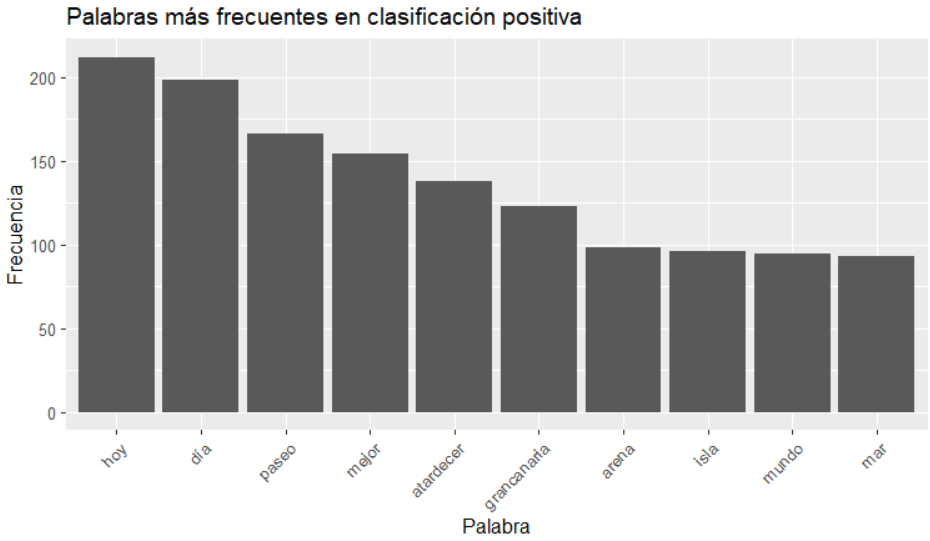

Ilustración 6.5: Palabras más frecuentes en clasificación positiva descartando *stopwords*

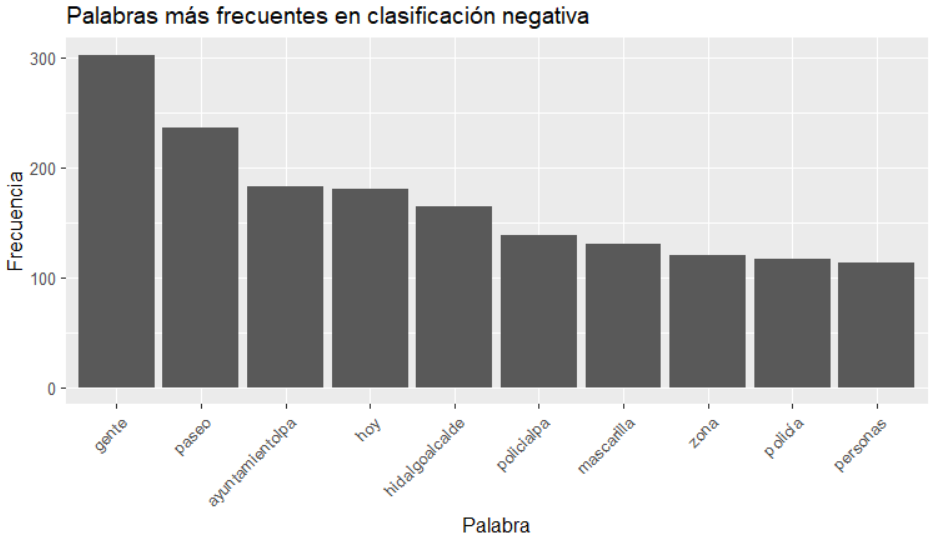

Ilustración 6.6: Palabras más frecuentes en clasificación negativa descartando *stopwords*

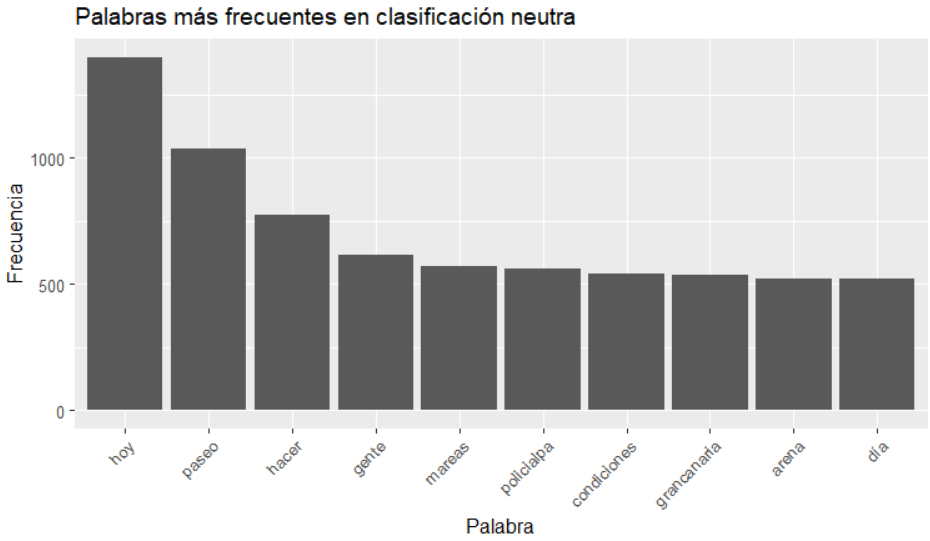

Ilustración 6.7: Palabras más frecuentes en clasificación neutra descartando *stopwords*

6.5. Modelado de sentimiento haciendo uso de diccionarios

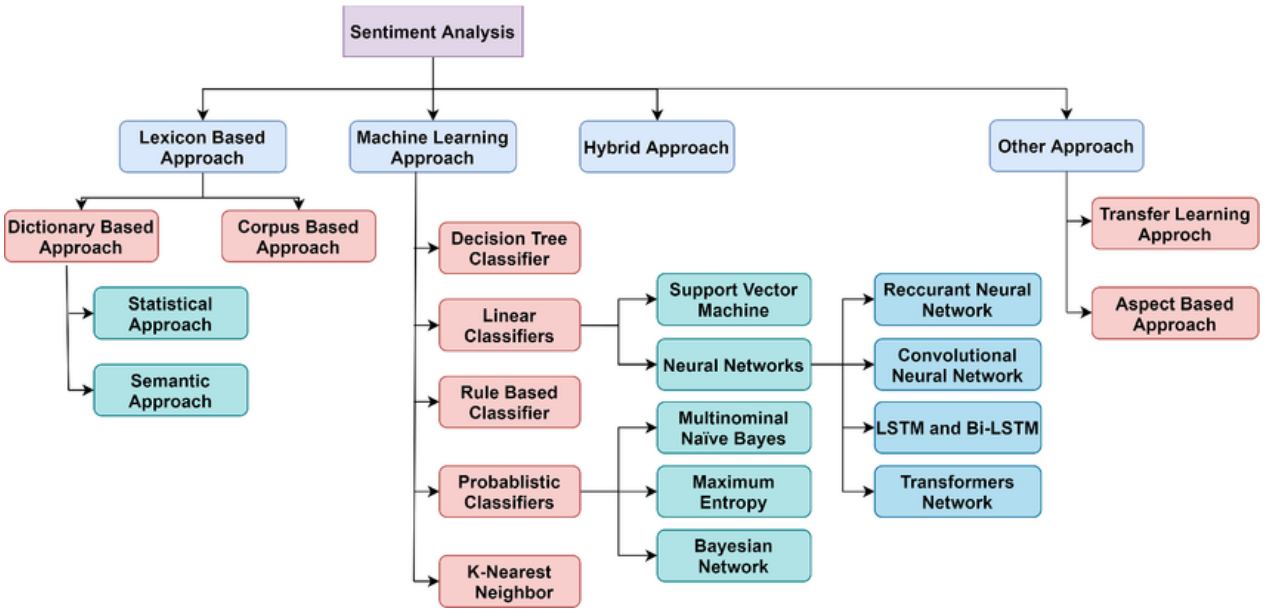

Ilustración 6.8: Approach of sentiment analysis [51]

"Hay tres enfoques principalmente utilizados para el Análisis de Sentimientos incluyen el Enfoque Basado en Lexicones, el Enfoque de Aprendizaje Automático y el Enfoque Híbrido.

Además, los investigadores están continuamente tratando de encontrar mejores formas de llevar a cabo la tarea con una precisión superior y un menor costo computacional."[51]

En esta sección para la realización de un análisis de sentimiento usando un **Enfoque Basado en Lexicones**, vamos a mirar el *dataset*, extrayendo los datos igual que en la sección de Análisis de sentimiento exploratorio excepto que aquí eliminaremos todo *hashtag* y referencias a otros usuarios, además que utilizaremos el *lexicon* de *emojis* para poder evaluar los *emojis* como palabras y los léxicos **NRC**, **AFINN** y **ML-Senticon**.

Se aplicará la metodología de **Enfoque Basado en Lexicones** en base de un *dataset* en el que se le aplicará un *inner join* de las palabras que aparecen en el texto del *dataset* con los léxicos **NRC**, **AFINN** y **ML-Senticon**. Para el cálculo manual de la polaridad de cada *tweet* usaremos las suma de las polaridades proporcionadas por cada *lexicon* y sacaremos la media según la longitud del *tweet*. Con este valor luego se comparará con las etiquetas ya evaluadas manualmente para observar el nivel de precisión que ofrece este enfoque.

6.5.1. Procesamiento de lenguaje natural con NRC

En esta sección se le aplica a los datos un filtro en el que extraemos todas las palabras que no aparecen en el *lexicon* **NRC**. Como podemos ver en comparación a la figura de distribución de filas en el *dataset* sin ningún *lexicon* aplicado 6.2 se puede ver que debido al descarte de algunas palabras, la cantidad de *tweets* disponible para el análisis se ha reducido como se puede ver en la figura 6.9 dejando un total de 11,474 siendo el *lexicon* que menos *tweets* ha descartado.

Si miramos en las figuras de frecuencias 6.10, 6.11 y 6.12, se puede observar que en los *tweets* positivo hay un uso más frecuente en el uso de *emojis* en comparación a los otros tipos de *tweets*.

Después de haber aplicado el filtro se hace un cálculo de la clasificación del sentimiento del *tweet* en base de la polaridad promedia de mas palabras en el texto. Con esto se realiza una evaluación de los resultados calculados usando únicamente la metodología de **Enfoque Basado en Lexicones**. Como se puede ver en la tabla 6.1 que representamos con una fila con los datos desequilibrados y en la otra aplicamos un *undersample* de los datos para tener una representación más realista, la precisión es bastante baja siendo el *lexicon* **NRC** el que mayor valor ha dado, al enfocarse exclusivamente en el uso exclusivo de *lexicon*.

Cuadro 6.1: Precisión según enfoque: lexicon NRC

Precisión %	**Positivo**	**Negativo**	**Neutro**	**Total**
Desequilibrado	13,25	6,21	13,86	33,33
Balanceado	25,85	14,36	6,35	46,56

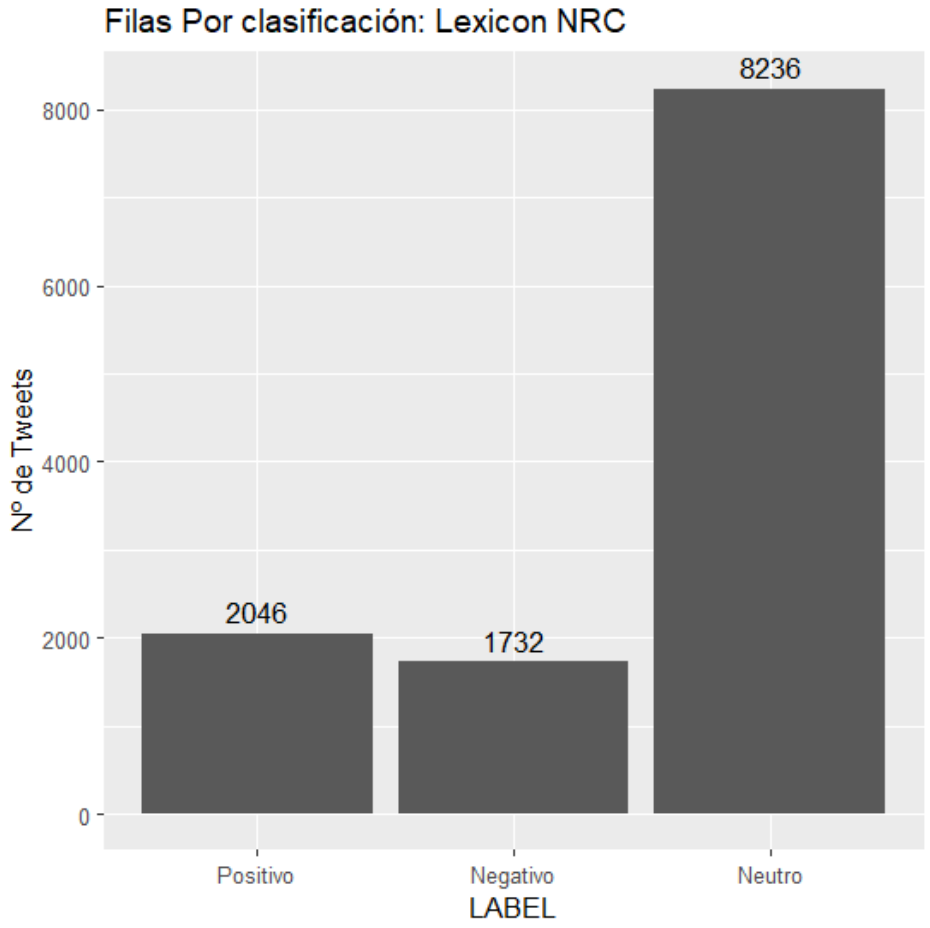

Ilustración 6.9: Distribución Filas Por Clasificación En Un Lexicon NRC

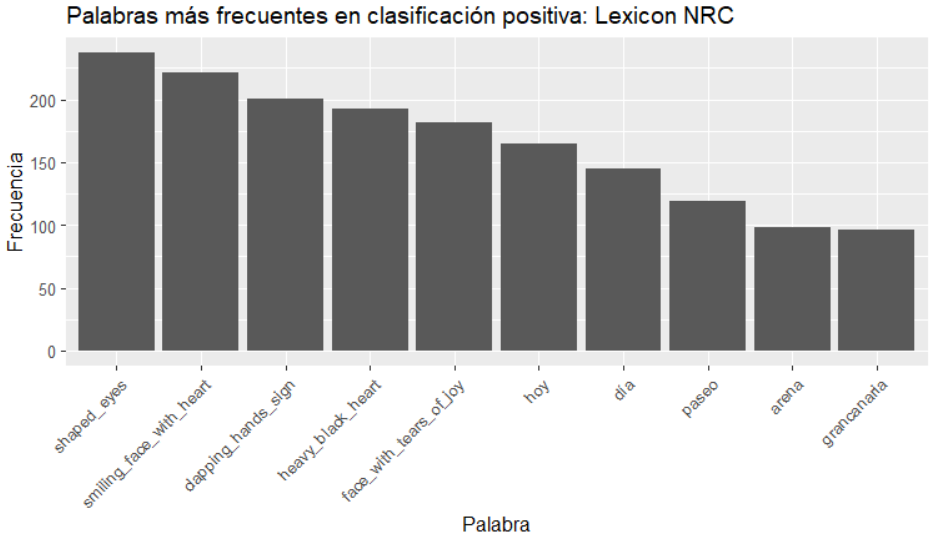

Ilustración 6.10: Frecuencia palabras positivas lexicon NRC

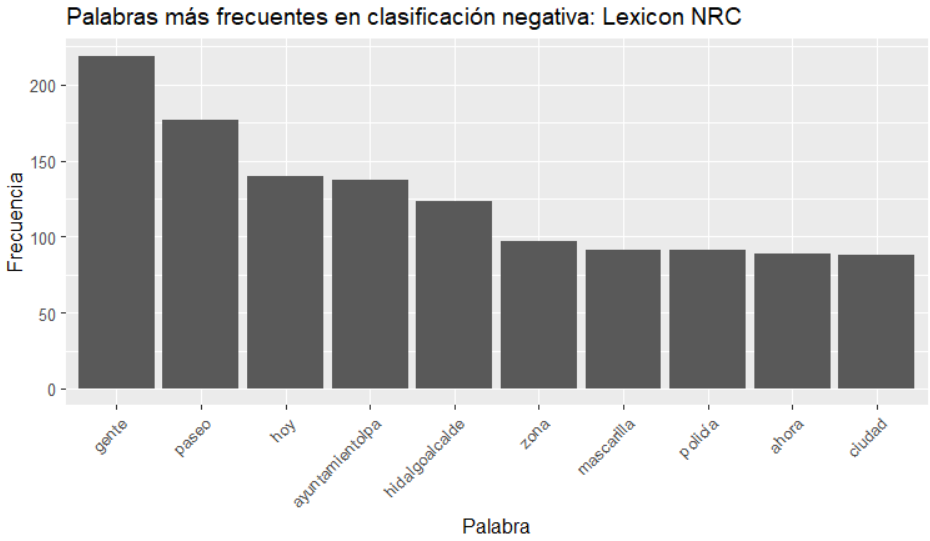

Ilustración 6.11: Frecuencia palabras negativas lexicon NRC

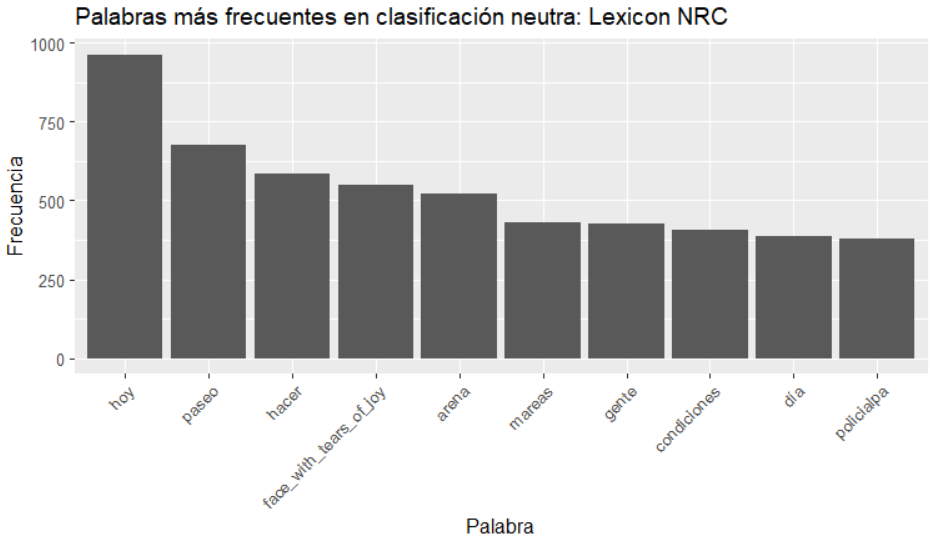

Ilustración 6.12: Frecuencia palabras neutras lexicon NRC

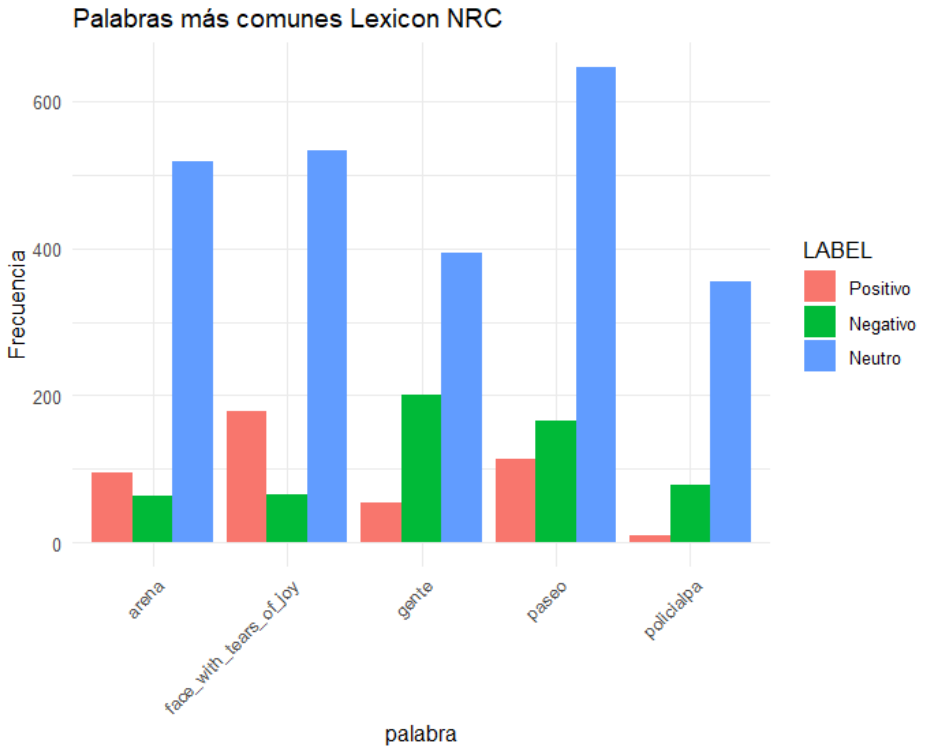

Ilustración 6.13: Apariencia de las palabras más comunes Lexicon NRC

6.5.2. Procesamiento de lenguaje natural con AFINN

En esta sección se le aplica a los datos un filtro en el que extraemos todas las palabras que no aparecen en el *lexicon* **AFINN**. Como podemos ver en comparación a la figura de distribución de filas en el *dataset* sin ningún *lexicon* aplicado 6.2 se puede ver que debido al descarte de algunas palabras, la cantidad de *tweets* disponible para el análisis se ha reducido como se puede ver en la figura 6.14 a un total de 8,918 siendo el *lexicon* que más *tweets* ha descartado.

Si miramos en las figuras de frecuencias 6.15, 6.16 y 6.17, se puede observar que en los *tweets* positivo hay un uso más frecuente en el uso de *emojis* en comparación a los otros tipos de *tweets*. Esto es algo que que también se puede observar en el *lexicon* **NRC**.

Después de haber aplicado el filtro se hace un cálculo de la clasificación del sentimiento del *tweet* en base de la polaridad promedia de mas palabras en el texto. Con esto se realiza una evaluación de los resultados calculados usando únicamente la metodología de **Enfoque Basado en Lexicones**. Como se puede ver en la tabla 6.2 que representamos con una fila con los datos desequilibrados y en la otra aplicamos un *undersample* de los datos para tener una representación más realista, la precisión es bastante baja al enfocarse exclusivamente en el uso exclusivo de *lexicon*.

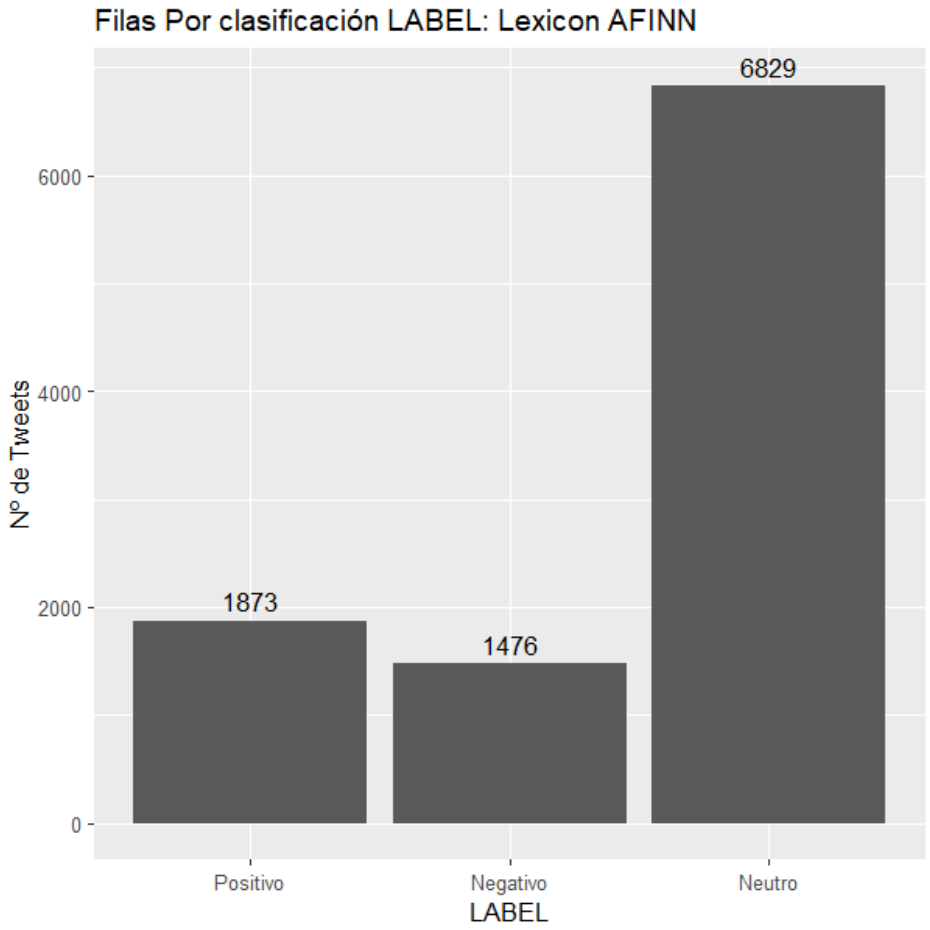

Ilustración 6.14: Distribución Filas Por Clasificación En Un Lexicon AFINN

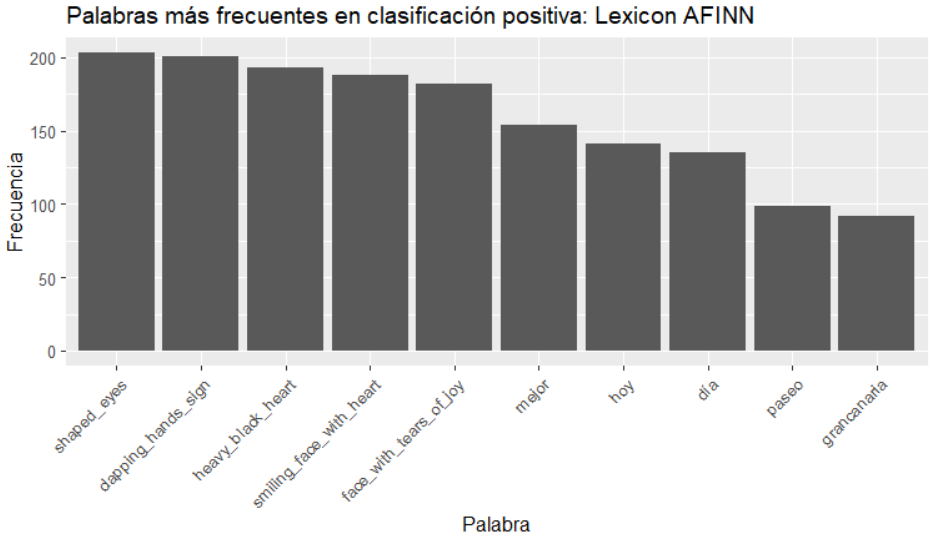

Ilustración 6.15: Frecuencia palabras positivas lexicon AFINN

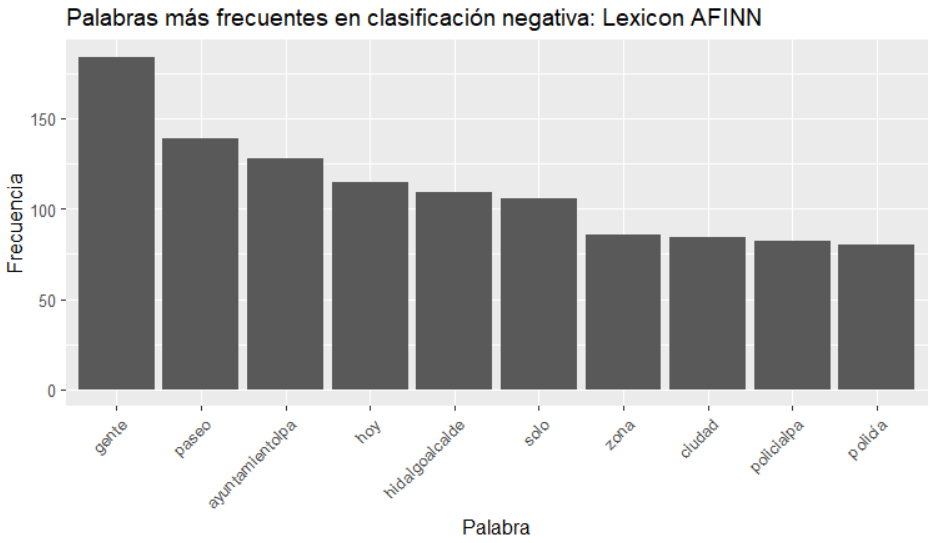

Ilustración 6.16: Frecuencia palabras negativas lexicon AFINN

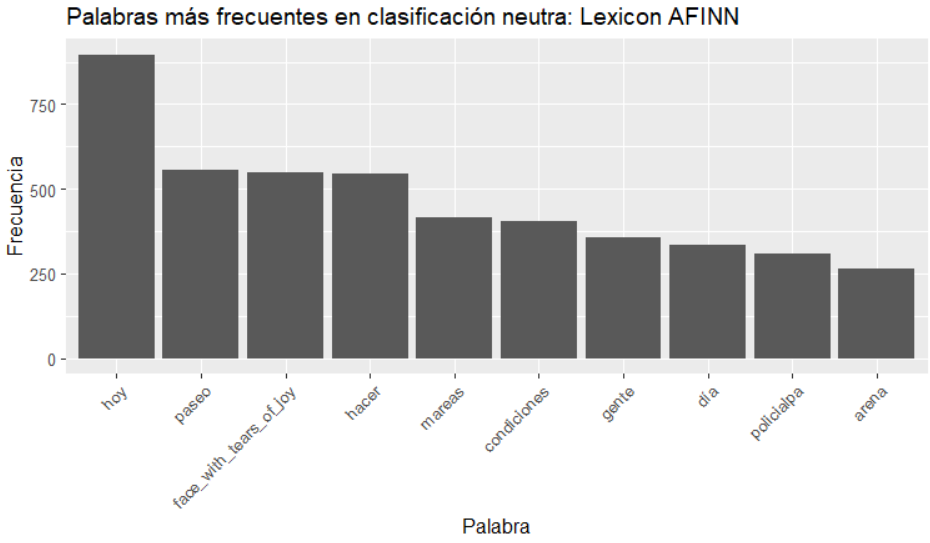

Ilustración 6.17: Frecuencia palabras neutras lexicon AFINN

Cuadro 6.2: Precisión según enfoque: lexicon AFINN

Precisión %	**Positivo**	**Negativo**	**Neutro**	**Total**
Desequilibrado	3,34	2,40	56,72	62,45
Balanceado	6,07	5,51	28,43	40,02

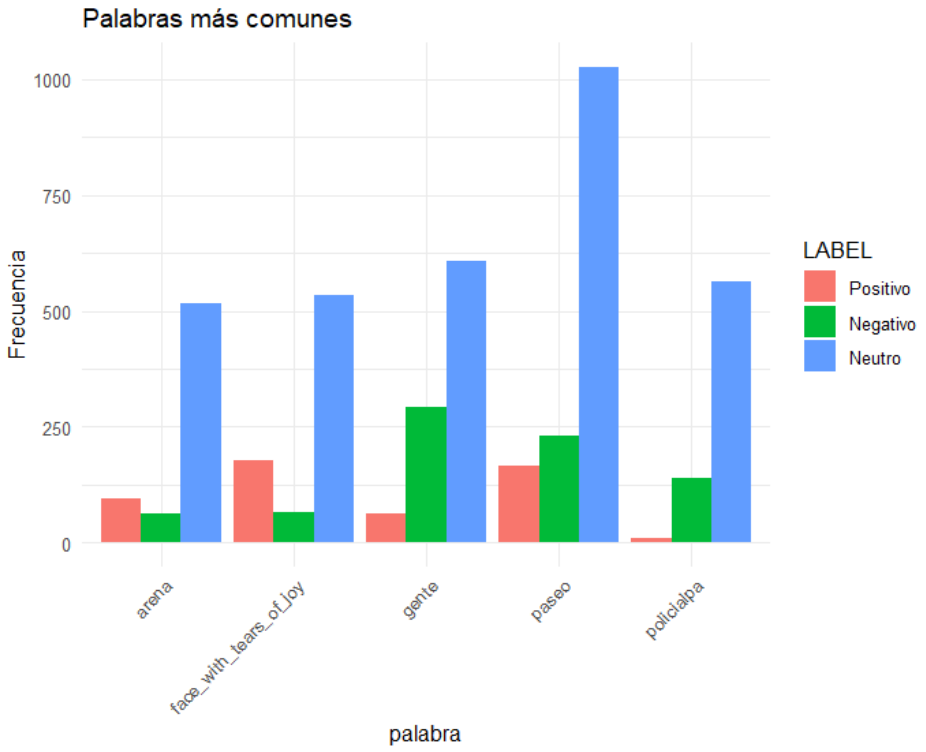

Ilustración 6.18: Apariencia de las palabras más comunes Lexicon AFINN

6.5.3. Procesamiento de lenguaje natural con ML-Senticon

En esta sección se le aplica a los datos un filtro en el que extraemos todas las palabras que no aparecen en el *lexicon* **ML-Senticon**. Como podemos ver en comparación a la figura de distribución de filas en el *dataset* sin ningún *lexicon* aplicado 6.2 se puede ver que debido al descarte de algunas palabras, la cantidad de *tweets* disponible para el análisis se ha reducido como se puede ver en la figura 6.19 a un total de 9,654.

Si miramos en las figuras de frecuencias 6.20, 6.21 y 6.22, se puede observar que en los *tweets* positivo hay un uso más frecuente en el uso de *emojis* en comparación a los otros tipos de *tweets*. Esto es algo que que también se puede observar en los anteriores *lexicons*.

Después de haber aplicado el filtro se hace un cálculo de la clasificación del sentimiento del *tweet* en base de la polaridad promedia de mas palabras en el texto. Con esto se realiza una evaluación de los resultados calculados usando únicamente la metodología de **Enfoque Basado en Lexicones**. Como se puede ver en la tabla 6.3 que representamos con una fila con los datos desequilibrados y en la otra aplicamos un *undersample* de los datos para tener una representación más realista, la precisión es bastante baja siendo el *lexicon* **ML-Senticon** el que menor valor ha dado, al enfocarse exclusivamente en el uso exclusivo de *lexicon*.

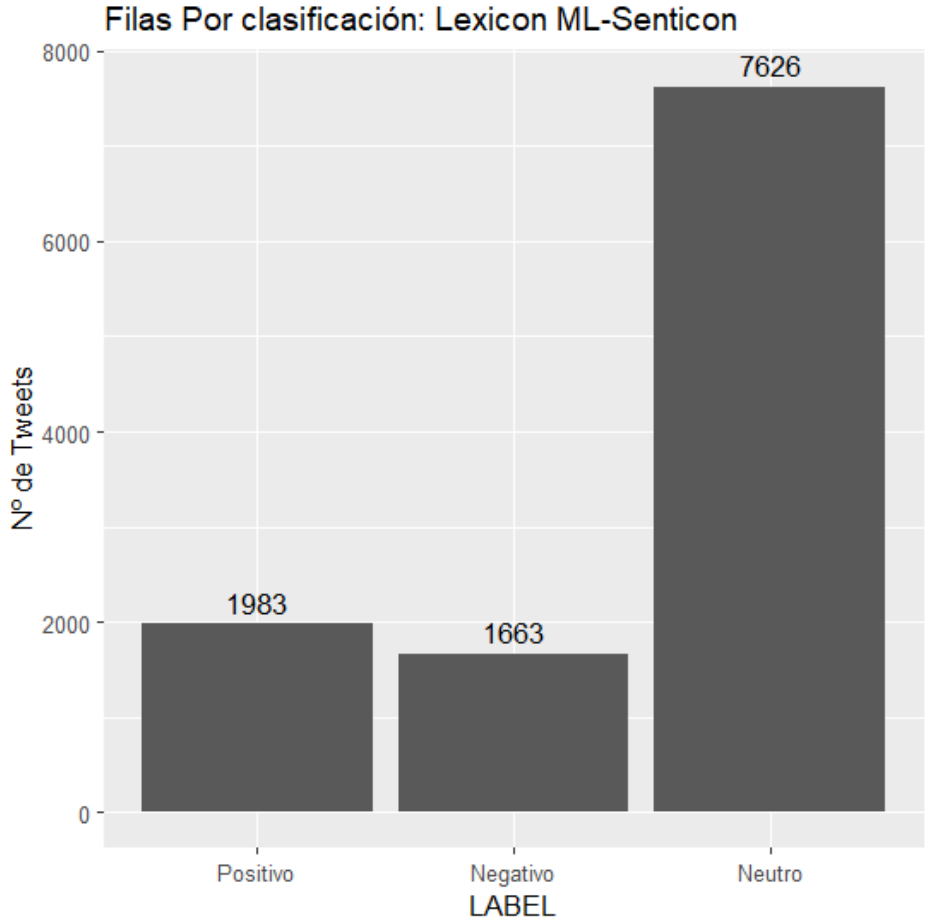

Ilustración 6.19: Distribución Filas Por Clasificación En Un Lexicon ML Senticon

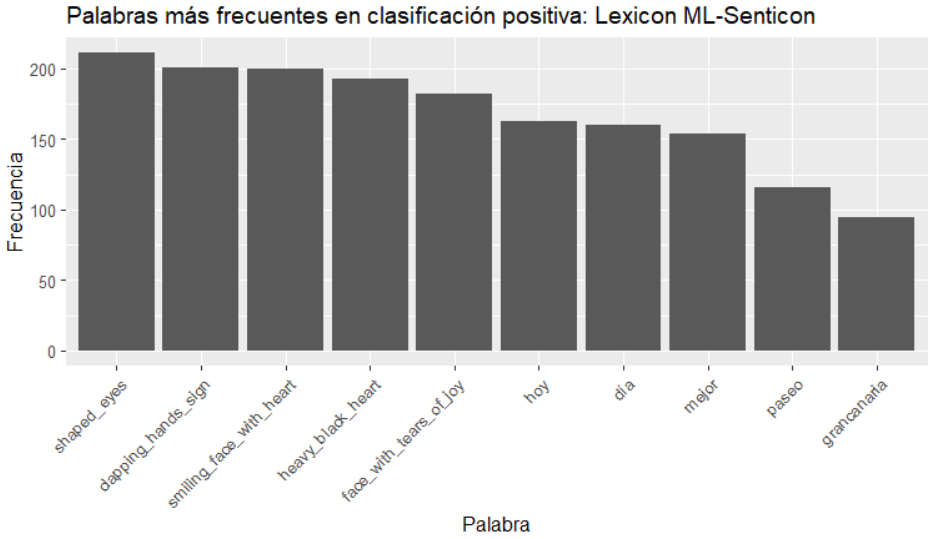

Ilustración 6.20: Frecuencia palabras positivas lexicon ML Senticon

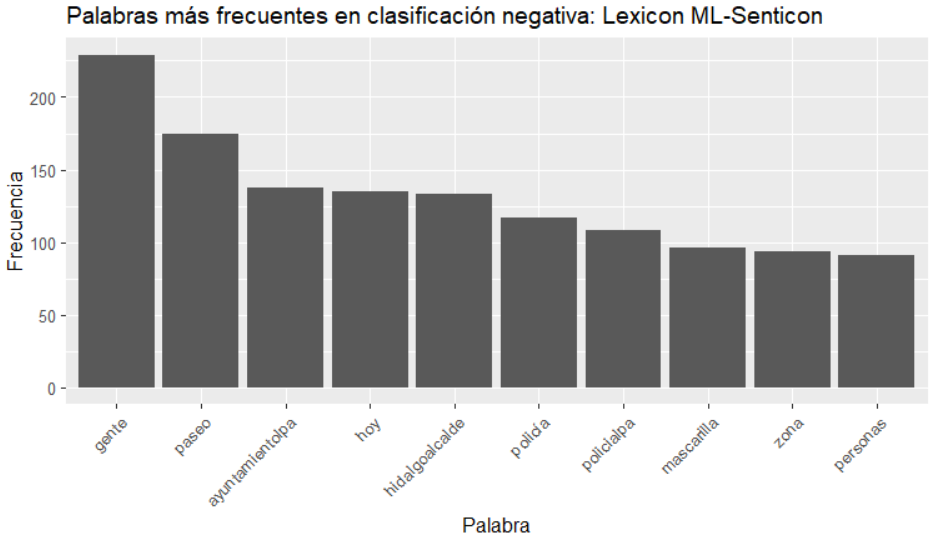

Ilustración 6.21: Frecuencia palabras negativas lexicon ML Senticon

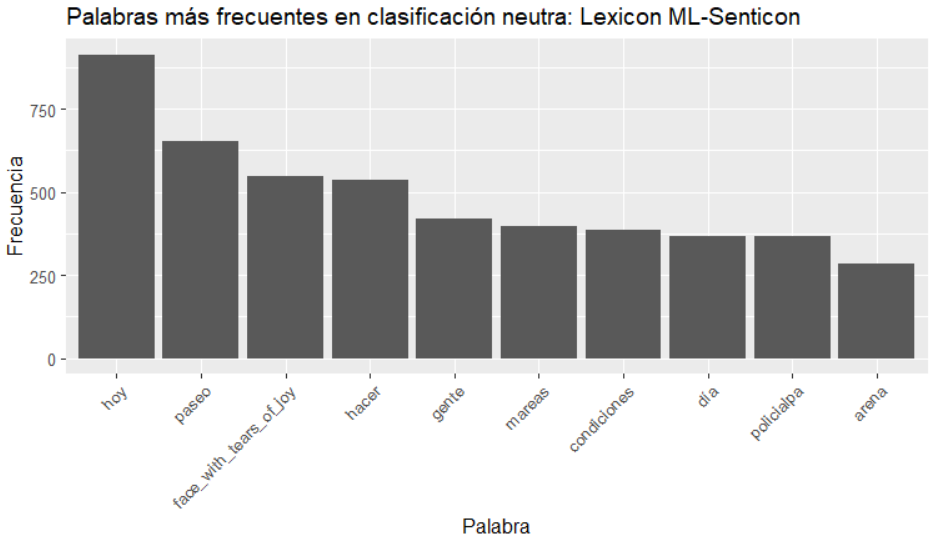

Ilustración 6.22: Frecuencia palabras neutras lexicon ML Senticon

Cuadro 6.3: Precisión según enfoque: lexicon ML-Senticon

Precisión %	**Positivo**	**Negativo**	**Neutro**	**Total**
Desequilibrado	11,39	2,03	31,98	45,40
Balanceado	22,29	4,84	15,60	42,73

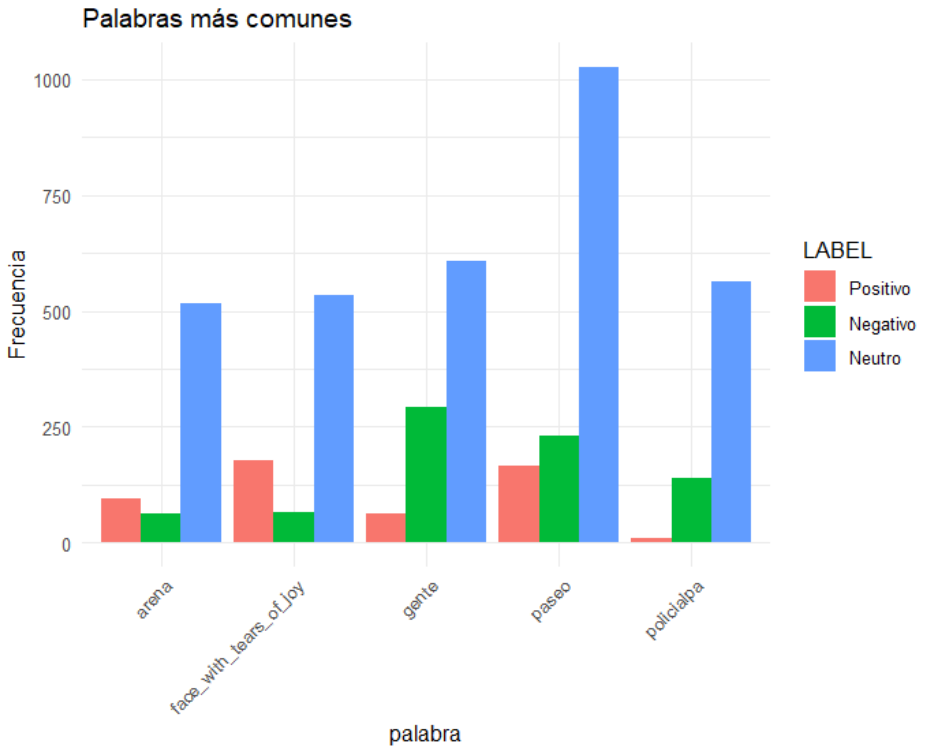

Ilustración 6.23: Apariencia de las palabras más comunes Lexicon ML-Senticon

6.5.4. Procesamiento de lenguaje natural con QDAP

"QDAP (Quantitative Discourse Analysis Package) es un paquete de R diseñado para asistir en el análisis cuantitativo de discursos. El paquete sirve de puente entre las transcripciones cualitativas del diálogo y el análisis y la visualización estadísticos."[43]

Es esta sección se realizará un análisis de sentimiento usando el paquete **QDAP**, dado a que es un paquete bastante completo, para realizar todas las fases de análisis de sentimiento. En el caso de este análisis, se trabajará con una distribución de los datos balanceados. A parte de ello para la creación de *lexicon* personalizado, en la que la polaridad de las palabras se calculará según la clasificación de los *tweets* en donde se encuentran. Para ello se extrae un 10 % de las filas para usar de referencia. A partir de ahí se empieza a preprocesar los datos. Pasos de preprocesamiento6.24:

1. Convertir a minúsculas
2. Eliminar marcas de puntuación inesperadas
3. Eliminar dígitos
4. Eliminar números enteros
5. Eliminar *stopwords*
6. Eliminar caracteres de escape

```r
clean_text <- tolower(clean_text)
clean_text <- gsub("[[:punct:]]", " ", clean_text)
clean_text <- gsub("[[:digit:]]", " ", clean_text)
clean_text <- gsub("\\b\\d+\\b", "", clean_text)
clean_text <- removeWords(clean_text, stopwords_tibble$palabra)
clean_text <- clean(clean_text)
clean_text <- Trim(clean_text)
clean_text <- scrubber(clean_text)
check_result <- qdap::check_text(clean_text)
if (any(check_result$missing_ending_punctuation == 1)) {
  clean_text <- paste(clean_text, ".", sep = "")
}
```

Ilustración 6.24: Preprocesado **QDAP**

```r
lexico_nrc_positivo = filter(lexico_nrc, sentimiento == "positivo")
lexico_nrc_neutral = filter(lexico_nrc, sentimiento == "neutral")
lexico_nrc_negativo = filter(lexico_nrc, sentimiento == "negativo")

custom_positivo_words <- positive.words
custom_negativo_words <- negative.words

custom_positivo_words <- c(custom_positivo_words, lexico_nrc_positivo$palabra)
custom_negativo_words <- c(custom_negativo_words, lexico_nrc_negativo$palabra)

custom.negators <- negation.words
custom.negators <- c(custom.negators, "nadie","nada","ni","jamas","nunca","tampoco", "no", "ninguno", "ninguna", "sin")

custom.amplification.words <- amplification.words
custom.amplification.words <- c(custom.amplification.words,"mucho","muchisimo", "mas","gran","grande","repleta","repleto")

custom.deamplification.words <- c(deamplification.words,"raro","poco","menos")

custom_positivo_words <- gsub("\\s+", "",custom_positivo_words)
custom_negativo_words <- gsub("\\s+", "",custom_negativo_words)

custom_pol <- sentiment_frame(custom_positivo_words,custom_negativo_words)



sentiment <- polarity(clean_text,polarity.frame = custom_pol,
                                          negators = custom.negators, amplifiers = custom.amplification.words,
                                          deamplifiers = custom.deamplification.words, constrain = T)
```

Ilustración 6.25: Implementación de palabras personalizada

7. Recortar espacios en blanco
8. Eliminar anomalías textuales
9. Añadir marcas de puntuación al final

Una vez que hayamos limpiado los datos probamos hacer un calculo de polaridad con la función que ofrece **QDAP**. Por defecto la biblioteca ya dispone de su propia lista de palabras personalizadas. Aunque, debido a que su uso es principalmente para un texto inglés, no dispone de suficientes palabras en castellano para realizar un calculo viable. Para intentar conseguir unos resultados más aptos para una evaluación de polaridad se implementará los *lexicons* **NRC**, **AFINN**, **ML-Senticon**, y el *lexicon* personalizado previamente mencionado. A parte de ello también se añaden a la lista de palabras negativas, de amplificación y atenuación usadas de manera común en el castellano6.25.

A partir de añadir estos parámetros se han sacado los resultados vistos en la tabla 6.4. Como se puede ver de el *lexicon* personalizado tiene la precisión total más alta. A pesar de

ello los resultados son bastante volátiles debido al hecho de que se basa de una extracción aleatoria de los datos. En cuanto a los *lexicons* **NRC**, **AFINN** y **ML-Senticon** se ve que la precisión total son bastantes similares con **NRC** dando el resultado más alto. En el caso del *lexicon* **AFINN** la distribución de precisión según la polaridad está bastante inclinada hacia los *tweets* neutros dejando los *lexicons* **NRC** y **ML-Senticon** como los más favorables. A pesar de todo ello los resultados no se alejan mucho de que sea lo mismo que dejarlo al azar.

Cuadro 6.4: Precisiónes qdap

Precisión %	**Positivo**	**Negativo**	**Neutro**	**Total**
Sin Lexicon	4,83	6,41	93,90	36,28
AFINN	21,28	17,80	81,74	41,92
NRC	44,35	31,65	53,78	42,74
ML-SENTICON	45,74	22,44	55,12	39,98
Lexicon Personalizado	32,24	83,46	30,07	51,19

6.5.5. Procesamiento de lenguaje natural con NLTK/VADER

VADER es un modelo sencillo basado en reglas para el análisis general de sentimientos cuyas siglas son las siglas de *Valence Aware Dictionary for sEntiment Reasoning*. Este modelo está diseñado con uno de sus enfoques siendo la evaluación textos en redes sociales([21]). VADER aplica un enfoque basado en *lexicons* y la aplicación de varías heurísticas. En el caso de este TFG, dado que el *dataset* está en castellano no se puede esperar una precisión de la clasificación tan alta como la de un texto ingles que ronda el 96 %[21] para textos positivos.

Dado a la naturaleza de este modelo en el que todo figura relevante en cuanto al calculo de polaridad, no se le aplicará ningún tipo de preprocesado a parte de un balanceo de los datos para facilitar la evaluación de precisión. Como previamente mencionado, el *lexicon* de Vader está diseñado para trabajar con textos en inglés. Asi que se ha intentando añadir los *lexicons* **NRC**, **AFINN** y **ML-Senticon**. Como se puede ver en la figura 6.5 la precisión es un poco mejor que usar el **QDAP**. Esto deja entender el que el uso de un *lexicon* adecuado es esencial para el correcto funcionamiento de estos modelos.

Cuadro 6.5: Precisiónes VADER

Precisión %	**Positivo**	**Negativo**	**Neutro**	**Total**
Sin Lexicon	41,83	58,81	37,12	42,80
AFINN	51,61	60,37	41,85	41,85
NRC	47,15	60,34	40,39	48,15
ML-SENTICON	35,89	50,96	42,45	41,55

6.6. Modelado de sentimientos basado en TFIDF

6.6.1. Preparación de los datos

En esta fase, su enfoque es "la selección de datos, limpieza de datos, construcción de nuevos datos, integración de datos y formato de datos"[23]. En el caso de este trabajo inicialmente todo el preprocesamiento de los datos desde su limpieza hasta su vectorización fue realizada usando el paquete *tidytext*; Éste paquete había resultado bastante práctico en cuanto a la fase exploratoria, pero resultó bastante problemático en las fases de vectorización. Debido a ello se usó el *framework Tidymodels*. Este *framework* introduce el concepto de *recipes*. Las *recipes*, ofrecen una implementación de las fases de la ingeniería de características conocidas como *steps*. Estas secuencias se utilizan para preparar los datos para el modelado, ofreciendo funciones que nos facilitan la limpieza, mutación, tokenización y vectorización de los datos. En el caso de este trabajo esto ha venido muy útil dado a las dificultades que suelen venir con trabajar con un texto.

No obstante para ciertos casos como el *stemming* se usó el *recipe step* de *step_stem* donde se aplica la función de *stemming* que se usó fue **wordStem** del paquete **SnowballC** que dispone del diccionario castellano. Esto nos permite transformar un token a su raíz. Luego en el caso de la lematización se tuvo que realizar una implementación utilizando el paquete **spacyr** que a través del paquete **reticulate** que nos permite ejecutar este paquete de **python**. Este paquete nos permite abrir una breve instancia de **python** para poder ejecutar sus funciones6.30.

6.6.2. Selección de datos

Para la selección de datos se tiene que establecer cual es el objetivo para los datos actuales. Se prepara un *dataset* en el que está compuesto de 3036 *tweets* positivos, 2344 *tweets* negativos y 14095 *tweets* neutros. Más adelante se hablará de como se resolverá este desequilibrio en los datos. Se mirará el proceso de aplicar el *stemming* y *lemming* al *dataset* que no se le integre un *lexicon*.

6.6.3. Undersampling y Oversampling

Dado a que el *dataset* disponible tiene una clasificación desequilibrada de datos donde podemos ver que tiene una gran mayoría de *tweets* etiquetados como neutral. Para ellos tenemos disponibles 3 posibles opciones. *undersampling*, *oversampling* por medio de aumentación de datos y crear unos parámetros de categorizar los valores de polaridad continua (como está en la grabación) que se usarán para el caso de NLP basado en puntos de polaridad del léxico, *undersampling* 6.26 es la técnica de reducir muestras de la clase mayoritaria mientras que el *oversampling* 6.26 es la duplicación de datos de la clase minoritaria en este caso en vez de

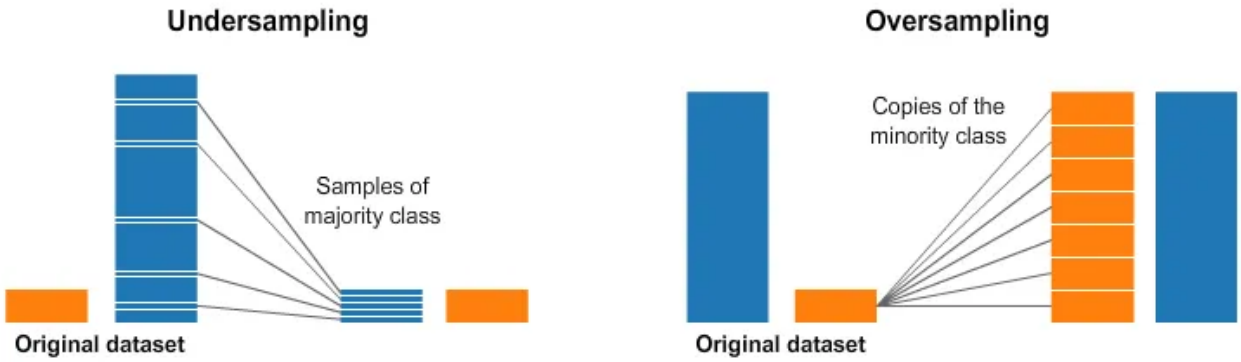

Ilustración 6.26: Undersampling and *oversampling*[46]

duplicar los *tweets* de las clases minoritarias se ha generado *tweets* basado en las palabras usadas en los *tweets* de su clase.

Para el caso de la generación de *tweets* y el establecimiento de umbrales de polaridad se usa una subsección de validación extraída del *dataset* de validación que usaremos como base. Este *dataset* de validación estará compuesto de una muestra aleatoria del 10 % del *dataset* original. Se implementa una función personalizada6.27 para la generación de *tweets* en el caso de una distribución así. Se extrae un vocabulario de los *tweets* en cada clase en el que se usarán las frecuencias de las palabras para generar aleatoriamente un *tweet*.

6.6.4. Limpieza de datos

Aquí se enfocó en la limpieza de los datos eliminando contenido que no sea parte de nuestro enfoque. En el caso de los *hashtags* y referencias a otros usuarios 6.28. Aunque se ha visto que los *hashtags* pueden tener un impacto en análisis de sentimientos [50] se ha decidido omitirlos6.29. También se ha decidido la eliminación de los *stopwords*. En nuestro texto procesado solo hemos extraídos las palabras y *emojis*.

6.6.5. Construcción de nuevos datos

Dado a la naturaleza desequilibrada se realizó un *downsampling* en el que se procedió a equilibrar la distribución de datos reduciendo el tamaño del *dataset* en base de la clase más baja que en este caso fue la negativa. Otra implementación que se aplicó fue un *oversampling* de los datos.

6.6.6. Integración de datos

En cuanto la integración de datos, debido al origen del *dataset* en uso, no se buscará ningún tipo de integración con datos de fuentes externos. En cuanto a los datos que ya están

```r
#En este asumimos que los datasets están en su paso final asi que pasaremos de filtros
shinGenTweets <- function(df_validation_tweets, df_training_test_tweets){
  #pillamos de la tabla de validación textos para usar de ejemplo para el generador de tweets
  text_column <- df_validation_tweets$text

  #sacando el promedio de longitud de tweets tenemos una idea de que longitud debería tener los tweets generados
  avg_word_per_tweet <- mean(sapply(
    strsplit(as.character(text_column), "\\s+"), length))
  avg_word_per_tweet <- round(avg_word_per_tweet)
  #Aqui vamos a generar los tweets según lo que nos haga falta
  positive_label_size <- as.numeric(table(df_training_test_tweets$LABEL)[1])
  negative_label_size <- as.numeric(table(df_training_test_tweets$LABEL)[2])
  neutral_label_size <- as.numeric(table(df_training_test_tweets$LABEL)[3])

  vbo <- validator_build_no_lexicon(df_validation_tweets)

  df_generated_positive_tweets <- generate_tweets(
    avg_word_per_tweet,
    vbo[[2]],
    neutral_label_size - positive_label_size, 1)

  df_generated_negative_tweets <- generate_tweets(
    avg_word_per_tweet,
    vbo[[3]],
    neutral_label_size - negative_label_size, 2)

  df_generated_negative_positive_tweets <- rbind(
    df_generated_negative_tweets,
    df_generated_positive_tweets)

  df_tweets_generated_balance <- rbind(
    df_generated_negative_positive_tweets %>%
      select(text_id = ID, text, LABEL), df_training_test_tweets) %>%
    mutate(text_id = 1:n())
  return(df_tweets_generated_balance)
}
```

Ilustración 6.27: Snippet función generación de tweets

```r
if(filtrar_usertag){
  print("filtrando usertags")
  df$text <- gsub("@\\w+", "", df$text)
}
```

Ilustración 6.28: Filtro de usertags

```r
if(filtrar_hashtag){
  print("filtrando hashtags")
  df$text <- gsub("#\\w+", "", df$text)
}
```

Ilustración 6.29: Filtro de hashtags

```r
if (!require(spacyr)) {
  install.packages("spacyr")
  library(spacyr)
  spacy_install()
}

innit_lema <- function(){
  spacy_download_langmodel(lang_models = "es_core_news_lg")
  spacy_initialize(model = "es_core_news_lg")
  es_core_news_lg
}
finalize_lema <- function(){
  spacy_finalize()
}
lemmatize_text <- function(text) {
  options(warn=-1)
  tokens <- spacy_parse(text)
  options(warn=0)
  lemmas <- tokens$lemma
  lemmatized_text <- paste(lemmas, collapse = " ")
  return(lemmatized_text)
}
```

Ilustración 6.30: Fragmento código lematizador

presentes, se podrá considerar la generación de *tweets* en el proceso de aumento de datos como una alternativa.

6.6.7. Formato de datos

En la fase de vectorización se hará una conversión de los tokens de palabras a valores numéricos para que los algoritmos de *machine learning* los puedan leer. En este caso se usarán técnicas de frecuencia de terminos como el *term frequency* (**TF**) y el *term frequency inverse document frequency* (**TF-IDF**).

6.6.8. Emojis

Para la conversión de *emojis* se utilizó el *Emoji Sentiment Ranking 1.0*[29]. Este *dataset* no ofrecía directamente la polaridad de los *emojis*, dado a que la tabla que proporcionaba el estudio no contenía el **sentiment score** estos valores se calcularon replicando los resultados que se pueden visualizar en la tabla [29, Emoji Sentiment Ranking v1.0]. Siendo **positive_ocurrences** la cantidad de veces que el *emoji* aparece en *tweets* positivos y **negative_ocurrences** la cantidad de veces que aparece el *emoji* en *tweets* negativos, podemos calcular el **sentiment_score** 6.6.8. Luego en base de este **sentiment_score** podemos asignarle polaridad al *emoji*6.6.8.

```
if(filtrar_locuciones){
  locuciones_lexicon <- locuciones_lexicon[grep("_", locuciones_lexicon$palabra), ]

  print("filtrando locuciones")
  palabracount = 0;
  for (palabra in locuciones_lexicon$palabra) {
    df$text <- gsub(paste0("\\b", gsub("_", " ", palabra), "\\b"),
                    palabra,
                    df$text)
    palabracount = palabracount + 1
    progress(palabracount, length(locuciones_lexicon$palabra))
  }

  rm(locuciones_lexicon)
}
```

Ilustración 6.31: Fragmento código locuciones

$$\text{sentiment\_score} = \frac{\text{positive\_ocurrences} - \text{negative\_ocurrences}}{\text{total\_ocurrences}}$$

$$\text{sentiment} = \begin{cases} \text{positive} & \text{if sentiment\_score} > \left(1 - 2\left(\frac{\text{Positive}}{\text{Occurrences}}\right)\right) \\ \text{negative} & \text{if sentiment\_score} < \left(-1 + 2\left(\frac{\text{Negative}}{\text{Occurrences}}\right)\right) \\ \text{neutral} & \text{otherwise} \end{cases}$$

6.6.9. Locuciones

En el castellano es bastante común el uso de las locuciones. Las locuciones son un "Grupo fijo de palabras que constituye una unidad léxica compleja"[11]. En el preprocesado de un texto las locuciones suelen ser descartadas ya que muchas de las palabras que componen una locución se pueden confundir con las *stopwords*. Para evitar la perdida de éstas palabras se utiliza el *lexicon ML-Senticon* ya que contiene una gran cantidad de locuciones. Después de haber definido correctamente las locuciones6.31 es cuando ya se puede comenzar el resto de los *steps* del preprocesado.

El *lexicon ML-Senticon* se obtiene a partir de un archivo **XML**, compuesto de 8 capas de lematización, en el que cada una está dividida por las categorías positivas y negativas, dentro de las categorías de positivo y negativo se dispone de una lista de palabras en las que vienen asignadas una polaridad representadas en un rango de -1 a 1 siendo -1 las polaridades negativas y 1 las polaridades positivas. Para este archivo se tuvo que crear una series de funciones para de-construir el **XML** a un formato **data.frame**.

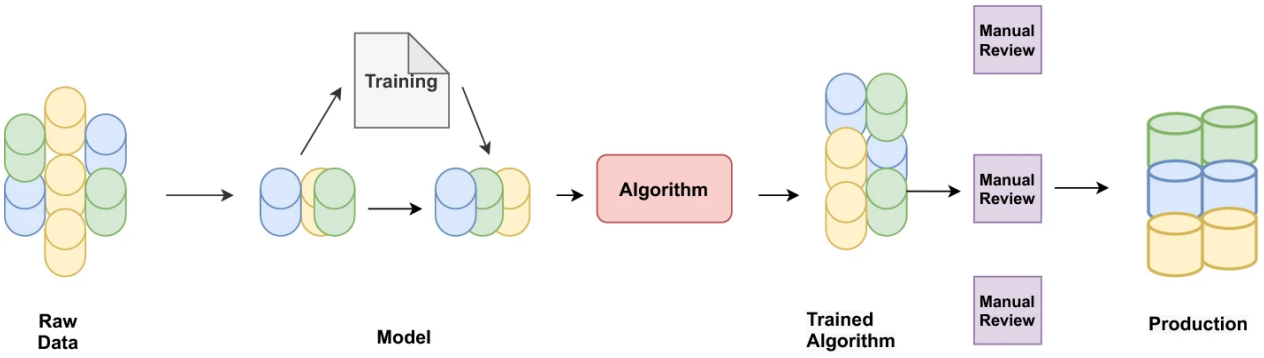

Ilustración 6.32: General procedure for sentiment analysis in supervised machine learning category [51]

6.6.10. Modelado

En esta fase, su enfoque es "la selección de técnicas de modelado, generación de un diseño de comprobación, generación de los modelos y la evaluación del modelo"[23].

Originalmente se experimentó entrenar los modelos usando *caret* y *tidymodels* dado a que la fase del prerocesado de datos fue realizado en R. Aun así, debido al tamaño de los datos y a que el soporte que ofrece R en cuanto al uso de del GPU está bastante limitado, no se pudo realizar un entrenamiento de los modelos eficaz en R. A pesar de ello se exploraron alternativas como el uso de la paralelización para utilizar todos los *cores* de la CPU. En estos casos, el tiempo que requiso el entrenamiento podía llegar a durar múltiples días incluso con el supercomputador de la universidad.

Se ha realizado el entrenamiento de modelos en python gracias a su capacidad de usar la tarjeta gráfica del ordenador ofreciendo un tiempo de ejecución bastante bajo; Ya que para múltiples modelos, python dispone de un conjunto de paquetes como el **cuML**. **cuML** ofrece algoritmos de *machine learning* cuya implementación es bastante similar la las que ofrece el **scikit-learn** pero con capacidad de cálculo usando la GPU. También existen bibliotecas como la propia del **XGBoost** que también permiten el uso de la GPU.

6.6.11. Estilos de clasificación y enfoques comunes encontrados en los modelos

Para la clasificación de un *dataset* existen 2 tipos6.33, la clasificación binaria donde se intenta clasificar entre una de las dos clases y la la clasificación multi-clase que intenta clasificar de múltiples clases. Las multi-clases resultan más difícil debido a la implicación de añadir múltiples clases. para ello se aplican distintos tipos de enfoques.

- ✔ ***One-to-One approach* 6.34**: Este enfoque se basa en establecer una separación binaria para cada clase. Para ello será necesario crear un modelo para cada clase, resultando en un entrenamiento más lento. Este enfoque suele implicar un coste alto en cuanto a la memoria debido a la ampliación de datos que implica si se usa un *dataset* grande. [1]

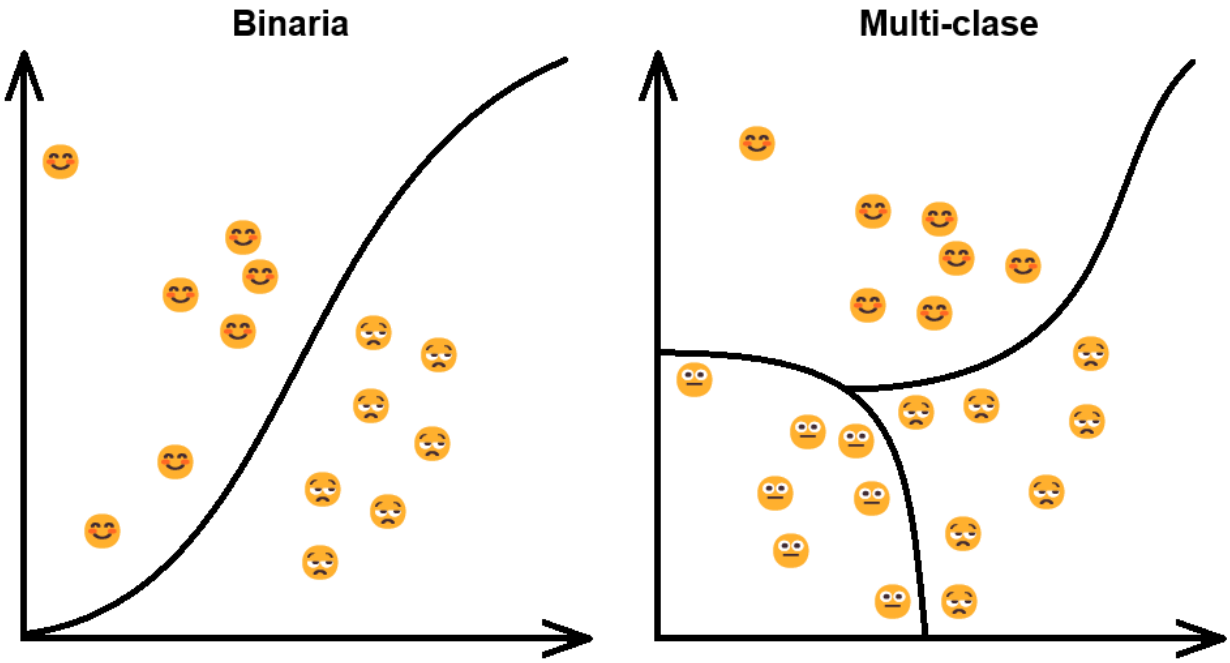

Ilustración 6.33: Clasificación Binaria/Multi-clase

- ✔ ***One-to-Rest approach* 6.35**: Este enfoque se basa en crear una separación entre una clase y todas las demás. En este caso la creación de modelos será mucho menos que en el *One-to-One*. Resultando en un entrenamiento menos lento que la alternativa a cambio de una separación más pequeña.[1]

6.6.12. Selección de técnicas de modelado

Se miran los modelos visitados y los elegidos. En esta fase de realiza una exploración de los algoritmos se *machine learning* visitados y los que finalmente han sido utilizados para entrenar el modelo. La base principal en cuanto a la decisión de que algoritmos explorar fue basado en el contexto de la clasificación multinomial y análisis de sentimientos.

6.6.13. Random Forest

El *Random forest* es una técnica de *machine learning* en la que está compuesta de un *ensemble* de *decision trees* en las que por medio del *bootstraping* y *feature selection* en el primer caso cada *decision tree* recibe aleatoriamente un subconjunto del *dataset* para tener una base única de entrenamiento, en el segundo caso cada arbol se le asigna una selección de distintos *features*. Una vez entrenado los *decision trees* del *Random forest* por el proceso del *bagging* se compila una suma de todos las predicciones.[26]6.36

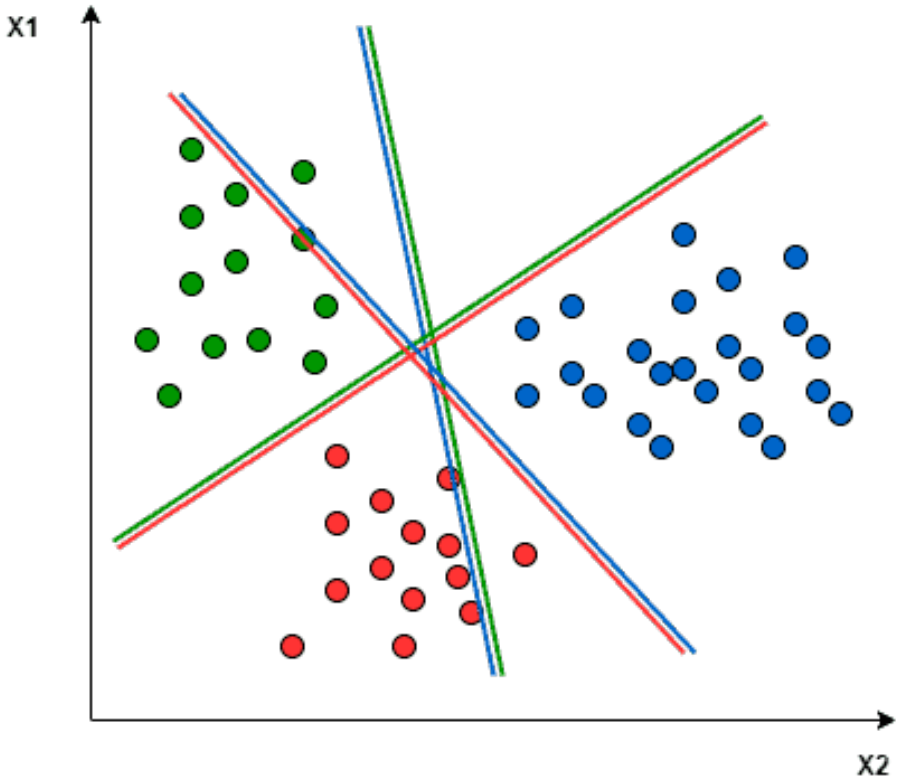

Ilustración 6.34: One-to-One approach[20]

6.6.14. Regresión lineal (LASSO)

"LASSO es el acrónimo de Least Absolute Shrinkage and Selection Operator (Operador de Selección y Reducción Mínima Absoluta). Se utiliza con frecuencia en el aprendizaje automático para manejar datos de alta dimensión, ya que facilita la selección automática de características con su aplicación."[25] LASSO es un tipo de regresión lineal en el que se enfoca en la reducción progresiva de los coeficientes irrelevantes.

6.6.15. XGBoost

"XGBoost es una librería distribuida optimizada de gradient boosting diseñada para ser altamente eficiente, flexible y portable. Implementa algoritmos de aprendizaje automático en el marco del Gradient Boosting. XGBoost proporciona un refuerzo de árbol paralelo (también conocido como GBDT, GBM) que resuelve muchos problemas de ciencia de datos de una manera rápida y precisa."[52] Similar al randomforest ésta tecnica se basa en un *algorítmo* de *ensemble gradient boosted decision tree* (GBDT). Compuesto de una combinación de múltiples *decision trees* con finalidad de obtener uno más compuesto con objetivo de colectivamente crear un modelo mejor.[7]

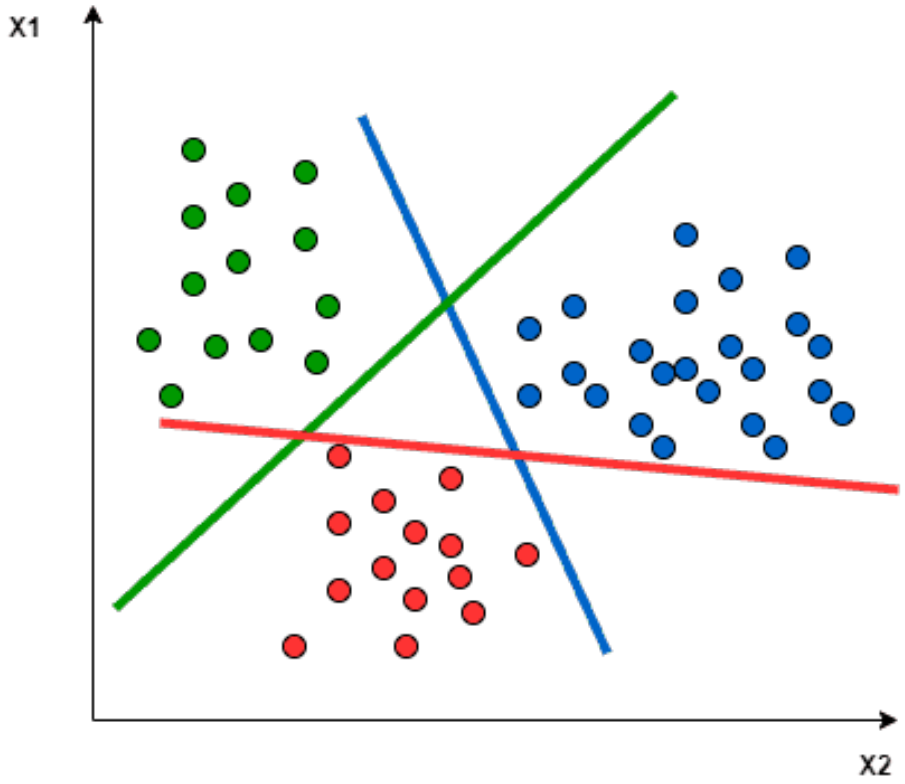

Ilustración 6.35: One-to-Rest approach[20]

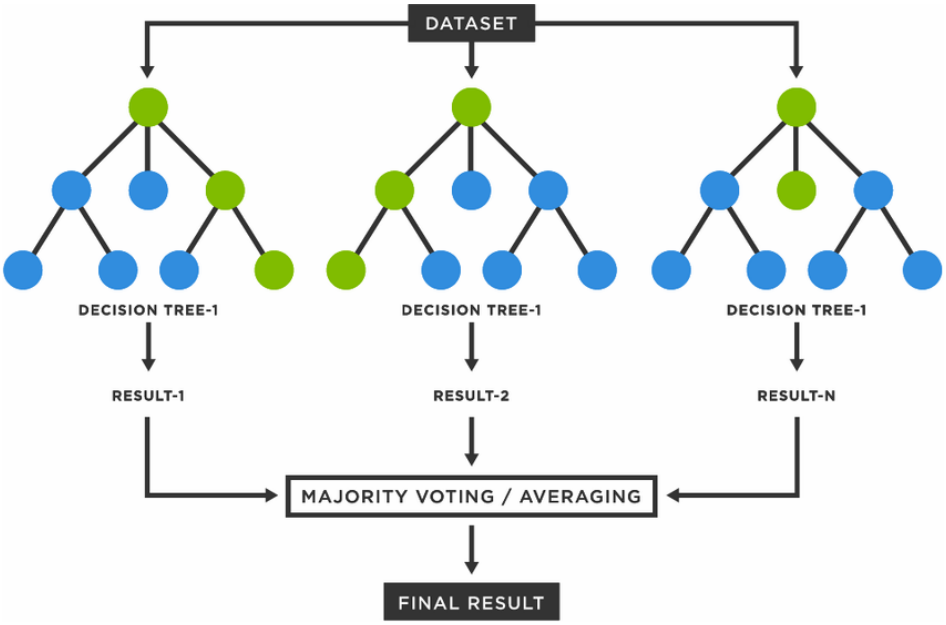

Ilustración 6.36: Workflow representation of a Random Forest model [18]

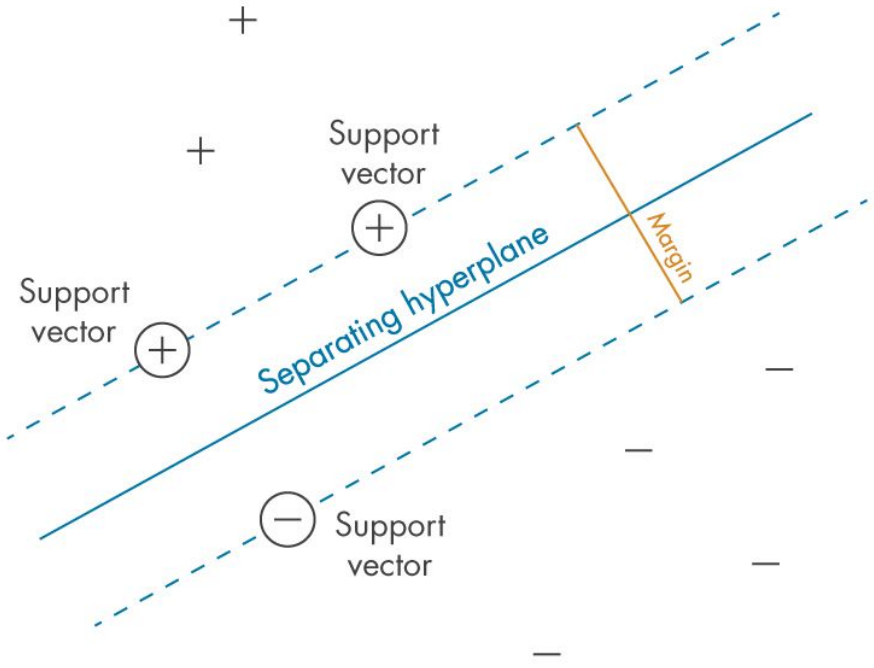

Ilustración 6.37: Definición del "margen" entre clases: el criterio que los SVM intentan optimizar. [47]

6.6.16. SVM

"Una máquina de vectores soporte (SVM) es un algoritmo de aprendizaje automático supervisado que clasifica los datos encontrando una línea o hiperplano óptimo que maximiza la distancia entre cada clase en un espacio de N dimensiones."[24] "El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. 'De la mejor forma posible' implica el hiperplano con el margen más amplio entre las dos clases, representado por los signos más y menos en la siguiente figura."[47]6.37 Se ha aplicado el motor de radial dado a su buen rendimiento con *datasets* de alta dimensionalidad, el SVM se suele usar en las tareas de **NLP** como el análisis de sentimientos.[24] A pesar de ello, debido a que nativamente SVM no dispone de una clasificación multi-clase6.6.11, por defecto se basa en una clasificación binaria 6.6.11. En el caso de la clasificación multi-clase se tendrá que aplica el enfoque *one-to-one*6.6.11([1]).

6.6.17. Naive bayes

El algoritmo Naive bayes es un algoritmo de machine learning supervisado basado en el algoritmo del teorema de Bayes. Es un clasificador probabilístico que se usa con frecuencia en tareas de NLP. Ya que por ejemplo, si una palabra se suele encontrar en textos clasificados como positivos. Se puede asumir que probablemente cualquier texto que contenga esa palabra va a tender a ser un texto positivo ([17]) ([49]). Este algoritmo se descartó debido a su bajo

rendimiento en comparación a los otros modelos en la fase exploratoria y la poca consideración de las dependencias que pueden tener las palabras.

6.6.18. Generación de un diseño de comprobación

Para estos modelos se estableció una proporción del 10 % de los datos para crear el *dataset* de validación con el resto destinado al *dataset* temporal de *training_testing*. Éste *dataset* se usará como semilla para el generador de *tweets* y establece el vocabulario de *features* que se usarán en la fase de vectorización de datos. En el caso del *dataset* de entrenamiento y pruebas, se establece una proporción de 80 % destinado al *dataset* de entrenamiento y el otro 20 % al de prueba.

En el proceso de investigación de los modelos experimentó con utilizar el *dataset* sin aplicar ningún tipo de equilibrio. En el *dataset* desequilibrado se observó que debido al nivel de inmenso de desequilibrio los modelos entrenados resultaron inclinado en predecir que el resultado siempre era la clase neutra. Por lo cual los datos pasarán por un proceso de *undersampling* y *oversampling* en el que se comparará la diferencia entre ellos. En el caso del *undersampling* los datos de entrenamiento y prueba se reducirán las clases positiva y neutra al mismo nivel que la clase negativa. Para el oversampling se realiza una aumentación de datos al *dataset* de entrenamiento, en el caso del *dataset* de pruebas no se le aplicará la aumentación de datos para evitar el *data leakage*.

6.6.19. Generación de los modelos

En esta fase se definen los parámetros y se procede al entrenamiento de los modelos. En python se han entrenado 4 modelos, Regresión lineal L1 (**Lasso**), **Random forest**, **svm** y **xgboost**. Todos se le han presentado un *dataset* vectorizado en **TFIDF** y aplicando una *cross-validation*.

Para el modelo de *LASSO* se aplicó una *cross validation* de 10 *folds* y los siguiente parámetros6.38:

1. **penalty**: Establece la regularización del L1 y L2, Siendo valores intermedios una mezcla de los dos. En el caso de este modelo se ha implementado el L1.
2. **max_iter**: Establece el numero máximo de iteraciones de optimización. Se implementó un máximo de 1000 iteraciones.
3. **C**: Regula la fuerza de regularización. Cuanto más pequeño es el valor, más fuerte es la regularización. Para el *dataset* con *undersampling* se le aplicó un valor de $C = 0{,}01$ y para el *dataset* con *oversampling* para reducir el overfitting se aplicó un $C = 0{,}001$
4. **tol**: La tolerancia permitida en cuanto a las iteraciones de optimización. Se estableció un valor de 1×10^{-3}.

```
model = cuLogisticRegression(penalty='l1', max_iter=1000, C=0.001, tol=1e-3, verbose=1)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
```

Ilustración 6.38: Generación modelo *LASSO*

Para el modelo de *Random forest* se aplicó una *cross validation* de 10 *folds*, 5 repeticiones y los siguiente parámetros6.41:

1. **n_estimators**: Establece el numero de arboles. En este caso se utilizaron 1000 arboles en el entrenamiento.
2. **max_depth**: Controla la cantidad máxima de la profundidad de los arboles. Más profundo implica arboles más complejos aumentando la posibilidad de *overfitting*. Tras múltiples iteraciones de entrenamiento se observó que para este caso una profundidad de 30 resultaba en la menor cantidad de *overfitting*.
3. **min_samples_split**:Establece la cantidad mínima de muestras necesarias para poder dividir un nodo. Se estableció una mínima de 2 muestras.
4. **min_samples_leaf**: Establece la cantidad mínima de muestras que deben tener las hojas de un árbol decisivo. Siendo de valores pequeños más precisos y los valores grandes más generalizados. Se suele ver como buena práctica ir ascendiendo en valores. En este caso se fijó un mínimo de 4 muestras.

```
model = cuRF(n_estimators=1000, max_depth=30,
             min_samples_split=2, min_samples_leaf=4)

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
```

Ilustración 6.39: Generación modelo *Random forest*

Para el modelo de *SVM* se aplicó una *cross validation* de 5 *folds* y los siguiente parámetros6.40:

1. **C**: regulariza el *training error*, un valor más pequeño permite mejorar la generalización de los datos. después de múltiples iteraciones de entrenamiento se observó que para este caso una regularización de 0,01 resultaba en la menor cantidad de *overfitting*.

```
model = svm_model = LinearSVC(C=0.01, penalty='l1')
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X_train.get(), y_train.get(), cv=cv, scoring='accuracy')
```

Ilustración 6.40: Generación modelo *SVM*

Para el modelo de *XGBoost* se aplicó una *cross validation* de 5 *folds* y los siguiente parámetros6.41:

1. **nrounds**: Determina la cantidad de rondas aumentadas. Se establece unas 1500 rondas.
2. **max depth**: Profundidad máxima de cada árbol. Según la profundidad se puede evitar el riesgo del *overfitting* y *underfitting*. Se observó que para este caso el valor más adecuado era una profundidad máxima de 4.
3. **eta**: Conocido como *shrinkage factor* y *learning rate* este parámetro permite equilibrar entre la rapidez de su aprendizaje y el rendimiento. Se vio que un *learning rate de 0,3* daba un buen balance en cuanto al rendimiento.
4. **gamma**: Establece una penalización por cada división de los nodos finales evitando posible *overfitting*. En este caso se fijó un valor de 0.
5. **subsample**: Similar al **colsample** pero con las filas. Permitiendo entre un rango de 0 a 1 siendo el 1 un uso completo de las filas. Se observó que en este caso un valor de 0,5 resultada el más adecuado.

```
params = {
    'objective': 'multi:softmax',
    'num_class': 3,
    'max_depth': 4,
    'eta': 0.3,
    'gamma': 0,
    'subsample': 0.5,
    'device': 'gpu'
}
```

```
start_time = time.time()
cv_results = xgb.cv(params, dtrain, num_boost_round=1500, nfold=5,
                    metrics={'merror', 'mlogloss', 'auc', 'aucpr'},
                    seed=42, verbose_eval=True)
end_time = time.time()
print(f"Duración de entrenamiento: {end_time - start_time} segundos.")

best_num_boost_round = cv_results['test-merror-mean'].idxmin()
final_model = xgb.train(params, dtrain, num_boost_round=best_num_boost_round)

y_train_pred = final_model.predict(dtrain)
y_test_pred = final_model.predict(dtest)
```

Ilustración 6.41: Generación modelo *XGBoost*

6.6.20. Evaluación del modelo

En esta fase, su enfoque es "la evaluación de los resultados, el proceso de revisión y determinación de los pasos siguientes"[23]. Esta es la fase en la que medimos el rendimiento de los modelos en base de las métricas del resultado de cada modelo.

6.6.21. Evaluación resultada

Al finalizar el entrenamiento de los modelos se termina con una serie de medidas estadísticas que se usará para su evaluación. Entre ellos se tendrán las siguientes medidas[30] [9]:

1. **Class**: Representan las clases de los 3 sentimientos. Siendo 0 el sentimiento positivo, 1 el negativo y 2 el neutro.
2. **Accuracy**: La precisión encontrada entre todas las clases en su total.
3. **Precision**: La proporción de predicción de las clasificaciones de verdaderos positivos para una clase.
4. **Recall**: La proporción de predicción de las clasificaciones de verdaderos positivos en todas las clases.
5. **F1_score**: Es la media de la precisión y el *recall*. Dispone de un rango de 0 a 1 siendo 1 el valor favorable.
6. **Support**: El numero de muestras disponibles en esa clase.

6.6.22. Evaluación *SVM*

Para el modelo SVM, se entrenó con el *dataset* sujeto a un proceso de *undersampling* y *oversampling*. Primero, se observan las métricas obtenidas usando el *dataset* con *undersampling*. Comparando las métricas de entrenamiento y de pruebas6.6 6.7 se puede observar que las diferencias son aceptables con un bajo nivel de *overfitting*. Luego se miran las métricas obtenidas usando el *dataset* con *oversampling*. Después de comparar las métricas de entrenamiento y de pruebas6.8 6.9 se puede observar que se está manifestando un nivel alto de *overfitting*, donde es especialmente notable en la métrica de *Recall*. Finalmente se puede ver que con un *dataset* balanceado por *undersampling*, ofrece una exactitud del 63 % en comparación al 55 % del *oversampling* que fue gravemente afectado por el *overfitting*.

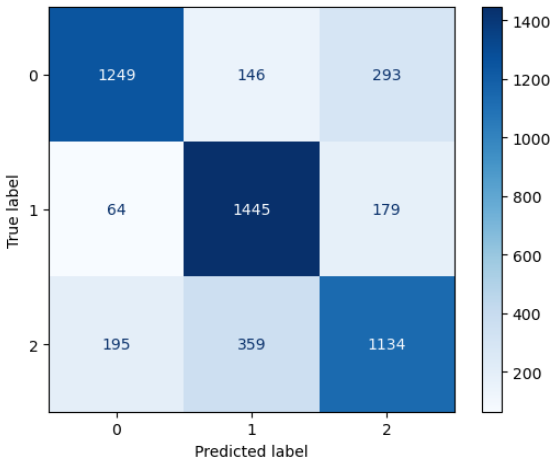

Ilustración 6.42: *Undersampled train SVM Confusion Matrix*

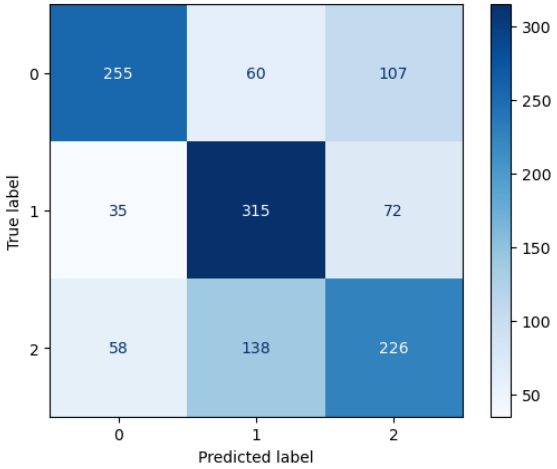

Ilustración 6.43: *Undersampled test SVM Confusion Matrix*

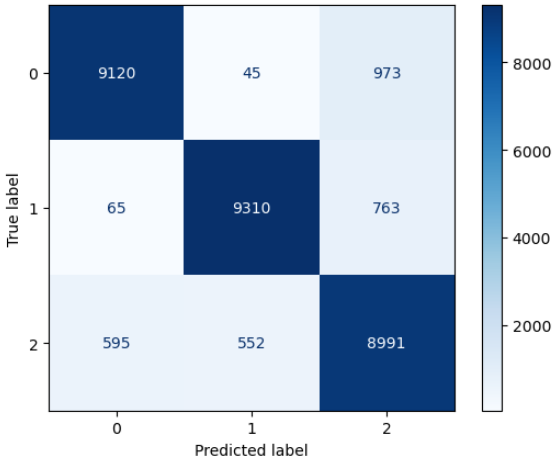

Ilustración 6.44: *Oversampled train SVM Confusion Matrix*

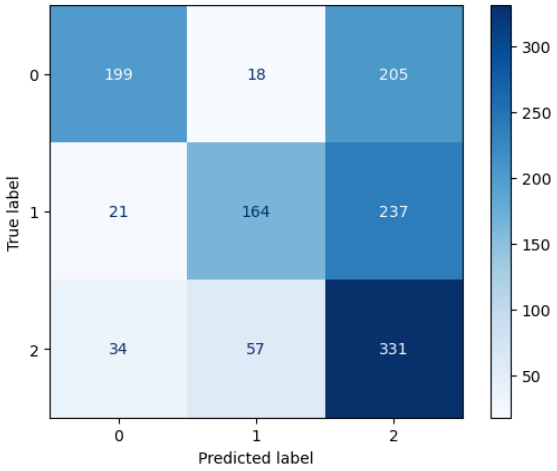

Ilustración 6.45: *Oversampled test SVM Confusion Matrix*

Class	Precision	Recall	F1-score	Support
0	0,83	0,74	0,78	1688
1	0,74	0,86	0,79	1688
2	0,71	0,67	0,69	1688
Accuracy			0,76	5064
Macro avg	0,76	0,76	0,75	5064
Weighted avg	0,76	0,76	0,75	5064

Cuadro 6.6: Métricas *SVM Undersampled Train*

Class	Precision	Recall	F1-score	Support
0	0,74	0,61	0,66	422
1	0,62	0,75	0,67	422
2	0,56	0,54	0,55	422
Accuracy			0,63	1266
Macro avg	0,64	0,63	0,63	1266
Weighted avg	0,64	0,63	0,63	1266

Cuadro 6.7: Métricas *SVM Undersampled Test*

Class	Precision	Recall	F1-score	Support
0	0,93	0,90	0,92	10138
1	0,94	0,92	0,93	10138
2	0,84	0,89	0,86	10138
Accuracy			0,90	30414
Macro avg	0,90	0,90	0,90	30414
Weighted avg	0,90	0,90	0,90	30414

Cuadro 6.8: Métricas *SVM Oversampled Train*

Class	Precision	Recall	F1-score	Support
0	0,78	0,47	0,59	422
1	0,69	0,39	0,50	422
2	0,43	0,78	0,55	422
Accuracy			0,55	1266
Macro avg	0,63	0,55	0,55	1266
Weighted avg	0,63	0,55	0,55	1266

Cuadro 6.9: Métricas *SVM Oversampled Ttest*

6.6.23. Evaluación *LASSO*

Para el modelo *LASSO*, se entrenó con el *dataset* sujeto a un proceso de *undersampling* y *oversampling*. Primero, se observan las métricas obtenidas usando el *dataset* con *undersampling*. Comparando las métricas de entrenamiento y de pruebas6.10 6.11 se puede observar que las diferencias son aceptables con un minúsculo nivel de *overfitting*. Luego se miran las métricas obtenidas usando el *dataset* con *oversampling*. Después de comparar las métricas de entrenamiento y de pruebas6.12 6.13 se puede observar que similar al *dataset* con *undersampling* se está manifestando un nivel muy bajo de *overfitting*. Finalmente se puede ver que con un *dataset* balanceado por *undersampling*, ofrece una exactitud del 58 % en comparación al 51 % del *oversampling*.

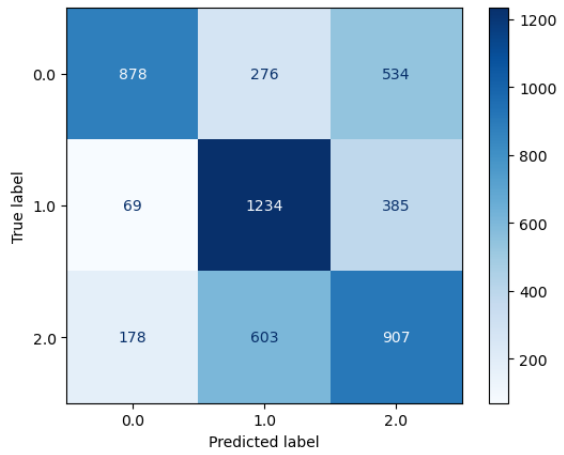

Ilustración 6.46: *Undersampled LASSO Train Confusion Matrix*

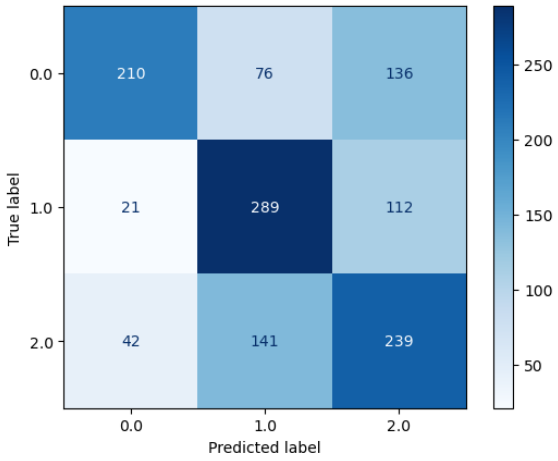

Ilustración 6.47: *Undersampled LASSO Test Confusion Matrix*

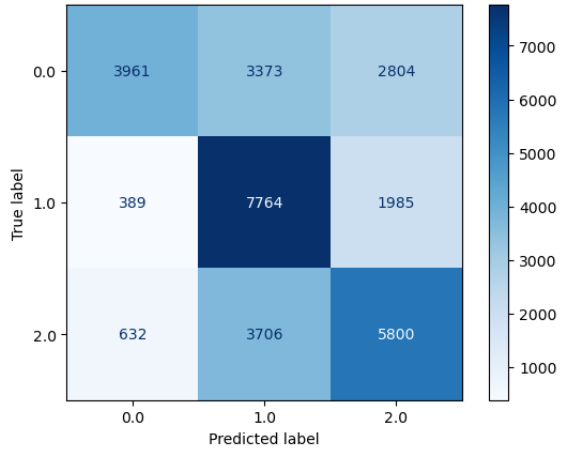

Ilustración 6.48: *Oversampled LASSO Train Confusion Matrix*

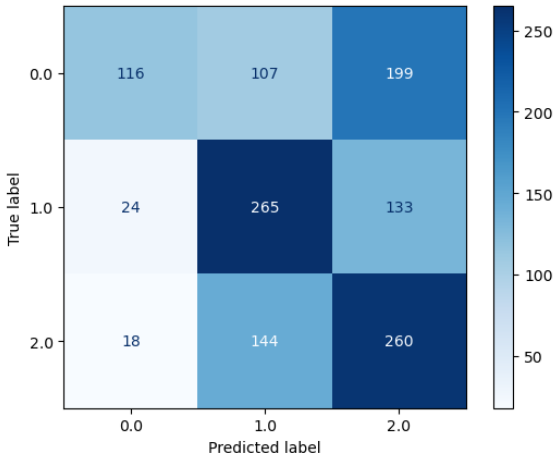

Ilustración 6.49: *Oversampled LASSO Test Confusion Matrix*

Class	**Precision**	**Recall**	**F1-score**	**Support**
0	0,78	0,52	0,62	1688
1	0,58	0,73	0,65	1688
2	0,50	0,54	0,52	1688
Accuracy			0,60	5064
Macro avg	0,62	0,60	0,60	5064
Weighted avg	0,62	0,60	0,60	5064

Cuadro 6.10: Métricas *LASSO Undersampled Train*

Class	Precision	Recall	F1-score	Support
0	0,77	0,50	0,60	1688
1	0,57	0,68	0,62	1688
2	0,49	0,57	0,53	1688
Accuracy			0,58	5064
Macro avg	0,61	0,58	0,58	5064
Weighted avg	0,61	0,58	0,58	5064

Cuadro 6.11: Métricas *LASSO Undersampled Test*

Class	Precision	Recall	F1-score	Support
0	0,80	0,39	0,52	10138
1	0,52	0,77	0,62	10138
2	0,55	0,57	0,56	10138
Accuracy			0,58	30414
Macro avg	0,62	0,58	0,57	30414
Weighted avg	0,62	0,58	0,57	30414

Cuadro 6.12: Métricas *LASSO Oversampled Train*

Class	Precision	Recall	F1-score	Support
0	0,73	0,27	0,40	422
1	0,51	0,63	0,57	422
2	0,44	0,62	0,51	422
Accuracy			0,51	1266
Macro avg	0,56	0,51	0,49	1266
Weighted avg	0,56	0,51	0,49	1266

Cuadro 6.13: Métricas *LASSO Oversampled Test*

6.6.24. Evaluación *Random forest*

Para el modelo *Random forest*, se entrenó con el *dataset* sujeto a un proceso de *undersampling* y *oversampling*. Primero, se observan las métricas obtenidas usando el *dataset* con *undersampling*. Comparando las métricas de entrenamiento y de pruebas6.14 6.15 se puede observar que las diferencias son aceptables con un nivel muy bajo de *overfitting*. Luego se miran las métricas obtenidas usando el *dataset* con *oversampling*. Después de comparar las métricas de entrenamiento y de pruebas6.16 6.17 se puede observar que se está manifestando un nivel medio alto de *overfitting*, donde es especialmente notable en la métrica de *Recall*. Finalmente se puede ver que con un *dataset* balanceado por *undersampling*, ofrece una exactitud del 60 % en comparación al 52 % del *oversampling* que fue notablemente afectado por el *overfitting*.

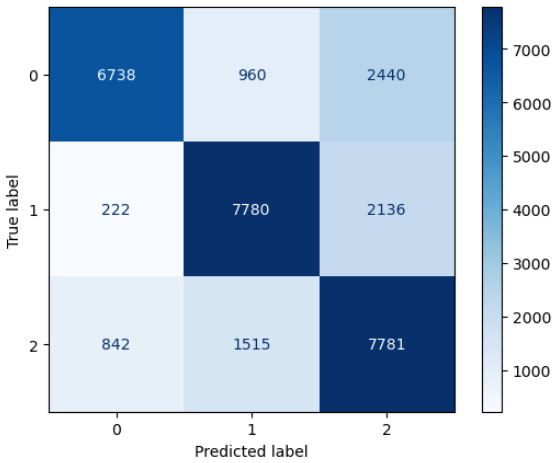

Ilustración 6.50: *Oversampled train Random forest Confusion Matrix*

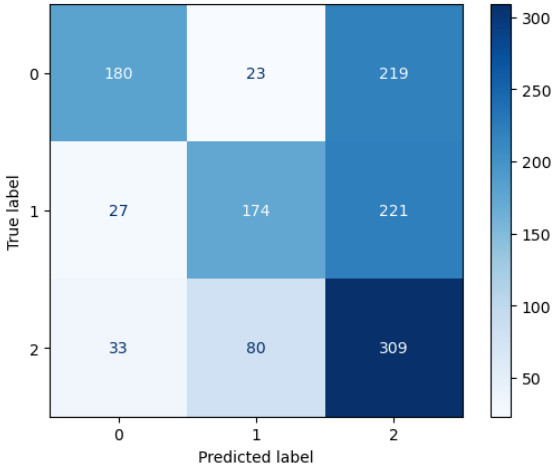

Ilustración 6.51: *Oversampled test Random forest Confusion Matrix*

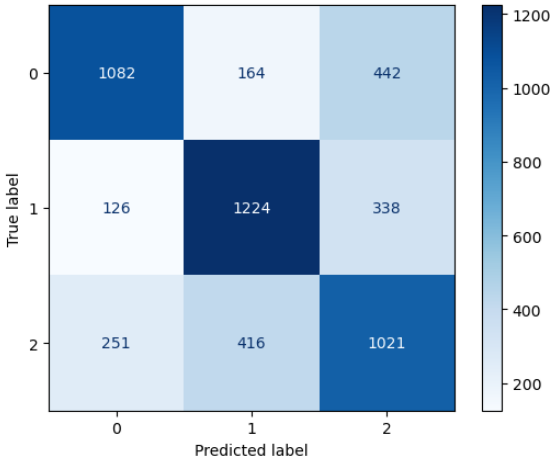

Ilustración 6.52: *Undersampled train Random forest Confusion Matrix*

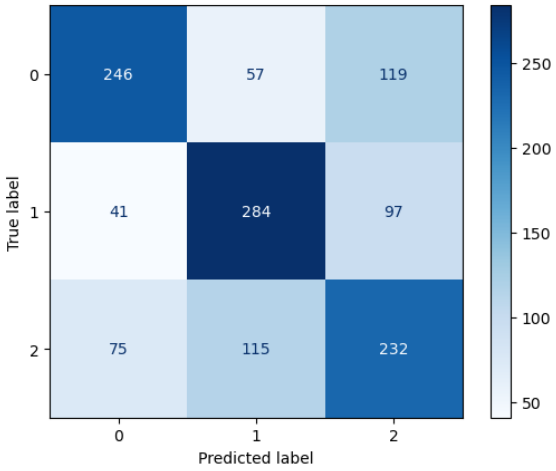

Ilustración 6.53: *Undersampled test Random forest Confusion Matrix*

Class	Precision	Recall	F1-score	Support
0	0,74	0,64	0,69	1688
1	0,68	0,73	0,70	1688
2	0,57	0,60	0,59	1688
Accuracy			0,66	5064
Macro avg	0,66	0,66	0,66	5064
Weighted avg	0,66	0,66	0,66	5064

Cuadro 6.14: Métricas *Random forest Undersampled Train*

Class	Precision	Recall	F1-score	Support
0	0,68	0,58	0,63	422
1	0,62	0,67	0,65	422
2	0,52	0,55	0,53	422
Accuracy			0,60	1266
Macro avg	0,61	0,60	0,60	1266
Weighted avg	0,61	0,60	0,60	1266

Cuadro 6.15: Métricas *Random forest Undersampled Test*

Class	Precision	Recall	F1-score	Support
0	0,86	0,66	0,75	10138
1	0,76	0,77	0,76	10138
2	0,63	0,77	0,69	10138
Accuracy			0,73	30414
Macro avg	0,75	0,73	0,74	30414
Weighted avg	0,75	0,73	0,74	30414

Cuadro 6.16: Métricas *Random forest Oversampled Train*

Class	Precision	Recall	F1-score	Support
0	0,75	0,43	0,54	422
1	0,63	0,41	0,50	422
2	0,41	0,73	0,53	422
Accuracy			0,52	1266
Macro avg	0,60	0,52	0,52	1266
Weighted avg	0,60	0,52	0,52	1266

Cuadro 6.17: Métricas *Random forest Oversampled Test*

6.6.25. Evaluación *XGBoost*

Para el modelo *XGBoost*, se entrenó con el *dataset* sujeto a un proceso de *undersampling* y *oversampling*. Primero, se observan las métricas obtenidas usando el *dataset* con *undersampling*. Comparando las métricas de entrenamiento y de pruebas 6.18 se puede observar que las diferencias son aceptables con un nivel muy bajo de *overfitting*. Luego se miran las métricas obtenidas usando el *dataset* con *oversampling*. Después de comparar las métricas de entrenamiento y de pruebas 6.18 se puede observar que se está manifestando un nivel medio alto de *overfitting*, donde es especialmente notable en la métrica de *Recall*. Finalmente se puede ver que con un *dataset* balanceado por *undersampling*, ofrece una exactitud del 58,8 % en comparación al 46,7 % del *oversampling* que fue notablemente afectado por el *overfitting*.

Dataset	Accuracy	Precision	Recall
Training Data Undersampled	0,657	0,674	0,657
Testing Data Undersampled	0,588	0,605	0,588
Training Data Oversampled	0,744	0,760	0,744
Testing Data Oversampled	0,467	0,576	0,467

Cuadro 6.18: Métricas para *Undersampled* y *Oversampled*

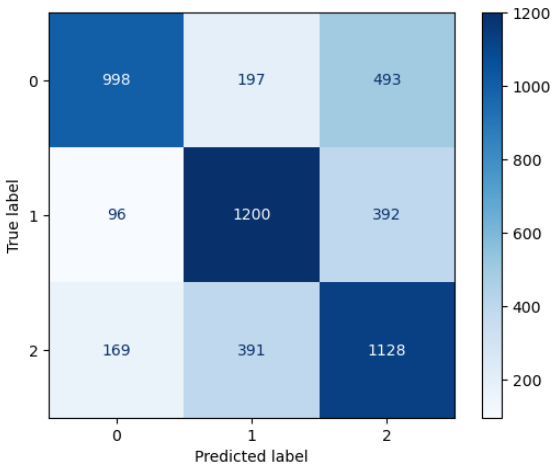

Ilustración 6.54: Métricas *XGBoost Undersampled Train*

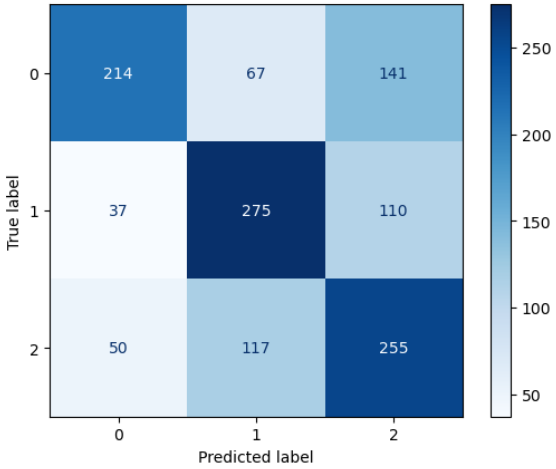

Ilustración 6.55: Métricas *XGBoost Undersampled Test*

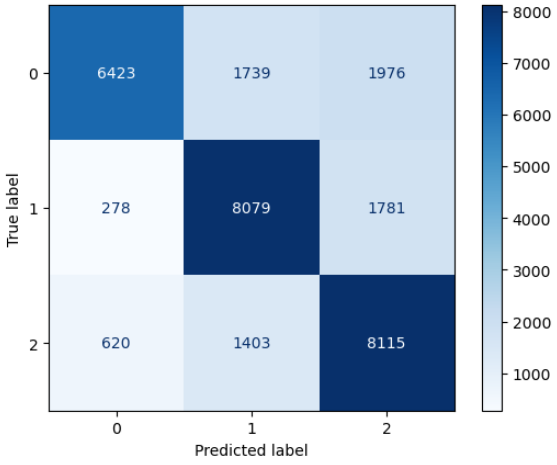

Ilustración 6.56: Métricas *XGBoost Oversampled Train*

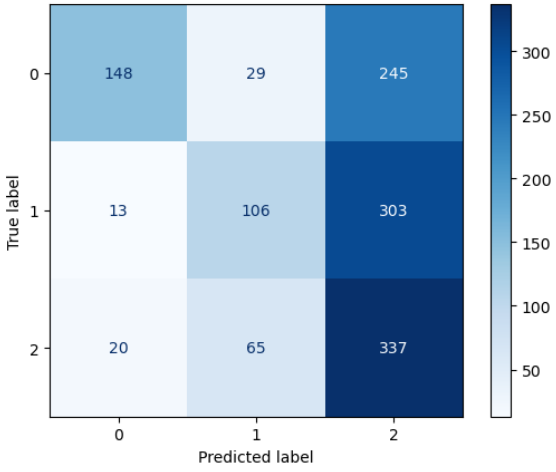

Ilustración 6.57: Métricas *XGBoost Oversampled Test*

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusión

Este trabajo comenzó con un análisis exploratorio de un *dataset* de *tweets* facilitados por el Centro de Innovación para la Sociedad de la Información (CICEI). En él hemos identificado las peculiaridades de los datos, como la naturaleza de las palabras que componen un *tweet* según su sentimiento. Como en el caso de los *tweets* positivos que mostraban un amplio uso de *emojis* y adjetivos calificativos. En el caso de los *tweets* negativos se podía ver que había un enfoque inclinado más hacia hablar de las personas y mascarillas. Luego, en el caso de los *tweets* neutros se podía observar que tenían un enfoque más informativo con el objetivo de comunicar algún tipo de información. También observamos el desequilibrio notable hacia los *tweets* neutros 6.2 que contiene este *dataset*.

Esto nos llevó a investigar posibles medidas que podríamos tomar para reducir este desbalance de clases. Para ello hemos optado por aplicar un balanceo en el que extraemos aleatoriamente unas muestras de cada clase hasta llegar a tener tantas muestras de las clases positivas y neutras como las que se ven en la clase negativa. La otra medida que aplicamos fue una aumentación de datos en la cual, según un fragmento de los textos de las clases positivas y negativa, se generaban *tweets* sintéticos de palabras con alta frecuencia en sus respectivas clases.

En aras de implementar los diferentes modelos de análisis de sentimiento convencionales, realizamos dichos análisis aplicando un enfoque basado en *lexicons* de origen inglés o traducidos al castellano, donde comparamos los *lexicons* **NRC**, **AFINN** y **ML-Senticon** para ver cuál de ellos ofrecen el mejor resultado. En la tabla 7.1 vimos que ninguno de estos *lexicons* por su cuenta fueron capaz de ofrecer una precisión favorable en cuanto a la identificación del sentimiento de un *tweet*. Por otro lado, se exploró el uso de las herramientas para el análisis de texto **QDAP** y **VADER**. En el caso del **QDAP** se experimentó con los *lexicons* **NRC**, **AFINN** y **ML-Senticon** donde vimos que hay una diferencia notable entre utilizar **QDAP** y la implementación manual. La gran ventaja que ofrece esta herramienta está en la capacidad de poder fijar palabras como las negaciones, amplificaciones y desamplificaciones.

Cuadro 7.1: Precisión según *lexicon*

Precisión %	**Positivo**	**Negativo**	**Neutro**	**Total**
NRC	25,85	14,36	6,35	46,56
AFINN	6,07	5,51	28,43	40,02
ML-Senticon	22,29	4,84	15,60	42,73

La gran desventaja que hemos visto en esta herramienta es su dificultad en identificar las palabras neutras.

En el caso de **VADER** no se vio mucha diferencia en cuanto a las metodologías basadas en *lexicons* ya probadas. Aun así, a pesar de tener una precisión total bastante similar a la del **QDAP**, el **VADER** ofrece una precisión mucho más equilibrada, además de su capacidad de tomar en cuenta el uso de *emojis* y puntuaciones sin ningún tipo de preprocesado. La desventaja que hemos visto con el **VADER** es que tiene una fuerte dependencia en que sea usado en un contexto inglés encontrado en redes sociales. Adicionalmente en cuanto a las opciones disponibles para el procesamiento de lenguaje natural, **VADER** ofrece el mejor rendimiento de las opciones disponibles. No obstante, estos resultados dependen mucho en los *lexicons* utilizados.

A continuación, exploramos distintos algoritmos de *machine learning* en los cuales evaluamos cuál de ellos podría ser el más adecuado para calcular automáticamente el sentimiento de textos provenientes de redes sociales.

Hemos explorado múltiples modelos de clasificación en los que optamos por usar LASSO, SVM, *Random forest* y XGBoost. Entre estos modelos el que mejor resultado 7.2 obtuvo en contexto de un dataset balanceado por *undersampling* fue LASSO con un 63 % de aciertos. Este porcentaje en un contexto de clasificación entre 3 posibles clases significa que el modelo mejora (duplica) los resultados que se podrían obtener con la hipótesis nula (clases fijas o al azar).

En cuanto a la aumentación de datos por medio del *oversampling* todos los modelos excepto el LASSO se encontraron con casos de *overfitting*.

Modelo	**Exactitud**
LASSO	63 %
SVM	58 %
Random Forest	60 %
XGBoost	58,8 %

Cuadro 7.2: Comparativa final de los modelos

Finalmente, hemos visto que el análisis de sentimiento de un texto no escasea de complejidad y su cálculo depende de múltiples factores, connotaciones culturales y contextuales, las cuales pueden invertir el significado de un texto. También hemos visto que el idioma tiene una importancia crítica para las herramientas de análisis de textos. Además, trabajar con

lexicons obtenidos por medio de traducciones desde un idioma más documentado, puede no ser lo ideal, pues se puede perder una gran cantidad de contexto.

El lenguaje es uno de los pilares de la humanidad, siendo una representación de las experiencias que hemos vivido como sociedad.

7.2. Nivel personal

Siendo un estudiante de la mención de ingeniería de software. Este trabajo fue mi primera experiencia realizando un análisis de sentimiento. Las metodologías aplicadas en la minería de datos tienen un aire bastante distinto a las metodologías de desarrollo del software. Además, que el proceso de investigación ha sido una experiencia que me ha cambiado de perspectiva en como uno puede afrontar un problema. En mis previos desarrollos de software, el análisis siempre ha sido nada más que un simple paso en el que se tiene caminar para llegar al objetivo. A pesar de que este trabajo también tenía fijado sus series de objetivos, había más lugar para contemplar posibles ideas alternas.

7.3. Trabajo futuro

Después de ver el gran potencial que ofrecen herramientas como **VADER**, podemos explorar el desarrollo de un *lexicon* castellano especializado en lenguaje digital. En cuanto al nivel de *machine learning* también podemos explorar posibles alternativas de preprocesamiento y vectorización en las que se ponen un mayor peso en ciertas características del texto.

Capítulo 8

Código disponible

El código de este TFG está compuesto de una serie de *jobs* y documentos dinámicos. Han sido desarrollados en R y python para los distintos modelos que fueron implementados en esta investigación con objetivo de determinar cuáles herramientas de análisis de texto y modelos ofrecía el mejor resultado. Este código ha sido entregado junto con esta memoria donde se podrá encontrar un *Readme.txt* con una breve explicación de los directorios y archivos utilizados. El *dataset* usado no será publicado excepto junto con la memoria para la evaluación del TFG.

Bibliografía

[1] baeldung (2024). Multiclass classification using support vector machines.

[2] boe, ley, proteccion, datos (2018).

[3] colaboradores de Wikipedia (2024). Overleaf.

[4] Corp., X. (2022a). About the x api.

[5] Corp., X. (2022b). X research for academics and marketers.

[6] Corporation, N. (2023). Welcome to cuML's documentation! — cuml 24.04.00 documentation.

[7] Corporation, N. (2024). What is XGBoost? .

[8] Cruz Mata, F., Troyano Jiménez, J. A., Pontes Balanza, B., and Ortega Rodríguez, F. J. (2014-09). Ml-senticon: un lexicón multilingüe de polaridades semánticas a nivel de lemas.

[9] DataHeadhunters (2024). Sensitivity vs Specificity: Performance Metrics in Classification.

[10] Diaz, G. (2016).

[11] Española, R. A. (2023). Diccionario de la lengua española. *REAL ACADEMIA ESPAÑOLA*.

[12] Estefanía Narrillos, Yasmina Yakimova, J. O. (2024). La eurocámara aprueba una ley histórica para regular la inteligencia artificial.

[13] Europeo, P. (2024). Resolución legislativa del parlamento europeo, de 13 de marzo de 2024, sobre la propuesta de reglamento del parlamento europeo y del consejo por el que se establecen normas armonizadas en materia de inteligencia artificial (ley de inteligencia artificial) y se modifican determinados actos legislativos de la unión (com(2021)0206 – c9-0146/2021 – 2021/0106(cod)).

[14] Foundation, P. S. (2024a). Python.

[15] Foundation, T. R. (2024b). The r project for statistical computing.

[16] Foundation, T. R. (2024c). What is r?

[17] Frank, E. and Bouckaert, R. R. (2006). Naive bayes for text classification with unbalanced classes. In *Knowledge Discovery in Databases: PKDD 2006: 10th European Conference on Principles and Practice of Knowledge Discovery in Databases Berlin, Germany, September 18-22, 2006 Proceedings 10*, pages 503–510. Springer.

[18] Fuster-Palà, A., Luna-Perejon, F., and Domínguez-Morales, M. (2024). Disease screening using artificial intelligence.

[19] GitHub (2024). Github.

[20] Goyal, C. (2024). Understanding multiclass Classification using SVM.

[21] Hutto, C. and Gilbert, E. E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI.

[22] IBM (2021a). The data mining life cycle.

[23] IBM (2021b). Introducción al crisp-dm.

[24] IBM (2023). What is support vector machine? — ibm.

[25] IBM (2024a). What is lasso regression? — IBM.

[26] IBM (2024b). ¿Qué es un bosque aleatorio? — IBM.

[27] jvera (2017). Spanish stopwords for tidytext package.

[28] Kolodny, L. (2023). Elon musk says twitter, now x, is moving to monthly subscription fees and has 550 million users.

[29] Kralj Novak, P., Smailović, J., Sluban, B., and Mozetič, I. (2015). Emoji sentiment ranking 1.0. Slovenian language resource repository CLARIN.SI.

[30] Kuhn, M. (2019a). 17 Measuring Performance — The caret Package.

[31] Kuhn, M. (2019b). The caret Package.

[32] ley, politica, privacidad, twitter (2023).

[33] Liu, B. (2022). *Sentiment analysis and opinion mining*. Springer Nature.

[34] Microsoft (2021). Visual Studio Code - Code editing. Redefined.

[35] Mohammad, S. M. (2022). Nrc emotion lexicon.

[36] Murzone, F. (2020). Procesamiento de lenguaje natural: Stemming y lemmas.

[37] Overleaf (2024). Overleaf, online latex editor.

[38] Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135.

[39] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011a). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[40] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Édouard Duchesnay (2011b). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

[41] Posit (2024). Rstudio ide the most trusted ide for open source data science.

[42] proteccion, datos (2020).

[43] Rinker, T. W. (2023). qdap.

[44] Singh, T. and Kumari, M. (2016). Role of text pre-processing in twitter sentiment analysis. *Procedia Computer Science*, 89:549–554. Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India.

[45] Stokel-Walker, C. (2024). Under elon musk, x is denying api access to academics who study misinformation.

[46] Teslenko, D., Sorokina, A., Khovrat, A., Huliiev, N., and Kyriy, V. (2023). Comparison of dataset oversampling algorithms and their applicability to the categorization problem. *Innovative Technologies and Scientific Solutions for Industries*, pages 161–171.

[47] The MathWorks, I. (2024). Support vector machine (svm).

[48] Tidyverse (2024). Tidyverse.

[49] Troussas, C., Virvou, M., Espinosa, K. J., Llaguno, K., and Caro, J. (2013). Sentiment analysis of facebook statuses using naive bayes classifier for language learning. In *IISA 2013*, pages 1–6.

[50] Wang, X., Wei, F., Liu, X., Zhou, M., and Zhang, M. (2011). Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *Topic sentiment analysis in Twitter: A graph-based hashtag sentiment classification approach*, pages 1031–1040.

[51] Wankhade, M., Rao, A. C. S., and Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7):5731–5780.

[52] xgboost developers (2022). Xgboost documentation — xgboost 2.0.3 documentation.

[53] y Sara Lana-Serrano y Eugenio Martínez-Cámara y José Carlos González-Cristóbal, J. V.-R. (2013). Tass - workshop on sentiment analysis at sepln. *Procesamiento del Lenguaje Natural*, 50(0):37–44.

[54] Zimbra, D., Abbasi, A., Zeng, D., and Chen, H. (2018). The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. *ACM Trans. Manage. Inf. Syst.*, 9(2).