FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2020/2021

Trabajo de Fin de Máster

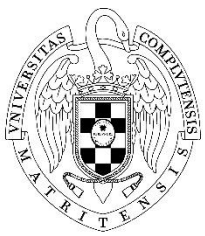
TÍTULO: Estudio de la Percepción Pública de la Vacuna contra la COVID-19 mediante Técnicas de PLN y de Aprendizaje Automático

***Alumno*: Daniel Povedano Álvarez**

***Tutores*: Javier Portela García-Miguel**

Esteban Alejandro Armas Vega

Julio de 2021



UNIVERSIDAD COMPLUTENSE
MADRID

Agradecimientos

Primero, quiero agradecer a Javier Portela García-Miguel y a Esteban Armas Vega los directores de este trabajo, todo el apoyo que me han proporcionado para la realización del mismo. Segundo, agradecer a los miembros de mi familia, a mis padres y amigos, (en especial a Alejandra y a Ana) que me apoyaron y animaron durante todo el trabajo. Por último, agradecer a todos aquellos equipos de desarrolladores que han elaborado las herramientas y documentación necesarias para la comunidad y que han servido como soporte para la realización de este trabajo.

Índice General

Índice de Figuras

Índice de Tablas

Resumen

La pandemia de COVID-19 causada por el nuevo coronavirus SARS-CoV-2 ha tenido un impacto significativo en la sociedad, tanto por los graves efectos sanitarios y económicos como por los efectos de las medidas sanitarias para evitar su propagación. Gracias a las técnicas de PLN se ha podido investigar las actitudes hacia la vacunación, siendo particularmente oportuno en estos momentos ante la llegada de las vacunas contra la COVID-19. Este trabajo tiene una doble finalidad, por un lado estudiar la percepción hacia la vacunación contra la COVID-19, mediante técnicas de Procesamiento de Lenguaje Natural y por otro, la construcción de un clasificador de sentimientos interpretable mediante técnicas de Aprendizaje Automático. Para ello se utilizaron 4.000.000 de *tweets* relacionados con la vacunación en el periodo comprendido entre el 15 de noviembre de 2020 y el 16 de diciembre de 2020 como conjunto de datos. El periodo de tiempo seleccionado es esencial porque durante este tiempo se publicaron los primeros resultados de las vacunas contra la COVID-19, como Pfizer y Moderna, surgiendo un debate público. El análisis de la percepción sugiere que hay un número significativo de *tweets* negativos que pueden poner en peligro el objetivo de alcanzar la inmunidad de rebaño. En cuanto a los resultados del clasificador de sentimientos multiclase se obtuvo un 92% de ROC-AUC con el algoritmo *LinearSVC*.

Palabras clave

Análisis de Sentimientos, Análisis de Texto, Covid-19, Machine Learning, Procesamiento del Lenguaje Natural, Redes Sociales, Vacunación.

Abstract

The COVID-19 pandemic caused by the new SARS-CoV-2 coronavirus has had a significant impact on society, both in terms of serious health and economic effects and in terms of the effects of health measures to prevent its spread. Thanks to Natural Processing Language techniques, it has been possible to investigate attitudes towards vaccination, which is particularly timely at this time in view of the arrival of vaccines against COVID-19. This work has a dual purpose: on the one hand, to study the perception towards vaccination against COVID-19, using PLN techniques, and on the other hand, the build of a sentiment classifier explainable using Machine Learning techniques. For this purpose, 4,000,000 tweets related to vaccination in the period between 15 November 2020 and 16 December 2020 were used as a dataset. The selected period is essential because during this time the first results of COVID-19 vaccines, such as Pfizer and Moderna, were published, sparking a public debate. Perception analysis suggests that there are a significant number of negative tweets that may put herd immunity at risk. As for the sentiment classifier results, a 92% ROC-AUC was obtained with the LinearSVC algorithm.

Keywords

Covid-19, Machine Learning, Natural Language Processing, Sentiment Analysis, Social Networks, Text Analysis, Vaccination.

Lista de Acrónimos

ANN	Artificial Neural Network
API	Application Programming Interfaces
AS	Análisis de Sentimientos
AUC	Area Under The Curve
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
COVID	Coronavirus Desease
CPU	Central Processing Unit
CSV	Comma Separated Value
CV	Count Vectorizer
DAL	Dictionary of Affect in Language
DL	Deep Learning
EE. UU	Estados Unidos
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Units
HDF5	Hierarchical Data Format 5
HDP	Hierarchical Dirichlet Process
HPV	Human Papilloma Virus
HTML	HyperText Markup Language
IA	Inteligencia Artificial
IDF	Inverse Document Frecuency

IoT	*Internet of Things*
JSON	JavaScript Object Notation,
KNN	K-Nearest-Neighbour
LDA	*Latent Dirichlet Allocation*
LIME	Local Interpretable Model-Agnostic Explanations
LSA	Análisis Semántico Latente
LTSM	Long Short-Term Memory
MERSCoV	Middle East Respiratory Syndrome Coronavirus
ML	Machine Learning
NB	Naïve Bayes
NER	Name Entity Recognizer
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
OMS	Organización Mundial de la Salud
PCA	Principal Component Analysis
PLN	Procesamiento de Lenguaje Natural
POS	Part of Speech
POSIX	Portable Operating System Interface
RAM	Random Access Memory
RP	Random Projections
RBF	Radial Basis Function
RF	Random Forest
RFE	Recursive Feature Elimination
RFECV	Recursive Feature Elimination Cross Validation
RL	Regresión Logística
ROC	Receiver Operating Characteristics

SARS-CoV-2	Coronavirus del Síndrome Respiratorio Agudo Severo 2
SEMMA	Sample, Explore, Modify, Model, Access
SHAP	Shapley Additive exPLanations
SVD	Single Value Descomposition
SVM	Support Vector Machine
TF	Term frequency
TF-IDF	Term frequency – Inverse document frequency
TN	True Negative
TP	True Positive
TPR	True Positive Rate
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UTF-8	8-bit Unicode Transformation Format
VADER	Valence Aware Dictionary and Sentiment Reasoner
XAI	Explainable Artificial Intelligence
XML	Extensible Markup Language

1. Introducción

1.1. Justificación del Proyecto

La pandemia causada por el coronavirus del síndrome respiratorio agudo severo 2 (SARS-CoV-2) ha infectado a más de 180 millones de personas en más de 200 países[1]. La COVID-19, enfermedad producida por el virus SARS-CoV-2, se está extendiendo actualmente de forma masiva por todo el mundo y está causando millones de infecciones y muertes entre la población humana. El SARS-CoV-2 se detectó en China a finales deg 2019 y acabó infectando a millones de personas. Sin embargo, las nuevas tendencias tecnológicas, especialmente el papel de la informática, han demostrado con creces su contribución en la medicina, como enfermedades infecciosas y brotes [AZZA20]. Entre las fuentes actuales y realmente asequibles para obtener estos datos se encuentran las plataformas de los medios sociales, que proporcionan una cantidad ingentes de datos. Estos datos sirven como base para llevar a cabo la minería de opinión y el análisis de sentimientos.

Debido al avance de nuevas variantes[2], se hace necesario concienciar al público, en especial a los más jóvenes, sobre la vacunación, a los funcionarios gubernamentales y a los responsables de las políticas de salud pública para que implementen estrategias de comunicación, educación y aplicación de políticas más eficaces para llegar al público objetivo. Actualmente no existe ningún tratamiento curativo para la infección por COVID-19 [Sye21]. Por ello, se necesita urgentemente una vacunación segura y eficaz para contener la pandemia, que ha tenido repercusiones médicas, sociales y económicas devastadoras. Hasta la fecha se han desarrollado y aprobado varias vacunas para la inmunización de emergencia, como por ejemplo Pfizer-Biontech, Moderna, AstraZeneca entre otras. Los países y gobiernos de todo el mundo han gastado miles de millones de dólares en la compra de vacunas para inmunizar a la población de sus respectivos países.

Los programas de vacunación pueden conseguir la tan ansiada inmunidad de rebaño sin necesidad de que una proporción significativa de la población esté infectada, aunque dicha inmunidad requiere que se vacune una proporción suficiente de la población. Si bien la vacunación está efectivamente reconocida como una forma eficaz de reducir y eliminar la gravedad de la COVID-19, su eficacia depende de la voluntad de la población de vacunarse.

En este trabajo, se utilizan datos de la red social Twitter, recolectados entre el 15 de noviembre y el 16 de diciembre de 2020, para conocer y describir el sentimiento del público respecto a la vacunación contra la COVID-19 aplicando técnicas de Procesamiento del Lenguaje Natural (PLN) tales como el análisis de sentimientos, análisis de *n*-gramas, nubes de palabras y el modelado de temas. Debido al gran volumen de información recolectada, se hace inviable el etiquetado manual, por lo que los datos han sido previamente etiquetados con la librería VADER.

Además, se utilizan técnicas de ML para la construcción de un clasificador de sentimientos interpretable que sea capaz de identificar los sentimientos asociados a los *tweets* como negativos, neutros o positivos. El resto del trabajo se organiza como sigue: En el Capítulo 2 se presenta el marco teórico y las principales técnicas de PLN y ML. En el Capítulo 3 se estudian los principales antecedentes de la investigación así como las técnicas más recientes en la literatura. En el Capítulo 4 se describe la metodología utilizada para el desarrollo de la investigación. En los Capítulos 0 y 6 se presentan las contribuciones y los resultados y, finalmente, en el Capítulo 7, se presentan las conclusiones, las limitaciones y el trabajo futuro.

1.2. Objetivos

Este trabajo tiene dos objetivos principales, por un lado un estudio de la percepción de la vacunación contra la COVID-19 en Twitter (Objetivo I) para conocer y describir el sentimiento del público hacia la vacuna, y por otro, la construcción de un clasificador de sentimientos (negativo, neutro y positivo) interpretable (Objetivo II). En este sentido, se investigará si con un corpus de tamaño medio se puede construir un modelo con buena capacidad de generalización, interpretable y que simplifique la respuesta de los clasificadores basados en reglas (como VADER y *Textblob*), cuya respuesta es un valor numérico (polaridad) y no un valor categórico.

Para el Objetivo I, se pretende:

- Filtrar el conjunto de datos por palabras clave (*vaccine*, *corona vaccine*¸ etc.).
- Etiquetar el *corpus* con alguno de los Clasificadores de Sentimientos como VADER.
- Realizar un preprocesamiento del texto (limpieza de ruido, normalización, etc.).
- Realizar un análisis descriptivo de las principales variables de la API de Twitter.
- Extracción y ponderación de características.
- Ejecutar un análisis de *n*-gramas y un modelado de temas para extraer nuevos conocimientos o *insights*.

Para el Objetivo II, se pretende:

- Comparar el rendimiento entre los diferentes tipos de preprocesamiento, diferentes métodos de vectorización y diferentes subconjuntos de características.
- Búsqueda de los mejores parámetros para cada algoritmo de ML.
- Realizar una comparación entre los diferentes modelos obtenidos en base a su rendimiento y su interpretabilidad.
- Interpretabilidad del modelo seleccionado.

2. Marco Teórico

Este capítulo aborda los principales conceptos teóricos que aparecen en este trabajo. Se describe de forma breve que es la Inteligencia Artificial (IA) y de una forma más detallada, se repasan los conceptos y aplicaciones del Procesamiento del Lenguaje Natural (PLN). En la Sección 2.1 se da una visión general sobre IA. En la sección 2.2 se introducen las nociones básicas de PLN, revisando las principales características, los casos de uso y las técnicas de preprocesamiento de texto. En la Sección 2.3 se estudian los conceptos, aplicaciones y problemáticas del Análisis de Sentimientos. En la Sección 2.4 se explican brevemente los principales conceptos, técnicas de ML y métodos de vectorización así como los principales algoritmos de clasificación supervisada. En la Sección 2.5, se describe la API de Twitter y la aplicación *Hydrator* necesaria para recolectar los datos en bruto (*tweets*) procedentes de una lista de identificadores de *tweets*. Por último, en la Sección 2.6, se presenta la librería VAEX, necesaria para manejar grandes volúmenes de datos.

2.1. Inteligencia Artificial

"La Inteligencia Artificial (IA) es la nueva electricidad" - Andrew Ng

La IA se ha estudiado durante décadas y sigue siendo uno de los temas más complejos de abordar en la informática. La IA consiste en simular la inteligencia humana en máquinas programadas para pensar como los humanos e imitar sus acciones. Tiene aplicaciones en casi todos los ámbitos en los que utilizamos los ordenadores en la sociedad. A medida que la tecnología avanza, la investigación en el campo de la IA también crece, por lo que los anteriores puntos de referencia que definen la IA se están quedando obsoletos. Debido a estos avances [Fahm20], surgieron nuevos términos como Aprendizaje Automático, del inglés *Machine Learning* (ML) y el Aprendizaje Profundo, del inglés *Deep Learning* (DL). Pero a veces, hay solapamientos entre la IA, el ML y el DL, por lo que la diferencia entre ellos puede ser muy poco clara. En la Figura 2.1 se observa como ML es parte de la IA y el DL es parte del ML.

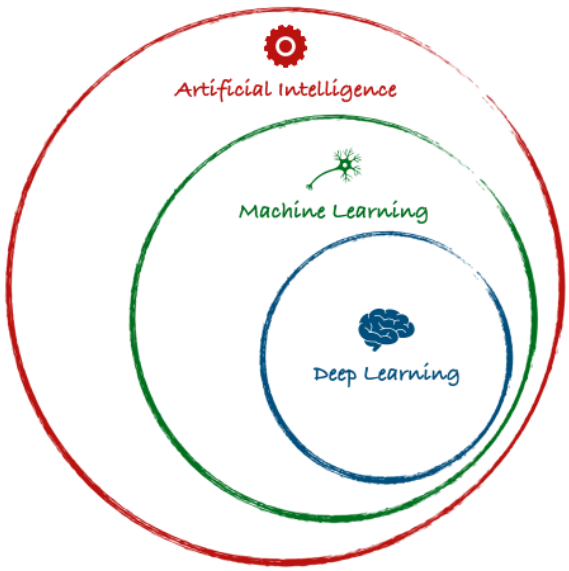


Figura 2.1: Inteligencia Artificial, Machine Learning y Deep Learning[3]

En sus inicios, los enfoques de la IA eran la lógica formal y los sistemas expertos. Estos métodos dominaban la IA en aquella época. Sin embargo, a raíz del desarrollo de la potencia de cálculo de los ordenadores, un mayor énfasis en la solución de problemas específicos, y también nuevos vínculos entre la IA y otros campos, surge un nuevo aprendizaje: el Aprendizaje Automático o ML. En este sentido, ML proporciona a los maquinas la capacidad de aprender y mejorar automáticamente, basándose en la experiencia. Estos sistemas transforman los datos en conocimiento o *insights* con el objetivo de mejorar la toma de decisiones en prácticamente cualquier ámbito. El ML está estrechamente relacionado con la estadística computacional, que se centra en hacer predicciones. La minería de datos también está relacionada con este estudio, enfocada más en el análisis exploratorio de datos. Debido a los grandes avances estos últimos años en ML, se ha podido aplicar estas técnicas para mejorar el desempeño en diversas áreas de conocimiento, como por ejemplo en el reconocimiento de imágenes, llamada Visión por Computador (del inglés *Computer Vision*) o en el campo del PLN (área en la que se centrará este trabajo) entre otros.

2.2. Procesamiento del Lenguaje Natural

2.2.1. Introducción

El PLN [KKKS17] es una vertiente de la IA (Figura 2.2) y la Lingüística dedicada a la comprensión por parte de los ordenadores, de los enunciados o palabras escritas en lenguaje natural, con el objetivo de obtener conocimientos a partir de datos en formato de texto .

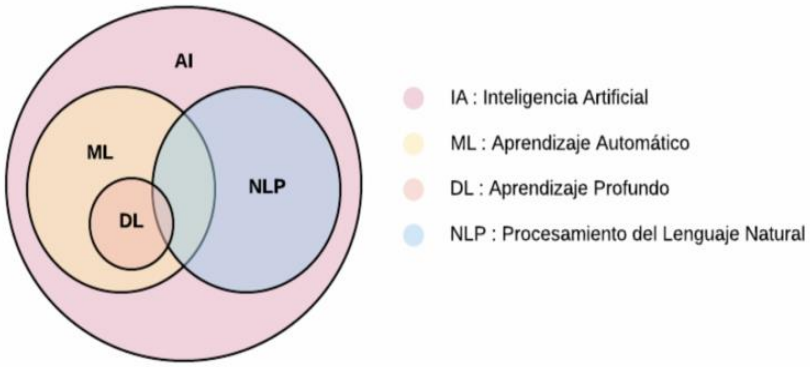


Figura 2.2: El PLN solapa con otros campos de estudio dentro de la IA[4].

La principal finalidad del PNL [SSAA19] es recopilar conocimientos sobre el modo en que los seres humanos entienden y utilizan el lenguaje, de modo que puedan desarrollarse las herramientas y técnicas adecuadas para que los sistemas informáticos entiendan e interactúen utilizando los lenguajes naturales para realizar diversas tareas.

Durante las primeras décadas de historia del PLN las computadoras emulaban la comprensión de lenguaje natural aplicando una colección de reglas elaboradas de forma manual. El aumento en el poder de computación y aprendizaje favoreció el uso de métodos estadísticos y probabilísticos, consiguiendo una gran mejora en los resultados. Además con el crecimiento de la web, *smartphones*, dispositivos IoT, cantidades ingentes de datos sin etiquetar empezaron a estar disponibles como datos de entrenamiento para modelos de aprendizaje semi y no supervisado. En el presente, gracias al DL se están produciendo grandes avances en el PLN con la aparición de modelos pre-entrenados como BERT (*transformers*) de Google [DCLT19]. El PLN se puede clasificar básicamente en dos grupos

[KKKS17] : Comprensión del Lenguaje Natural y Generación del Lenguaje Natural, que desarrolla la tarea de comprender y generar el texto (Figura 2.3).

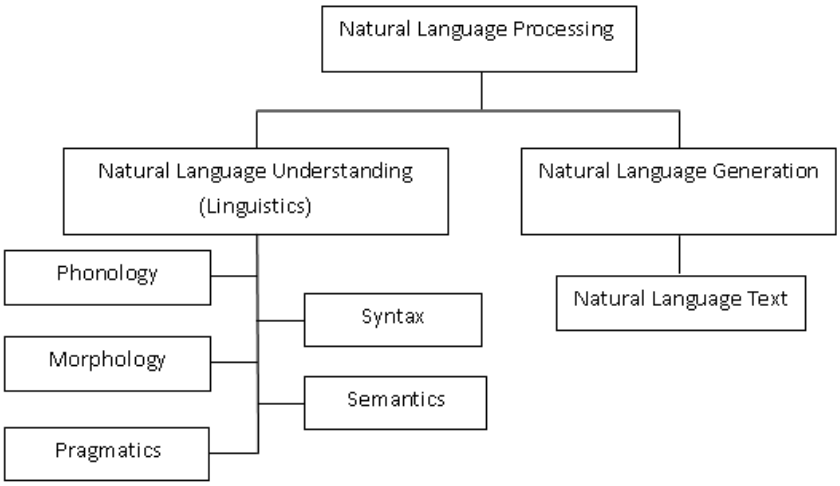


Figura 2.3: Clasificación general del PLN

La lingüística es la ciencia del lenguaje que estudia la fonología (sonido), la morfología (formación de la palabra), la sintaxis (la estructura de la oración), la semántica (significado) y la pragmática (comprensión).

Se puede definir el lenguaje como un conjunto, potencialmente infinito, de oraciones y secuencias de palabras construidas mediante las reglas gramaticales (en sus distintos niveles). Atendiendo su propósito y forma en que se originó, se pueden clasificar los lenguajes en dos grandes grupos: Lenguaje natural y Lenguaje formal.

Noah Chomsky [Chom14], uno de los primeros lingüistas del siglo XX, que inició las teorías sintácticas, marcó una posición única en el campo de la lingüística teórica ya que revolucionó el área de la sintaxis. Dicha área se puede clasificar a grandes rasgos en dos niveles Nivel Superior, que incluye el reconocimiento del habla, y Nivel Inferior, que corresponde al lenguaje natural.

Se puede definir la lengua natural [Laca14] como la lengua o idioma que nace espontáneamente de un grupo de hablantes por la mera necesidad de establecer comunicación verbal. Está ligado a la cultura de cada civilización y evoluciona según su uso por parte de su comunidad de hablantes, que debe ponerse de acuerdo en la manera de usar el lenguaje para que cualquier hablante del mismo pueda interpretar y producir mensajes con el mismo sentido en que se originó. Son los idiomas de uso común para expresar ideas y sentimientos: español, inglés, francés, latín, griego, etc. Es muy importante enmarcar el ámbito de empleo de la lengua natural en la comunicación humana. Por el contrario, los lenguajes formales, difieren de las naturales en que no han surgido espontáneamente, sino que han sido diseñados para un ámbito de aplicación concreto (normalmente en las ciencias) y se definen de manera que sean precisos y libres de cualquier ambigüedad. Ejemplos de lenguas formales son:

- Lenguaje matemático y lógico
- Lenguajes de programación (C, Java, Fortran, ...)
- Lenguaje de la música

Algunas de las tareas investigadas de la PLN son la Resumen Automático, el Análisis de Sentimiento, el Análisis del Discurso, la Traducción Automática, la Segmentación Morfológica, el Reconocimiento de Entidades Nombradas, el Reconocimiento Óptico de Caracteres (OCR), el Etiquetado de Partes del Lenguaje (del inglés *Part Of Speech Tagging* o POS *Tagging*), etc. Algunas de estas tareas tienen aplicaciones directas en el mundo real, como la Traducción Automática, el NER, el OCR, etc.

El campo del PLN está relacionado con diferentes teorías y técnicas que tratan de resolver los problemas de la comunicación entre los humanos y las máquinas. La ambigüedad es uno de los principales problemas del lenguaje natural que se suele presentar en el nivel sintáctico, el cual tiene como subtareas el léxico y la morfología, que se ocupan del estudio de las palabras y de su formación. Cada uno de estos niveles puede producir ambigüedades que pueden resolverse mediante el conocimiento de la frase completa.

La corrección de ambigüedades no es una tarea trivial y por tanto es necesario un tratamiento exhaustivo en todas las fases [Laca14] (léxico, estructural, pragmático…). La polisemia de las palabras, diferentes recursos literarios (anáforas, elipsis, hipérbaton...), acentos extranjeros, regionalismos, errores ortográficos y la intención o carga emocional de las sentencias (ironía, sarcasmo, etc.), son algunos de los elementos que hay que tener en consideración al analizar cualquier texto. En algunos casos son difíciles de identificar incluso para las personas, lo que dificulta aún más su formalización y su posterior interpretación por parte de las máquinas. Los elementos básicos de cualquier técnica de PLN son [KKKS17]:

- **Análisis fonológico**: La fonología es la parte de la Lingüística que se estudia la ordenación sistemática del sonido.
- **Análisis morfológico o "léxico"**: Consiste en analizar las palabras que integran las sentencias para obtener sus lemas o raíces, unidades léxicas compuestas, rasgos flexivos, etc. La finalidad de cualquier análisis léxico o morfológico es examinar y dividir el texto que se va a analizar en una serie de componentes léxicos, también conocidos como *tokens* o símbolos, propios del lenguaje original en el que está escrito. En este sentido, hay que prestar especial atención a los separadores de los componentes léxicos definidos por la lengua en cuestión. Por ejemplo, en lenguajes formales como pueden ser los lenguajes de programación, los separadores léxicos pueden ir desde los espacios en blanco hasta los símbolos no representables, como los saltos de línea. En el caso de las lenguas naturales, los separadores léxicos suelen limitarse a los espacios en blanco y a los símbolos de puntuación. Otra de las tareas realizadas como parte del análisis léxico es el etiquetado morfológico. El etiquetado morfológico, también denominado POS *tagging*, permite asignar etiquetas morfológicas a los elementos léxicos identificados según su categoría gramatical. Estas etiquetas permiten denotar el tipo (artículo, sustantivo, verbo, adjetivo, adverbio, etc.), género, número, tiempo o modo de cada una de las palabras en cuestión.
- **Análisis sintáctico**: La finalidad de un analizador sintáctico (del inglés *parser*) es determinar si una secuencia de componentes léxicos o *tokens* se ajusta a una estructura gramatical, es decir, comprueba que una frase dada está bien construida según las reglas gramaticales del lenguaje analizado.
- **Análisis semántico**: El objetivo del análisis semántico es determinar, de forma inequívoca, el significado de las frases.
- **Análisis del discurso** Si bien la sintaxis y la semántica trabajan con unidades de longitud de frase, el nivel de discurso del PNL se centra en las unidades de texto más largas que

una frase, es decir, no identifica los textos de varias frases como simples secuencias de frases. En su lugar, el discurso se centra en las propiedades del texto en su conjunto que trasmiten un significado al relacionar las conexiones entre las sentencias que lo componen.

- **Análisis pragmático**: Se centra en el análisis del contexto donde se encuentra inmerso el texto analizado y en cómo éste influye en el significado del texto.

Este trabajo se centra en el primer y segundo componente mencionado anteriormente. La aplicación de estos componentes suele ser de manera secuencial y en el orden indicado en la Figura 2.4. Esto da lugar a que los resultados de un componente sean utilizados como entrada del siguiente componente.

Figura 2.4: Componentes básicos del análisis de textos

2.2.2. Principales Tareas y Aplicaciones

El PLN agrupa diversas técnicas para interpretar el lenguaje humano, desde métodos estadísticos y de ML hasta los enfoques basados en reglas y algorítmicos.

Las tareas básicas de PLN [Sas20] implican la simbolización y el análisis sintáctico, lematización/derivación, POS, detección del lenguaje e identificación de relaciones semánticas. En general, las tareas de PLN dividen el lenguaje en trozos o piezas elementales más cortas o *tokens*, con la idea de comprender las relaciones entre dichos *tokens* y explorar cómo funcionan estas piezas juntas para crear significado. Dichas tareas implícitas se emplean a menudo en recursos PLN de más alto nivel [KKKS17], [Sas20], como:

- **Categorización de contenido**: Un resumen del contenido del documento basado en la lingüística, incluyendo búsqueda e indización, alertas de contenido y detección de duplicación. Un ejemplo claro de la categorización de contenidos son los filtros de spam.
- **Descubrimiento y modelado de temas o del inglés *Topic Modeling***: Analiza cómo las palabras y frases se relacionan entre sí y automáticamente "selecciona o aprende" grupos de palabras que mejor caracterizan esos documentos. Estos conjuntos de palabras representan un tema o un tópico. Con la introducción de la minería de texto, se han realizado investigaciones para analizar temas y tendencias importantes en la recopilación de documentos. El uso de la Asignación Latente de Dirichlet (en inglés LDA) o la Descomposición en Valores Singulares (en inglés SVD) para el análisis de tendencias en la minería de texto es uno de los métodos de análisis de tendencias más precisos.
- **Extracción contextual**: Extrae de forma automática datos estructurados de fuentes basadas en texto.
- **Clasificación de texto**: La clasificación es una las tareas básicas en el análisis de texto y se utiliza ampliamente en una variedad de dominios y aplicaciones. La premisa de la clasificación es sencilla: dada una variable categórica objetivo, la finalidad es extraer los patrones existentes entre las instancias compuestas por variables independientes y su relación con el objetivo. Dado que el objetivo se da por adelantado, se dice que la clasificación es supervisada ya que se entrena un modelo para minimizar el error entre las categorías predichas y las reales en los datos de entrenamiento. Una vez que el modelo

de clasificación se ajusta, se asignan las etiquetas categóricas a las nuevas instancias basándose en los patrones detectados durante el entrenamiento.

- **Análisis de sentimiento**: Identificación de opiniones subjetivas y emociones en grandes volúmenes de texto, incluyendo minería de sentimiento. En la práctica, el Análisis de Sentimientos (AS) es una Clasificación de Texto. Esta última área está muy de moda hoy en día, ya que tiene diversas aplicaciones, como por ejemplo: medición de la satisfacción del cliente, el sentimiento hacia un producto o la identificación de sentimientos hacia un tema específico en las Redes Sociales, etc.
- **Conversión de habla a texto y de texto a habla o del inglés *Speech Recognition***. Transformación de comandos de voz en texto escrito y viceversa.
- **Resumen de documentos**: Generación automática de resúmenes de grandes volúmenes de documentos.
- **Traducción automática**: Traducción automática de texto o habla de un idioma a otro.
- **Reconocimiento de Entidades Nombradas o** NER: El NER tiene como objetivo localizar y clasificar en categorías predefinidas, como, organizaciones, personas, lugares y cantidades, las entidades nombradas encontradas en un texto dado.
- **ChatBots**.
- **Interfaces en lenguaje natural**.
- **Reconocimiento óptico de caracteres, del inglés Optical Character Recognition** (OCR)

En todos estos casos, la finalidad es convertir el texto en crudo del lenguaje y aplicar técnicas de lingüística computacional y algoritmos para transformar o enriquecer el texto para extraer conocimientos o *insights*.

2.2.3. Técnicas y Herramientas de Preprocesamiento

En esta sección se presentan las técnicas de preprocesamiento de PLN [Orei20]. Estás técnicas consisten en una serie de tareas enfocadas a preparar o a limpiar el *corpus* original (datos en bruto) para las diferentes tareas que ocupa la minería de texto. Al conjunto o colección de palabras en forma de texto no estructurado utilizado para entrenar los algoritmos de ML se le denomina *corpus*. En el contexto del PLN los siguientes términos se utilizan habitualmente [Pai20] y es importante su comprensión:

- ***Corpus***: Se denomina *corpus* al conjunto de documentos sobre el que se realiza el análisis. El plural del *corpus* es *corpora*, que también es su derivación latina, que significa "cuerpo". Cuando el corpus está etiquetado y estructurado adecuadamente, se denomina corpus etiquetado.
- "***Token***": Conjunto de fragmentos o trozos de caracteres que representa la mínima unidad en el análisis de texto.
- **Documento**: La representación escrita de una idea, concepto o diálogo se le denomina documento. Un documento está compuesto por varios *tokens*. Ejemplos de documentos son un *tweet*, un artículo de una publicación científica, etc.
- **Vocabulario**: El conjunto de tokens únicos que se obtienen como resultado al "tokenizar" nuestro *corpus* completo.

Como se ha mencionado anteriormente, el preprocesamiento de datos consiste en una serie de pasos que pueden aplicarse (o no) a una tarea determinada, pero que generalmente se engloban en las categorías generales de *tokenization*, *normalización* y *sustitución*.

En términos generales, el principal objetivo será obtener un *corpus* de texto bruto predeterminado y realizar sobre él algunos análisis y transformaciones básicas con el fin de obtener características que sean mucho más útiles para realizar alguna tarea analítica posterior más significativa. En nuestro caso, esta tarea será la construcción de un clasificador de sentimientos. Así pues, como se ha mencionado anteriormente, se tres componentes principales en el preprocesamiento de textos (Figura 2.5):

- Separar palabras del texto en entidades denominadas *tokens* (*Tokenization*)
- Normalización
- Sustitución o *noise removal*

Al establecer un marco para abordar el preprocesamiento, se debe tener presentes estos conceptos de alto nivel.

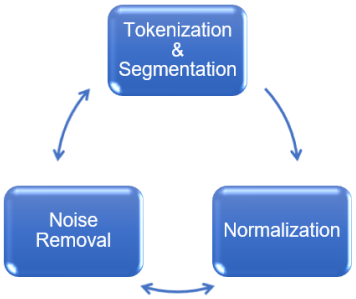


Figura 2.5: Etapas del preprocesamiento de textos.

En las siguientes secciones, se describen de forma más detallada las diferentes etapas que componen en el preprocesamiento.

2.2.3.1. Tokenization

La “tokenización” o *tokenization* es proceso de separar las cadenas de texto más largas en trozos más pequeños o *tokens*. Los trozos de texto más grandes pueden dividirse en frases, las frases pueden dividirse en palabras, etc. La manera de seleccionar los *tokens* del texto depende de la naturaleza del problema que se afronte, en algunas ocasiones una simple “tokenización”, como la de separar el texto por los espacios, puede ser suficiente (Figura 2.6)

```
text = 'Seguramente ya habrás escuchado algunas veces, o tu mismo has dicho \"Hey Google\" o
'\"Hey Siri\", o tal vez le has echado un ojo a tu filtro de spam...'"'

simple_tokens = text.split(" ")
print([token for token in simple_tokens])
```

```
['Seguramente', 'ya', 'habrás', 'escuchado', 'algunas', 'veces,', 'o', 'tu', 'mismo', 'has', 'dicho', '"Hey',
 'Google"', 'o', '"Hey', 'Siri",', 'o', 'tal', 'vez', 'le', 'has', 'echado', 'un', 'ojo', 'a', 'tu', 'filtro',
 'de', 'spam...']
```

Figura 2.6: Ejemplo simple de "tokenización"

El procesamiento posterior se realiza generalmente después de que un trozo haya sido debidamente “tokenizado”. El proceso de “tokenización” también es conocido como segmentación del texto o análisis léxico. A menudo, la segmentación se emplea para hacer referencia al desglose de un trozo de texto en piezas más grandes que las palabras (por ejemplo, párrafo u oraciones), mientras que la “tokenización” se limita al proceso de desglose que resulta exclusivamente en palabras. Este proceso de división en pequeños trozos o

palabras no es trivial, ya que dependiendo del separador utilizado para dividir las palabras de una frase, se tendrá un resultado u otro. Por ejemplo, se podría emplear una estrategia de segmentación que identifique (correctamente) un límite particular entre los *tokens* de la palabra como el apóstrofe en la palabra she's (una estrategia que "tokenice" sólo los espacios en blanco no serían suficiente para reconocer esto). Por lo tanto, dependiendo del idioma en el que se esté trabajando, el proceso de "tokenización" será diferente.

2.2.3.2. Limpieza de Ruido

Mientras que los procesos de "tokenización" y de normalización se aplican generalmente a casi cualquier trozo de texto o *token*, la eliminación de ruido es una sección mucho más específica.

La eliminación de ruido es una tarea de normalización específica del texto que suele producirse antes de la "tokenización". No es un proceso lineal, cuyos pasos deban aplicarse exclusivamente en un orden determinado. Por lo tanto, la eliminación del ruido puede producirse antes o después de las secciones anteriormente descritas o en algún punto intermedio.

Por ejemplo, si obtiene un *corpus* de internet alojado en un formato web sin procesar, seguramente el *corpus* contendrá texto envuelto en etiquetas HTML o XML.

Algunas de las tareas más habituales en la eliminación de ruido son (algunas específicas de la red social Twitter) [Gane19]:

- **Eliminar los encabezados y pies de página** de los archivos de texto.
- **Eliminar marcas y metadatos** de lenguajes como HTML, XML, etc.
- **Extraer datos valiosos de otros formatos**, como JSON o de bases de datos
- **Mapeo de contracciones/contracciones en expansión**: Las contracciones son una versión abreviada de palabras o un grupo de palabras, bastante común tanto en el lenguaje hablado como escrito. En inglés, y sobre todo en las redes sociales, son bastante comunes las contracciones *"I'm"*, *"didn't"*, *"haven't"*, etc. El mapeo de estas contracciones a su forma expandida ayuda en la estandarización de texto.
- **Eliminación de signos de puntuación**: Los signos de puntuación y caracteres especiales introducen ruido en el *corpus*, ya que en la mayoría de los mensajes no son relevantes a la hora de determinar la polaridad del mensaje. Además, dado que uno de los objetivos de este trabajo es la construcción de un clasificador de sentimientos mediante bolsas de palabras o BOW (explicada en la sección 2.4.2.2), es necesario reducir el número de palabras que formaran el conjunto de entrenamiento de los algoritmos de ML. Por lo que a efectos prácticos, términos como genial, genial!!!, genial!, etc., serán tratados de forma equivalente.
- **Eliminación números**, o en su defecto, conversión de números a texto.
- **Eliminación menciones, enlaces y hashtags**: Estos tres elementos son específicos de la red social Twitter. Las menciones hacen referencia a otros usuarios mediante su nombre de usuario (cuenta en Twitter) precedido del símbolo "@". Los *hashtags* son cadenas de texto con un significado asociado precedida del símbolo "#". Finalmente, los usuarios pueden añadir enlaces a sus mensajes. Debido a la gran variabilidad en el uso de estos elementos por parte de los usuarios de esta red social, se hace necesario realizar un tratamiento específico, ya sea sustituyendo estos elementos por espacios en blanco o por

algún tipo de carácter, ya que no parece que puedan ayudar a determinar el sentimiento de los expresado por usuarios en sus *tweets*.

- **Eliminación de *retweets***: Cuando un usuario quiere replicar un mensaje (*retweet*), el mensaje incorpora la palabra reservada "RT". Dicha palabra reservada no aporta ningún tipo de información al sentimiento del texto, por lo que es necesario eliminarlo.

Para llevar a cabo estas operaciones, suele emplearse de expresiones regulares (en este trabajo se ha empleado el módulo *re* de Python)

2.2.3.3. Normalización del texto

La normalización suele referirse a una serie de tares relacionadas entre sí con el objetivo de equiparar todo el texto, como por ejemplo: conversión de minúsculas o mayúsculas, eliminación de los signos de puntuación, conversión de números a sus equivalentes en palabras, etc. La normalización establece todas las palabras en igualdad de condiciones y permite que el preprocesamiento sea uniforme. Los caracteres y símbolos especiales contribuyen e la generación de ruido adicional en el texto no estructurado. Se recomienda usar expresiones regulares para eliminarlos como paso previo. A continuación se lista los principales métodos de normalización del texto [Gane19] [KCTH20]:

- **Revisión ortográfica:** Los documentos de un corpus son propensos a errores ortográficos. Realizar una limpieza del texto es una buena práctica y para ello se puede ejecutar un corrector ortográfico y corregir los errores ortográficos antes de continuar con los siguientes pasos. En determinados *corpus*, como puede ser el obtenido de una red social, no siempre este paso da buenos resultados ya que esta corrección puede introducir ruido adicional al tener que lidiar con expresiones propias de las redes sociales como '*omg'* ('*oh my god*') o '*lol*' ('*laughing out loud*').
- **Radicación o Stemming** [KGVI15]: Se llama *stemming* al proceso de conversión de una palabra a su raíz, es decir, permite eliminar afijos (sufijos, prefijos, infijos…) de una palabra para obtener la raíz de la misma. Estas raíces son la parte invariable de palabras relacionadas sobre todo por su forma. De cierta manera se parece a la lematización (como se verá posteriormente), pero los resultados (las raíces) no tienen por qué ser palabras de un idioma o valores lingüísticos válidos. Por ejemplo, el algoritmo de *stemming* puede decidir que la raíz de amamos no es "am-" (la raíz) sino "amam-" (cosa que desconcertaría a más de uno). Desde el punto de vista del procesamiento, el *stemming* es mucho más rápido que la lematización. Además, tiene como ventaja el reconocimiento de relaciones entre palabras de distinta clase. Podría reconocer por ejemplo, que picante y picar tienen como raíz pic-. En definitiva, el *stemming* puede reducir el número de elementos que componen nuestras oraciones. La principal desventaja del *stemming* es que su implantación es más simple que la lematización. Como principal desventaja es que pueden "recortar" demasiado la raíz y encontrar relaciones entre palabras que realmente no existen (*overstemming*). También puede suceder que deje raíces demasiado extensas o específicas, y que se tenga más bien un déficit de raíces (*understemming*), en cuyo caso palabras que deberían convertirse en una misma raíz no lo hacen.
- **Lematización o Lemmatization** [BaLl00]: La lematización está relacionada con la derivación, con la diferencia de que la lematización es capaz de capturar formas canónicas basadas en el lema de una palabra. Por ejemplo, la palabra "*better*" daría como resultado lo siguiente:

    - Better → Good
    - Was → Be
    - Studies → Study

  La lematización también ayuda a emparejar sinónimos mediante el uso de un diccionario de sinónimos, de modo que cuando uno busca "*hot"* (caliente) la palabra "*warm"* (cálido) también coincide. De la misma manera, una búsqueda de *car* producirá tanto *cars* como "*automobile"*. La técnica de lematización se ha utilizado en varios idiomas para la recuperación de información.

  Este proceso es más costoso computacionalmente que la radicación o *stemming*, pero suele dar mejores resultados ya que reduce las formas de las palabras a lemas válidos lingüísticamente hablando.
- Otros procesos de normalización de texto:
    - **Palabras vacías o Stopwords**: Las *stopwords* son aquellas palabras que normalmente se filtran antes de seguir procesando el texto (sobre todo en tareas como el Análisis de *n*-gramas o *Topic Modeling*), ya que estas palabras contribuyen poco al significado global, dado que suelen ser las más comunes en un idioma. Existen diversas librerías, como por ejemplo NLTK o Spacy que incorporan una lista bastante amplia de *stopwords*. Por ejemplo, "el", "y" y "a", aunque son palabras necesarias en un texto concreto no suelen contribuir en gran medida a la comprensión del contenido. Como ejemplo sencillo, el siguiente texto es igual de legible si se eliminan (palabras ~~tachadas~~) las stopwords:

      "The quick brown fox jumps over the lazy dog" ("El rápido zorro marrón salta sobre el perro perezoso.")
    - **Convertir todos los caracteres en minúsculas**. Aunque es fácil identificar que "casa" y "CASA" tienen el mismo significado, los algoritmos de ML las procesan como palabras diferentes. Para evitar esto, todos los textos del corpus serán convertidos a su equivalente en letras minúsculas.
    - **Tratamiento de la duplicidad de caracteres**: Es común, sobre todo en las redes sociales, la repetición de caracteres para dar intensidad a lo que se intenta explicar. Es importante realizar un tratamiento en estos casos con el objetivo de establecer relaciones entre términos que realmente son iguales, como por ejemplo: "calooooor", "calor", "caaaaaaaaalor", etc., son la misma palabra: 'calor'.
    - **Eliminar los espacios en blanco, normalización de jerga, conversión de emojis/emoticonos a texto**, etc.

Todas estas operaciones se pueden ejecutar con las librerías de Python de expresiones regulares (*re*), NLTK y *Spacy*. Gracias al preprocesamiento del texto, la dimensionalidad del conjunto de datos de entrenamiento se verá reducida y el Análisis de *n*-gramas o el *Topic Modeling* se verán beneficiados, ya que el objetivo principal es encontrar términos similares en el *corpus* para que el desempeño de estas tareas sea mejor, tanto en rendimiento como en tiempo.

2.2.4. Herramientas y Librerías

A continuación se listan los principales *toolkits* de PLN para Python [SSAA19]:

- ***Spacy***: Se trata de una biblioteca programada en Python cuyo objetivo es el procesamiento avanzado de lenguaje natural. Proporciona *pipelines* pre-entrenados y tiene soporte para más de 60 idiomas. Permite realizar etiquetados, "tokenización",

análisis o reconocimiento de entidades con nombre (NER) entre otras funcionalidades. En la última actualización (3.0) se han añadido soporte para trabajar con *transfomers* [WDSC19], un modelo de DL de vanguardia que utiliza un mecanismo de atención (una técnica que imita la atención cognitiva), sopesando la influencia de diferentes partes de los datos de entrada. Se utiliza principalmente en PNL pero investigaciones recientes también han desarrollado su aplicación en otras tareas como la comprensión de videos. Para poder utilizarlo en español basta con añadir el modelo para este idioma desde un programa Python.

- NLTK**:** Es un conjunto de bibliotecas y programas de PLN, en Python. Cuenta con distintas interfaces para poder trabajar con más de 50 corpus y recursos léxicos que proporciona. Está diseñado para utilizarse principalmente en inglés, pero tiene soporte para varios idiomas bien entrenándolo con archivos que incorpora el propio NLTK.
- ***Stanza* (*The Standford CoreNLP*)***: Stanza*, anteriormente conocido como *Stanford NLP*, es una biblioteca desarrollada en Java que ofrece diversas herramientas para tareas de PLN, así como el acceso a *CoreNLP*, una librería que proporciona tuberías o *pipelines* (Figura 2.7) con las que poder realizar operaciones sobre el texto de una forma más automatizada (“concatenando operaciones o métodos”).
- ***Scikit-Learn***: Librería de Python que proporciona métodos de vectorización (conversión de texto a número), como *Count Vectorizer* o *TF-IDF.*
- ***Gensim***: *Gensim* es una biblioteca de Python para modelado de temas, indexación de documentos y recuperación de similitudes con grandes *corpora*. Además, incorpora implementaciones de otros algoritmos como Latent Semantic Analysis (LSA)**,** Latent Dirichlet Allocation (**LDA**)**,** Random Projections (RP)**,** Hierarchical Dirichlet Process (HDP) or word2vec deep learning**.**

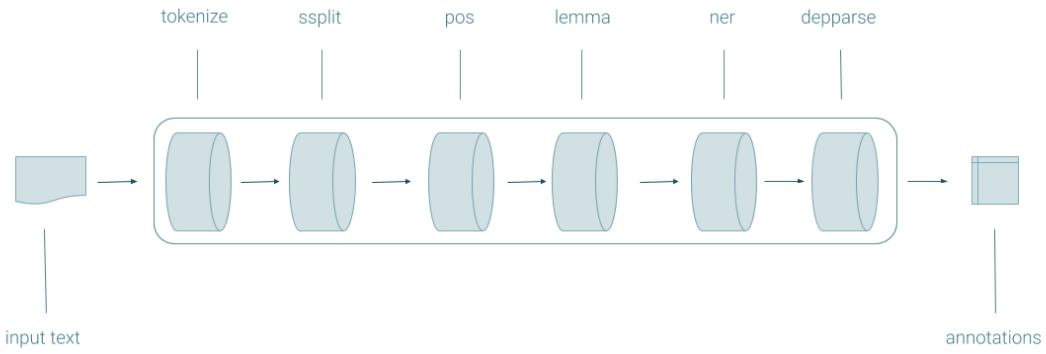


Figura 2.7: Estructura del pipeline de CoreNLP[5]

Finalmente, se presenta en la Tabla 2.1 las principales características de las librerías mencionadas anteriormente.

Tabla 2.1: Tabla resumen principales librerías PLN.

Librerías	Tokenización	Lematización	Stemming	POS Tagging	Soporte en español	Lenguaje
NLTK	Sí	Sí	Sí (Snowball, Porter, Lancaster, etc)	Si	Sí	C++ (API Python
Spacy	Sí	Sí	Sí	Si	Sí	Python
Stanza	Sí	Sí	No	Si	Sí	Java (API Python)
Gensim	Sí	Sí	Sí (Porter Stemming Algorithm)	Si	No del todo	Fortran/C (API Python)

2.3. Análisis de sentimientos

Una de las áreas de PLN con más auge en los últimos años es el Análisis de Sentimiento AS (o del inglés *Sentiment Analysis*). En esta sección se describirá los principales conceptos del AS, así como sus aplicaciones y sus limitaciones.

2.3.1. Introducción

El AS [LiLi12] es el área de estudio que analiza las opiniones, sentimientos, actitudes y emociones de las personas a partir del lenguaje escrito. En la actualidad es una de las áreas de investigación más activas en el PLN y ampliamente estudiada en la minería de datos, la minería web y la minería de textos. La creciente relevancia del AS coincide con el crecimiento de los medios sociales, como por ejemplo las reseñas, las discusiones en foros, los blogs, los *microblogs*, Twitter, etc. El auge del AS y de la IA van de la mano, ya que hoy en día se dispone de un gran volumen de datos de opinión en formato digital permitiendo a las grandes empresas aplicar estas técnicas en diferentes áreas. Los sistemas de AS se están aplicando en casi todos los sectores empresariales y sociales ya que las opiniones son fundamentales en casi todas las actividades humanas y son factores clave que influyen en nuestros comportamientos. En [YaSZ17] diferencian la minería de opinión u *opinion mining* de la minería de emociones u *emotion mining*. La minería de opinión u *opinión mining* implica el uso del PLN y el ML para determinar la actitud de una persona hacia un tema. La minería de emociones (*emotion mining*) también utiliza tecnologías similares pero se ocupa de detectar y clasificar las emociones de las personas hacia eventos o temas.

El AS se le conoce también como minería de opinión, extracción de opiniones, minería de sentimientos, análisis de subjetividad, análisis de afectos, análisis de emociones, minería de reseñas, etc. Sin embargo, en la actualidad todas se engloban bajo el paraguas del AS o la minería de opinión. Mientras que en la industria se utiliza más el término AS, en el mundo académico se emplean con frecuencia tanto el análisis de sentimientos como la minería de opinión. En [LiLi12], utilizan los términos AS y minería de opinión indistintamente. El análisis de sentimientos y la minería de opinión se centran en las opiniones que expresan o implican sentimientos positivos o negativos. La industria que rodea al AS también ha florecido debido

a la proliferación de aplicaciones comerciales. El PLN comenzó realizando tareas de clasificación a nivel de documento y el análisis se realizaba a nivel de frase. En los últimos años, [AgBM09] llevaron el PLN a un nuevo nivel utilizando la puntuación léxica con el análisis de *n*-gramas para modelar el efecto del contexto con el fin de realizar la polaridad a nivel de frase y el análisis de *n*-gramas. Teniendo en cuenta los avances en PLN y el análisis de textos, el AS se ha convertido en una herramienta cada vez más beneficiosa para explorar los patrones de actitud de los individuos hacia diferentes temas. Varios estudios han aplicado el AS para explorar las tendencias de sentimiento sobre la vacunación a través de medios sociales como Twitter. El AS de un documento puede realizarse en tres niveles [Katr19] en función a la granularidad, profundidad y detalle requeridos. Estos niveles son:

- **Análisis a nivel de documento**: Se estudia el sentimiento global de un documento como un conjunto indivisible, clasificándolo como positivo, negativo o neutro o empleando otro sistema de calificación.
- **Análisis a nivel de oración**: El documento se descompone en frases individuales y se extrae la opinión contenida en cada una de ellas. La opinión de cada frase puede ser categorizada como positiva, negativa o neutra o bien tomar un valor en base a cualquier otro tipo de medida.
- **Análisis a nivel de aspecto y entidad**: Es el nivel de análisis más detallado, en el que una entidad se compone por distintos elementos o aspectos y se expresa una opinión sobre cada uno de ellos, cuya polaridad puede ser diferente en cada caso.

2.3.2. Metodología y Tareas de Análisis de Sentimientos

En [AZZA20], el proceso de AS se divide en cuatro etapas (Figura 2.8):

- **Primer paso**: **Plataformas de medios sociales**. Esta primera fase se selecciona la fuente de donde se desea extraer los sentimientos. Por ejemplo, en el área de medios sociales se pueden elegir varias fuentes en línea, como Facebook, Twitter y Reddit ([ChHZ15]).
- **Segundo paso**: **Proceso de recogida de datos**. En esta fase se utilizan ciertas palabras clave o *hashtags* (es decir, #) para adquirir la información que se desea en función de las preferencias (en este trabajo se ha empleado tanto las palabras clave o *keywords* como las etiquetas o *hashtags*. Esta información tiene diferentes formas (por ejemplo, *tweets*, posts, noticias y textos) ([ChHZ15]).
- **Tercer paso**: **Preprocesamiento**. En esta fase se procesa la información extraída para preparar los datos para la siguiente fase. Esta etapa incluye la extracción de características (estructuras gramaticales y características de minería), la “tokenización” (proceso de convertir el texto en tokens antes de transformarlo en vectores) y limpieza (es decir, la normalización y limpieza de ruido).
- **Cuarto paso**: **Análisis de datos**. En esta fase, todos los datos “preprocesados” se utilizan para los fines previstos, como la identificación de la polaridad, el AS o análisis de frecuencia o *n*-gramas. Para la identificación del sentimiento existen dos enfoques principales: i) enfoque basado en el léxico y ii) basados en ML (ambos métodos se describirán en la sección 2.3.3).

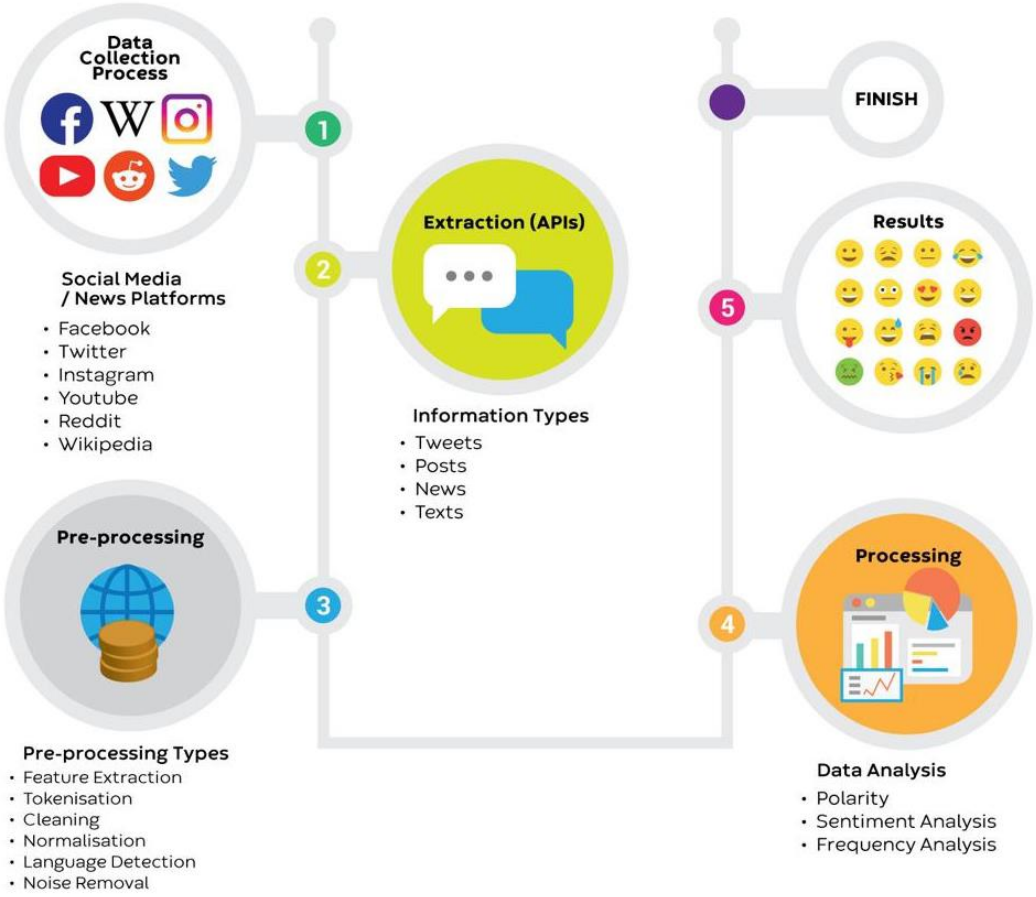


Figura 2.8: Flujo del Análisis de Sentimientos[6]

2.3.3. Clasificadores de Sentimientos

Como se ha comentado anteriormente, existen dos enfoques principales para la identificación del sentimiento en un texto [Prak19]:

- **Basado en léxico**: Un léxico es el vocabulario de una persona, lengua o rama del conocimiento. El método basado en el léxico se denomina diccionario de sentimiento con palabras de sentimiento. La puntuación del sentimiento se asigna a las palabras de sentimiento que representan positivo, negativo y neutro. La colección de términos de sentimiento, frases e incluso expresiones idiomáticas genera la expresión del sentimiento del léxico. Hay dos tipos de enfoques basados en el léxico: la clasificación basada en diccionario y la clasificación basada en el corpus:
  - **Clasificación basada en diccionario:** En este tipo de clasificaciones, los datos se recogen manualmente y la información se busca en sinónimos y antónimos del diccionario de sentimientos. Estos diccionarios son el diccionario WordNet [Lisi69] y el diccionario sentiwordNet [BaES10].
  - **Clasificación basada en corpus**: Este enfoque se basa en los objetivos de los diccionarios relacionados con el dominio específico. Las palabras están relacionadas con métodos estadísticos y semánticos que son el Análisis Semántico Latente (LSA) y el método basado en la semántica.
  - **Clasificación basada en diccionario y en reglas**: El AS basado en el léxico [Gyal20] puede ser tan simple como por ejemplo unas palabras con etiqueta positiva menos

palabras con etiqueta negativa con el objetivo de ver si un texto tiene un sentimiento positivo. También puede ser muy complejo, con reglas de negación, cálculos de distancia, variación añadida y varias reglas adicionales. Una de las principales diferencias entre la PLN basada en reglas y la PLN estadística es que en la PLN basada en reglas, el investigador es completamente libre de añadir cualquier regla que considere útil. Por lo tanto, en la PLN basada en reglas hay expertos altamente capacitados que desarrollan reglas basadas en la teoría de un dominio particular y las aplican a un problema en particular. En este sentido, la librería VADER [HuGi14] es una excelente herramienta para clasificar sentimientos, ya que es un clasificador basado en diccionario y en reglas, especialmente adaptada para el AS en redes sociales. Por cada una de las frases de entrada se obtienen diferentes puntuaciones: neg (negativa), neu (neutral), pos (positiva) y *compound*. La puntuación *compound* se obtiene sumando la puntuación de cada palabra en el conjunto léxico. Además, se ajusta a las reglas definidas y se normaliza el valor entre -1 y 1 (de más negativo a más positivo). Se recomienda un *threshold* o umbral de +/- 0.05 en la puntuación para determinar la clase, por ejemplo, si un documento obtiene una puntuación entre (-0.05;0.05), entonces se clasificaría con un sentimiento neutro. Por el contrario, si la puntuación obtenida es mayor o igual que 0.05, sería clasificado con un sentimiento positivo y si es menor o igual que 0.05 entonces el sentimiento del documento se clasificaría como negativo. Este valor se desglosa para obtener la probabilidad de que el texto analizado sea positivo, negativo o neutro.

  - **Otra herramienta muy utilizada es TextBlob** [Text19], una librería que proporciona una API simple para introducirse en las tareas más comunes de PLN como el AS, la clasificación y la traducción entre otras. La librería *TextBlob* va encontrando palabras y frases a las que puede asignar polaridad y subjetividad y promediando todas para obtener un texto más largo. Para ello, se crea un diccionario de palabras (adjetivos) y puntuaciones de polaridad (positiva/negativa) a partir del léxico (en-sentiment.xml, un documento XML). El valor de cada palabra POS-tag es una tupla con valores de polaridad (-1,0-1,0), subjetividad (0,0-1,0) e intensidad (0,5-2,0).

- **Clasificación basada en** ML: Los métodos de ML son un método muy útil para clasificar los sentimientos que se clasifican en positivo, negativo y neutro (dependiendo del número de categorías se tiene una clasificación binaria o multiclase). El ML requiere un conjunto de datos de entrenamiento y de prueba. Los textos se clasifican mediante algoritmos de ML (en este trabajo se emplearán algoritmos de clasificación supervisados).

En la Figura 2.9 se muestra el funcionamiento básico de los dos tipos de clasificadores.

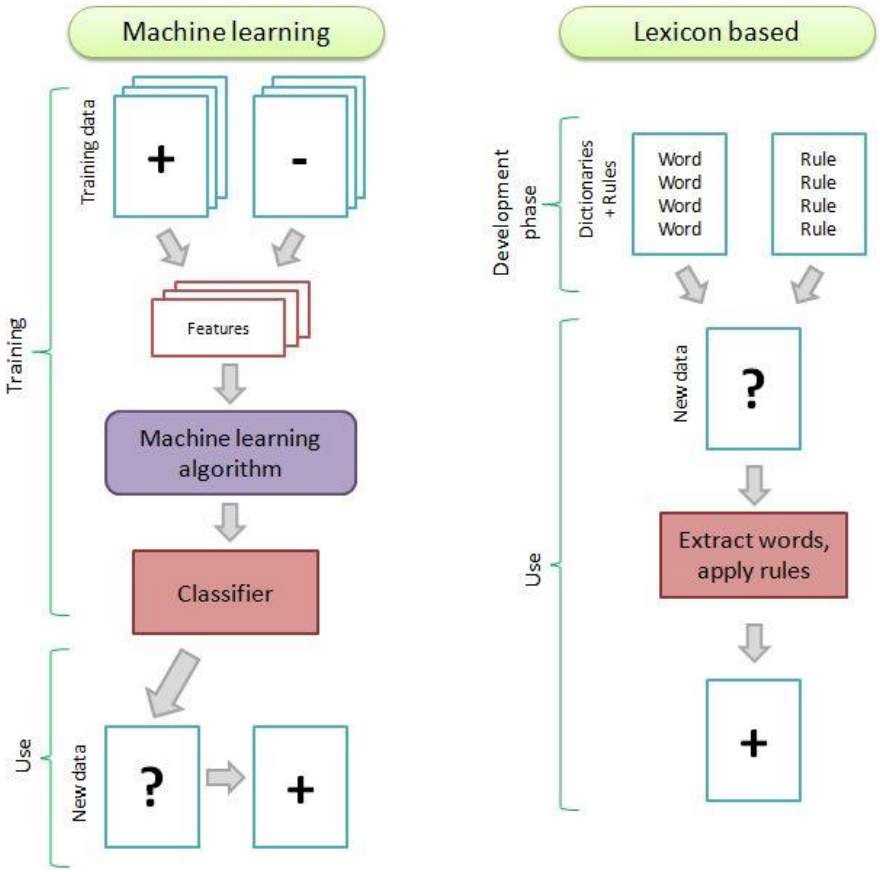


Figura 2.9: Tipos de clasificadores de sentimientos [7].

2.3.4. Problemática en el Análisis de Sentimientos

En términos generales, el AS en texto tiene que lidiar con los siguientes problemas [Erem19]:

- **La ironía y el sarcasmo**: El empleo del sarcasmo en los textos permite engañar fácilmente a los modelos de AS, a menos que estén entrenados específicamente para tener en cuenta este factor, pues aparecerán palabras positivas para expresar sentimientos negativos (y viceversa), por ejemplo: “La verdad que eres un genio. (A alguien que dio una idea poco brillante)”. El sarcasmo aparece con mayor frecuencia en los contenidos generados por los usuarios, como los comentarios de Facebook, Twitter, etc. La detección del sarcasmo en el AS es muy difícil de lograr si no se tiene en cuenta el contexto de la situación, el tema específico, el entorno o no se cuenta con un *corpus* lo suficientemente grande y variado.
- **Tipos de negaciones**: En lingüística, la negación es una forma de invertir la polaridad de las palabras, frases e incluso oraciones. Los investigadores emplean una serie de reglas lingüísticas diferentes para identificar si se produce la negación, pero también es importante determinar el alcance de las palabras que se ven afectadas por la negación. No hay un tamaño prestablecido para el alcance de las palabras afectadas. Por ejemplo, en la frase "El espectáculo no era interesante", el ámbito es sólo la siguiente palabra después de la palabra de negación. Pero en frases como "No considero que esta película sea una comedia", el efecto de la palabra de negación "no" es hasta el final de la frase. El significado original de las palabras cambia si una palabra positiva o negativa cae dentro del ámbito de la negación, en ese caso se devolverá la polaridad opuesta. Tener un

número elevado de observaciones con diferentes tipos de negaciones, aumentará la calidad del *corpus* de entrenamiento.

- **Ambigüedad de palabras**: La ambigüedad de las palabras es otro de los obstáculos en el AS. El problema de la ambigüedad de las palabras es la imposibilidad de definir la polaridad por adelantado porque la polaridad de algunas palabras depende en gran parte del contexto de la frase. Los enfoques de AS basados en léxicos son populares entre los métodos existentes. Como se ha visto anteriormente, un léxico de opinión contiene palabras de opinión con su valor de polaridad. Hay algunos léxicos de opinión públicos disponibles en Internet: *WordNet* [Lisi69], *SentiWordNet* [BaES10] o VADER [HuGi14] entre otros. Como la polaridad de las palabras varía en los distintos ámbitos, es imposible desarrollar un léxico de opinión universal que tenga una polaridad para cada palabra. Por ejemplo: "La historia es impredecible" y la frase 2, "El volante es impredecible". En la primera frase, la palabra "impredecible" es una palabra de opinión que puede tener un sentimiento positivo y la misma palabra "impredecible" en la segunda frase, expresa un sentimiento negativo.
- **Multipolaridad**: A veces, una frase o un documento presenta multipolaridad. En estos casos, tener sólo el resultado total del análisis puede ser erróneo, de forma muy parecida a cómo una media puede ocultar a veces información valiosa sobre todos los elementos que la componen. Es común que dentro de un texto algunos temas sean criticados y otros alabados. En ese caso, la polaridad total del sentimiento carecerá de información clave. Por eso es necesario extraer todas las entidades o aspectos de la frase con etiquetas del sentimiento asignado y calcular la polaridad total sólo si es necesario.

Para hacer frente a estas situaciones, un modelo de AS debe asignar una polaridad a cada aspecto de la frase.

2.4. Aprendizaje Automático

En las secciones 2.1 y 2.2 se describieron de forma general la IA y el PLN respectivamente. En esta sección se abordará el ML, repasando brevemente algunos conceptos básicos para posteriormente introducir conceptos clave en el tratamiento de texto previo al entrenamiento de los algoritmos de ML, como la extracción de características, la ponderación de características y la reducción o selección de características.

2.4.1. Métodos de Aprendizaje

Como se aprecia en la Figura 2.10, existen básicamente tres grandes grupos de ML [Port20]:

- **Aprendizaje supervisado**: En este tipo de aprendizaje los datos de entrada para el entrenamiento de los algoritmos han sido previamente etiquetados, por lo que existe un conocimiento a priori.
- **Aprendizaje no supervisado**. Este tipo de aprendizaje, por el contrario, los datos de entrada para el entrenamiento no están etiquetados, es decir, no hay un conocimiento a priori (en el caso de clasificación, sin necesidad de estar la muestra etiquetada con su clase).
- **Aprendizaje por refuerzo**. El objetivo de este tipo de aprendizaje es la generalización o extrapolación de situaciones que no estén presentes en los datos de entrenamiento. Para ello, se enfocan en aumentar la señal de recompensa (prueba y error).

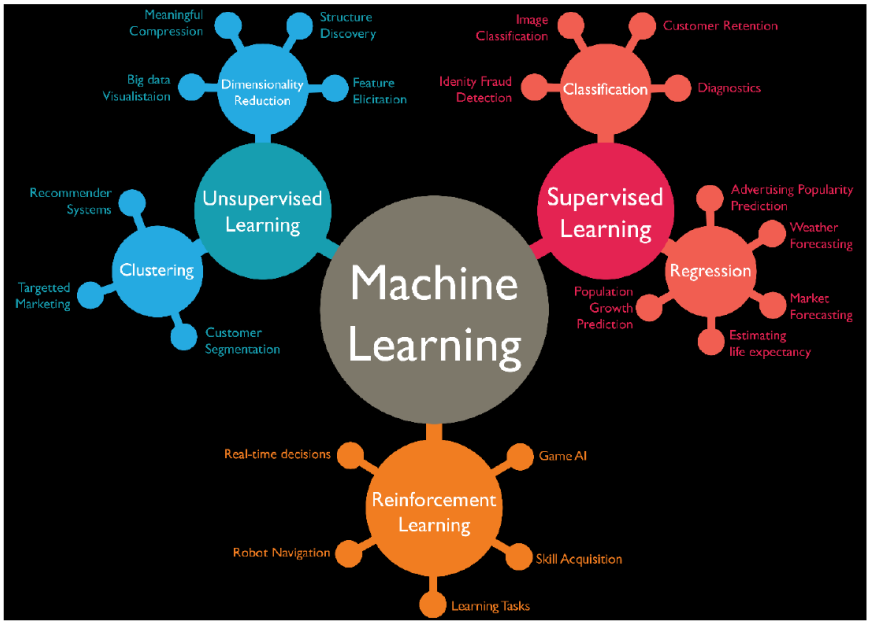


Figura 2.10: Tipos de aprendizaje en ML.

Los principales algoritmos para cada método de aprendizaje se muestran en la Tabla 2.2.

Tabla 2.2: Algoritmos por cada método de aprendizaje

Supervisados	**No supervisados**	**Por refuerzo**
• K-Vecinos más cercanos (KNN). • Máquinas de Vectores de Soporte (SVM). • Árboles de Decisión. • *Random Forest*. • *Multinomial Naive Bayes*. • *Gradient Boosting*. • *Regresión Logística*. • *XGBoost*. • Redes Neuronales Artificiales (ANN).	Agrupamiento jerárquico.	*Q-Learning*

2.4.2. Extracción de características

A partir de la etapa de preprocesamiento, los textos que componen el *corpus* no son computables por los algoritmos de ML (los algoritmos trabajan con datos numéricos), por lo que deben transformarse en datos numéricos, como por ejemplo el modelo de espacio vectorial [8]. Esta transformación se denomina generalmente extracción de características de los documentos. La extracción de características tiene dos métodos principales: la Bolsa de Palabras o BOW (es el método empleado en este trabajo para la extracción de características) y la incrustación de palabras o *Word Embbedings*. Ambos son comúnmente utilizados y tienen diferentes enfoques.

2.4.2.1. Incrustación de Palabras

La incrustación de palabras o *Word Embbedings* [Eiki19] son una de las posibles representaciones de documentos de textos en el modelo de espacio vectorial. Captura parte del contexto y la semántica de las palabras, a diferencia del modelo BOW. La BOW sólo representa el número de palabras que aparecen en el documento (frecuencia) sin ninguna relación ni contexto. Por otro lado, los *Word Embbedings* conserva parte del contexto y las relaciones de las palabras, de modo que identifica las palabras similares con mayor precisión. Estos modelos generan incrustaciones que son independientes del contexto, es decir, sólo hay una representación vectorial (numérica) para cada palabra. Los diferentes sentidos de la palabra (si los hay) se combinan en un único vector. Para profundizar más sobre los *WordEmbbeding*, ver la sección A del Anexo.

2.4.2.2. Bolsa de Palabras

Se define Bolsa de Palabras o Bag of Words (BOW) [Eiki19] al conjunto de palabras existente en todo el *corpus*. Todas las palabras se disponen en una “bolsa” donde se mezclan. La disposición original en el texto se pierde y solo se tiene en cuenta la frecuencia de los términos. En este caso se recomienda eliminar aquellas palabras con poca frecuencia de distribución dentro del corpus (si esté es muy grande), con el objetivo de reducir el vocabulario, mejorando el rendimiento y el tiempo de ejecución del entrenamiento de los modelos así como reducir el posible sobreajuste (*overfitting)* del modelo. Hay que diferenciar los “modelos de representación” de datos como BOW de los "métodos de cálculo o vectorización” para ponderar la importancia de las palabras en el documento, como por ejemplo la Frecuencia de Términos o *Term Frecuency* (TF) o la Frecuencia de Términos-Frecuencia de Documentos Inversa *Term Frequency – Inverse Document Frequency* (TF-IDF) que se describirán más adelante. A continuación se muestra un ejemplo partiendo de tres documentos (que en conjunto forman nuestro *corpus*):

- “Fui a comer tacos de suadero. Juro que es el suadero más delicioso de mi vida. #suadero”
- “Taco de delicioso suadero con bolsa de plástico para que no ensucie el plato.”
- “Tengo ganas de comprarme unos tacos de fútbol, ir a la cancha y jugar hasta la noche”

Una vez se tenga “tokenizado” (eliminando *stopwords,* signos de puntuación, etc.) cada documento se tiene el siguiente vocabulario: bolsa, cancha, comer, comprarme, delicioso, ensuciar, fútbol, ganar, jugar, juro, noche, plato, plástico, suadero, taco y vida.

En la Figura 2.11 se muestra la BOW del corpus (tres documentos). Cada fila es un documento y cada columna un *token*:

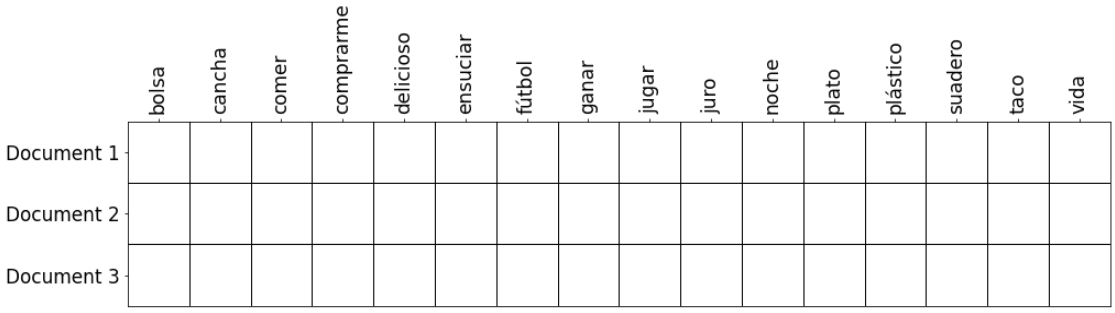


Figura 2.11: Representación BOW[9].

2.4.2.3. Bolsa de *n*-gramas (*n*-BOW)

Según la Wikipedia, "un *n*-grama es una secuencia continua de *n* elementos de una determinada secuencia de texto o discurso". En otras palabras, los n-gramas [Kim18] son simplemente todas las combinaciones de palabras o letras adyacentes de longitud *n* que se pueden encontrar en el texto fuente. La Figura 2.12 representa bien cómo se construyen los *n*-gramas a partir de un texto fuente.

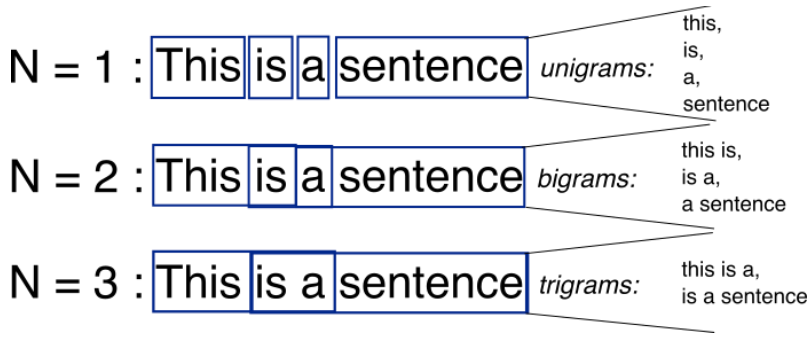


Figura 2.12: Construcción de *n*-gramas a partir de un texto dado[10].

Por ejemplo la frase “el cielo es azul”, los 1-grama serían: “el”, “cielo”, ‘es, “azul”. Un 2-grama (o “bigrama”) es una secuencia de dos palabras, siguiendo el ejemplo anterior se tiene: "el cielo”, “el es”, “el azul”, “cielo es”, “cielo azul”, “es azul” y un 3-grama (o “trigrama”) es una secuencia de tres palabras: “el cielo es”, “el azul es”, etc. Este método en sí mismo no es una representación de un texto, pero puede utilizarse como característica para representar un texto. Extendiendo este concepto para *n*, se pueden crear modelos tan complejos como se quiera. Sin embargo, este tipo de método puede ser insuficiente en determinadas tareas ya que en el lenguaje hay palabras que dependen de otras que están bastante más atrás en la misma oración.

La BOW es una representación de un texto utilizando un 1-grama, perdiendo el orden original del texto. Es muy fácil de obtener y el texto puede representarse mediante un vector, generalmente de tamaño manejable. Por lo tanto se puede considerar los *n*-grama como características y un *n*-BOW como una representación de un texto utilizando los *n*-gramas que contiene. La mayoría de las veces, los “bigramas” y “trigramas” pueden capturar más información que un 1-grama. Por ejemplo, "la hamburguesa sabe mal" o "me gustan las hamburguesas" puede transmitir más información, que sólo "hamburguesa". El problema es que en un *corpus* hay muchos más “bigramas” que 1-gramas, y es posible que muchos de ellos aparezcan una sola vez, por lo que no es útil comparar elementos entre ellos. Sin embargo, si se trata de clasificar un texto para conocer, por ejemplo, el gusto de la gente hacia una comida, puede que no sea suficiente con utilizar un 1-grama. Para clasificar el sentimiento, tal vez se podría representar un texto con BOW y 2-gramas (“bigramas”), donde una de las palabras es una apreciación de las personas hacia la comida.

2.4.3. Ponderación de características

Las características extraídas en la sección 2.4.2 pueden tratarse todas con el mismo peso o importancia u asignarles diferentes pesos (*weighted*) en función de algún tipo de criterio. Aunque existen diversos métodos de ponderación, en este trabajo se emplearán dos modelos muy populares en la Clasificación de Textos y que tienen su origen en el área de la Recuperación de la Información:

- Frecuencia de Términos (TF)
- Frecuencia de Términos – Frecuencia Inverso de Documento (TF-IDF)

Estos pesos establecen la relevancia de cada característica dentro del texto al que pertenecen y, por tanto, repercutirán en los resultados obtenidos en los entrenamientos de los algoritmos de ML.

2.4.3.1. Frecuencia de Términos

La frecuencia de términos o *Term Frecuency* (TF) [Roha20] se define como el valor de la frecuencia absoluta de ese término o palabra en un documento concreto. Cada característica tendrá un peso igual a su frecuencia absoluta en un documento. Matemáticamente se puede representar como se muestra en la Ecuación 2.1.

$$tf\ (w, D)\ =\ f\_\{wD\} \tag{2.1}$$

donde $f\_\{wD\}$ es la frecuencia de la palabra *w* en el documento *D*, convirtiéndose en la Frecuencia de Términos o TF. En otras palabras, TF es el número de veces que una palabra *w* aparece en un documento *D* dividido por el número total de palabras del documento. Cada documento tiene su propia frecuencia de términos.

A veces también se puede estandarizar o normalizar la frecuencia absoluta utilizando logaritmos o promediando la frecuencia. En este trabajo se utilizará la frecuencia absoluta sin normalizar. Partiendo del ejemplo de la sección 2.4.2.2 y utilizando la función *Count Vectorizer* de la librería *Scikit-Learn*, se obtiene la matriz de frecuencia de términos mostrada en la Figura 2.13.

	bolsa	cancha	comer	comprarme	delicioso	ensuciar	fútbol	ganar	jugar	juro	noche	plato	plástico	suadero	taco	vida
Document 1	0	0	1	0	1	0	0	0	0	1	0	0	0	3	1	1
Document 2	1	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0
Document 3	0	1	0	1	0	0	1	1	1	0	1	0	0	0	1	0

Figura 2.13: Matriz de frecuencia de términos [11].

2.4.3.2. Frecuencia de Términos - Frecuencia Inversa de Documento

Pueden existir algunos problemas potenciales con el modelo BOW cuando se utiliza en *corpus* grandes [Roha20]. En los modelos basados en TF (frecuencias absolutas de los términos), es posible que haya algunas palabras que aparezcan con frecuencia en todos los documentos y que éstos tiendan a eclipsar otros términos en el conjunto de características, sobre todo palabras que no aparecen con tanta frecuencia pero que podrían ser más interesantes y eficaces como características para identificar ciertas categorías. *Term frequency – Inverse document frequency* o TF-IDF son las siglas de Frecuencia de Términos-Frecuencia Inversa de Documento. Es una combinación de dos métricas, por un lado la frecuencia de términos (TF) mencionada anteriormente y por otro la frecuencia inversa de documentos (IDF). Con este modelo, se compensa la frecuencia de cada palabra en el documento con la frecuencia de la palabra en la colección de documentos, lo que permite manejar el hecho de que ciertas palabras sean más comunes que otras. Este método se ideó

originalmente como una métrica para clasificar los resultados de los motores de búsqueda en función de las consultas de los usuarios y se ha convertido en una pieza fundamental para de la recuperación de la información y la extracción de características de texto. La fórmula matemática se muestra en la Ecuación 2.2.

$$tfidf\ (t,d,D) = tf(t,d) * idf(t,D) \tag{2.2}$$

donde, $tf(t,d) = f_{t,d}$ es la cantidad de veces que el *token* t aparece en el documento *d*., $idf(t,D) = \log \frac{|D|+1}{|\{d \in D: t \in d\}|+1} + 1$, siendo |*D*| la cantidad total de documentos en el *corpus* y $|\{d \in D : t \in d\}|$ es la cantidad de documentos en las que aparece el *token t.* Dependiendo del paquete o librería utilizado, se puede encontrar un parámetro llamado "suavizado" o *smooth*, que consiste en sumar una cantidad (normalmente 1 ó 0.1) que ayudan a prevenir divisiones entre cero y términos que puedan tener un IDF igual a cero. A continuación, se listan los pasos necesarios [Taco20] para el cálculo del TF-IDF de la palabra "suadero" del ejemplo de la sección 2.4.2.2:

- Seleccionamos el *token "suadero*" en el primer documento (denotado como *documentos₁*):
- $tfidf$(suadero, $documentos_1, corpus$) = tf(suadero, $documentos_1$) × idf(suadero, $corpus$)
- $tfidf$(suadero, $documentos_1, corpus$) $= 3 * (log \frac{3+1}{2+1} + 1)$
- $tfidf$(suadero, $documentos_1, corpus$) $= 3 * (log \frac{4}{3} + 1)$
- $tfidf$(suadero, $documentos_1, corpus$) $= 3 * (\log 1.333 + 1)$
- $tfidf$(suadero, $documentos_1, corpus$) $= 3.8630$

En la Figura 2.14 se muestra la matriz de términos TF-IDF para cada uno de los documentos que forman el *corpus* del ejemplo de la sección 2.4.2.2.

	bolsa	cancha	comer	comprarme	delicioso	ensuciar	fútbol	ganar	jugar	juro	noche	plato	plástico	suadero	taco	vida
Document 1	0.00	0.00	1.69	0.00	1.29	0.00	0.00	0.00	0.00	1.69	0.00	0.00	0.00	3.86	1.00	1.69
Document 2	1.69	0.00	0.00	0.00	1.29	1.69	0.00	0.00	0.00	0.00	0.00	1.69	1.69	1.29	1.00	0.00
Document 3	0.00	1.69	0.00	1.69	0.00	0.00	1.69	1.69	1.69	0.00	1.69	0.00	0.00	0.00	1.00	0.00

Figura 2.14: Matriz TF-IDF[12].

Los algoritmos de ML normalmente funcionan mejor cuando los datos de entrenamiento se encuentran normalizados (para que el entrenamiento del modelo sea menos sensible a la magnitud de las características). Esto permite que los modelos converjan mejor y más rápido logrando un modelo más preciso. La normalización hace que las características sean más coherentes entre sí, lo que permite al modelo predecir los resultados con mayor precisión. Por ello, se utilizará la función TF-IDF de la librería *Scikit-Learn* para normalizar (normalización euclídea) cada fila como se muestra en la Ecuación 2.3.

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{v_1^2 + \cdots + v_n^2}} \tag{2.3}$$

Por ejemplo, para el término "suadero" del *documentos₁* se tiene la Ecuación 2.4.

$$v_{norm} = \frac{3.86}{\sqrt{0^2 + \cdots + 1.69^2 + \cdots + 1.29^2 + \cdots + 1.69^2 + \cdots + 3.86^2 + 1^2 + 1.69^2}} = 0.75 \quad (2.4)$$

Finalmente, siguiendo los ejemplos anteriores, en la Figura 2.14 se muestra la matriz TF-IDF normalizada.

	bolsa	cancha	comer	comprarme	delicioso	ensuciar	fútbol	ganar	jugar	juro	noche	plato	plástico	suadero	taco	vida
Document 1	0.00	0.00	0.33	0.00	0.25	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.75	0.20	0.33
Document 2	0.43	0.00	0.00	0.00	0.32	0.43	0.00	0.00	0.00	0.00	0.00	0.43	0.43	0.32	0.25	0.00
Document 3	0.00	0.40	0.00	0.40	0.00	0.00	0.40	0.40	0.40	0.00	0.40	0.00	0.00	0.00	0.23	0.00

Figura 2.15: Matriz TF-IF normalizada con *Scikit-Learn*.

Como se observa en la Figura 2.15, en muchas ocasiones se tienen más ceros (valores en blanco) que valores reales, es decir, los vectores (documentos) son dispersos (o en ingles *sparse vectors*). Esto podría llegar a ser un problema (sobre todo de memoria) cuando se tenga un vocabulario de tamaño considerable.

2.4.4. Modelado de Tópicos

Como detalla en la sección 2.2.2, una de las principales aplicaciones del PLN es el Modelado de Tópicos o *Topic Modeling*, el cual puede definirse como una técnica no supervisada para descubrir temas en varios documentos de texto [Deve13]. Estos temas son de naturaleza abstracta, es decir, las palabras que están relacionadas entre sí forman un tema. Del mismo modo, puede haber múltiples temas en un documento individual. El Modelado de Tópicos se puede ver como una “caja negra”, como se ilustra en la Figura 2.16.

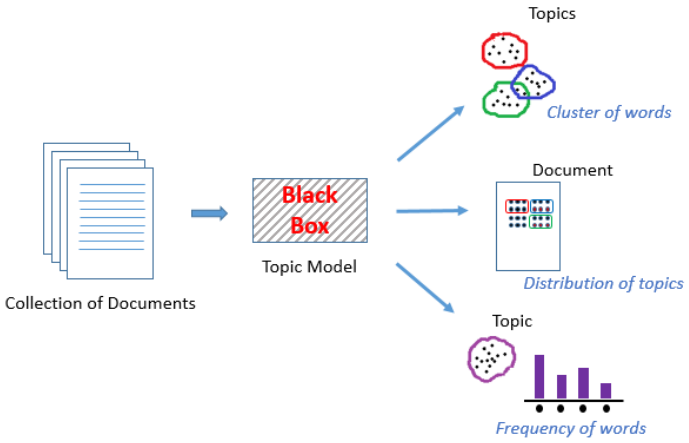


Figura 2.16: Modelado de Tópicos[13].

Este modelo de “caja negra” forma grupos de términos similares y relacionados entre sí, que se denominan temas o tópicos. Estos temas tienen una determinada distribución en un documento y cada tema se define por la proporción de términos diferentes en el grupo. Aquí es donde entra en escena el LSA o SVD, que intenta aprovechar el contexto que rodea a las palabras para captar los conceptos ocultos, también conocidos como temas. Esta técnica tiene la capacidad de identificar las correlaciones más fuertes entre los términos, mediante la

reducción de dimensiones utilizando una descomposición de la matriz término-documento en valores singulares (SVD).

En este trabajo se utilizará el nodo *Text Topic* de SAS Miner, asignando una puntuación a cada documento y término de cada tema. A continuación, se utilizarán indicadores para decidir si la asociación es lo suficientemente fuerte como para determinar que el documento o el término es un tema. Como resultado, los documentos y términos pueden pertenecer a más de un tema o a ninguno. El número de temas que se establezca debe estar directamente relacionado con el tamaño de la colección de documentos (por ejemplo, un número elevado para una colección grande). La tarea que más memoria requiere es el cálculo de *Single Value Descomposition* o SVD de la matriz de frecuencia término-documento (TF-IDF). Partiendo de la matriz de términos TF-IDF, se reducen las dimensiones de la matriz a *k* (número de temas o tópicos deseados) dimensiones, utilizando el algoritmo *Single Value Descomposition* o SVD (en la sección 2.4.6.1 se describirá de forma más detallada este algoritmo). Además del SVD, existen otras técnicas avanzadas y eficientes de modelado de temas, como el Latent Dirichlet Allocation (LDA) e lda2Vec.

2.4.5. Reducción de Características

A la hora de construir un modelo de ML, es muy importante el espacio vectorial de entrada de los algoritmos. Con un espacio vectorial demasiado grande, el tiempo de ejecución de los algoritmos será mayor y se podría producir un sobreajuste. En tal caso, se pueden eliminar aquellas características que no aportan gran información o que no sirven para diferenciar una clase de otra con el objetivo de mejorar el rendimiento tanto en tiempo como en espacio de memoria. A esta técnica se le llama reducción de características o *feature selection*.

Una de las formas más simples es eliminar aquellas características que tengan poca o ninguna varianza (observaciones con el mismo valor o valores muy cercanos). A esta técnica se le denomina Análisis de Componentes Principales (o del inglés *Principal Component Analysis* PCA) [Alon20] y permite “agrupar” la información aportada por múltiples variables en solo unas pocas componentes. Otra manera de reducir el número de características sería estudiando la relación de las variables con el objetivo (Test Chi-Cuadrado).

Por último, también se podría utilizar un algoritmo de optimización cuyo objetivo sea encontrar el subconjunto de funciones con mejor desempeño. Para ello, se crea repetidamente varios modelos y se deja de lado la mejor o la peor característica de rendimiento (según la métrica de evaluación elegida) en cada iteración. Este método se le conoce como Eliminación de Características Recursivas o del inglés *Recursive Feature Elimination* (RFE).

2.4.5.1. Método Chi-Cuadrado

El estadístico Chi-Cuadrado (X^2) [Calv20] es un método muy común para detectar relaciones entre dos variables de clase, siendo especialmente útil para relaciones no lineales. Los datos de entrada requeridos para calcular este estadístico consisten en una tabla de contingencia, por lo que las variables de intervalo han de ser discretizadas previamente. Para una explicación más detallada, ir a la sección B del Anexo.

Chi-Square Test con Python

Se emplearán dos métodos para la selección de características mediante el Chi-Cuadrado en Python:

- Por un lado, se seleccionan las variables con un p-valor < 0.05 (rechazamos la H0).
- Por otro lado, se seleccionan las *n* variables cuyos valores Chi-Cuadrado (*scores*) sean los más elevados.

2.4.5.2. Eliminación de Características Recursivas con Validación Cruzada

La eliminación recursiva de características (RFE) [IgEl03] es básicamente una selección hacia atrás o *backward* de los predictores. Esta técnica comienza construyendo un modelo (con un estimador elegido por el usuario) con todo el conjunto de variables y calculando una métrica de importancia para cada predictor. A continuación, se eliminan los predictores menos importantes y se vuelve a construir el modelo, calculándose de nuevo la métrica de importancia. Dado un estimador externo que asigna pesos a las características (por ejemplo, los coeficientes de un modelo lineal), el objetivo del RFE es seleccionarlas considerando recursivamente conjuntos cada vez más pequeños de características. En primer lugar, el estimador se entrena con el conjunto inicial de características y la importancia de cada una de ellas se obtiene a través de algún atributo específico (como *coef_* o feature_*importances_*). A continuación, se eliminan las características menos importantes del conjunto actual de características. Este procedimiento se repite recursivamente en el conjunto reducido hasta que se alcanza el número deseado de características a seleccionar (programado por el usuario). El tamaño mínimo del subconjunto es un parámetro de ajuste para la RFE. El subconjunto óptimo se utiliza entonces para entrenar el modelo final. Para conocer los parámetros del algoritmo, ir a la C del Anexo.

2.4.5.3. Análisis de Componentes Principales

El Análisis de Componentes Principales (PCA) es una técnica de extracción de características en la que combinan las entradas de una manera específica pudiendo eliminar algunas de las variables "menos significativas", conservando la parte más importante de todas las variables [Alon20]. Como valor añadido, después de aplicar el PCA se logra en cierto grado, que todas las nuevas variables sean independientes una de otra. A continuación se describen las diferentes etapas del algoritmo:

- Estandarizar los datos de entrada (o Normalización de las Variables) entre 0 y 1.
- Calcular los autovectores y autovalores de la matriz de covarianza
- Ordenar los autovalores de mayor a menor y seleccionar los “k” autovectores que se correspondan con los autovectores “k” más grandes (donde “k” es el número de dimensiones del nuevo subespacio de características).
- Construir la matriz de proyección W con los “k” autovectores seleccionados.
- Transformación del conjunto de datos original “X estandarizado” mediante W para obtener las nuevas características k-dimensionales.

Para conocer los parámetros del algoritmo, ir a la sección D del Anexo.

2.4.6. Algoritmos de Clasificación No Supervisados

Los algoritmos no supervisados utilizan conjunto de datos no etiquetados para construir *clusters* o grupos. Para el modelado de tópicos se utilizará SAS MINER, en concreto el nodo *Text Topic Node* que emplea a su vez el algoritmo SVD que se describirá a continuación.

2.4.6.1. Descomposición de Valores Singulares

Cada entrada de la matriz de documento-término representa el número de veces que un término aparece en un documento. Para una colección de varios miles de documentos, la matriz de frecuencia término-documento puede contener cientos de miles de palabras. Se requiere por tanto mucho tiempo y espacio de cálculo para analizar esta matriz de forma eficaz [Deve13]. Además, tratar con datos con alta dimensionalidad es intrínsecamente complejo para el proceso de modelado. Para mejorar el rendimiento, se puede aplicar la descomposición de valores singulares (del inglés *Single Value Descomposition* o SVD) para reducir las dimensiones de la matriz de frecuencia término-documento, transformando la matriz en una forma de menor dimensión más compacta e informativa. Un número elevado de dimensiones SVD suele resumir mejor los datos, pero cuanto mayor sea el número más recursos informáticos se necesitarán. El cálculo de la SVD es en sí mismo una tarea que requiere mucha memoria. En el caso de problemas con un número de elevado de observaciones y dimensiones, SVD puede realizar automáticamente una muestra aleatoria de los documentos para evitar que se agote la memoria.

Para cualquier *k* (en nuestro caso *k* es el número de temas o tópicos a buscar) dado, el resultado de la transformación será la factorización de la matriz con *k* dimensiones que mejor se aproxima a la matriz original. Un valor más alto de *k* da una mejor aproximación a la matriz *A*. Sin embargo, la elección de un valor demasiado grande para *k* podría dar lugar a una dimensión demasiado alta para el proceso de modelado. En general, el valor de *k* debe ser lo suficientemente grande como para preservar el significado de la colección de documentos, pero no tan grande como para capturar el ruido. Los valores entre 10 y 200 son apropiados a menos que la colección de documentos sea pequeña. Para su aplicación específica en minería de textos, es posible que se desee comparar los resultados para varios valores de *k*. Como regla general, los valores más pequeños de *k* (de 2 a 50) son útiles para la agrupación, y los valores más grandes (de 30 a 200) son útiles para la predicción o la clasificación. La SVD descompone la matriz de frecuencia término-documento en otras tres matrices. SVD factoriza la matriz de frecuencia término-documento grande y dispersa (*sparse matrix*) calculando una SVD truncada de la matriz. Suponiendo que *A* es la matriz de frecuencia término-documento grande y dispersa con entradas ponderadas. La SVD de una matriz *A* es una factorización de *A* en tres nuevas matrices *U*, *D* y *V* con la Ecuación (2.5):

$$A = UDV^T \tag{2.5}$$

Las matrices *U* y *V* tienen columnas ortonormales mientras que *D* es una matriz diagonal de valores singulares. La SVD calcula sólo las primeras *k* columnas de estas matrices (*U, D* y *V*). Una vez calculada la SVD, cada columna (o documento) de la matriz de frecuencias término-documento puede proyectarse sobre las primeras *k* columnas de *U*. Matemáticamente, esta proyección forma un subespacio *k*-dimensional que es el que mejor describe el conjunto de datos. Cada fila de la matriz *U* (matriz de término-documento) es la representación vectorial del correspondiente documento. La longitud de estos vectores es *k*,

que es el número de temas deseados. Así, la SVD genera los vectores para cada documento y término del conjunto de datos.

2.4.7. Algoritmos de Clasificación Supervisados

Todos los algoritmos de clasificación que se han utilizado para construir el mejor modelo pertenecen a la librería de Python *Scikit-Learn*. A continuación, se describe los algoritmos utilizados y sus principales parámetros.

2.4.7.1. Regresión Logística

La regresión logística [Calv20] es conceptualmente similar a la regresión lineal, donde la regresión lineal estima la variable objetivo a partir de un conjunto de variables independientes xi. Sin embargo, en lugar de predecir valores continuos, como en la regresión lineal, la regresión logística estima las probabilidades de que se produzca un determinado evento. El modelo de regresión logística, para el caso de variable objetivo-binaria, se representa en la Ecuación (2.6).

$$p_1 = P(Y = 1|x_1, x_2, \ldots, x_m) = \frac{1}{1 + e^{-(\beta 0 + \beta 1 x1 + \cdots + \beta m x m)}} \tag{2.6}$$

Lo que implica que:

$$\log\left(\frac{p_1}{1 - p_1}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m$$

El término de la izquierda de la ecuación recibe el nombre de *logit* y es el logaritmo del *odds ratio*. Este concepto representa el cociente entre los *odds*, es decir, el cociente entre la probabilidad de que ocurra el suceso y la probabilidad de que no ocurra. Los principales parámetros en Python son:

- ***Penalty***: Se utiliza para especificar el criterio de penalización.
- ***Tol***: Tolerancia para el criterio de parada.
- ***C***: Inverso de la "fuerza" de la regularización. Como en las SVM, los valores más pequeños especifican una regularización más fuerte.
- ***Solver***: algoritmo usado en proceso de optimización.
- ***L1 ratio***: parámetro de ajuste de Elastic-Net.

2.4.7.2. Clasificador Naive Bayes para Modelos Multinomiales

Los clasificadores de la familia de *Naive Bayes* [Roma19] son muy usados en PLN, en particular en la clasificación de textos, ya que suelen producir resultados comparables con los obtenidos por otros métodos más sofisticados y son bastantes sencillos de implementar. Uno de los aspectos desfavorables de estos clasificadores es que presuponen que los términos que figuran en un documento son todos independientes entre sí, lo cual puede no ser totalmente cierto debido a la naturaleza del lenguaje. Muchos estudios han tratado de proporcionar información adicional al clasificador para suavizar las consecuencias de la suposición de independencia. El clasificador *Naive Bayes* simple [Deve00] considera la probabilidad de ocurrencia de cada término dada la clase de forma binaria, es decir, el término se da o no se da y luego su probabilidad condicional dada la clase es o no considerada. En este sentido, el clasificador *Naive Bayes* Multinomial suele mejorar su rendimiento ya que considera el número de apariciones del término para evaluar la contribución donde la

probabilidad condicional dada la clase con lo que el modelado de cada documento se ajusta mejor a la clase a la que pertenece.

Multinomial NB de Python implementa el algoritmo de *Naive Bayes* para datos con una distribución multinomial, siendo una de las dos variantes clásicas de *Naive Bayes* utilizadas en la clasificación de textos (donde los datos se representan típicamente como recuentos de vectores de palabras, aunque también se sabe que los vectores TF-IDF funcionan bien en la práctica).

Estos son sus principales parámetros:

- ***alpha*:** Parámetro de suavizado aditivo (*Laplace/Lidstone*) (0 para no suavizar).

2.4.7.3. K-Nearest Neighbors

En clasificación, algoritmo KNN [Port20] trata de asignar a cada observación el valor más frecuente de los *k* observaciones más próximas a ella. Algunas de las medidas de distancia habituales son euclídea (continuas) o la distancia de *Hamming* (discretas). El funcionamiento del KNN (Figura 2.16) depende de la distancia entre las observaciones, por lo que normalizar los valores de las características puede mejorar drásticamente los resultados. Elegir un buen valor para *k* es importante ya que normalmente grandes valores pueden reducir la cantidad de ruido pero a la vez puede exponerse a un *overfitting*. Por el contrario, un valor más bajo favorecerá que la separación entre clases sea más nítida.

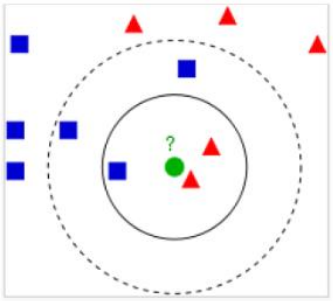


Figura 2.16: Ejemplo de clasificación con KNN.

Estos son sus principales parámetros:

- ***Neighbors*:** Número de vecinos.
- ***Weights*:** Función de pesos empleada en la predicción**,** *'uniform'* para dar el mismo peso a cada vecino o *'distance'* para que los pesos sean inversamente proporcionales a la distancia entre vecinos.
- ***Algorithm*:** Algoritmo utilizado para calcular los vecinos más cercanos '*kd_tree*', *'ball_tree*', *'brute'* o *'auto'*.
- ***Metric*:** Métrica de distancia. Las más habituales son: *Euclídea*, *Minkowsky*, *Hamming*, *Jaccard* o *Manhattan*.

2.4.7.4. Árboles de Decisión

Los árboles de decisión son algoritmos iterativos [Port20] que consisten en dividir los datos en regiones basadas en intervalos de variables independientes. Además, son capaces de explicar variables de una forma sencilla al tratarse de métodos no paramétricos, no siendo necesaria se cumpla ningún tipo de distribución específica. En términos generales, el proceso

se inicia a partir de un nodo raíz [Calv20] que contiene todas las observaciones del conjunto de datos de entrenamiento, posteriormente, se dividen las observaciones en subconjuntos o nodos hijos, con el objetivo de encontrar una segmentación con la mayor homogeneidad posible respecto a la variable objetivo. Cuando se quiere predecir una nueva observación, se recorre el árbol según el valor de sus predictores hasta alcanzar uno de los nodos hoja. La predicción del árbol es la media de la variable respuesta de las observaciones de entrenamiento que están en ese mismo nodo hoja. Los árboles de decisión consisten en:

- Se empieza por encontrar un punto de corte en cada variable (obteniendo dos intervalos) y se observa el error cometido en la predicción fijando valores constantes. Se seleccionan la variable y el punto de corte óptimos. En el caso de las variables categóricas, se establecen agrupaciones de categorías en lugar de puntos de corte.
- Dentro de las divisiones obtenidas en el apartado anterior, se subdivide el árbol hasta alcanzar algún criterio de parada (número de hojas finales, por ejemplo). Además, es necesario establecer una metodología para crear las regiones mencionadas anteriormente, es decir, decidir dónde se establecen las divisiones, sobre qué predictores y sobre qué valores de los mismos. En este trabajo, dado que uno de los objetivos es la construcción de un clasificador multiclase (variable objetico de clase), los criterios de división empleados por los árboles son los siguientes:
- **Test de la χ^2(*ProbChisq*):** Consiste en utilizar el p-valor asociado al estadístico chi-cuadrado explicado en el apartado 2.4.5.1. La idea es encontrar las variables que sean dependientes de la variable objetivo, mediante el contraste de hipótesis (rechazando o manteniendo la hipótesis nula sobre la variable seleccionada)
- **Índice de Gini**: Este índice permite identificar la homogeneidad de las observaciones de un nodo u hoja con relación a una variable categórica. Un nodo "puro" tendrá un índice de 0 y uno con la misma proporción de eventos y no eventos tendrá un índice de 0,5. Cuando la variable objetivo es binaria, el Índice de Gini se expresa con la Ecuación (2.7).

$$IG(nodo\ j) = 1 - \left(\frac{n_{ev}^{j}}{n_j}\right)^2 - \left(\frac{n_{no}^{j}}{n_j}\right)^2 \tag{2.7}$$

- **Entropía**: Permite medir la homogeneidad de un nodo, si bien la expresión es diferente a la del Índice de Gini. Viene dada por la Ecuación (2.8).

$$E(nodo\ j) = -p_{ev}^{i}\log_2\left(p_{ev}^{j}\right) - p_{no}^{j}\log_2\left(p_{no}^{j}\right) \tag{2.8}$$

donde $p_{ev}^{j} = \frac{n_{ev}^{j}}{n_j}$ y $p_{no}^{j} = \frac{n_{no}^{j}}{n_j}$

La entropía de un nodo "puro" será igual a 0 mientras que un nodo con igual proporción de eventos y no eventos, tendrá un valor igual a 1.

2.4.7.5. Random Forest

El algoritmo de *Random Forest* [Port20] es una modificación del método de *bagging*, el cual consiste en incorporar aleatoriedad en las variables utilizadas para segmentar cada nodo del árbol. Los modelos de *Random Forest* [Amat20] están formados por un conjunto de ***árboles de decisión*** individuales, de modo que los valores de cada árbol son entrenados con una muestra aleatoria extraída de los datos de entrenamiento originales mediante reemplazamiento o *bootstrapping*. Por lo tanto, cada árbol se entrena con datos ligeramente diferentes, ya que en cada árbol individual las observaciones se van distribuyendo por nodos,

creando la estructura de árbol hasta alcanzar un nodo hoja. La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo. El objetivo [Port20] es reducir el lograr un equilibrio entre *bias* y varianza incorporando dos métodos de variabilidad, por un lado el *remuestreo* de las observaciones y por otro la aleatoriedad de las variables, utilizadas para segmentar los nodos de los árboles. Este algoritmo consiste en:

- Repetir *m* veces las siguientes acciones: Seleccionar *N* observaciones con reemplazamiento de los datos originales.
- Aplicar un árbol de la siguiente manera: En cada nodo, seleccionar *p* variables de las *k* originales y de las *p* elegidas, seleccionar la mejor variable para la partición del nodo.
- Obtener predicciones para todas las observaciones originales *N*.
- Promediar las *m* predicciones obtenidas en el apartado 1.

Las principales ventajas y desventajas del *Random Forest son*:

Ventajas:

- Los árboles son fáciles de interpretar aun cuando las relaciones entre predictores son complejas.
- Encuentran relaciones no lineales y mejoran su capacidad predictiva cuando las los predictores son categóricos, además de evitan el problema de la existencia de variables predictoras muy dominantes. Logran reducir la varianza en el modelo final gracias a la aleatoriedad en las variables de cada nodo, generando árboles diferentes unos de otros.
- Los árboles pueden, en teoría, manejar tanto predictores continuos como categóricos sin tener que crear variables *dummies* o *one-hot-encoding*.
- Al tratarse de métodos no paramétricos, no es necesario que se cumpla ningún tipo de distribución específica.
- En general, requieren mucha menos limpieza y preprocesamiento de datos en comparación con otros métodos de ML (no requieren estandarización).
- No se ven muy influenciados por *outliers*.
- Son muy útiles en la exploración de datos, permiten identificar de forma rápida y eficiente las variables (predictores) más importantes.
- Son capaces de seleccionar predictores de forma automática.
- Pueden aplicarse a problemas de regresión y clasificación.

Desventajas:

- Son sensibles a datos de entrenamiento desbalanceados (una de las clases domina sobre las demás).
- Cuando tratan con predictores continuos, pierden parte de su información al categorizarlos en el momento de la división de los nodos.

Los parámetros para configurar en el *Random Forest utilizando* la librería de *Scikit-Learn* de Python son:

- ***Bootstrap***: Si se utilizan muestras *Bootstrap* o reemplazamiento al construir los árboles. Si es Falso, se utiliza todo el conjunto de datos para construir cada árbol.
- ***Max depth***: Máxima profundidad del árbol.
- ***Max features***: Número de variables que se sortean en cada división del árbol.
- ***Min samples leaf***: Número de observaciones que habrá como mínimo en una rama-nodo, es decir, al tamaño mínimo de la hoja.

- ***Min samples split***: Número de observaciones a sortear por nodo.
- ***N estimators***: Número total de árboles.
- ***Criterion***: Función para medir la calidad de una división. Los criterios admitidos son "*gini*" para la impureza de *Gini* y "entropía" para la ganancia de información.

2.4.7.6. XGBoost

XGBoost (*Extra Gradient Boosting*) es una modificación del *Gradient Boosting,* convirtiéndose en uno de los algoritmos más famosos entre los concursos de predicción o clasificación. *XGBoost* emplea varios clasificadores débiles o *weak learnings*, en este caso Árboles de Decisión, pero potenciando los resultados de estos gracias al procesamiento iterativo de los datos con una función de pérdida o coste, la cual minimiza el error iteración tras iteración. La principal mejora introducida frente al *Gradient Boosting* es la incorporación de factores de regularización para reducir el sobreajuste. Esta regularización consiste en controlar los valores de los parámetros para que no obtengan valores excesivamente grandes o pequeños. De esta manera, la función que se genera para predecir estará normalizada. Los principales parámetros para controlar en Python son:

- ***Eta o learning rate****:* Hace referencia al *shrinkage*, es decir, la velocidad de ajuste.
- ***Gamma***: Es el parámetro de regularización, que penaliza las predicciones complejas que pudiera tener una hoja de un árbol.
- ***Min_child_weight***: Número mínimo de observaciones que tendrán los nodos finales.
- ***Max_depth***: Profundidad máxima del árbol, es decir, la distancia entre el nodo raíz y la hoja más lejana.
- ***Subsample***: Número de observaciones que se utilizan para construir cada árbol.

2.4.7.7. Máquinas de Soporte Vectorial

El algoritmo de Maquina de Vector Soporte (SVM) [Port20] trata de buscar el hiperplano que divida los datos en diferentes grupos de acuerdo con sus características. En la Figura 2.17 se muestra un ejemplo de hiperplano en 2 dimensiones. Se podría pensar que hay múltiples soluciones para dividir un plano con una recta, pero la SVM resuelve este problema cogiendo el punto de cada grupo más alejado, también llamado vector de soporte e intentando maximizar su margen (distancia que lo separa de la recta que divide cada agrupación de datos). Además, se puede tener más de un vector de soporte por grupo, en ese caso se tendrá que elegir el que mayor margen tenga. Idealmente una SVM separará los grupos de forma perfecta usando los hiperplanos mencionados anteriormente. Esto no siempre es posible, y si lo es, este modelo no podrá ser utilizado en otro conjunto de datos, ya que se consideraría que ha sobre aprendido.

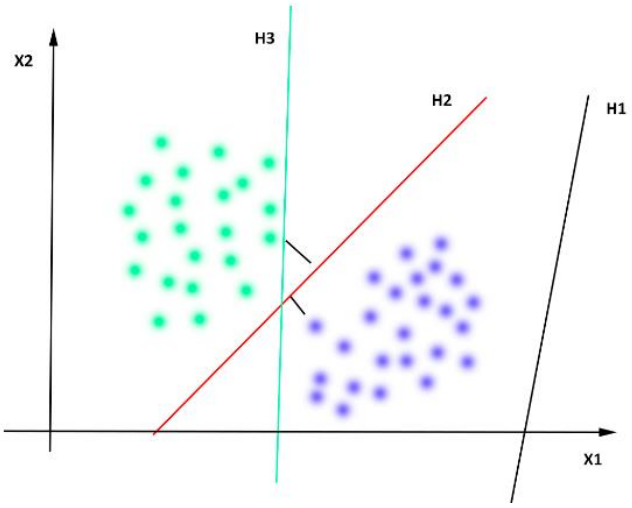


Figura 2.17: Hiperplanos que dividen dos grupos de puntos

La separación entre clases en muchos problemas no es lineal. Es posible trabajar, a pesar de ser un algoritmo de separación lineal, en un espacio de alta dimensionalidad. El problema es que este aumento de dimensión hace a menudo impracticables los cálculos. Este problema se soluciona con el llamado "truco *Kernel*" ("*the Kernel trick*"): cualquier algoritmo que dependa solo de los productos escalares (como es el caso del SVM) permite trabajar computacionalmente en una dimensión controlada a través de una función llamada *Kernel*. A continuación se muestra los diferentes parámetros asociados a las SVM (La compatibilidad multiclase se gestiona según el esquema *one-vs-one*):

- ***Kernel***:
  - **RBF**: Aumentar gamma en la función RBF implica menor sesgo y mayor sobreajuste.
  - **Polinomial**. Aumentar el grado del polinomio implica menor sesgo y mayor sobreajuste.

  En la Figura 2.18 se muestran las expresiones matemáticas de cada clase de *Kernel*.

Linear function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j$
Polynomial function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)^d$
RBF function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ $= \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma = \frac{1}{2\sigma^2}$
Sigmoid function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \tanh\left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)$ $\tanh(x) = \frac{e^x}{e^x + 1}$

Figura 2.18: Expresiones matemáticas asociadas a cada *Kernel*.

- ***C***: Con el fin de contemplar un poco de flexibilidad, SVM cuenta con un parámetro *C*, es un parámetro de regulación, es el único parámetro que puede ser ajustado a la hora de construir un clasificador con SVM. Su rango depende mucho de los datos. Aumentar *C* implica menor sesgo y mayor sobreajuste. Este parámetro regula cómo de precisos se quiere que sea nuestra máquina, por lo tanto puede controlar el nivel de aprendizaje.
- ***Degree***: Grado del polinomio (válido para el *kernel* polinomial).
- ***Gamma***: Coeficiente para el *kernel Radial Basis Function* o RBF y Polinomial. Cuando *gamma* es muy pequeña, el modelo está demasiado restringido y no puede capturar la complejidad o la "forma" de los datos. Cuando *gamma* es muy grande, el radio del área

de influencia de los vectores de soporte sólo incluye el propio vector de soporte y ninguna regularización con *C* podrá evitar el sobreajuste.

- ***Max_iter***: Límite estricto de iteraciones dentro del solucionador dado, o -1 si no hay límite (por defecto max_iter = -1).

La mayoría de las implementaciones (R, Python, etc.) están basadas en el artículo *libsvm* de [ChLi11]. En 2008, [FCHW08] desarrollaron una implementación para resolver problemas de clasificación lineal a gran escala, soportado regularizaciones basados en RL y SVM. Para grandes volúmenes de datos los resultados obtenidos con/sin mapeos no lineales son similares. Sin utilizar *kernels*, se puede entrenar rápidamente un conjunto mucho más grande a través de un clasificador lineal. La clasificación de documentos es una de esas aplicaciones. En [FCHW08], se midió el rendimiento de LIBSVM y LIBSVM sobre 20,242 instancias y 47,236 características. Obtuvieron resultados similares en términos de *accuracy*, pero el tiempo de la validación cruzada se vio reducido significativamente al usar LIBLINEAR.

Esta implementación, es semejante a SVM con el parámetro *kernel="linear"* (único *kernel* soportado). En lugar de estar implementado con LIBSVM, lo está con LIBLINEAR, lo que supone más flexibilidad en la asignación de penalizaciones y funciones de coste. Según la documentación de *Scikit-learn*, esta implementación debería ofrecer mejor rendimiento que SVM cuando se trabaja con gran número de muestras y con matrices dispersas (*sparse matrix,* como es el caso de los métodos de vectorización vistos en la sección 2.4.3). Dado que la implementación SVM conlleva un tiempo de entrenamiento elevado (del orden de θ^2 y θ^3), se probará la implementación LIBLINEAR con el objetivo no solo de reducir los tiempos de ejecución sino también de mejorar los resultados.

2.4.7.8. Redes Neuronales Artificiales

Las redes neuronales artificiales o en inglés ANN [Port20] son una técnica de modelización inspirada en el funcionamiento del cerebro humano (conexiones neuronales) que permite aprender mediante el aprendizaje a partir de datos representativos que describen un fenómeno físico o un proceso de decisión. Una característica única de las ANN es que son capaces de establecer relaciones empíricas entre variables independientes y dependientes, además de extraer información relevante y conocimientos complejos de conjuntos de datos representativos. La relación entre las variables independientes y las dependientes pueden establecerse sin necesidad de asumir ninguna representación matemática. Los modelos de ANN aportan ciertas ventajas sobre los modelos basados en regresión (linealidad), entre ellas su capacidad para lidiar con datos ruidosos (no linealidad). Las redes neuronales están compuestas por nodos interconectados (neuronas), donde cada unidad procesa determinada información aprovechando las conexiones establecidas con el resto de las neuronas para "combinar" dicha información. A continuación, se muestra la arquitectura de una ANN (Figura 2.19), donde se establece una capa *input* que se puede conectar con una o más capas ocultas (entonces se habla de DL). Finalmente, los nodos de las capas ocultas se conectan con la capa output (dependiendo del tipo de variable objetivo, se tendría uno o varios nodos).

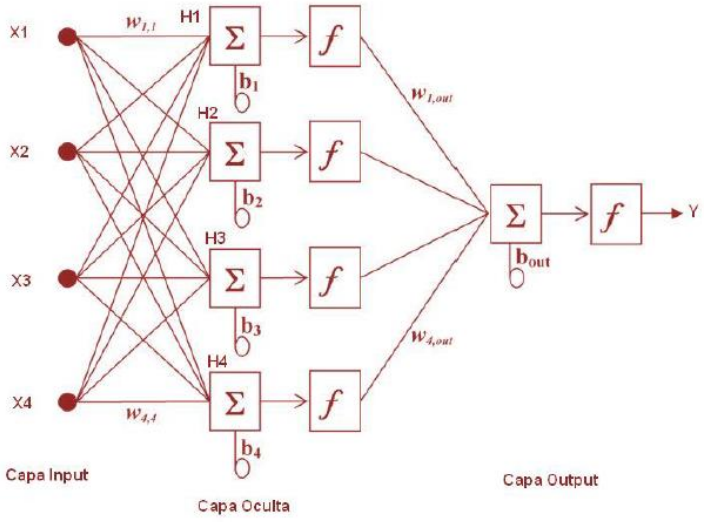


Figura 2.19: Arquitectura de una Red Neuronal para Regresión.

Los nodos de la capa de *input* pasan información a los nodos de la capa oculta mediante el "disparo" de funciones de activación. Los nodos de la capa oculta se "disparan" o permanecen inactivos dependiendo de los valores de sus entradas. Los nodos de las capas ocultas aplican funciones de ponderación (suma ponderada) y cuando el valor de un nodo o conjunto de nodos concreto de la capa oculta alcanza algún umbral, pasan la información a uno o más nodos de la capa *output*. La conexión entre los nodos o neuronas la establecen los pesos, que hacen el rol de parámetros a ajustar (de manera parecida a la regresión lineal). Los parámetros principales que se pueden configurar en Python con la librería Keras son los siguientes:

- ***Neurons***: Número de neuronas, es decir, los nodos que se encuentran en la capa oculta.
- ***Layer* o capa**: Número de capas ocultas.
- **Función de activación**: La función de activación decide si una neurona debe activarse o no calculando la suma ponderada. Las más utilizadas son *Relu*, *Tanh* (tangente hiperbólica) o *Softmax*. En este trabajo, se utilizará por defecto *Relu* y se probará *Tanh*. Dado que uno de los objetivos de este trabajo es la construcción un clasificador de sentimientos multiclase, la función de activación de la última capa será *Softmax*.
- **Función de pérdida o de coste**: La finalidad de las funciones de pérdida es calcular la cantidad que un modelo debe tratar de minimizar durante el entrenamiento. En el caso de una clasificación multiclase, se utilizará *Categorical Crossentropy Function*.
- **Algoritmo de optimización**: Es el encargado de optimizar los valores de los pesos o parámetros con la finalidad de minimizar la función de error. Los más comunes son *Adam*, *Levenberg-Marquardt* (LEVMAR), *Back Propagation* (*BPROP*), *Quasi-Newton* (*QUANEW*).
- ***Decay* de los pesos de la red o *learning rate***: Controla la velocidad de aprendizaje, es decir, la cantidad en la que se actualizan los pesos durante el entrenamiento.

  Drop out: Es una técnica que consiste en ignorar nodos o neuronas seleccionadas al azar durante el entrenamiento. Esto significa que su contribución a la activación de las neuronas de las capas posteriores se elimina de forma temporal.

Según [Port20], se puede calcular el número de nodos aproximados que tendrá la capa oculta. Para ello, se recomienda que al menos cada nodo debe tener 20 observaciones por parámetro (Ecuación (2.9)).

$$h(k+1)+h+1=\frac{nºobservaciones}{20} \tag{2.9}$$

donde, *h* es el número de nodos ocultos y *k* el número de nodos de entrada.

2.5. API de Twitter y *DocNow Hydrator*

El conjunto de datos inicial está formado por una lista de identificadores (*IDs*) de *tweets*. Por temas de privacidad [Song20], la mayoría de conjunto de datos de *Twitter* que se pueden extraer de la API de *Twitter* como de base de datos de terceros, son *tweets* "deshidratados". En vez de recopilar el contenido de los *tweets*, las geolocalizaciones, el texto y otra información sobre los *tweets*, lo que se obtiene es un archivo de texto plano que consiste en una lista de identificadores únicos de cada uno de los *tweets* (*tweets IDs*). Dichos identificadores permiten recuperar (con una cuenta de desarrollador) todos los metadatos del *tweet*, incluyendo el texto, convirtiéndose en una fuente significativa de investigación. El gran tamaño de los metadatos relacionados con los *tweets* es una de las razones principales por las que los conjuntos de datos sólo ofrecen identificadores "deshidratados". Así, un archivo que sólo proporciona una serie de números (*IDs*) es mucho más manejable que por ejemplo una hoja CSV con miles de *tweets* con sus correspondientes metadatos. Los datos en bruto de estos identificadores sólo pueden descargarse a través de una cuenta de desarrollador de *Twitter*. Una vez se cuente con las credenciales (API *Key*), se procede a "hidratar" el conjunto de datos.

Todas las APIs de *Twitter*, proporcionan datos codificados utilizando la Notación de Objetos de JavaScript (JSON). Los objetos JSON se basan en pares clave-valor, con atributos nombrados y valores asociados. Cada *tweet* tiene un autor, un mensaje, un *ID* único, una marca de tiempo de cuando fue publicado y a veces metadatos geográficos compartidos por el usuario. Cada usuario tiene un nombre de *Twitter*, un *ID*, un número de seguidores y, a menudo, una biografía de la cuenta. El la Figura 2.20, se muestra la estructura de estos objetos JSON y algunos de sus atributos.


```
{
  "created_at": "Thu Apr 06 15:24:15 +0000 2017",
  "id_str": "850006245121695744",
  "text": "1\/ Today we\u2019re sharing our vision for the future of the Twitter API platform!\nhttps:\/\/t.co\/XweGngmxlP",
  "user": {
    "id": 2244994945,
    "name": "Twitter Dev",
    "screen_name": "TwitterDev",
    "location": "Internet",
    "url": "https:\/\/dev.twitter.com\/",
    "description": "Your official source for Twitter Platform news, updates & events."
                   "Need technical help? Visit https:\/\/twittercommunity.com\/ \u2328\ufe0f #TapIntoTwitter"
  },
  "place": {
  },
  "entities": {
    "hashtags": [
    ],
    "urls": [
      {
        "url": "https:\/\/t.co\/XweGngmxlP",
        "unwound": {
          "url": "https:\/\/cards.twitter.com\/cards\/18ce53wgo4h\/3xo1c",
          "title": "Building the Future of the Twitter API Platform"
        }
      }
    ],
    "user_mentions": [
    ]
  }
}
```


Figura 2.20: Estructura del objeto JSON devuelto por la API de *Twitter*.

Existen seis clases de objeto: *Tweet object*, *User object*, *Geo objects*, *Entities object*, *Extended entities object* y *Example payloads*. En este trabajo, solo se utilizarán los tres primeros (*Tweet object*, *User object*, *Geo objects*).

En este trabajo se ha utilizado la aplicación *Hydrator* para "hidratar" el conjunto de datos de Twitter. DocNow Hydrator [Summ20] es un programa de código abierto utilizado para hidratar los identificadores de los *tweets*, pudiéndose descargarse desde *Github*. Antes de comenzar a utilizar *Hydrator* se debe vincular la cuenta de desarrollador *Twitter* en *Settings* (Configuración). Finalmente, se pueden guardar los *tweets* "hidratados" en formato JSON o CSV.

2.6. VAEX

VAEX [BrVe18] es una librería de Python para manejar conjuntos de datos tabulares extremadamente grandes, como catálogos astronómicos como el catálogo de Gaia o cualquier otro conjunto de datos regulares que se pueda estructurar en filas y columnas. Los cálculos rápidos de estadísticas en cuadrículas N-dimensionales regulares permiten el análisis y la visualización en el orden de mil millones de filas por segundo. Usan algoritmos de flujo, archivos mapeados en memoria y una política de copia en memoria cero (*zero memory*) para permitir la exploración de conjuntos de datos más grandes que la Memoria de Acceso Aleatorio (RAM), como por ejemplo algoritmos fuera del núcleo (*out-of-core*). Vaex permite realizar transformaciones (matemáticas) arbitrarias utilizando expresiones normales de Python y (un subconjunto de) funciones *Numpy* que se evalúan de forma "perezosa" (*lazy*) y se calculan cuando se necesitan en pequeños trozos, lo que evita el consumo de RAM. Las expresiones booleanas (que también se evalúan perezosamente) se pueden utilizar para explorar subconjuntos de los datos, denominados selecciones. VAEX utiliza una *Application Programming Interfaces* (API) de *DataFrame* similar a la de Pandas, lo que facilita la migración desde Pandas. La visualización es uno de los puntos clave de VAEX, y se realiza utilizando estadísticas en 1d (por ejemplo, histograma), en 2d (por ejemplo, histogramas en 2d con mapeo de colores) y en 3d (utilizando renderizado de volumen).

3. Estado del Arte

Los enfoques de PLN más avanzados hoy en día suelen adoptar algoritmos de ML y en particular de DL. Los intentos anteriores de procesamiento lingüístico se basaban en reglas, las cuales se sintetizan para llevar a cabo diferentes tareas de PLN. A diferencia del enfoque basado en reglas, el enfoque ML captura automáticamente las reglas a partir de los datos de entrenamiento. A continuación se exponen los diferentes trabajos relacionados con el AS y en concreto con el AS en redes sociales sobre el COVID-19. Además, se describirán los últimos métodos de representación de palabras en vectores de ML y en concreto de DL.

En [AgBM09] se presenta un clasificador para predecir la polaridad (negativa, neutra y positiva) contextual de frases subjetivas. La clasificación se lleva a cabo de manera automática mediante el diccionario *Dicionary of Affect in Language (*DAL*)* y ampliada a través de *WordNet*. Además, aumentan la puntuación léxica con un análisis de *n*-gramas para captar el efecto del contexto. Tuvieron como resultado un alto *accuracy (73%)* en la clasificación *multi-clase* de expresiones positivas, negativas y neutrales. Una de las limitaciones de su sistema de puntuación es que no maneja la polisemia, ya que a las palabras en DAL no se les proporciona el resto de las partes de la oración. Dado que uno de los principales obstáculos en el AS es la presencia de sarcasmos en los texto, en [Akul21] propusieron un método novedoso para detectar el sarcasmo en conversaciones textuales de varias plataformas de redes sociales y medios de comunicación online. Para ello, desarrollaron un modelo interpretable de DL que utiliza la autoatención o *self-attention* y *Gated Recurrent Unit* o GRU. El módulo de *self-attention* ayuda a identificar las palabras sarcásticas cruciales de la entrada y las unidades recurrentes aprenden las dependencias de largo alcance entre las palabras clave para clasificar mejor el texto de entrada. Además, las modelos entradas con su enfoque son fácilmente interpretables, permitiendo identificas las palabras sarcásticas en el texto de entrada que contribuyen a la puntuación final de la clasificación. El mecanismo de *self-attention* permite descubrir patrones en la entrada que son cruciales para resolver la tarea dada. En DL, *self-attention* es un mecanismo de atención para secuencias que ayuda a aprender la relación especifica de la tarea entre los diferentes elementos para producir una mejor visualización de la secuencia. En cuanto a los resultados, gracias a los pesos del módulo de atención aprendidos para interpretar el modelo entrenado, mostraron una alta efectividad para identificar palabras en el texto de entrada que proporcionan indicios de sarcasmo.

En [KuBJ19] compararon las herramientas de AS como VADER [HuGi14], *TextBlob* [Text19] y el *toolkit* NLTK. La finalidad era clasificar críticas de películas descargadas de la web http://www.rottentomatoes.com mediante las tres librerías mencionadas anteriormente. Los resultados experimentales confirmaron que VADER supera a TextBlob y NLTK. Si el sentimiento es la única finalidad que se planea hacer para un *corpus* procedente de redes sociales o *microblogging* y si necesita ser procesado rápidamente, entonces VADER es una mejor opción considerando un umbral de +/- 0,05.

En el marco de AS en *Twitter*, en [KoTA00] compararon los dos principales enfoques para el AS: los métodos basados en el léxico y el método de ML. Los resultados muestran que el método de ML (ambos métodos se describen en la sección 2.3.3) basado en clasificadores SVM y Naive Bayes supera al método del léxico. Para ello, presentan un nuevo método conjunto que emplea una puntuación de sentimiento basada en el léxico como característica de entrada para el enfoque de ML. El método combinado demostró producir clasificaciones más precisas. También mostraron que el empleo de un clasificador sensible a los costes (*cost-*

sensitive) para conjuntos de datos muy desequilibrados produce una mejora del rendimiento de la clasificación de sentimientos de hasta un 7%.

En cuanto a los trabajos relacionados con enfermedades infecciosas, en [Zarr14] se recolectaron 1.5000.000 de *tweets* mediante técnicas de Big Data con el objetivo de analizar las opiniones de las personas sobre el efecto de la infección por coronavirus (MERSCoV) en Arabia Saudí. Para etiquetar los sentimientos empelaron la técnica basada en diccionarios o léxicos. Otro trabajo interesante de [LLSS20], analizó la COVID-19 examinando las tendencias mundiales de *Twitter* sobre la pandemia, empleando más de 20.325.929 *tweets*. El enfoque aplicado por los autores fue el del lexicón o diccionarios y el algoritmo "*CrystalFeel*". Se analizaron cuatro emociones (miedo, ira, tristeza y alegría). Se observo como las emociones negativas fueron las dominantes en el resultado final.

Los autores de [KaMa20] crearon una herramienta llamada *CoronaVis*, cuya finalidad es la recolección de *tweets* para encontrar temas o tópicos mediante el Modelado de Tópicos o *Topic Modeling*. Desarrollaron una aplicación en tiempo real para monitorizar los *tweets* sobre la COVID-19 generados desde los EE. UU. También compartieron un conjunto de datos limpio y procesado llamado *CoronaVis Twitter* (centrado en EE. UU) disponible para la comunidad investigadora en https://github.com/mykabir/COVID19. En este trabajo, se analizaron más de 200 millones de *tweets* desde el 5 de marzo de 2020. Además, se realizó un análisis de frecuencias en el que se podían encontrar entre las palabras más frecuentes términos como: "*virus*", "*people*", "*cases*", "*trump*", "*deaths*". En cuanto al Modelado de Temas, seleccionaron temas para el estado de Nueva York, siendo "*cases*", "*mask*" y "*death*" los temas más "populares". Finalmente, se realizó un extenso AS (utilizando el clasificador de sentimientos del *toolkit* NLTK) de los *tweets* utilizando tres categorías (todos los *tweets*, *tweets* por perfil verificado y *tweets* por perfil no verificado). Los resultados demostraron un equilibrio entre los tres sentimientos (negativo, neutro y positivo) pero una puntuación más elevada (y por tanto una mayor "agresividad") en los *tweets* negativos en comparación con los *tweets* con sentimiento positivo y neutros.

En [ONMS20] analizaron los comentarios relacionados con la COVID-19 recogidos de seis plataformas diferentes de medios sociales. El objetivo fue identificar frases claves o *keyphrases* y sus respectivas polaridades o sentimientos de más de un 1 millón de comentarios seleccionados aleatoriamente. Tras aplicar técnicas de preprocesamiento y tras haber etiquetado las muestras con VADER, descubrieron 32 temas negativos y 20 positivos entornos a la pandemia. En [APSN20] desarrollaron un *dashboard* con el fin de acelerar los descubrimientos sobre la COVID-19. Utilizando el repositorio de la OMS, realizaron un análisis exploratorio en el que incluían las famosas *word cloud* (nubes de palabras) y un análisis de frecuencia. También, utilizaron los *word embbeding* para visualizar, gracias a la similitud del coseno, la relación entre los términos más frecuentes.

En cuanto al sentimiento o percepción general sobre las vacunas, en [RaRR20] se realizó un estudio de 9681 *tweets* relacionados con las vacunas en el periodo comprendido entre el 1 de enero de y el 5 de abril de 2019, periodo en el que hubo un brote de sarampión en EE. UU y se produjo un debate público. Utilizando AS, se agruparon los datos en temas o tópicos usando TF-IDF. Los análisis sugirieron que la mayoría (alrededor del 77%) de los *tweets* se centraban en la búsqueda de nuevas o mejores vacunas para enfermedades como el virus del ébola, el virus del papiloma humano (HPV) y la gripe. Del resto, aproximadamente la mitad se refería al reciente brote de sarampión en EE. UU y cerca de la mitad formaba parte de los debates entre partidarios y detractores de la vacunación contra el sarampión en particular. Si

bien estas cifras sugieren actualmente un papel relativamente pequeño de la desinformación sobre las vacunas, el concepto de inmunidad de grupo pone ese papel en contexto. Para el etiquetado del *corpus*, utilizaron VADER. Posteriormente, realizaron un análisis de *n*-gramas para visualizar las palabras más frecuentes. En cuanto a los resultados, un 43,3% de los *tweets* se clasificaron con sentimiento negativo, un 40,4% positivo y un 16,3% como neutro. El número de *tweets* con sentimientos negativos fue sólo ligeramente superior al de los que tenían positivos. Los sentimientos negativos se centraron en la relación entre la vacuna y el autismo, específicamente en como la vacuna es una causa del autismo y que la vacuna causa daños a los niños. Los sentimientos positivos se referían a la existencia de una vacuna contra el sarampión, a la eficacia de la vacuna y a que la vacuna salva vidas.

En cuanto a los trabajos relacionados con la construcción de clasificadores de sentimientos utilizando técnicas de ML y DL, cabe destacar los trabajos de [RKAR21][CBBP20][MüSK20]. En primer lugar, en [RKAR21] emplearon diferentes algoritmos de ML, como *Naive Bayes*, SVM, Regresión Logística y *LinearSVC* entre otros, sobre la matriz de términos TF-IDF (utilizando "unigramas", "bigramas" y "trigramas") y la matriz resultante del *word embbeding Doc2Vec*. Utilizaron dos conjuntos de datos, uno con 226.673 *tweets* y otro con 22.985 *tweets*. Las muestras fueron etiquetadas con la librería *TextBlob* y *AFFIN*. Para el primer conjunto de datos, el mejor modelo fue con RL utilizando "trigramas" con TF-IDF, con un *accuracy* del 81%y para el segundo conjunto, la RL fue el mejor modelo pero esta vez usando "bigramas" con TF-IDF, obteniendo un 75% de *accuracy*.

Los autores de [CBBP20] evaluaron diferentes clasificadores de ML de *tweets* sobre COVID-19. En este caso, se evaluaron los modelos mediante las siguientes métricas: *Precision*, *Recall*, *Accuracy* y *F1-Score*. Con la finalidad de intentar mejorar los resultados, además de los modelos clásicos de ML, emplearon las redes Long Short-Term Memory o LSTM de DL. Los resultados dieron como ganador a los modelos clásicos de ML (en particular el modelo *Extra Trees Classifiers*, que alcanzó un 93% de *accuracy*) utilizando una concatenación de la matriz de términos TF-IDF y la matriz de términos BOW (dando mejores resultados que VADER, TextBlob y las matrices TF-IF y BOW por separado).

Finalmente, en [MüSK20] presentaron *COVID-Twitter-BERT* (CT-BERT), un modelo basado en las redes *transfomers* [WDSC19], pre-entrenado con un gran *corpus* de mensajes de *Twitter* sobre el tema de la COVID-19. Este modelo creado por Google, a diferencia de otros *word embbeding*, genera "incrustaciones" que permiten tener múltiples (más de una) representaciones vectoriales (numéricas) para la misma palabra en función del contexto en el que se utiliza. Por lo tanto, las "incrustaciones" de BERT dependen del contexto. El modelo propuesto, muestra una mejora marginal entre el 10% y el 30% en comparación con el modelo base, *BERT-LARGE* [DCLT19] , en cinco conjuntos de datos de clasificación diferentes (*COVID-19 Category, Vaccine Sentiment, Maternal Vaccine Stance, Stanford Sentiment Treebank 2* y *Twitter Sentiment SemEval*). Los modelos pre-entrenados basados en *transfomers*, como CT-BERT, se entrenan en un dominio objetivo específico y pueden utilizarse para una amplia variedad de tareas de PLN, como la clasificación de texto, la respuesta a preguntas y los *chatbots*. CT-BERT está optimizado para su uso en contenidos COVID-19, en particular de las redes sociales. CT-BERT fue entrenado con un *corpus* de 160 millones de *tweets* sobre los coronavirus recogidos entre el 12 de enero y el 16 de abril de 2020. Los *tweets* se trataron como documentos individuales y se segmentaron en *tokens* con la librería *spaCy* de Python. Por último, compararon los resultados de los cinco conjuntos de datos con *BERT-LARGE* y *CT-BERT* utilizando la métrica *F1-Score*. La media de la mejora total entre *BERT-LARGE* y *CT-BERT* fue de un 17,57%.

4. Metodología

El desarrollo teórico de este trabajo se basa en la metodología SEMMA [Calv20]. Esta metodología se divide en 5 etapas:

1. **Muestreo (Sample):** En esta etapa se extrae una muestra lo suficientemente representativa de los datos, que en este trabajo la muestra es el *corpus*).
2. **Explorar (Explore):** En esta etapa se lleva a cabo un análisis univariante y multivariante para estudiar las relaciones entre los elementos de los datos e identificar las posibles "lagunas" de los mismos. Además, se analizan todos los factores que pueden influir en el resultado del trabajo, recurriendo en gran medida a la visualización de los datos, con la finalidad de realizar un análisis descriptivo de los datos.
3. **Modificar (Modify):** En esta etapa, se utiliza lo aprendido en la fase de exploración para corregir los posibles errores que se hayan detectado. En definitiva, los datos se analizan y se limpian, para luego pasar a la fase de modelado. Además en esta etapa se seleccionan y transforman las variables de cara a la modelización. En este trabajo se utilizan las técnicas de vectorización vistas anteriormente para la conversión de texto a número.
4. **Modelizar (Model):** Con los datos ya refinados y limpios, en la etapa de modelado se aplican una variedad de técnicas de ML con el fin de construir un Clasificador de Sentimientos. Se ejecutan varios algoritmos con varias configuraciones (rejillas) diferentes para encontrar el mejor modelo posible.
5. **Evaluar (Assess):** En esta etapa final de SEMMA, se evalúa el modelo con la finalidad de medir su utilidad y fiabilidad para el tema en cuestión. Los datos pueden ahora probarse y utilizarse para medir su rendimiento. En este sentido, la evaluación se realiza sobre datos de *test*, es decir, sobre los datos que se separan del conjunto de datos original (datos no vistos) con el objetivo de evaluar el rendimiento de los mejores modelos obtenidos.

4.1. Métricas y Métodos de Evaluación de los Resultados

A la hora de evaluar el rendimiento o la *performance* de cualquier modelo de ML, es necesario establecer las métricas adecuadas para la correcta evaluación e los resultados. En una clasificación multiclase, en comparación con una clasificación binaria, es necesario tener en cuenta las siguientes diferencias [Deve20]:

- *Micro-Average*: Calcula las métricas de forma global contando el total de verdaderos positivos, falsos negativos y falsos positivos (tiene en cuenta el número de muestras por etiqueta).
- *Macro-Average*: Calcula las métricas para cada etiqueta y trata todas las clases por igual (mismo peso). No tiene en cuenta el desequilibrio de las etiquetas.
- *Ponderada o weighted*: Calcula las métricas para cada etiqueta y encuentra su media ponderada por el soporte (el número de instancias verdaderas para cada etiqueta).

Para el objetivo II (Construcción de un Clasificador de Sentimientos Multiclase) puesto que no hay una clase de interés específica y el conjunto de datos de entrada está balanceado, se utilizan las siguientes métricas:

- ***ROC-AUC* (macro) y curva ROC** como principal métrica para evaluar los diferentes modelos: La curva ROC-AUC es una medida de rendimiento para los problemas de clasificación con distintos ajustes de umbral. La ROC es una curva de probabilidad y la AUC representa el grado o la medida de separabilidad. Indica en qué medida el modelo

es capaz de distinguir entre clases. Cuanto más alto sea el AUC, mejor será las predicciones del modelo (el modelo predecirá “ceros” como “ceros” y “unos” como “unos”. La curva ROC (Figura E.1) se representa con la TPR frente al FPR, donde el TPR está en el eje “*y*” y el FPR en el eje “*x*”.

Para más información sobre otras métricas ver el apartado E.

4.2. Remuestro y Optimización de modelos

Las técnicas de “remuestreo” son necesarias en la construcción de cualquier modelo de ML o DL con el objetivo de evitar el *overfitting*. En este sentido, se suelen emplear dos técnicas:

- **Training-Test repetido**: Mediante esta técnica, se dividen los datos de entrada en dos grupos: datos de entrenamiento y datos de *test*. De este modo, los modelos se entrenan con la parte de entrenamiento y se prueban con la partición de *test* con "semillas" aleatorias en repetidas ocasiones. Este método es adecuado cuando el número de observaciones disponibles es grande.
- **Validación cruzada o *Cross-Validation***: La validación cruzada es una técnica para evaluar los modelos de ML y DL que consiste en dividir los datos en varios subconjuntos (*k-folds*) de entrenamiento y evaluar el modelo con el subconjunto complementario de los datos (Figura 4.1). Mediante la validación cruzada, hay muchas posibilidades de detectar el sobreajuste con facilidad.

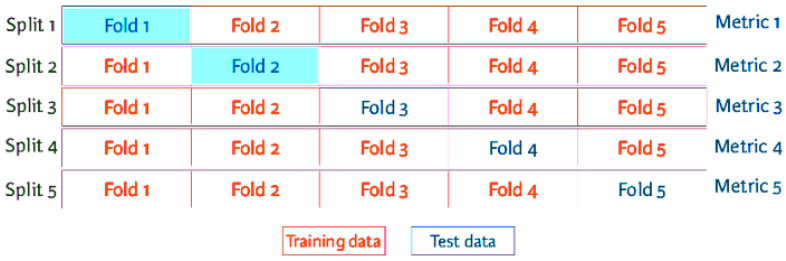


Figura 4.1: Ejemplo de *Cross-Validation* con 5 particiones o *k-fold* [14]

En este trabajo, se ha optado por el método de validación cruzada o *cross-validation* con 5 particiones o *k-folds* sin repetición sobre el 70% del *corpus*, puesto que se cuenta con un *corpus* de entrenamiento lo suficientemente elevado como para estar seguros de que los resultados obtenidos en cada partición no han sido fruto del azar, evitando así un posible sobre entrenamiento u *overfitting*.

En los algoritmos como XGBoost y Redes Neuronales se utiliza la técnica de *Early Stopping*. Esta técnica permite especificar un número arbitrario de iteraciones o *epochs* de entrenamiento y detener lo una vez el rendimiento del modelo deje de mejorar en el conjunto de datos de validación.

Finalmente, se configura una rejilla o *grid*, utilizando la librería *GridSearchCV* de Python o *RandomizeSearchCV* lo suficientemente variada para poder evaluar diferentes modelos con diferentes configuraciones (*hiperparámetros*) con la finalidad de determinar el mejor modelo posible. En caso de que el número de parámetros sea muy elevado, no se prueban todos los valores, sino que se prueban un número fijo de ajustes o *fits* de los parámetros a partir de las

distribuciones especificadas. El número de ajustes de parámetros que se prueban viene dado por *n_iter*.

4.3. Metodología para el Objetivo I. Percepción Pública de la Vacuna contra la Covid-19 en Twitter

Para el primer objetivo de este trabajo se utilizan las herramientas *Hydrator App* (sección 2.5) para la extracción de los metadatos de los *tweets* que forman el corpus. Posteriormente, se utiliza la librería VAEX vista en la sección 2.6 para el filtrado de los tweets que contengan los siguientes *keywords* o *hashtags*: "*vaccine*", "*vaccines*", "#vaccine", "*#vaccines*", "*corona vaccine*", "*corona vaccines*", "*#coronavaccine*", "*#coronavaccines*". Con los *tweets* filtrados, se emplean las técnicas preprocesamiento de texto vistas en las secciones 2.2.3 y 2.3 para etiquetar el *corpus* en tres clases (sentimiento negativo, neutro y positivo) de forma automática con el clasificador VADER visto en la sección 2.3.3. Tras el etiquetado de las muestras, se realiza un análisis de la Percepción de Pública de la Vacuna contra la COVID-19 y finalmente un análisis de frecuencia (*unigramas*, *bigramas* y *trigramas*) y modelado de tópicos (*Topic Modeling*) con SAS.

4.4. Metodología para el Objetivo II. Construcción de un Clasificador de Sentimientos

El segundo objetivo consiste en construir un clasificador de sentimientos multiclase (negativo, neutro y positivo) interpretable a partir de un subconjunto del *corpus* etiquetado del objetivo I. Dado que los algoritmos de ML funcionan con valores numéricos, se utilizan las técnicas de vectorización vistas en el apartado 2.4.3. (*Count Vectorizer* y *TF-IDF*). Además, es necesario realizar algún tipo de selección o reducción de variables sobre los dos conjuntos de datos generados por los dos tipos de vectorización mencionados anteriormente, ya que un número de características muy grande podrían suponer un conjunto de datos de entrada demasiado complejo para los algoritmos de ML (con el riesgo de sobreajuste que ello supone) y además suponer un alto coste en el tiempo de ejecución. Finalmente, se selecciona el mejor modelo en términos de número de variables, rendimiento e interpretabilidad.

5. Percepción Pública de la Vacunación contra la Covid-19 en Twitter

5.1. Fuente de Datos

El conjunto de datos original fue descargado de la web https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset entre el 15 de noviembre y el 16 de diciembre de 2020. El *feed* (contenido) de Twitter ha sido monitorizado en tiempo real recolectando *tweets* relacionados con el coronavirus utilizando más de 90 palabras clave y *hashtags* diferentes relacionados con la pandemia. La política de *Twitter* restringe la compartición de datos que no sean los identificadores de *tweets* o *IDs*, por lo tanto, sólo los *IDs* de los *tweets* se liberan a través de este conjunto de datos. Es necesario "hidratar" los *ID*s de los *tweets* para obtener los datos completos. El trabajo asociado a este conjunto de datos es [Lams20]. Los IDs de los tweets del conjunto de datos pertenecen a *tweets* en inglés.

Este conjunto de datos incluye archivos CSV (uno por mes) que contienen *IDs* y puntuaciones de sentimiento de los *tweets* relacionados con la pandemia COVID-19 etiquetados con la librería *Textblob*. El conjunto de datos descargado contiene 97.841.744 *IDs* y tiene un tamaño de aproximadamente de 3 GB, siendo el tiempo de carga en memoria (con Pandas) de 1 min aproximadamente (Figura H.2). Para poder realizar posteriormente la carga y análisis de los *tweets* "hidratados" se utilizó la librería VAEX (Sección 2.6). Para beneficiarse de las ventajas de VAEX, es necesario convertir el archivo CSV a HDF5. En la Figura H.2 se muestra como el tiempo de carga con VAEX fue de tan solo 3.99 ms (utilizando la técnica de mapeo de la memoria o *memory-mapping* nunca copiará todos los datos en memoria a menos que se solicite explícitamente, esto permite trabajar con conjuntos de datos del tamaño de su disco duro). A partir de este punto (y salvo tareas puntuales) se utiliza la librería VAEX para el filtrado de filas y para la exploración y análisis de los datos.

Como se ha mencionado anteriormente, para conseguir los datos en crudo, se "hidrataron" el conjunto de *IDs* mediante la aplicación *Hydrator App*. En la Figura H.3 se muestra el número de *tweets* finales recuperados mediante la aplicación *Hydrator App.* Como se observa en la figura anterior, no se han podido "hidratar" o recuperar aproximadamente el 21% de los *IDs* del conjunto inicial (muchos de los *tweets* han podido ser eliminados por los usuarios o las cuentas de los usuarios a los que pertenecen esos *tweets* hayan cambiado su nivel de privacidad).

Posteriormente, se ejecutó un script en Python que filtró los campos "hidratados" anteriormente seleccionando los campos que se muestran en la Tabla H.1 descartando los campos restantes.

5.2. Corpus

El *corpus* original está formado por un total de 77.332.093 *tweets*. Antes de realizar cualquier tarea, como el Análisis Exploratorio y del Análisis *n*-gramas¸ es necesario realizar un filtrado de *hashtags* y *keywords* de los *tweets* que estén relacionados con la vacuna, ya que el Objetivo I de este trabajo es realizar un estudio de los sentimientos hacia la vacuna contra la COVID-19. Para ello, primero se pasa a minúsculas el corpus y posteriormente se filtra mediante expresiones regulares las palabras clave o *keywords* y *hashtags* mencionados en la sección 4.3. Finalmente, se realiza una limpieza del *corpus* para eliminar duplicados, *retweets* y posibles *tweets* que provengan de cuentas *spam.* En la Tabla H.2 y

Tabla H.3 se muestra el número de observaciones existentes por cada columna y el número de datos faltantes o *missings*.

Tras realizar el filtrado de los datos mediante expresiones regulares, el número de observaciones finales del *DataFrame* fue de 10.523.999. Como se observa en la Tabla H.4 y Tabla H.5, las únicas columnas con valores faltantes o *missings* fueron: *"location*", "*coordinates_0*", "*coordinates_1*", "*place_name*", "*place_country_code*" y "*source*". Las variables nombradas anteriormente contienen datos de geolocalización (salvo "*source*") y por tanto solo estan disponibles cuando los usuarios hayan activado o compartido campo en el momento de realizar el *tweet*.

5.3. Preprocesamiento y Limpieza del Corpus

Tras haber realizado el filtrado de *tweets*, es necesario realizar una pequeña limpieza en busca de duplicados y posibles *retweets* dentro del conjunto de datos.

1. **Eliminar Duplicados**: Mediante la función *drop_duplicates ()* de VAEX se eliminaron las observaciones repetidas. Para ello, se buscan las observaciones con un "*id_str*" repetido. Tras la ejecución de la función, el número de observaciones se redujo de 10.523.999 a 10.523.187 tweets.
2. **Eliminar Retweets**: Este paso se divide en dos, por un lado nos quedaremos con las observaciones cuya columna "retweeted_status_id_str" sean "*missings*" ya que estos *tweets* no han sido "retweetados" en ningún momento (y por tanto son *tweets* originales). Tras este paso, el número de *tweets* fue de 4.032.929. Por otro lado, de los 6.490.258 *tweets* restantes que sabemos que son *retweets*, eliminamos primero las observaciones cuyos "retweeted_status_id_str" estén repetidos, dando lugar a 490.658 *tweets*. Finalmente, buscamos y eliminamos los "*id_str*" restantes de la columna "*retweeted_status_id_str*" en la columna "*id_str*" (tweets originales) para asegurarnos que esos *retweets* únicos no están presentes en los 4.032.929 *tweets* que sabemos que son *tweets* originales. Finalmente, de los 490.658 tweets *retweets* únicos, sólo 44.440 no estaban presentes en el conjunto de *tweets* originales, por lo que el total de *tweets* quedó en 4.077.369.
3. **Eliminar spam:** Es necesario analizar aquellos *tweets* que podrían pertenecer a cuentas *spam* (cuentas que aprovechen los *keywords* o *hashtags* que son tendencia en un momento dado para *Twitter* sobre otros temas). Tras la eliminación de los duplicados y de los *retweets* vamos a ejecutar un "*join*" (por la columna "*id_str*") del *DataFrame* obtenido del filtrado anterior con el conjunto de datos original (formado por dos columnas, "*id_str*" y el *"score*" asociado). En la Tabla 5.1 se muestra un pequeño resumen estadístico de la nueva columna "*sentiment_score*".

Tabla 5.1: Resumen estadístico de la columna *sentiment score*.

	Sentiment_score
Count	4.077.369
Missings	0
Mean	0.082367
Std	0.251441
Min	-1.0
Máx	1.0

En la Figura 5.1 se resumen los pasos seguidos para la eliminación de duplicados y de *retweets*.

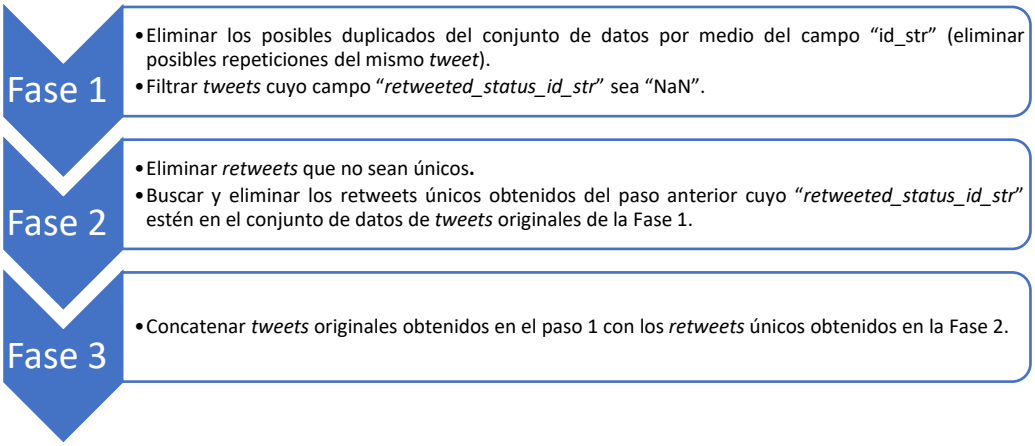


Figura 5.1: Fases para la eliminación de duplicados y de *retweets*.

Como se observa en la Tabla 5.1, la media de las puntuaciones de los sentimientos del calculados mediante la librería *TextBlob* (Sección 2.3.3) y tras un pequeño preprocesamiento según los autores [Lams20], es ligeramente superior a cero, por lo que o bien la tenemos una mayor número de *tweets* positivos o *tweets* con una grado de positividad más elevada (cerca de 1). Posteriormente, se comparan los sentimientos obtenidos con VADER con los sentimientos originales obtenidos con *TextBlob* y se realiza un análisis descriptivo más detallado.

Para detectar posibles cuentas *spam*, vamos a agrupar los usuarios por la columna "*user_screen_name*" y calcularemos el número de *tweets* total por usuario al igual que la media de las puntuaciones obtenidas por *TextBlob*. Como se observa en la Tabla H.9, se cuenta con un total de 1.457.046 usuarios que han "tweetado" al menos un *tweet*, un máximo de 3208 y una media total de *tweets* por usuario de 2.79. Como vemos, es necesario realizar un análisis con la finalidad de encontrar cuentas con un número de *tweets* aproximadamente por encima del doble de la desviación estándar, es decir, nos quedaremos con el 95% de los datos.

El siguiente paso consiste en calcular la media y la desviación típica de los *tweets* por usuario. Además, no se consideran *spam* aquellos usuarios que hayan "tweetado" una sola vez, ya que en un periodo de un mes, sería lo esperable. De los 1.457.046 usuarios, 906.502 "tweetado" solo una vez, es decir, el 62.22% de los usuarios solo aparecen una sola vez en el corpus mientras que el 38% (550.544) aparece 2 o más veces, siendo estos últimos las cuentas de usuario a analizar. En la Tabla H.10 se muestra el resumen estadístico de la agrupación por usuarios con su media de *tweets* por día. Por ejemplo, se observa como la media de *tweets* correspondiente al usuario que ha "tweetado" 3208 veces en un mes, es de 100.25, lo cual nos hace sospechar que podría tratarse de una cuenta *spam*.

Finalmente, se filtran los usuarios cuya media de *tweets*/día superan el doble de la media más dos veces la desviación típica.

$$df_{spam} = mean\_tweets\_by\_day > (mean\_tweets\_by\_day.mean() + 2 * (mean\_tweets\_by\_day.std())$$

En la Tabla H.11 se muestran el número de cuentas y el número de *tweets* tras ejecutar el filtrado anterior. En ella, se observa cómo se ha reducido el número de usuarios sospechosos de ser *spam* (6984), el mínimo y el máximo de las puntuaciones de los sentimientos (tanto

negativo como positivo). Además, la media de *tweets* diarios ha subido de 5.75 (Tabla H.10) a 94.13 (Tabla H.11), lo cual nos indica que las cuentas de usuarios del ultimo filtrado o bien son cuentas de perfiles de noticias o pueden ser cuentas *spam*. El número de *tweets* correspondientes a los 6984 usuarios es de 657.407.

En la

Tabla 5.2 se muestran el top 15 de cuentas ordenadas por la media de *tweets* diarios mayor a menor. Por ejemplo, la cuenta “PharmacyFridge” sigue a cero personas y tiene cero seguidores y los *tweets* consisten en información sobre temperaturas de refrigeradores de vacunas (Figura 5.2).

Tabla 5.2: Top 10 de posibles cuentas *spam*

User_screen_name	Tweets	Sentiment_score	Mean_tweets_by_day
PharmacyFridge	3208	0.0678055	100.25
realBenTalks	3073	0.377663	96.0312
africanaffairs	1967	0.0509096	61.4688
newsfilterio	1566	0.0619295	48.9375
FinTechZoom	1428	0.133566	44.625
Pehal_News	1360	0.0747232	42.5
iWeller_health	1321	0.0802017	41.2812
ExBulletinUk	1315	0.0755599	41.0938
mlnangalama	1292	0.117993	40.375
AJBlackston	1231	0.0771499	38.4688
MKucala	1202	0.103885	37.5625
DigbyKale	1194	0.0080521	37.3125
KevinScott4all	1157	-0.0311356	36.1562
ZyiteGadgets	1149	0.0823815	35.9062
dev_discourse	1134	0.0834753	35.4375

created_at	text	user_screen_name
Sun Dec 13 21:43:16 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" return...	PharmacyFridge
Sun Dec 13 13:53:08 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" is too...	PharmacyFridge
Sun Dec 13 13:30:07 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" is too...	PharmacyFridge
Sun Dec 13 23:17:09 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" is too...	PharmacyFridge
Sun Dec 13 03:42:28 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" return...	PharmacyFridge
...	...	...
Fri Dec 11 21:01:21 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" return...	PharmacyFridge
Tue Dec 15 19:51:13 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" return...	PharmacyFridge
Fri Dec 11 14:53:57 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" return...	PharmacyFridge
Sun Dec 13 08:08:28 +0000 2020	'Temperature at "Vaccine Fridge [Bromic]" is too...	PharmacyFridge
Thu Dec 10 17:09:51 +0000 2020	'Temperature at "Vaccine Fridge 2" returned to n...	PharmacyFridge

Figura 5.2: Tweets de la cuenta "PharmacyFridge"

En la tabla anterior podemos observar como el sentimiento medio de los *tweets*, en su mayoría, están cerca de 0 (sentimiento neutro), lo cual tiene sentido, ya que la mayoría de las cuentas son portales de noticias, como por ejemplo “FinTechZoom”, “Pehal_News”, “ExBulletinUk”, etc.

En la Figura 5.6 se muestra como a partir de 20 *tweets*/día el número de usuarios o cuentas baja notablemente, por lo que eliminaremos los *tweets* pertenecientes a los usuarios que cuya media de tweets sea superior a 20. El número de usuarios que tienen un promedio superior a 20 *tweets* al día es de 48, correspondientes a 50953 *tweets* u observaciones del conjunto de datos.

Finalmente, se eliminó los 50.953 *tweets* de los 4.077.369 *tweets* sin duplicados, obteniendo un conjunto final de 4.026.416 *tweets* sin duplicados y sin los *tweets* pertenecientes a las cuentas analizadas anteriormente.

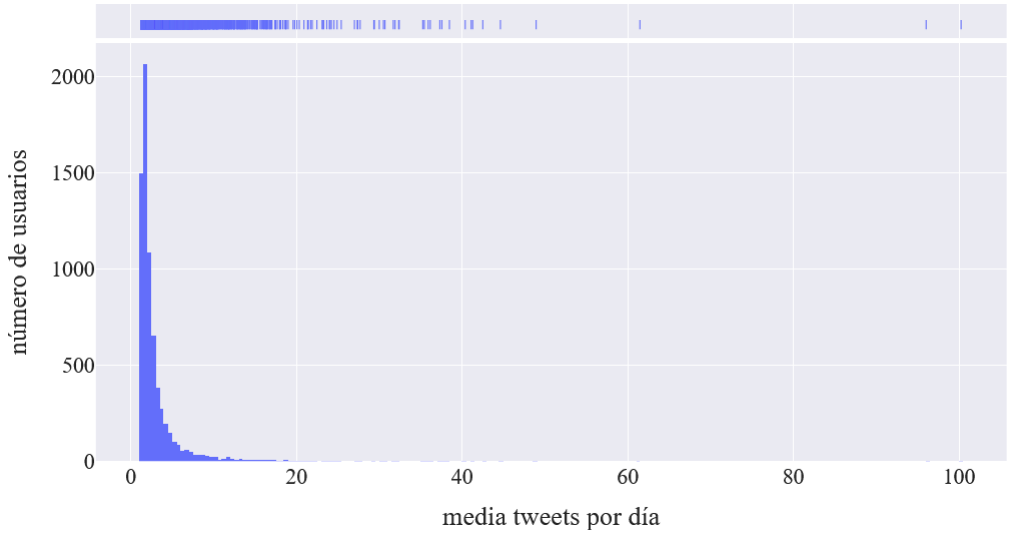


Figura 5.6: Número de usuarios cuya media de tweets al día es superior al 95% de los usuarios.

5.4. Etiquetado del Corpus

Aunque el *corpus* del conjunto de datos utilizado en este trabajo ha sido etiquetado previamente con la librería *TextBlob*, vamos a volver a etiquetarlo utilizando la librería VADER por dos motivos:

- Como se explicó en la sección 2.3.3, la librería VADER está optimizada etiquetar textos provenientes de redes sociales, como es nuestro caso.
- Control del preprocesamiento. Realizar una comparación entre diferentes preprocesamientos con la finalidad de ver si estos afectan a la puntuación final del sentimiento. Los autores de [Lams20] realizaron un preprocesamiento básico, como la eliminación de "#", "@" , URL, espacios en blanco adicionales así cómo los saltos de párrafo. Los signos de puntuación, emojis y números se conservaron.

Como se ha mencionado anteriormente, VADER está optimizado para procesar texto de redes sociales y manejar signos de puntuación, emojis, mayúsculas (la presencia de mayúsculas puede acentuar más la puntuación del sentimiento), etc. Los tres tipos de preprocesamiento se muestran en la Tabla 5.3. Podemos ver como el preprocesamiento 1 realiza el tratamiento de etiquetas HTML. El preprocesamiento 2 es el más exhaustivo, ya que conlleva muchas de las tareas mencionadas en la sección 2.2.3, como por ejemplo la expansión de contracciones, conversión de número a texto y eliminación de signos de puntuación entre otros. proceso de corrección ortográfica suponía a la alta carga computacional por lo que finalmente no se realizó. Finalmente, el preprocesamiento 3 es el texto en “crudo”.

Tabla 5.3: Tipos de preprocesamiento

	Preprocesamiento 1	Preprocesamiento 2	Preprocesamiento 3
Pasar texto a minúsculas	NO	NO	NO
Eliminación de "#" y "RT"	SI	SI	NO
Eliminación de "@" y URLs	SI	SI	NO
Decodificar texto y etiquetas HTML	SI	SI	NO
Expandir contracciones	NO	SI	NO
Eliminar acentos	NO	SI	NO
Conversión de abreviaturas	NO	SI	NO
Eliminar signos de puntuación	NO	SI	NO
Número a palabras	NO	SI	NO
Eliminar palabras con un solo carácter	NO	SI	NO
Eliminar espacios en blanco adicionales	SI	SI	SI

Como podemos observar en las figuras Figura H.4, Figura H.5 y Figura H.6, se muestran las primeras diez filas de los tres conjuntos de datos "preprocesados" con la puntuación (o *compound*) obtenida con VADER, el texto original (*text*), el texto "preprocesado" (*processed_text_vader*) y la puntuación con *TextBlob* (*sentiment_score*). Se observa como VADER hace un buen trabajo aún con el texto en "crudo" (Figura H.6), ya que por ejemplo en las primeras 10 observaciones se ha obtenido la misma puntuación o *compound* con los tres tipos de preprocesamiento. Dado que no hay mucha diferencia entre los tres tipos de preprocesamiento, se elige el preprocesamiento 1, pues no conlleva un preprocesamiento tan exhaustivo como el preprocesamiento 2 pero al menos elimina ciertos símbolos especiales, como "RT", "#" y "@" y etiquetas HTML asegurando un mínimo de limpieza en el texto comparado con el preprocesamiento 3.

Finalmente, se observa cómo hay ciertas diferencias entre las puntuaciones de VADER y *TextBlob*. Por ejemplo, el *tweet* "*State and local government aid, REGARDLESS OF STATE, will be used to prop up vaccine dictators and their minions and plug holes in MASSIVE state pension deficits, not pave roads and fix schools. Worst run states will need the most money*."("*La ayuda del gobierno estatal y local, INDEPENDIENTEMENTE DEL ESTADO, se utilizará para apuntalar a los dictadores de las vacunas y sus secuaces y tapar agujeros en los déficits de las pensiones estatales MASIVAS, no para pavimentar carreteras y reparar escuelas. Los estados peor gestionados necesitarán más dinero*") tiene connotaciones negativas, pero la presencia de palabras en mayúsculas acentúa más este sentimiento si empleamos la librería VADER (-0.6249) en comparación con TextBlob (-0.125).

5.5. Análisis Exploratorio del conjunto de datos

En esta sección, se realizará un análisis exploratorio de las principales variables de la Tabla H.1. Se analizarán fecha, longitud, latitud, ciudad, etc., en relación con el sentimiento hacia la vacunada asociado a cada mensaje de Twitter contra la COVID-19 calculado en la sección 5.4. Además, gracias la ingeniería de características o *feature engineering* (el proceso de creación de características a partir de datos de texto sin procesar) se analizarán nuevas

variables con el objetivo de extraer información valiosa en términos descriptivos. Quizás se tiene la suerte de descubrir que una categoría es sistemáticamente más larga que otra y la longitud del texto se podría añadir al conjunto de datos que sirven como input para construcción del clasificador de sentimientos.

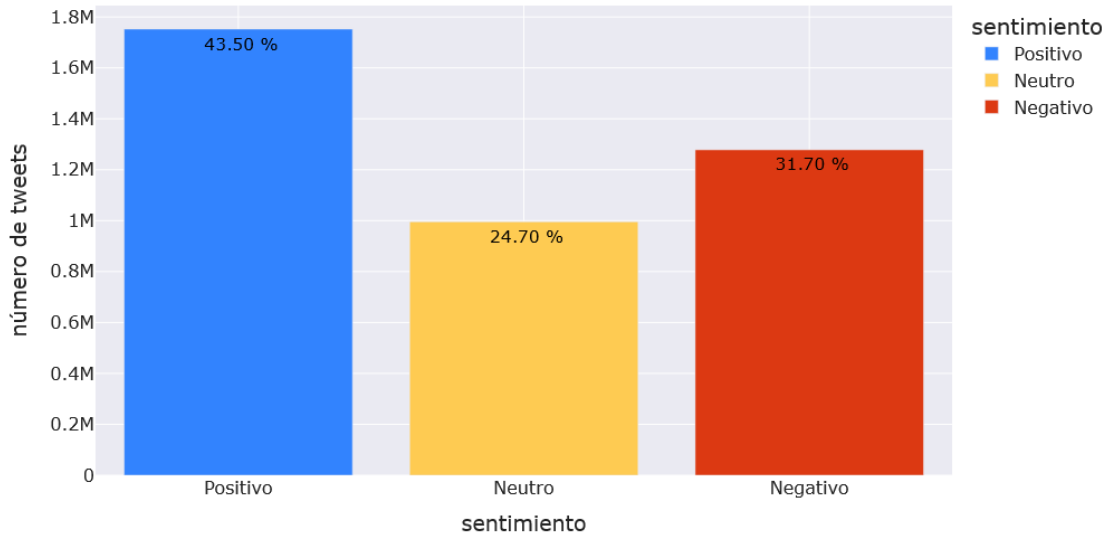
5.5.1. Estudio de la Variable Objetivo

Vamos a comenzar comparando la distribución de la variable objetivo obtenida con VADER con la obtenida con *TextBlob*. Para ello, se codificó la variable objetivo de la siguiente manera:

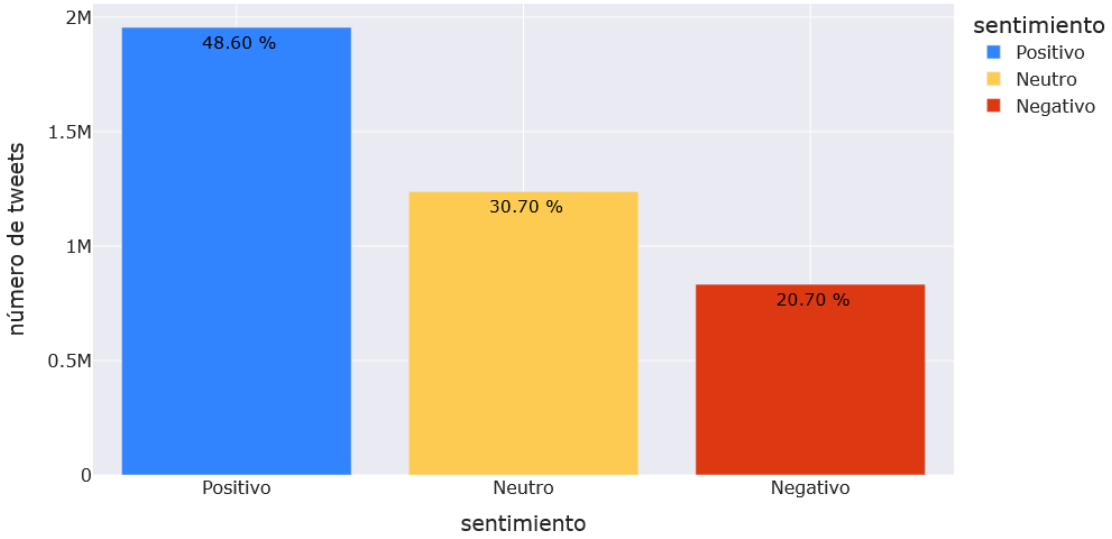
- Para la puntuación obtenida con VADER, se codificó como sentimiento negativo todas las observaciones con una puntuación <= - 0.05, neutro entre > -0.05 y < 0.05 y positivo >= 0.05 (según los autores de [HuGi14]).
- Para la puntuación original obtenida con *TextBlob*, se codificaron como sentimiento negativo todas las observaciones con una puntuación < 0, como neutro las puntuaciones = 0 y como positivo las observaciones > 0 (según los autores de [Text19]).

En la Tabla H.6, Tabla H.7 y Tabla H.8 se muestra un resumen descriptivo del conjunto de datos final.

Como se observa en la Figura 5.3 (a), con el etiquetado de VADER se han obtenido 1.752.383 (43.52 %) de *tweets* positivos, 995.701 (24.729 %) de *tweets* neutros y 1.278.332 (31.749 %) de *tweets* negativos. Por el contrario, en la Figura 5.3 (b) se han obtenido 1.810.986 (44.978%) *tweets* positivos, 1.500.641 (37.27 %) *tweets* neutros y 714.789 (17.75 %) *tweets* negativos. En general, el número de observaciones o *tweets* positivos es el mismo utilizando ambas técnicas pero VADER hace un mejor trabajo a la hora de identificar sentimientos negativos comparado con *TextBlob*. por lo que de aquí en adelante tanto para el Objetivo I como para el Objetivo II se utilizara el etiquetado obtenido con VADER (variable objetivo "*vader_sentiment*"), ya que como se ha dicho anteriormente está optimizada para redes sociales.



a) VADER



b) *TextBlob*

Figura 5.3: Distribución de la variable objetivo con VADER y *TextBlob*.

En la Figura 5.4 se muestra la distribución de las puntuaciones de los sentimientos obtenida con VADER. Como vemos, la mayoría de las observaciones con sentimiento neutro se localizan en el rango [0,0.05) (alrededor de un 97%) mientras unas pocas (alrededor del 3%) en el rango (-0.05 -0). En cuanto a las observaciones con sentimiento positivo, siguen prácticamente una distribución normal, mientras que las observaciones negativas presentan cierta asimetría negativa (o a la izquierda). La media de las puntuaciones en los *tweets* positivos es de 0.52105, -0.52265 para los *tweets* negativos y prácticamente 0 (0.003) en los *tweets* neutros. Por tanto, según las puntuaciones obtenidas con VADER, no se aprecia una gran positividad entre los *tweets* positivos (valores cercanos a 1) ni una gran negatividad en entre los *tweets* negativos (valores cercanos a -1), aunque se aprecia una mayor intensidad entre los *tweets* negativos (mayor número de *tweets* cerca de -1) en comparación con los *tweets* con sentimiento positivos en relación hacia la vacunación contra la COVID-19.

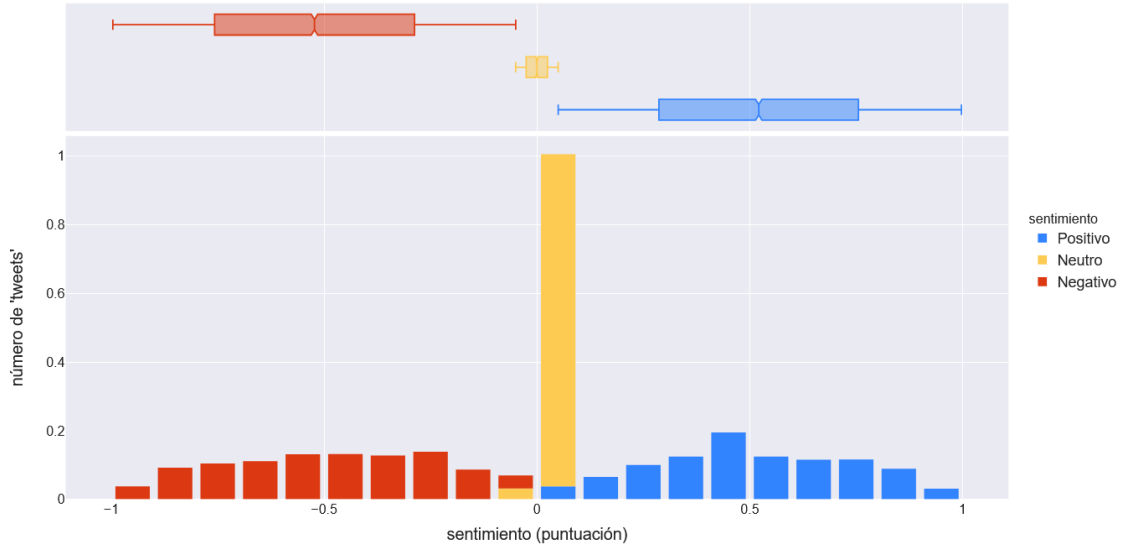


Figura 5.4: Distribución de la variable "*compound*" (puntuación de los sentimientos)

A continuación, en la Figura 5.5 se muestra en un diagrama de barras de la distribución de los *tweets* y los sentimientos por fechas (desde el día 15 de noviembre hasta el 16 de diciembre).

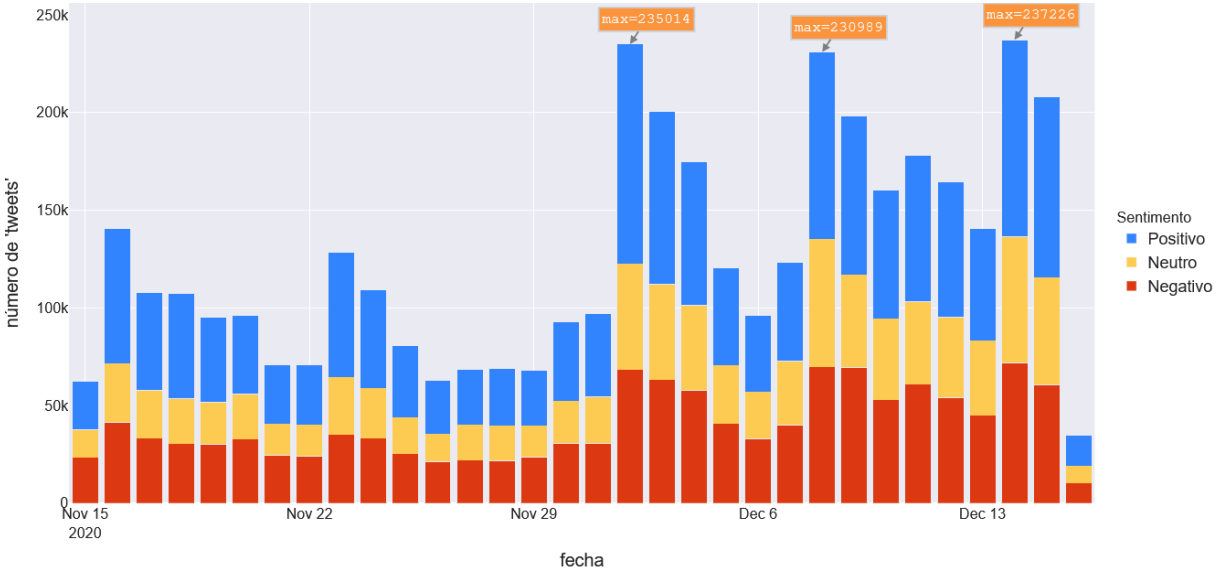


Figura 5.5: Distribución de sentimientos por fecha (I).

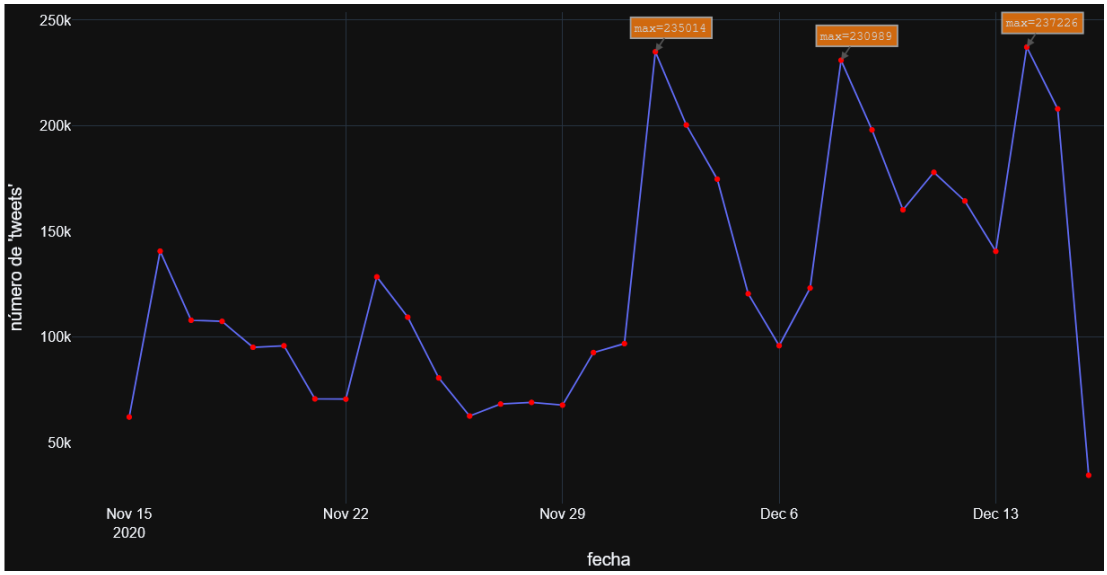


Figura 5.6: Distribución de sentimientos por fecha (II).

En la Figura 5.6, podemos ver un gráfico de serie temporal los *tweets* recolectados entre las fechas dichas anteriormente. Se ha señalado los 3 días con mayor tráfico. A continuación, se muestran los eventos principales relacionados con la vacunación en esos días (en la Tabla H.12 se muestra con mayor detalle):

- **Día 2 de diciembre del 2020**: Reino Unido aprobó la vacuna Pfizer / BioNTech Covid[15].
- **Día 8 de diciembre del 2020**: Primera persona en recibir la vacuna Pfizer en Reino Unido[16].
- **Día 14 de diciembre del 2020**: Un trabajador sanitario de Nueva York entre los primeros en recibir la vacuna COVID-19[17].

Como vemos, este incremento de *tweets* puede deberse a los eventos mencionados anteriormente. En general, se observa como a partir de diciembre hay un incremento en el número de *tweets* por día, coincidiendo con la aprobación de la vacuna y las primeras inoculaciones en diversos países. En la Figura H.7 se puede observar de mejor manera si se ha producido alguna variación o "picos" en los sentimientos durante el periodo analizado. Por ejemplo, se observa como el mayor incremento de *tweets* positivos (color azul) se producen los días 16 de noviembre,18 de noviembre y 23 de noviembre y (en la Tabla H.13 se muestran los porcentajes de forma más detallada). Posteriormente, se buscaron las noticias más relevantes que pudieron dar lugar a esta variación en las fechas mencionadas anteriormente. Por ejemplo, el 16 de noviembre Moderna anunció que su vacuna contra el coronavirus tiene una 'efectividad del 94.5% contra el virus', lo que la convierte en la segunda vacuna en los EE. UU que tiene una tasa de éxito asombrosamente alta[18].El 18 de noviembre se anuncia la que vacuna de Pfizer es '95% efectiva y segura' [19]. Finalmente, el 23 de noviembre, AstraZeneca comunicó que su candidata a vacuna contra el coronavirus tiene una 'efectividad del 70% en promedio'[20]. Por el contrario, los días con un mayor número de *tweets* negativos fueron los días 15 de noviembre, 21 de noviembre y 9 de diciembre (en la Tabla H.13 se muestra los porcentajes de forma más detallada). Entre las noticias más relevantes del 9 de diciembre relacionada con la vacuna cabe destacar aprobación del uso de la vacuna de Pfizer en Canadá[21]. En cuanto al resto de días, no se ha encontrado ninguna noticia relevante

5.5.2. Estudio de la Variable Place Country

A continuación, se van a agrupar aquellos *tweets* (86197 observaciones) que hayan activado la opción de localización "place_country" (Tabla H.7). Para ello, primero se san seleccionado los quince primeros países ordenados de mayor a menor por número de *tweets*. Dado que los *tweets* analizados son en inglés, en la Tabla 5.4 se observa como casi el 84% de los *tweets* provienen de EE. UU, Reino Unido y Canadá.

Tabla 5.4: Países con mayor número de *tweets* ordenados de mayor a menor (Top 15).

Country	Tweets
United States	49580
United Kingdom	16789
Canada	5932
India	3674
Australia	1188
Ireland	1169
South Africa	1057
Republic of the Philippines	434
Germany	383
Nigeria	329
Pakistan	307
Spain	287
Kenya	280
Malaysia	232
Mexico	207

En la Figura 5.7 se muestra en un diagrama de violín la distribución de los sentimientos por países (Top 10 de los países con mayor número de *tweets*). Vemos como la mayoría de los países tienen una mediana cerca de 0 (sentimiento ligeramente positivo). Las secciones más

amplias del diagrama de violín representan una mayor probabilidad de que los miembros de la población adopten el valor dado, por el contrario, las secciones más delgadas representan una probabilidad menor. Vemos como los *tweets* de EE. UU presentan una sección más amplia cerca de 0 comparado con el resto de los países mientras que muestra una sección más delgada alrededor de 0.25. También hay que destacar la presencia de una sección relativamente amplia en torno a valores cercanos a 0.5 o superiores, lo cual nos indica un grado alto de positividad.

La distribución de Reino Unido y Australia es parecida a la de EE. UU pero con una positividad mayor, ya que la sección cercana a 0 es más estrecha y con una sección un poco más amplia en los valores por encima de 0.5. El caso de India es interesante, ya que su mediana es superior a la del resto de países, lo cual nos indica un grado de positividad superior con respecto al resto de países. En el caso de España, el sentimiento es neutro o ligeramente positivo, ya que podemos ver como la sección de los valores negativos es más delgada que la sección cercana a 0. En general, podemos concluir que las puntuaciones de los *tweets* se encuentran cercanos a la neutralidad o a la positividad, salvo Australia, que presenta una distribución parecida entre los valores negativos y positivos.

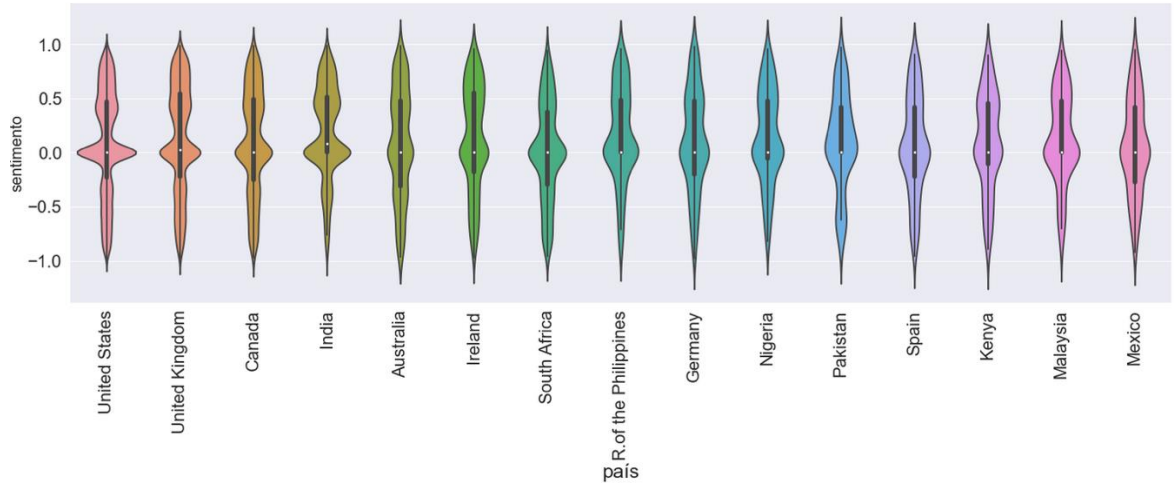


Figura 5.7: Diagrama de Violín de los sentimientos por países

Se ha incorporado un Diagrama de violín (Figura H.8) en el Anexo con la media del sentimiento por cada país.

5.5.3. Estudio de la Variable User Location

De los 2.735.125 de *tweets* analizados que tienen un valor en la columna "*user_location*", casi 13% (348.647) provienen de EE. UU. Dado que el conjunto de datos fue recolectado entre los meses de noviembre y diciembre, vamos a analizar los sentimientos por estado de los *tweets* cuya localización de usuario sea EE. UU. Además, como el periodo analizado fueron las elecciones presenciales de EE. UU (3 de noviembre) sería interesante comparar los resultados del estudio de los sentimientos por cada uno de los estados con los resultados de las elecciones. En este sentido, se comparó los sentimientos obtenidos por cada estado, tanto por la media como por el porcentaje total, con el porcentaje de vacunación a fecha de 7 de junio del 2021 (Figura H.10). El objetivo de esta comparación puede darnos una idea del rendimiento de la herramienta VADER y del preprocesamiento realizado y por lo tanto poder predecir qué estados de EE. UU pueden ralentizar el ritmo de vacunación, ya que como se

indica en el artículo de la CNN , casi un 36% de los adultos menores de 35 años no piensan vacunarse.

Para ello, haremos uso de la librería *city_to_state* de Python para extraer la abreviatura del estado de la columna "*user_location*". En esta columna puede venir tanto el nombre de la ciudad donde reside el usuario del *tweet* como el estado y el país. En la Figura 5.8 se muestra el mapa de EE. UU en el que cada estado ha sido coloreado según la media de los sentimientos (variable "*compound*"). Se observa como los estados del noroeste tienen un sentimiento positivo promedio mayor (colores azulados) y por tanto una percepción positiva hacia la vacunación, si lo comparamos con los estados del suroeste y del centro del país (colores más rojizos). Por otro lado, en Figura H.9 se muestran los resultados de las elecciones presidenciales del 2020, en el que se aprecia como en los estados del sur y suroeste ha ganado el Partido Republicano mientras que en los estados del oeste y noroeste del país ha ganado el Partido Demócrata (para más detalles, ver la Tabla H.14 situada en el Anexo).

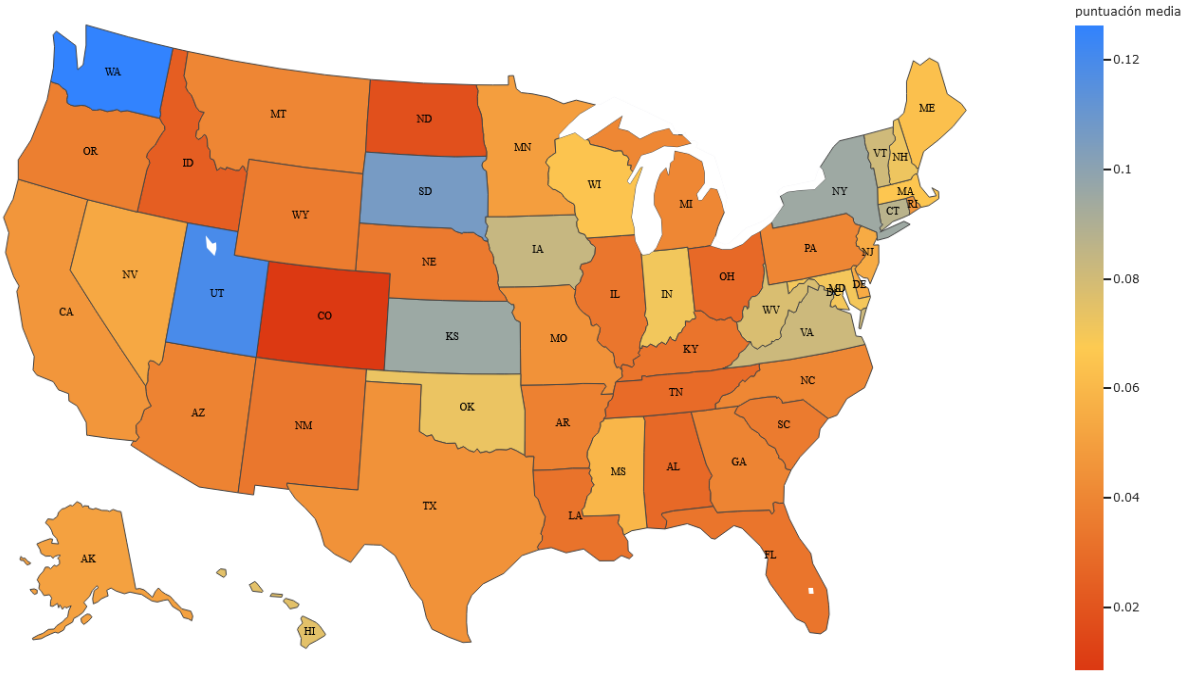


Figura 5.8: Percepción pública de la vacuna contra la COVID-19 en EEUU.

El cálculo de la media de las puntuaciones de los sentimientos mostrados en la Figura 5.8 puede mostrar una realidad distorsionada, ya que la presencia de valores muy positivos puede dar como resultado una media más alta. Esto no permite apreciar bien el número de observaciones negativas. Además de esto y dado que las elecciones presidenciales de EE. UU fueron el 3 de noviembre de 2020, se compararon los resultados electorales por estado con los sentimientos (porcentaje de tweets positivos y negativos) sobre la vacuna. En la Tabla H.21 y Tabla H.22, se muestran los 15 estados con mayor porcentaje de *tweets* negativos y positivos respectivamente. Los estados en rojo corresponden a los estados donde ganó el Partido Republicano y en azul el Partido Demócrata. De los 15 estados con mayor negatividad hacia la vacuna, en 12 ganaron los republicanos por solo 3 demócratas. En cuanto a los estados con mayor positividad, en 10 estados han ganado los demócratas y 5 los republicanos. Hay que resaltar la presencia de Utah y South Dakota como los dos estados

con mayor porcentaje de *tweets* positivos, siendo estos estados donde ganaron los republicanos.

En la Figura H.10 se muestra el ritmo de la vacunación en EE. UU a fecha de 7 de junio del 2020. Podemos ver como los estados con un mayor porcentaje de pauta completa (verde oscuro) son estados en los que ha ganado el Partido Demócrata (Figura H.9), mientras que los estados donde ha ganado el Partido Republicano tienen un porcentaje entorno al 34%-38% de personas vacunadas con pauta completa.

Finalmente, con el objetivo de comparar los sentimientos obtenidos por VADER con el ritmo de vacunación por estados (https://www.nytimes.com/interactive/2020/us/covid-19-vaccine-doses.html), en la Tabla H.23 y en Tabla H.24 se han ordenado de mayor a menor los 10 estados en el que la diferencia entre el porcentaje de *tweets* positivos y negativos es mayor y menor respectivamente. Además, se ha coloreado Tabla H.23 y Tabla H.24 según los estados del mapa de la Figura H.10, es decir, en la Tabla H.23 aparecer coloreados de un verde oscuro los estados que tienen un porcentaje de personas con la pauta completa (Washington, Oregon, Colorado, New Mexico, Minnesota, N.Y, Virginia, etc) mientras que en la Tabla H.24, aparecen de verde claro los estados con menor porcentaje de personas con la pauta completa (Idaho, Utah, Wyoming, Arkansas, Louisiana, Missisipi, Georgia, Tennessee, etc). Como vemos en la Tabla H.23, 7 de los 10 estados aparecen como los estados con mayor porcentaje de personas con la pauta completa comparándolo con la Figura H.10, mientras que en la Tabla H.24, solo se han "acertado" 5 de los 10 estados que aparecen son realmente estados con menor porcentaje de personas con la pauta completa si lo comparamos con la Figura H.10.

Se puede concluir como existe cierta relación entre los estados donde han ganado los demócratas y la negatividad o positividad hacia la vacunación frente a la COVID-19 y por otro, como podemos predecir qué estados van a presentar problemas para alcanzar la inmunidad de rebaño (70% por ciento de la población). En este sentido, hay que tener en cuenta que menos del 10% de los usuarios del conjunto de datos tienen activo el campo de la localización (*user_location*) y que el sentimiento hacia la vacuna seguramente haya cambiado con el paso del tiempo, ya que el conjunto de datos corresponde con fechas anteriores a los primeros "pinchazos" y por lo tanto la percepción puede ser que haya variado.

5.5.4. Estudio de la Variable Place Name

La variable "*place_name*", cuando aparece, nos indica el lugar desde donde se han "emitido" los *tweets*. Lamentablemente, solo 4998 observaciones del conjunto de datos tienen la variable "*place_name*" activa. El número de observaciones con la variable "*coordinates*" (longitud y latitud) activa es de tan solo 1144. Para que el número de observaciones sea mayor, se concatenó los *dataFrames* con la variable "*place_name*" activa y las variables "*coordinates_0*" y "*coordinates_1*", dando lugar a un *dataFrame* de 6141 observaciones únicas (sin duplicados). Posteriormente, utilizado la librería Geopy de Python, se ha extraído la longitud y latitud de las ciudades que aparecen en la columna "*place_name*", ya que los usuarios de estos *tweets* no han activado el campo de coordenadas.

En la Figura 5.9 se muestran las localizaciones (longitud y latitud) de EE. UU de los usuarios que hayan activado esta función en el momento de enviar el mensaje (en la Tabla H.15 se resume el número de *tweets* agrupado por sentimiento). Podemos ver de nuevo, como los sentimientos positivos (color azul) hacia la vacuna predominan entre los usuarios cuyos *tweets* provienen de la parte noroeste de EE.UU y de la costa Oeste, mientras que las zonas

del centro y sur están más divididas. En la Figura H.11, podemos ver las coordenadas (longitud y latitud) de Europa.

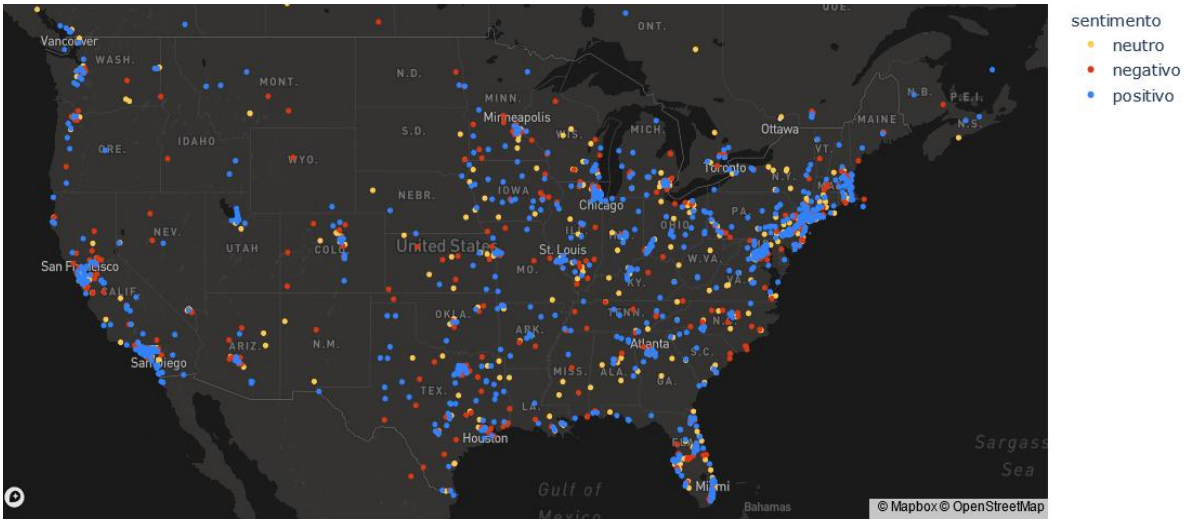


Figura 5.9: Mapa de sentimientos de EE. UU

Para concluir, en la Figura F.4 se observa como el ritmo de vacunación en EE. UU no llega al 70% de inmunidad en alguno de los grupos de edad a pesar de ser uno de los países con mayor número de vacunas inoculadas. Por el contrario, España tiene a la mayor parte de la población de riesgo inmunizada y parece que la percepción hacia la vacuna del COVID-19 es más positiva que en EE. UU. Como se vio en los análisis de la variable "*user_location*" y *"place_name*", existen ciertos estados de EE. UU donde el grado de negatividad y neutralidad hacia la vacuna es alto, por lo que se deben tomar las medidas oportunas para que al menos en los grupos de riesgo (mayores de 60 años) se alcance el 70% de personas vacunadas, si no, habría que vacunar a las personas más jóvenes y aun así pudiera darse el caso de que no se llegará a la inmunidad de grupo.

5.5.5. Análisis de Hashtags y Feature Engineering

En la sección F se analizan y se muestran los resultados del análisis de *hashtags* y en la sección G se muestra la matriz de correlación entre las nuevas variables creadas mediante *feature enineering* y la variable objetivo. Podemos decir

5.6. Preprocesamiento y Análisis Exploratorio de Unigramas, Bigramas y Trigramas

En esta sección, se realizará el preprocesamiento de texto [Orei20] (sección 2.2.3) que nos servirá para realizar el análisis de "unigramas", "bigramas" y "trigramas" más frecuentes utilizando las técnica TF y TF-IDF (explicadas en la sección 2.4.3). Para el análisis de *n*-gramas, es importante eliminar las *stopwords* o al menos las *stopwords* más frecuentes, normalizando el texto lo máximo posible mediante técnicas como la lematización, ya que el objetivo de este apartado es intentar extraer las opiniones (sentimientos) de los usuarios hacia la vacunación. Además, se construirá la "nube de palabras" o *WordCloud* de los "unigramas", y se analizaran los principales "bigramas" y "trigramas". Para ello, partiendo del preprocesamiento realizado en la sección 5.4, primero realizaremos una limpieza básica eliminando el "ruido" del texto mediante expresiones regulares del paquete *re* de Python. El enfoque de la limpieza de datos consiste en definir un conjunto de expresiones regulares e

identificar los patrones problemáticos y las reglas de sustitución correspondientes. Dichas funciones sustituyen primero todos los escapes HTML (por ejemplo, &) por su representación en texto plano para reemplazar posteriormente ciertos patrones por espacios. Por último, se eliminaron las secuencias de espacios en blanco. En la Tabla 5.5 se resume las diferentes operaciones realizadas para la limpieza de ruido.

Tabla 5.5: Expresiones Regulares para la limpieza de ruido del texto (*noise removal)*

Tarea	Expresión Regular o Función (librería "re" de Python y "html")
Pasar el texto a minúsculas	text.lower ()
Convertir los escapes de HTML como *&amp* en caracteres.	html.unescape(text)
Convertir etiquetas como "<tab>" en espacios en blanco.	re.sub(r'< [^<>]*>', ' ', text)
Eliminar URLs y mantener el texto asociado ([texto](https://...).	re.sub(r'[([^\[\]]*)\]\N-([^\(\)]*\)', r'\1', text)
Extraer texto o código entre paréntesis.	re.sub(r'\[[^\[\]]*\]', ' ', text)
Secuencias independientes de caracteres especiales	re.sub(r'(?:^|\s)[&#<>{}\[\]+|\\:-]{1,}(?:\s|$)', ' ', text)
Secuencias independientes de guiones como "---"o "==".	re.sub(r'(?:^|\s)[\-=\+]{2,}(?:\s|$)', ' ', text)
Secuencias de espacios en blanco.	re.sub(r'\s+', ' ', text)
Eliminar los espacios al principio y al final de la cadena.	text.strip()

Finalmente, en la Tabla H.16 se recoge las diferentes operaciones y funciones realizadas para la normalización del texto. Para ello, se han utilizado expresiones regulares del paquete *re* de Python, funciones de preprocesamiento de la librería *textacy* y el paquete *SpaCy* para realizar la lematización.

Tras realizar el preprocesamiento, se eliminaron los *tweets* que idénticos, ya que puede darse el caso de que *tweets* diferentes antes del preprocesamiento sean idénticos tras el preprocesamiento. Para ello, se filtrarán los *tweets* del preprocesamiento anterior mediante la librería Numpy de Python. El número de observaciones antes del filtrado es de 4.026.416 y tras el filtrado con Numpy 3.401.961.

5.6.1. Tokenizador

Para extraer los términos más frecuentes (TF) y la frecuencia inversa de documento (IDF), se siguieron los pasos de [Orei21]. Para ello, primero se "tokenizaron" cada uno de los *tweets d*el *corpus* obtenido del preprocesamiento anterior. En las lenguas occidentales, las palabras o términos suelen aparecer separadas por espacios en blanco y caracteres de puntuación. Por ello, el "tokenizador" más simple y rápido es el método nativo de Python "str.split()", que separa en espacios en blanco. En [Orei21], aplicaron un patrón simple. Las palabras en su definición consisten en al menos una letra, así como dígitos y guiones. Los números "puros" no deberían aparecer ya que fueron eliminados en el preprocesamiento anterior, aun así, se omitirán.

La expresión [A-Za-z], de uso frecuente, no es una buena opción para la búsqueda de letras, ya que omite las acentuadas como ä o â. Es mucho mejor la clase de caracteres POSIX

\p{L}, que selecciona todas las letras Unicode (se necesitóla biblioteca “regex" en lugar de “” para trabajar con las clases de caracteres POSIX). La expresión que se muestra en la Figura 5.10 coincide con aquellos *tokens* que consisten en al menos una letra (\p{L}), precedida y seguida por una secuencia arbitraria de caracteres alfanuméricos (“\w” incluye dígitos, letras y guión bajo) y guiones (-).

```
import regex as re

def tokenize(text):
    return re.findall(r'[\w-]*\p{L}[\w-]*', text)
```

Figura 5.10: “Tokenizador”[22]

A modo de ejemplo, en la Tabla H.17, se muestra el texto original del *tweet*, el texto tras el preprocesamiento, los *tokens* (con *stopwords*) y la puntuación *obtenida con* VADER (uno positivo y otro negativo). Se observa como en la columna del *tweet* “preprocesado”, no aparecen los signos de puntuación, se han eliminado las menciones (@...) y las palabras con una longitud menor de dos y, con el proceso de lematización, se han obtenido la raíz de las palabras, como por ejemplo “accomplishments” -> “accomplishment” y “mandates” -> “mandate”. Esta última etapa es muy importante (junto con la eliminación del *stopwords* que veremos posteriormente), ya que reducirá drásticamente la dimensión de la bolsa de palabras o BOW para el análisis de n-gramas y el conjunto de entrenamiento para la construcción del Clasificador de Sentimientos.

Finalmente, se creó una columna nueva en el *DataFrame* llamada *tokens,* que contenga el texto en minúsculas y “tokenizado “sin *stopwords* para cada documento. Para ello, se utilizó un patrón extensible para una cadena de procesamiento o *pipeline* que automatice todo el proceso. En este sentido, se pueden añadir otras operaciones simplemente ampliando el proceso. Hay que tener cuidado con las listas de *stopwords* que se utiliza para no eliminar información valiosa. Por ejemplo, si en la frase "No me gusta las vacunas" quitamos las *stopwords*, quedaría “me gustan las vacunas” (dependiendo de la librería utilizada para obtener la lista de *stopwords*). En nuestro caso, se creó una lista de *stopwords* concatenando las listas de las librerías de SpaCy, NLTK y Gensim de Python, dando lugar a 412 *stopwords*. En un principio, sólo se excluirá de la lista la palabra “*not*” para ciertos análisis (como los “unigramas” ya que se quiere identificar que palabras están asociadas a los sentimientos hacia la vacuna.

5.6.2. Análisis de n-gramas con TF y TF-IDF

Para el cálculo de TF y TF-IDF se implementaron funciones para contar palabras y calcular los valores IDF en vez de utilizar las clases predefinidas de *Scikit-Learn* como *CountVectorizer* y *TfidfVectorizer*. Al final tendríamos que escribir aproximadamente el mismo número de líneas de código para producir los resultados pero se perdería la oportunidad de introducir este importante concepto desde cero.

En la Figura H.15 del Anexo se muestra la función utilizada para el cálculo de TF e IDF. Por un lado, la función *count_words* recibe un *DataFrame*, el nombre de la columna sobre la cual se quiere computar o calcular el número de palabras o TF y un parámetro para filtrar el mínimo número de ocurrencias por término y devuelve un nuevo *DataFrame* con la nueva columna *freq* ordenada de mayor a menor. Por otro lado, la función *compute_idf* recibe un

DataFrame, el nombre de la columna sobre la cual se quiere computar o calcular el número de palabras o IDF y un parámetro para filtrar el mínimo número de ocurrencias por término y devuelve un nuevo *DataFrame* con la columna *idf*. Finalmente, para el cálculo TF-IDF es necesario multiplicar las columnas *freq* e *idf*. Para dibujar la nube de palabras, se ha creado una función (wordcloud) que utiliza a su vez la librería WordCloud con ciertas modificaciones:

- El número de *n*-gramas a dibujar es de 100 (100 *n-gramas* más frecuentes calculados con TF o TF-IDF)
- Se ha eliminado las *stopwords* mencionadas anteriormente (salvo la palabra "*not"* en los "bigramas" y "trigramas", ya que estos aportan algo más de contexto y pudiera ser relevante).

A continuación, se analizarán los "unigramas" más frecuentes mediante las nubes de palabras o *WordCloud* con TF y TF-IDF y los "bigramas" y "trigramas" más frecuentes mediante diagramas de barras.

En la Figura 5.11 se muestra de una forma más visual los pasos seguidos:

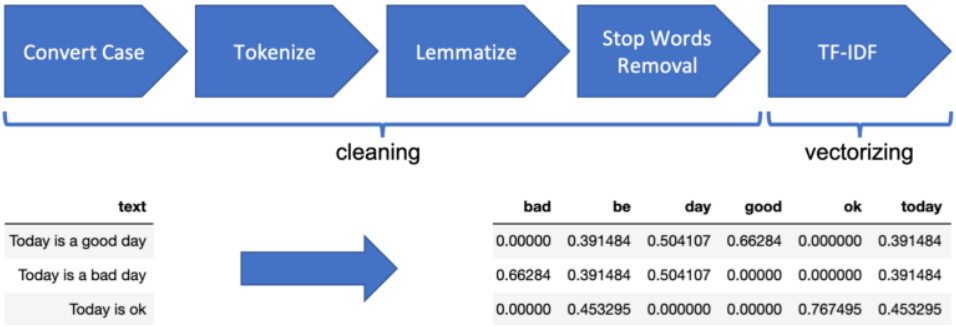


Figura 5.11: Resumen general de las etapas del preprocesamiento [23].

5.6.2.1. Visualización de *n*-gramas

En la Figura 5.12 y Figura 5.13 se muestran la nube de palabras de los "unigramas" calculados a partir de TF y TF-IDF respectivamente. Podemos ver como con método TF-IDF penalizamos algunos "unigramas" más frecuentes calculados con TF, dando lugar a una nube de palabras diferente. Por ejemplo, palabras como "*vaccine*" (vacuna), "*covid*", "*Trump*", "*need*", "*virus* "o *people"* aparecen en la Figura 5.13 con un tamaño diferente comparado con la Figura 5.12. La palabra "*vaccine*" desaparece del grupo de palabras más frecuentes en la Figura 5.13 (tamaño de letra más grande).

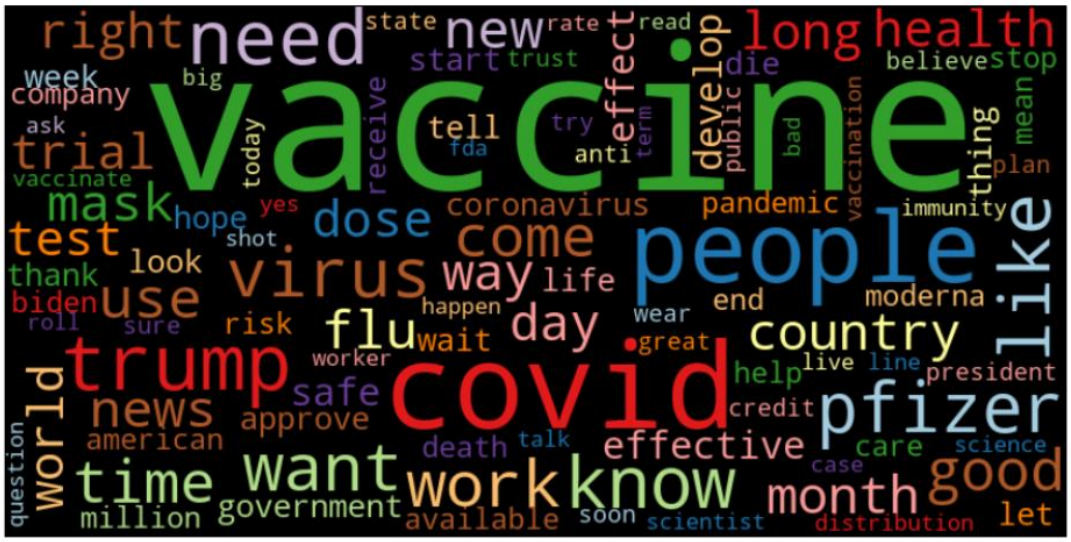


Figura 5.12: WordCloud “unigramas” con TF.

Figura 5.13: WordCloud “unigramas” con TF-IDF.

Las palabras más frecuentes calculadas con TF fueron “*vaccine*”, “*covid*”, “*people*”, “*trump*”, “*work*”, “*pfizer*”, “*like*”, “*virus*”, “*mask*”, “*flu*” y “*effective*”. En la nube de palabras calculada con TF-IDF se pueden encontrar palabras como “*covid*”, “*people*”, “*like*”, “*want*”, “*Pfizer*”, “*need*”, “*know*”, “*vaccine*”, “*use*”, “*effect*”, “*effective*”, “*mask*.

Además, de las figuras anteriores cabe destacar la presencia de palabras que pueden estar relacionadas con la vacuna como por ejemplo, “*pfizer*”, “*flu*”, “*dose”*, “refuse”, “approve”, “effective”, “*develop*”, “*available*” y “*distribution*”. El uso de “unigramas” para extraer conocimiento puede ser algo reducido, ya que nos faltaría añadir algo de contexto.

Examinar los *n*-gramas mediante las TF y TF-IDF de los “bigramas” y “trigramas” en lugar de palabras individuales tiene ventajas y desventajas. Los pares de palabras consecutivas pueden captar una estructura que no esté presente cuando se cuentan palabras sueltas y pueden proporcionar un contexto que haga más comprensibles los tokens. Sin embargo, los recuentos por “bigramas” y “trigramas” también son más escasos (un par típico de dos palabras es más raro que cualquiera de las palabras que lo componen). Por tanto, los

“bigramas” y “trigramas” pueden ser especialmente útiles cuando se dispone de un conjunto de datos de texto muy amplio.

Algunos de los “bigramas” más interesantes que podríamos destacar son “*vaccine not*”, “*not vaccine*”, “*not want*”, “*flu vaccine*”, “*long term*”, “*vaccine available*”,” vacccine *distribution*” y “*vaccine trial*”. Cabe destacar la alta presencia de “bigramas” que pueden considerarse negativos como “*not vaccine*”, “*vaccine not*” y sobre todo “*not want*”. También se refleja cierta preocupación sobre la efectividad, el funcionamiento de las vacunas (“*vaccine effective*” y “*vaccine work*”) y la forma en que será distribuida (*vaccine distribution*) y desarrollada (*vaccine develop*). Finalmente, hay que destacar el “bigrama” “*long term*” que podrá ser indicarnos la preocupación sobre los efectos posibles efectos secundarios de las vacunas.

En Figura 5.15, se muestra en el diagrama de barras los 30 “trigramas” más frecuentes con TF-IDF. Podemos ver como el “trigrama” *pfizer covid vaccine* es el más frecuente seguido por *operation warp speed*[24] (una asociación público-privada iniciada por el gobierno de EE.UU, para facilitar y acelerar el desarrollo, la fabricación y la distribución de vacunas, terapias y diagnósticos contra la COVID-19) y *long term effect*. Como vemos, hay una gran preocupación por los posibles efectos secundarios de la vacuna, lo puede llevar a que haya un gran número de personas no quieran vacunarse. Cabe destacar la presencia de “trigramas” negativos como “*not wear mask”* y “*not trust vaccine*”.

Por último, se analizaron los “trigramas” negativos y positivos más frecuentes. En este caso se añadió la palabra “*vaccine*” a la lista de *stopwords,* con el objetivo de descubrir otras combinaciones de palabras que nos aporten algo más de luz en el análisis de sentimientos (el filtrado llevado a cabo en la sección 5.3 dio como resultado un *corpus* en el que al menos siempre aparece un *keyword* o *hashtag* relacionado con la vacuna por lo que el hecho de eliminar la palabra “*vaccine*” de los “trigramas” no cambia el objetivo del análisis).

En la Figura H.16 se muestran los trigramas asociados a sentimientos positivos. Entre los “trigramas” más destacables fueron “*long term safety*”, “*know long term*”, “*long term care*”, “*light end tunnel*”, “*fda approve pfizer*” o “*virus survival rate*”. En la Figura H.17 se muestran los trigramas asociados a sentimientos negativos. Se observa una mayor frecuencia del “trigrama” “*long term effect*” comparado con la Figura H.16. Además, cabe destacar los “trigramas” “*history significant allergic*”, “*significant allergic reaction*”, “*severe allergic reaction*” y “*refuse wear mask*”.

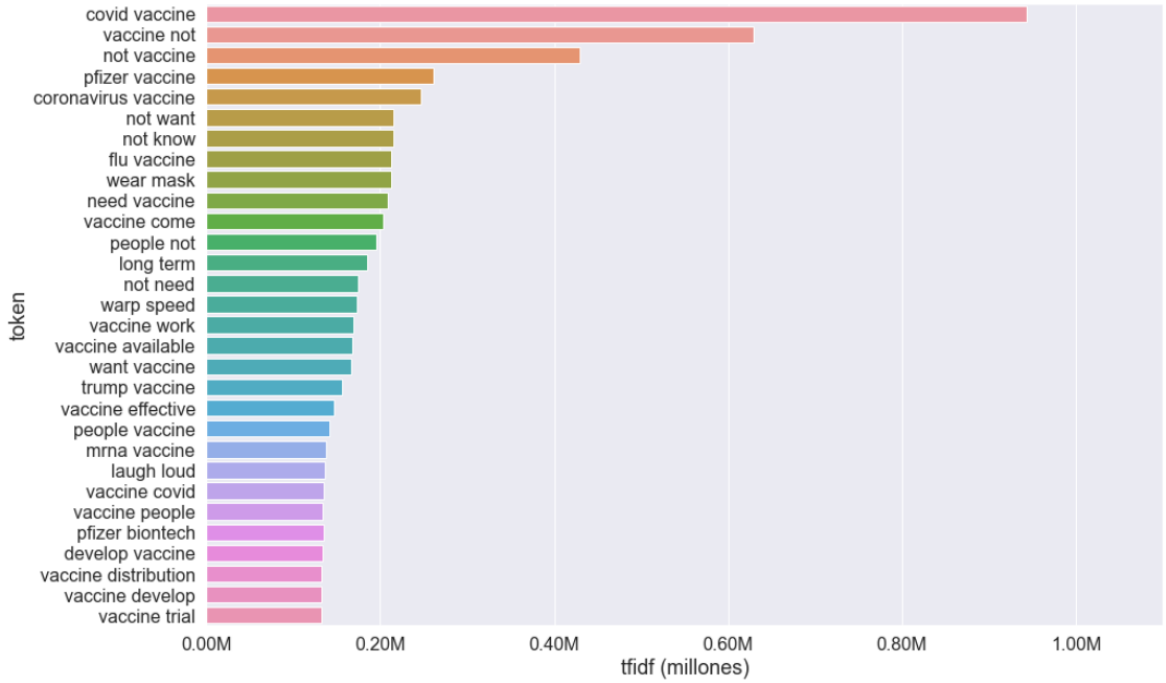


Figura 5.14: Gráfico de barras de los 30 “bigramas” más frecuentes calculados con TF-IDF

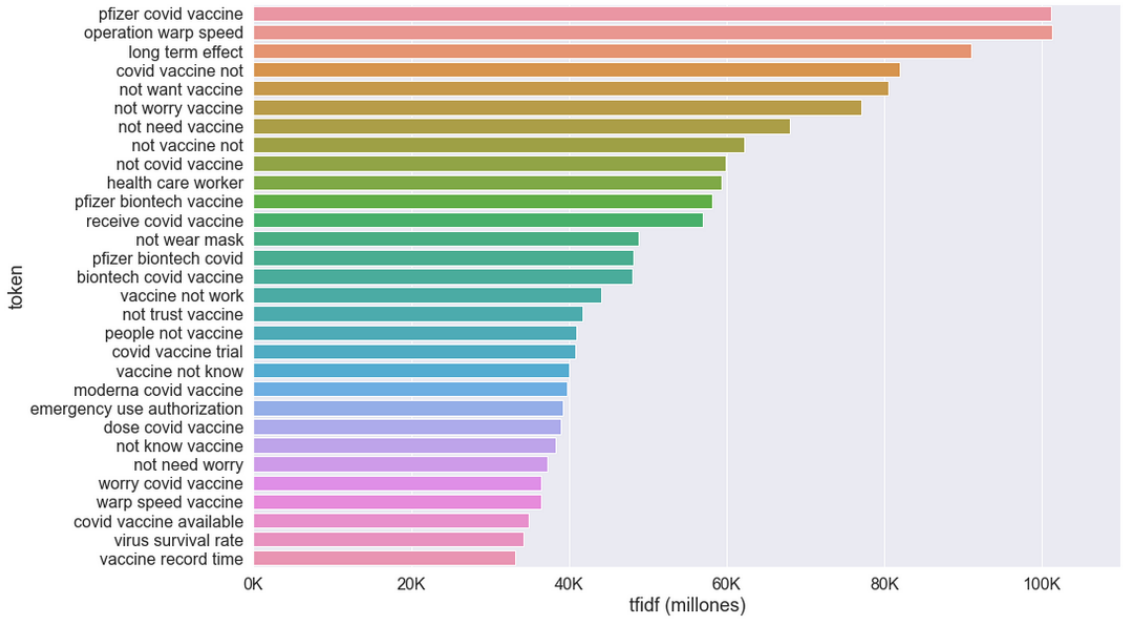


Figura 5.15: Gráfico de barras de los 30 “trigramas” más frecuentes calculados con TF-IDF.

5.7. Modelado de Tópicos con SAS MINER

En esta sección vamos a realizar un Modelado de Tópicos o *Tópic Modelling* (explicado en la sección 2.4.4) con SAS Miner. El objetivo es agrupar automáticamente conjuntos de palabras formando temas o tópicos. El conjunto de datos de entrada para el Modelado de Tópicos será el *corpus* obtenido del preprocesamiento de la etapa anterior (*n*-gramas). Tabla La elección de este preprocesamiento se debe a la necesidad de reducir al máximo el número de palabras que realmente tengan un significado semántico (por ejemplo, no queremos tener en un mismo grupo diferentes formas verbales de la misma palabra). Finalmente, el *corpus* que se cargará en SAS Miner para el Modelado de Tópicos consta de 4.026.416 *tweets*.

Para el Modelado de Tópicos con SAS, vamos a seguir la guía de SAS[25] para el Modelado de Temas. En la Figura 5.16 se muestran los nodos que forman el "pipeline" para el Modelado de Tópicos con SAS (el nodo de filtrado del texto es opcional). La configuración de los nodos "*parsing* del texto", "filtrado del texto" y "tema del texto" se muestran en la Figura H.18, Figura H.19 y Figura H.20 respectivamente. El nodo "*Parsing* del texto" se ha configurado (Figura H.19) para que ignore los determinantes, las preposiciones, las interjecciones y las conjunciones. Además, se filtraron las *stopwords* (salvo el término *not* por el mismo motivo que en la sección 5.6.2.1*)* ya que no aportan ningún valor en el Modelado de Tópicos. También se eliminaron los signos de puntuación al igual que los números y se activó la opción "términos de tema" (esta opción especifica si los términos diferentes con la misma raíz se van a tratar como equivalentes). En la Figura H.21 se muestran las frecuencias de los términos agrupadas por "roles" (parte del discurso del término, clasificación de entidad del término o el valor Grupo de sustantivo).

Con el nodo "Filtrado del texto" (Figura H.20) se puede reducir el número total de términos o documentos para que solo se considere la información más relevante. Además, se puede realizar una corrección ortográfica y configurar el tipo de peso o ponderación de las características (en nuestro caso usaremos TF-IDF). Utilizando el Visor Interactivo de Filtros del panel de propiedades del Nodo de Filtrado de Texto, se muestran los enlaces de conceptos que indican la fuerza de asociación de algunos términos. El grosor de la línea define la fuerza de la asociación, y una línea más ancha indica una asociación más fuerte entre esos términos. Cada nodo vinculado también puede ampliarse para ver sus "subnodos" vinculados para tener una mejor idea de la relación entre los dos nodos principales.

Mediante el Visor Interactivo de Filtros se examinaron los enlaces relacionados con la vacunación o la vacuna. En la Figura H.22 se observa algunos de los enlaces de conceptos más importantes que indican la fuerza de asociación entre algunos términos. El análisis se centró en la vacunación, vacuna y sus efectos secundarios. En la Figura H.22 (a) se muestra el enlace conceptual del término "*vaccine"* en el que podemos observar cómo hay una fuerte asociación con el nodo *"order vaccine*" y "*pfizer biontech vaccine*". Además, se ha expandido el nodo del "bigrama" "*question vaccine*" para extraer que tipo de cuestiones son las más importantes respecto a la vacuna. La presencia de términos como "*safety vaccine*", "*efficacy*" nos hace pensar en que uno de los temas principales del *corpus* puede ser la seguridad y eficiencia de las vacunas, además de cómo será distribuida (nodo "*distribute*"). En la Figura H.22 (b) se muestra los enlaces del término "*know*". Se observa como las principales preocupaciones son los posibles efectos adversos, como alergias o como estas podrían afectar a la fertilidad. En la Figura H.22 (c) se muestran algunos de los efectos secundarios, que pueden ser sobre la vacuna o sobre la COVID-19, ya que con un solo término (*"*unigrama*")* no se puede inferir si los efectos son de uno o de otro. Podemos ver como las reacciones adversas más comunes son dolor de cabeza, fiebre y fatiga. Por último, En la Figura H.22 (d) se muestra los enlaces del término "*pfizer*". Se observaron enlaces muy fuertes en los nodos cuyos términos hacen referencia a la distribución de la vacuna, tales como "*quick distribution*" y "*distribute*", además de otros que hacen referencia a los ensayos y a su autorización.

Una vez se configuró los nodos anteriores, se configuró la Tabla H.18, en la que se describe las opciones del nodo "Tema de texto".

Como se muestra en la Figura H.20, se configuró el nodo "Tema del texto" para que identificara 10 temas o tópicos (por simplicidad se seleccionó este número de temas). En Figura 5.17 se muestran los resultados del visor de temas del nodo "Tema del texto". En la

Tabla 5.6, se ha describen los posibles temas o tópicos teniendo en cuenta los *keywords* identificados en el nodo “Tema de Texto”. Algunos temas están bastante claros, como los “efectos secundarios”, “efectividad de la vacuna de Pfizer y Moderna” y la “opinión política sobre la distribución de la vacuna”. Por el contrario, hay tres grupos que no queda claro un tema específico y se ha dejado como “sin identificar”.

Por último, en la Figura H.23 se muestran los documentos y los términos con mayor peso del tópico 7 (efectos secundarios).

Tabla 5.6: Temas encontrados.

Tópico	Descripción	Términos del tema (keywords) *
1	Sin identificar	not, +vaccine, +want, +worry, want
2	Sin identificar	covid, covid vaccine, +vaccine, +receive, +worry
3	Opinión política sobre la distribución de vacunas	+trump, president, biden, credit, +vaccine
4	Sin identificar	+people, + die, +want, vaccine, +vaccine
5	Efectividad de la vacuna de Pfizer y Moderna	pfizer, moderna, effective, biontech, +vaccine
6	Sin identificar	vaccine, covid, news, first, work
7	Efectos secundarios	+know, effect, side, +long, term
8	Vacuna contra la gripe estacional	+flu, year, flu vaccine, +vaccine, +shoot
9	Efectividad medidas anti-covid	+virus, mask, +vaccine, effective, rate
10	Sin identificar	first, mask, year, +vaccine, wear

(*) El resultado tabular de la ventana Resultados del nodo *Parsing* del texto es la tabla de términos, que muestra información sobre los términos de nivel superior analizados (símbolo (+) si el término es un término padre; en blanco en caso contrario). Tenga en cuenta que si hay más términos que la configuración de la propiedad Número de términos a mostrar del nodo *Parsing* del texto, sólo se incluirá ese número de términos más frecuentes.

Figura 5.16: Pipeline del Modelado de Temas en SAS MINER

Tema	Categoría	Corte del término	Corte del documento	Número de términos	Nº docs
not,+vaccine,+want,+worry,want	Múltiple	0.003	0.095	674	152585
covid,covid vaccine,+vaccine,+receive,+worry	Múltiple	0.003	0.083	422	108433
+trump,president,biden,credit,+vaccine	Múltiple	0.003	0.075	925	78656
+people,+die,+want,+vaccine,vaccine	Múltiple	0.003	0.073	827	103289
pfizer,moderna,effective,+vaccine,biontech	Múltiple	0.003	0.071	944	88669
vaccine,covid,news,first,work	Múltiple	0.003	0.069	884	164743
+know,effect,+long,side,term	Múltiple	0.003	0.069	780	90994
+flu,year,flu vaccine,+vaccine,+shoot	Múltiple	0.003	0.068	610	66417
+virus,mask,effective,+vaccine,rate	Múltiple	0.003	0.067	977	86122
first,mask,year,+vaccine,wear	Múltiple	0.003	0.063	1394	121125

Figura 5.17: Resultados del nodo “Tema del texto”.

6. Construcción de un Clasificador de Sentimientos

El objetivo de este apartado es construir un Clasificador de Sentimientos interpretable basado en técnicas de ML. Dado que contamos con *corpus* etiquetado y una variable objetivo categórica (negativo, neutro y positivo), nos encontramos ante un problema de clasificación multiclase. Es necesario identificar el problema, ya que para resolverlo se empelarán una familia de algoritmos determinada. Además, se estudiarán las características (palabras) más importantes del mejor modelo. Las etapas para la construcción un clasificador de sentimientos (texto) multiclase basado en un sistema de aprendizaje automático supervisado son las siguientes:

- Preprocesamiento del *corpus* (tres métodos diferentes).
- "Tokenización".
- División del conjunto de datos en conjunto de entrenamiento y validación (70%) y conjunto de *test* (30%).
- Extracción y ponderación de las características.
- Reducción o selección de características (varios conjuntos de datos de entrenamiento).
- Selección del mejor conjunto de datos (mejor subconjunto de características).
- Ajustes de los "hiperparámetros" de los modelos (SVM, Random Forest, Deep Learning, etc.) y validación cruzada (*Hyperparameter optimization or tuning and cross validation*)
- *Test* sobre el modelo seleccionado
- Interpretabilidad del modelo seleccionado

6.1. Selección del Conjunto de Datos

Con el objetivo de seleccionar el mejor conjunto de datos posible, se realizó una comparación entre los algoritmos de vectorización (TF y TF-IDF) sobre "unigramas", "bigramas y "trigramas" y sus diferentes configuraciones, como por ejemplo, el número de características o de palabras. Además, se compararon diferentes tipos de preprocesamiento (analizando el rendimiento con las *stopwords* o sin ellas). Partiendo del conjunto de datos de la sección 5.4, se realizaron dos preprocesamientos (Tabla H.19). Como se observa en la Tabla H.19, no se eliminaron las *stopwords*, ya que se quiso realizar una comparación entre los estos dos preprocesamientos y los dos métodos de vectorización para crear varios modelos de línea de base o *baseline model* con los que poder comparar los rendimientos. Para este fin, se empleó un algoritmo rápido y robusto, como la Regresión Logística, y la métrica ROC-AUC para la comparación de los modelos obtenidos.

En primer lugar, se eliminaron los *tweets* que hayan podido quedar idénticos tras la ejecución de los dos tipos de preprocesamiento para posteriormente realizar un "*join*" de los dos *DataFrames* por la columna "*id_str*" (Tabla H.20). El "*join*" entre los dos tipos de preprocesamiento es clave, ya que el objetivo es realizar una comparación entre los distintos algoritmos de vectorización sobre el mismo subconjunto de *tweets*, por lo que debemos quedarnos con las observaciones que aparezcan en ambos conjuntos de datos.

En segundo lugar, dado que el objetivo es probar varios algoritmos de ML y debido al gran volumen de *tweets* del conjunto de datos de entrada, (más de 4 millones de *tweets*), se seleccionaron de forma aleatoria dos subconjuntos por cada método de preprocesamiento. Finalmente, se seleccionaron 50.000 observaciones (seleccionadas al azar utilizando la librería Pandas de Python) por cada clase (negativo, neutro y positivo). Posteriormente, se

separaron en dos *DataFrames*, “df_preproceso_1” y “df_preproceso_2” (Figura H.24 y Figura H.25 respectivamente), compuesto por 150.000 observaciones cada uno de ellos.

6.2. Extracción y Ponderación de Características

Antes de extraer y ponderar el *corpus*, se dividió (la división se realizó de forma estratificada conservando el mismo número de muestras por cada clase) previamente el conjunto de datos en *train* y *test*, ya que no se recomienda realizar esta división después de realizar la vectorización ya que algunas palabras sólo se encontrarían en el conjunto de *train* (105.000 observaciones) y otras sólo en el conjunto de *test* (45.000 observaciones).

Una vez dividido los dos conjuntos de datos en *train* y *test*, se ejecutaron los experimentos que aparecen en la Tabla H.26. Dado que por defecto las funciones *CountVectorizer* y *TfidfVectorizer* no establecen un límite en el número de características que formarán la matriz de características (matriz resultante tras la ponderación con los métodos de vectorización), se ejecutaron todos los experimentos con un mínimo de 1000 palabras (características) y un máximo de 10.000. Por lo tanto, cada experimento consistirá en un bucle cuyo valor inicial fue de 1000 (mínimo número de variables) y cuyo valor final fue de 10.000 (número máximo de variables) incrementando el contador de 1000 en 1000 en cada iteración.

En cuanto a las *stopwords*, se utilizó la lista de las librerías Gensim, NLTK y SpaCy de la sección 5.6.1 y otra lista personalizada con las 20 palabras más frecuentes por cada conjunto de datos (Tabla H.25), puesto que no siempre se recomienda eliminar todas las *stopwords* del *corpus*. En todos los experimentos, se realizó una validación cruzada con 5 *k-folds* o grupos sobre el conjunto *train*.

En las Figura H.26 y Figura H.27, se muestran los resultados de los modelos 1 y 2 de la tabla anterior. Se observa como los modelos sin *custom stopwords* (en rojo) tienen un ROC-AUC más alto que sin todas las *stopwords* y con *stopwords.* En la Figura H.28 se muestra como el modelo 2 con “unigramas” tiene un rendimiento mayor que los “bigramas” y “trigramas. En la Figura H.29 y Figura H.30, de nuevo se observa como los modelos 3 y 4 sin *custom stopwords* (en rojo) tienen un ROC-AUC más alto que sin las *stopwords* y con *stopwords.* En la Figura H.31 se observa como el modelo 4 con “unigramas” tiene un rendimiento mayor que los “bigramas” y “trigramas.

En la Tabla 6.1 se muestra la configuración de los mejores modelos de *CountVectorizer* y *TfidfVectorizer* respectivamente.

Tabla 6.1: Comparación entre *CountVectorizer* y *TfidfVectorizer* con diferentes *n*-gramas

Modelo	Preprocesamiento	Tipo de vectorización	*n*-gramas	*stopwords*	Ganador
2	df_preproceso_2	CountVectorizer (TF) (1000,10000, 1000)	Unigramas Bigramas Trigramas	Sin custom stopwords	• Sin custom stopwords • Unigramas
4	df_preproceso_2	TfidfVectorizer (TFIDF) (1000,10000, 1000)	Unigramas Bigramas Trigramas	Sin custom stopwords	• Sin custom stopwords • Unigramas

Finalmente, en la Figura 6.1, se muestran los mejores modelos obtenidos con *CountVectorizer* y *TfidfVectorizer* sobre "unigramas", "bigramas" y "trigramas" respectivamente. Se observa cómo el ROC-AUC, salvo en los modelos con "bigramas", es levemente mejor en los modelos con *CountVectorizer*, y en particular, el modelo con "unigramas".

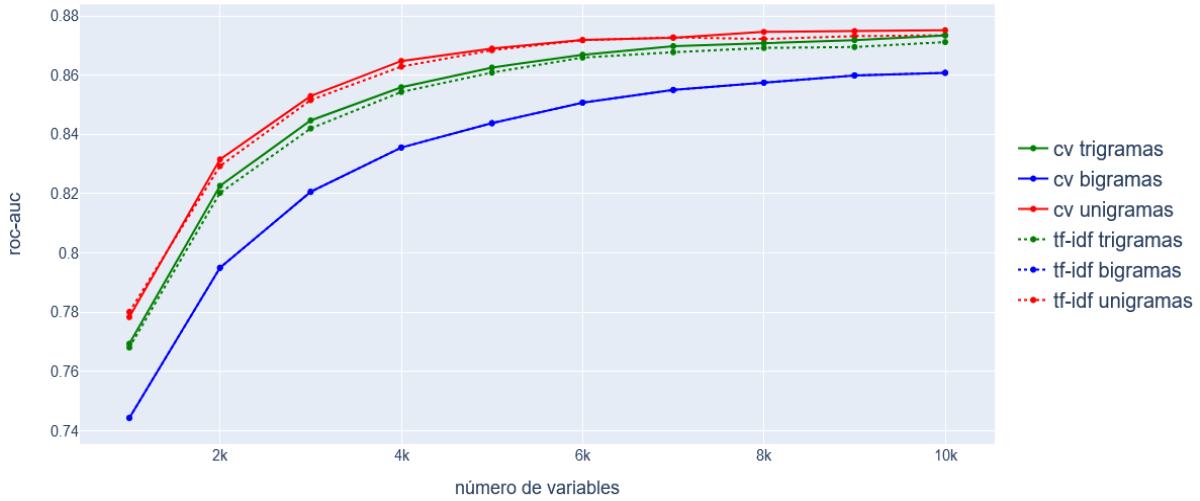


Figura 6.1: *CountVectorizer* (N-gram (1-3)) vs *TfidfVectorizer* (N-gram (1-3))

En la Figura H.32 y Figura H.33, se muestra el diagrama de cajas de los resultados de la validación cruzada correspondientes a los modelos ganadores con *CountVectorizer* y *TfidfVectorizer*. Se observa como en líneas generales casi todos los modelos son bastante robustos (poca varianza) y dependiendo del número de características seleccionadas, un sesgo bajo, llegando a valores cercanos del 90% en términos de ROC-AUC. En la Tabla 6.2 y Tabla 6.3 se muestran de forma más detallada la media de las validaciones cruzadas por cada configuración. Se obserca como con 10.000 características o palabras alcanzamos el porcentaje de ROC-AUC más alto, con un 0.897971.

Tabla 6.2: Validación cruzada del mejor modelo con *CountVectorizer* (modelo 2).

Modelo	Nº de caraterísticas	Roc-Auc (media)
1	1000	0.774386
2	2000	0.837364
3	3000	0.863036
4	4000	0.878521
5	5000	0.886193
6	6000	0.890986
7	7000	0.894364
8	8000	0.896493
9	9000	0. 897614
10	10000	0. 897971

Tabla 6.3: Validación cruzada del mejor modelo con *TfidfVectorizer* (modelo 4).

Modelo	Nº de caraterísticas	Roc-Auc (media)
0	1000	0.774329
1	2000	0.835500
2	3000	0.859450
3	4000	0.871664
4	5000	0.877079
5	6000	0.879293
6	7000	0.881307
7	8000	0.881443
8	9000	0.881364
9	10000	0.881686

En la siguiente sección, con el objetivo de evitar el *overfitting* y reducir los tiempos de ejecución de la búsqueda de los mejores parámetros para los algoritmos de ML, partiendo del conjunto de datos del modelo 2 y el modelo 4 (ambos con 10.000 características o variables), buscaremos otros subconjuntos de características con los que podamos obtener un rendimiento parecido con un menor número de características.

6.3. Selección de Variables

En las siguientes secciones se realizará una selección de características con diferentes algoritmos de selección sobre los dos conjuntos de datos obtenidos en el apartado anterior. Finalmente, se comparará los mejores modelos obtenidos en cada uno de los métodos mediante un diagrama de cajas. Una vez más, se utilizará la Regresión Logística y a métrica ROC-AUC y la validación cruzada con 5 *k*-folds o grupos para evaluar los resultados de cada método de selección.

6.3.1. Método Chi2

Como se vio en la sección 6.3.1, por un lado se seleccionaron las variables con un *p*-valor < 0.05 (rechazamos la H0). En la Figura H.34 se muestran los resultados de la selección de las características cuyo *p*-valor es < 0.5. Se observa como con el modelo con *CountVectorizer* se obtiene un ROC-AUC y con una varianza muy parecida a la obtenida con el modelo con *TfidfVectorizer*. El número de características del modelo con *CountVectorizer* (modelo 2) de 3912 y de 1475 con el modelo *TfidfVectorizer* (modelo 4).

Por otro lado, la librería *SelectKBest* de *sklearn* permite seleccionar las *n* variables cuyos valores Chi-Cuadrado sean los más elevados. En este caso, como el objetivo es encontrar un subconjunto de características con el que se obtenga un rendimiento parecido al obtenido por el modelo 2 y el modelo 4 pero con menos variables, se seleccionaron las 1000, 2000, 3000, 4000 y 5000 características con mayor *score*. En la Figura H.35 (a) y Figura H.35 (b) se muestra como de nuevo los modelos con *CountVectorizer* (modelo 2) son ligeramente mejores que con *TfidfVectorizer* (modelo 4). El cuanto al modelo ganador, se seleccionaron los modelos con 2000 características, tanto con *CountVectorizer* como *TfidfVectorizer*, ya que son los dos tienen mejor equilibrio entre rendimiento y complejidad (número de características).

6.3.2. Método RFECV

Como se vio en la sección 2.4.5.2, RFECV (misma versión que RFE pero con validación cruzada) es una selección hacia atrás o *backward* de los predictores. Esta técnica, como ya se ha mencionado anteriormente, comienza construyendo un modelo (con un estimador elegido por el usuario) con todo el conjunto de variables y calculando una métrica de importancia para cada predictor. Los parámetros y la configuración seleccionados se muestran en la Tabla H.31. Se seleccionó de nuevo la RL como estimador, un mínimo de 1000 características (ya que como se ha observado en los modelos anteriores, con menos de 1000 características el rendimiento decrece de forma importante), un *step* o paso de 1 (número de características a eliminar en cada iteración) y una validación cruzada con 5 *k*-fold. En la Figura H.36 (a) se muestran los resultados de la validación cruzada (media) con *CountVectorizer* (modelo 2) en función del número de características. Se observa como el "pico" más alto de ROC-AUC (0.91292) se obtiene con 1327 características. En la Figura H.36 (b) se muestran los resultados de la validación cruzada (media) con *TfidfVectorizer* (modelo 4) en función del número de características. Se observa como el "pico" más alto de ROC-AUC (0.88534) se obtiene con 1208 características.

6.3.3. Método PCA

Mediante la técnica de PCA, se intentó reducir el número de características que explicasen el 90% de la varianza. En el caso del conjunto de datos con *CountVectorizer*, es necesario estandarizar como paso previo a la ejecución del algoritmo (en el caso de *TfidfVectorizer* no es necesario ya que los valores de TF-IDF están entre 0 y 1). En la Figura H.37 (a) se observa cómo se alcanza el 90% de la varianza del conjunto de datos con *CountVectorizer* (modelo 2) con 2878 características, mientras que en la Figura H.37 (b) se muestra cómo se alcanza el 90% de la varianza del conjunto de datos con *TfidfVectorizer* (modelo 4) con 4723 características. Una vez se obtuvieron los subconjuntos de características, tanto con *CountVectorizer* (modelo 2) como para *TfidfVectorizer* (modelo 4), se ejecutó una validación cruzada con 5 *k*-folds con RL.

Finalmente, en la Figura H.38 se observa como de nuevo el rendimiento es mejor con el conjunto *CountVectorizer* (modelo 2), tanto en ROC-AUC (0.872) como en número de características, ya que el conjunto *CountVectorizer* (modelo 2) cuenta con 2878 características contra las 4723 del *TfidfVectorizer* (modelo 4).

6.3.4. Resultados finales

Por último, en la Figura 6.2 se muestra en un diagrama de cajas con los mejores modelos obtenidos en cada uno de los métodos de selección de características. Se aprecia como con los modelos, salvo los modelos obtenidos con el método PCA, se oobtuvieron mejores resultados (en términos de ROC-AUC) con menos características. Finalemente, se ha seleccionado el modelo obtenido con el método RFCEV con el conjunto de datos del modelo 2 (*CountVectorizer*), ya que es el mejor equilibrio tiene entre rendimiento (ROC-AUC) y número de caracteristicas (Tabla H.32).

Figura 6.2: Comparación entre los mejores modelos obtenidos según el método de selección de características.

6.4. Modelización con Python

6.4.1. Conjunto de Datos

El conjunto de datos que se utilizó para construir el mejor modelo fue el modelo 5 (Tabla H.32), es decir, el conjunto de datos sin lematización, sin *custom stopwords*, con "unigramas" y con el método de vectorización *CountVectorizer*. La metodología para encontrar el mejor modelo posible para cada algoritmo será la siguiente:

1. En caso de que el algoritmo cuente con muchos parámetros, con el objetivo de reducir el tiempo de ejecución de la búsqueda de los mejores parámetros, se realizaró un estudio previo entre los principales parámetros.
2. Configuración de la rejilla.
3. Estudio de los mejores modelos.
4. Validación cruzada con 5 *k*-fold de los mejores modelos.
5. Diagrama de cajas con los resultados de la validación cruzada y lección del mejor modelo (se seleccionaró el modelo con mejor equilibrio entre sesgo y varianza).
6. Dibujar la curva ROC-AUC y matriz de confusión.
7. Finalmente, se compararon los mejores modelos de todos los algoritmos y se seleccionó un ganador.
8. Obtener las características más importantes (*features importance*) del modelo ganador.

Por último, en la Tabla H.33 se muestran los parámetros y los valores de la rejilla asociados a cada algoritmo.

6.4.2. Regresión Logística

Tras la ejecución de la rejilla utilizando la función *Gridsearch* de Python, en la Tabla H.34 se muestran los diez mejores modelos.

En la Figura H.39 (a) se muestra una comparativa entre el parámetro *C* y el parámetro *solver*. Se observa como con *solver* "***saga"***, prácticamente para cualquier valor de *C*, se alcanza el ROC-AUC más alto. Además cuanto mayor sea el parámetro *C* mayor ROC-AUC para cualquier tipo de "*solver*". En la Figura H.39 (b), se estudia el parámetro "*penalty*" y el tipo de "*solver*". Vemos como con un "*penalty*" de tipo "*l1*" se obtienen los mejores resultados.

Una vez se seleccionaron los mejores modelos, se ejecutó una validación cruzada con 5 *k*-fold para poder comparar el sesgo y la varianza entre los modelos seleccionados (Figura 6.3). En la Figura 6.3 se observa como los modelos LR2, LR3, LR7, LR8 y LR9 son los que menos varianza tuvieron y un sesgo muy parecido al obtenido por mejor modelo (LR1). Finalmente, se seleccionó el modelo LR7 (modelo 1 para la comparación final), ya que es el modelo más equilibrado entre sesgo y varianza y cuyo parámetro *C* tiene menor valor (Tabla H.34), con lo que el riesgo de *overfitting* será menor. Una vez se seleccionó el mejor modelo, en la Figura H.40 se muestra la Curva ROC sobre el conjunto de datos *test*. Se observa como la clase con un área AUC más bajo (87%) es la clase 0 (sentimiento negativo) y la clase con área AUC más alto (94%) es la clase 2 (sentimiento positivo). Finalmente, en la Figura H.41 se muestra la Matriz de Confusión, en la que se aprecia de nuevo como la clase 0 (sentimiento negativo) es la que presenta un menor número de verdaderos positivos y un mayor número de falsos positivos.

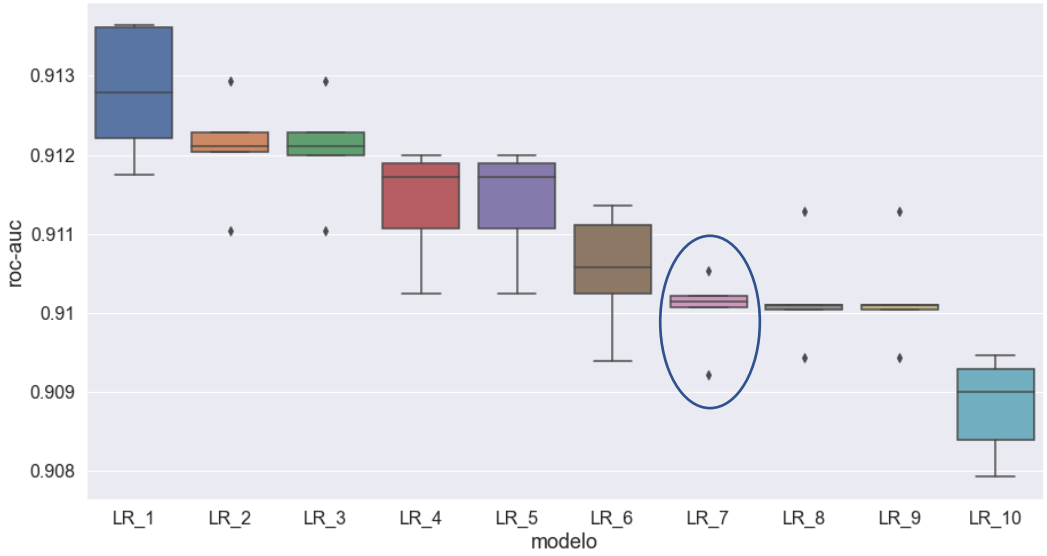


Figura 6.3: Resultados de la validación cruzada con RL.

6.4.3. Clasificador Naive Bayes para Modelos Multinomiales

Después de ejecutar la rejilla con *GridSearch*, en la Tabla H.35 se muestran los resultados de los mejores modelos. En la Figura H.42 se muestra cómo va variando el ROC-AUC según parámetro *Alpha*. Se observa como a para valores *alpha* < 3*,* el porcentaje de ROC-AUC es prácticamente el mismo, mientras que para valores *alpha* > *3*, los valores ROC-AUC son superiores hasta estancarse alrededor del 66%.

Una vez se seleccionaron los mejores modelos, se ejecutó una validación cruzada con 5 *k*-fold para poder comparar el sesgo y la varianza entre los modelos seleccionados (Figura 6.4). En la Figura 6.4, los modelos NB_3, NB_4 y NB_5 son menos robustos (tienen una mayor varianza) que los modelos NB_1 y NB_2, por lo que seleccionaremos el modelo NB_2 (modelo 2 para la comparación final), con un 66% de ROC-AUC, como modelo ganador en este apartado, ya que es el modelo más estable de los cinco. En la Figura H.43 se muestra la Curva ROC sobre el conjunto de datos *test*. Se observa como la clase con un área AUC más bajo (45%) es la clase 1 (sentimiento neutro) y la clase con un área AUC mayor (90%) es la clase 2 (sentimiento positivo). Finalmente, en la Figura H.44 se muestra la matriz de confusión, en la que podemos ver como todas las observaciones pertenecientes a la clase neutra del *test* las ha clasificado como clase 0 (sentimiento negativo).

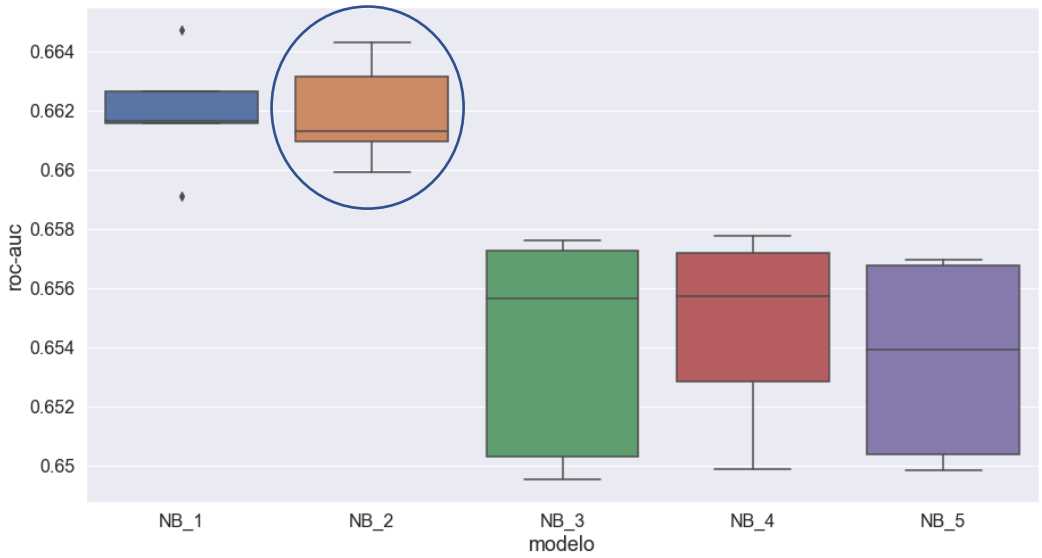


Figura 6.4: Resultados de la validación cruzada con MultinomialNB.

6.4.4. K-Nearest Neighbors

Dado que el algoritmo KNN se basa en el cálculo de distancias, fue necesario estandarizar (librería *StandardScaler* de Python) los datos de *train* y de *test* antes de ejecutar la rejilla. En la Tabla H.36 se muestran los resultados de los mejores modelos tras la ejecución de la rejilla. En la Figura H.45 se muestra cómo va variando el ROC-AUC según el número de vecinos, agrupados por el tipo de métrica o distancia. Se observa como aumenta el ROC-AUC a partir de 4 vecinos, para luego mantenerse en torno al 88% con 5, 6, 7 y 8 vecinos. En cuanto al parámetro "métrica", prácticamente no hay diferencias en este apartado aunque con la métrica "*cosine*", se obtiene una leve mejora.

Posteriormente, se ejecutó una validación cruzada con 5 *k*-fold para poder comparar el sesgo y la varianza entre los mejores modelos (Figura 6.5). En la Figura 6.5, se observa como casi todos los modelos, salvo el modelo KNN_1, son bastante robustos. Se seleccionó el modelo KNN_3 ya que es el modelo más simple (*k*-vecinos = 4) con prácticamente el mismo ROC-AUC (88,11 %) que el mejor modelo (KNN_2). En la Figura H.46 se muestra la Curva ROC sobre el conjunto de datos *test*. Se observa como la clase con un área AUC más bajo (83%) es la clase 0 (sentimiento negativo) y la clase con un área AUC mayor (94%) es la clase 1 (sentimiento neutro). Finalmente, en la Figura H.47 se muestra la matriz de confusión, en la que podemos ver como la clase con mejor ratio de verdaderos positivos es la clase neutra, seguida con la clase positiva y finalmente la clase negativa.

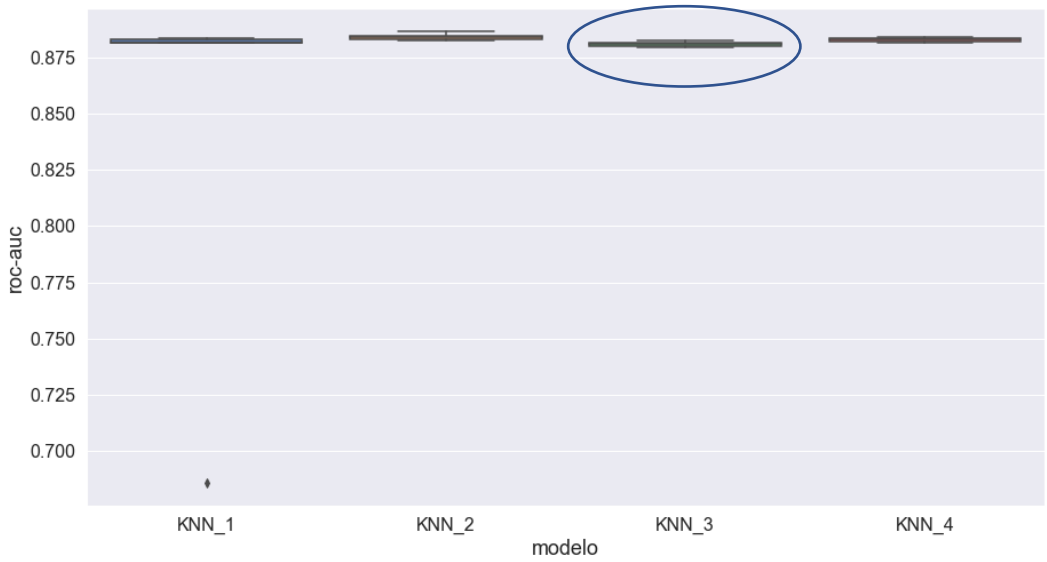


Figura 6.5: Resultados de la validación cruzada con KNN.

6.4.5. Random Forest

RF pertenece a la familia de los algoritmos basado en árboles y por lo tanto tiene un número elevado de parámetros para "tunear". Debido a ello, primero se ejecutó la rejilla de la Tabla H.33 (mediante la función *RandomizeSearchCV* de la librería *Scikit*-Learn) para estudiar los principales parámetros. Finalmente, se ejecutó una nueva rejilla (Tabla H.37) "tuneando" el parámetro *criterion*. En la Figura H.48 se muestra los resultados de la rejilla inicial. Se observa en la Figura H.48 (a) como *max_features* (número de características a sortear) igual a "*log2*" funciona mejor que con la opción "*auto*" con cualquier profundidad (*max_depth)*. Por otro lado, se aprecia como las configuraciones con una profundidad (*max_depth*) mayor a 40 el valor ROC-AUC va aumentando, llegando a superar el 80%. En la Figura H.48 (b) de nuevo el valor "*log2*" para el parámetro *max_features* funciona mejor con cualquier valor de *min_samples_leaf*. Cuando el tamaño de hoja (*min_samples_leaf*) es superior a 20 (en caso de que *max*_features sea "*log2*") los valores de ROC-AUC decrecen, ya que ajustando valores altos para este parámetro se evita que el modelo tienda al *overfitting* y se ajuste demasiado al conjunto de *train* (por contra de tener un modelo más simple). En la Figura H.48 (c) y Figura H.48 (d) se muestra como de nuevo la variable *max_features* con "*log2*" mejora a *auto*, sea cual sea los valores de *min_samples_split* y *n_estimators*. En cuanto a los valores de *min_samples_split*, parece que con valor pequeños (modelo más complejo) y altos (modelo más simple) se obtienen mejores resultados. Finalmente, con *n_estimators* igual a 400 se obtiene un ROC-AUC superior al 80%. Según los resultados del estudio anterior, se estableció la rejilla mostrada en la Tabla H.37. Los resultados de la segunda rejilla se muestran en la Tabla H.38 (el parámetro *n_estimators* se ha fijado en 400, el parámetro *min_samples_leaf* en 10 y max*_features en 'log2'*).

Una vez se seleccionaron los mejores modelos, se ejecutó una validación cruzada con 5 *k*-fold para poder comparar el sesgo y la varianza entre los mejores modelos (Figura 6.6). En la Figura 6.6, se observa como el modelo RF_3 es el más robusto de los cinco, por lo que será el mejor modelo en este apartado. En la Figura H.49 se muestra la Curva ROC sobre el conjunto de datos *test*. Se observa como la clase con un área AUC más bajo (75%) es la clase 0 (sentimiento negativo) y la clase con un área AUC mayor (86%) es la clase 1 (sentimiento neutro). Finalmente, en la Figura H.47 se muestra la matriz de confusión, en la que se muestra

como la clase con mejor ratio de verdaderos positivos es la clase neutra, seguida con la clase positiva y finalmente la clase negativa. Además, se ha clasificado casi un 40% de las observaciones negativas como neutras, por lo que no identifica demasiado bien esta clase.

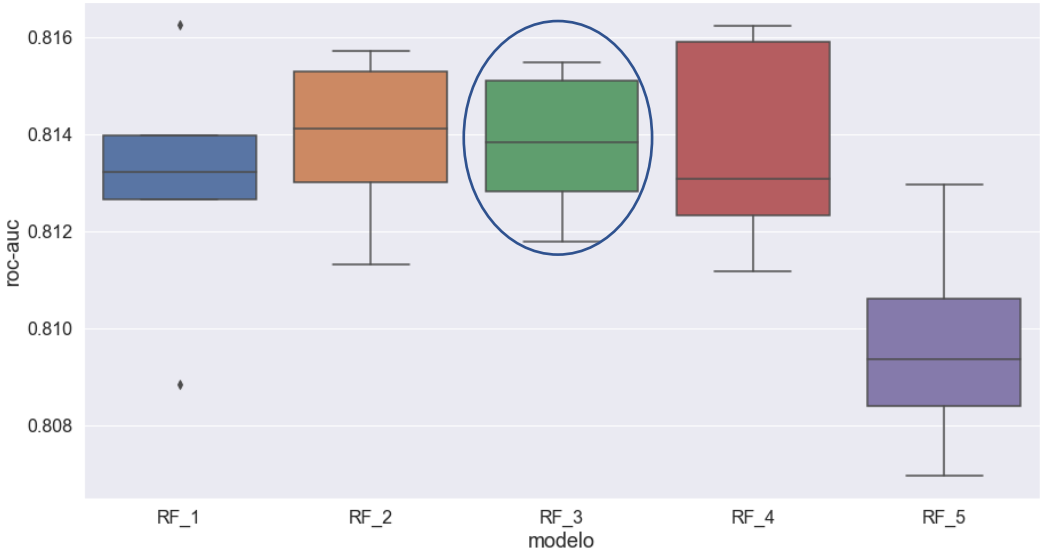


Figura 6.6: Resultados de la validación cruzada con RF.

6.4.6. XGBoost

Como en el apartado anterior, XGBoost pertenece a la familia de los algoritmos basado en árboles y por lo tanto, poseen un número elevado de parámetros para “tunear”. Por ello, primero se ejecutó la rejilla de la Tabla H.33 (mediante la función *GridSearchCV* de la librería *Scikit*-Learn) y posteriormente se estudió los principales parámetros. Con el objetivo de acelerar el proceso de entrenamiento, se activó la opción *gpu_hist* para utilizar la GPU en vez de la CPU. Finalmente, se ejecutó una nueva rejilla (Tabla H.39) “tuneando” el parámetro *gamma*. En la Figura H.51 se muestra los resultados de la rejilla inicial. En la Figura H.51 (a) se observa como el incremento del número mínimo de muestras por hoja (*min_child_weight*) se traduce en una bajada del ROC-AUC (modelos más robustos), ya que las líneas verticales (modelos con el mismo valor de *min_child_weight* pero con diferentes valores en otros parámetros) son más pronunciadas hacia abajo que hacia arriba. Además, con un *learning_rate* de 0.1 se obtienen mejores resultados (valores por encima del 83%). En la Figura H.51 (b) , se observa como los modelos con un *learning*_rate igual a 0.1 funcionan mejor (sea cual sea el valor del parámetro *n_estimators*) siendo los modelos con valores del parámetro *n_estimators* superiores a 300 los que ven incrementado su ROC-AUC de forma notable. En la Figura H.52 se muestran los parámetros estudiados en la Figura H.51 pero comparando con los valores obtenidos en el conjunto *train*. En general, no hay una gran separación entre los conjuntos de *train* y *val* salvo en el parámetro *min_child_weight*. Se observa como para valores altos de *min_child_ weight* la diferencia entre *train* y *val* es menor, a costa de aumentar el sesgo. A continuación, en la Tabla H.39 se muestra la configuración de la segunda rejilla con el objetivo de estudiar el parámetro *gamma* sobre la mejor configuración obtenida en la rejilla anterior. En la Figura H.53, las líneas de del conjunto *val* y *train* son más próximas, ya que aumentando el parámetro *gamma* seremos más conservadores, por lo que esté será el valor que se utilizó para regularizar la configuración con la que se ha obtenido mejores resultados (Tabla H.39).

Posteriormente, se ejecutó una validación cruzada con 5 *k*-fold para poder comparar el sesgo y la varianza entre los mejores modelos (Figura 6.7). En la Figura 6.6, el modelo XGB_1 es el mejor modelo en términos de ROC-AUC (85.7%), con una varianza muy parecida al resto de modelos por lo que será el modelo ganador. Una vez se seleccionó el mejor modelo, en la Figura H.54 se muestra la Curva ROC sobre el conjunto de datos *test*. La clase con un área AUC más bajo (80%) es la clase 0 (sentimiento negativo) y la clase con un área AUC mayor (89%) es la clase 1 (sentimiento neutro) y la clase 2 (sentimiento positivo). Finalmente, en la Figura H.55 se muestra la matriz de confusión, en la que se observa ver como la clase con mejor ratio de verdaderos positivos es la clase neutra, seguida de la clase positiva y finalmente la clase negativa. Además, el modelo clasificó casi un 25% de las observaciones negativas como neutras.

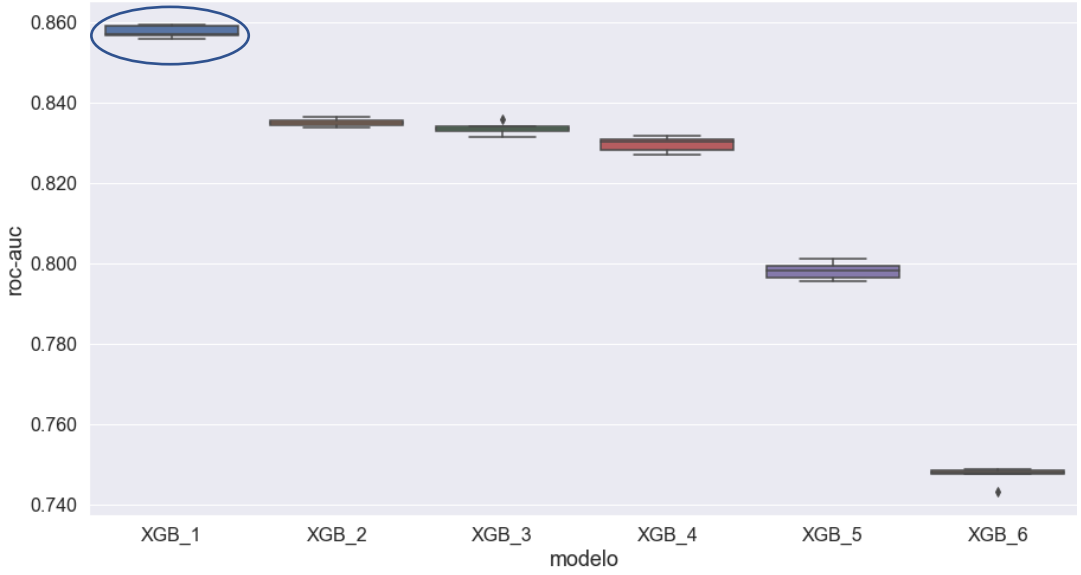


Figura 6.7: Resultados de la validación cruzada con XGBoost.

6.4.7. Support Vector Machine

Primero, al igual que en la sección 6.4.4 (KNN), fue necesario estandarizar los datos de entrada. A continuación, se ejecutaó la rejilla de la Tabla H.33 (mediante la función *GridSearchCV* de la librería *Scikit*-Learn) y posteriormente se estudiaron los principales parámetros. Con el objetivo de acelerar el proceso de entrenamiento, se configuró el parámetro *max_iter* (número máximo de iteraciones que tiene el *solver* para resolver el problema de optimización cuadrática con el objetivo de encontrar el hiperplano de margen máximo) igual a 10.000. En la Figura H.56 se muestra los resultados de la rejilla inicial. En la Figura H.51 (a) se muestran los resultados de la rejilla con el *kernel* lineal. Se observa cómo no se obtiene ninguna mejora para valores de *C* mayores de 0.01 y como con *C* igual a 0.001 se obtiene el ROC-AUC más alto (88%). En Figura H.51 (b) se muestran los resultados de la rejilla para el *kernel poly*. Observamos como para el valor más pequeño del parámetro *gamma*, obteniendo mejores resultados cuando el parámetro *C* es mayor de 0.01, mientras que para el resto de los valores del parámetro *gamma* se obtienen mejores resultados cuando el parámetro *C* es pequeño. En la Figura H.51 (c) se muestran los resultados de la rejilla para el *kernel rbf*. Se aprecia como para valores de *gamma* mayores de 0.01, el parámetro *C* apenas varía el desempeño del algoritmo, salvo en el caso de *gamma igual a 0.1 y C* igual a 1, en el que se aprecia una bajada importante de ROC-AUC. Por el contrario, para valores de *gamma* muy pequeños, se incrementa el parámetro *C*, aumentando el ROC-AUC. Por último,

en la Figura H.51 (d) y Figura H.51 (e) se muestran los resultados obtenidos para la rejilla con *LinearSVC*. En general, se obtienen mejores resultados con el parámetro *loss* igual a *hinge* y con valores de *C* menores a 1.

A continuación, en la Tabla H.41 se muestra la configuración de la segunda rejilla con el objetivo de estudiar el parámetro *gamma* sobre la mejor configuración obtenida en el estudio anterior. Una vez se seleccionaron los modelos, se ejecutó una validación cruzada con 5 *k*-fold (Figura 6.8). En la Figura 6.8, se observa como el modelo *LinearSVC* es el mejor modelo en términos de ROC-AUC, ya que presenta un porcentaje superior del 92% y una varianza muy pequeña, por lo que será el modelo ganador. Una vez se seleccionó el mejor modelo, en la Figura H.57 se muestra la Curva ROC sobre el conjunto de datos *test*. Se aprecia como la clase con un área AUC más bajo (88%) es la clase 0 (sentimiento negativo) y la clase con un área AUC mayor (95%) es la clase 2 (sentimiento positivo) siendo este modelo con el que se ha obtenido mejores resultados en este apartado. Finalmente, en la Figura H.58 se muestra la matriz de confusión, en la que podemos ver como la clase con mejor ratio de verdaderos positivos es la clase neutra, seguida con la clase positiva y finalmente la clase negativa. Además, la ratio falsos positivos en todas las clases es el menor de todas los modelos hasta ahora.

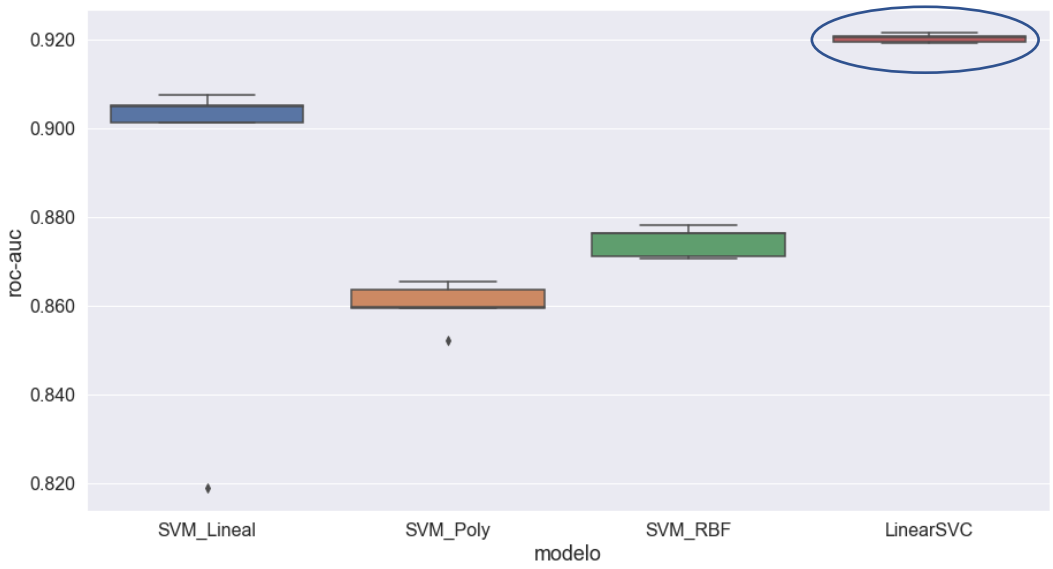


Figura 6.8: Resultados de la validación cruzada con SVM.

6.4.8. Redes Neuronales

Como en el apartado anterior, se debe estandarizar los datos entrada, ya que tener los datos en la misma escala pueden ayudar en la convergencia de los algoritmos de optimización empleados en las ANN. Como se vio en la sección 2.4.7.8, uno de los principales parámetros que se deben configurar es el número de nodos de la capa oculta, que dependerá tanto del número de características como de su complejidad. Según la ecuación 2.9:

$$1329h + 1 = \frac{105.000\ observaciones\ (train)}{20} = 4\ nodos \text{ (aprox)}$$

Debido al alto coste computacional que conlleva entrenar las ANN, primero se estudió el número de *epochs* (*early stopping)* (Tabla H.42). En la Figura H.59, Figura H.59 (b) y Figura H.59 (c) se aprecia ver como a partir de 20-30 *epochs* (aproximadamente) la función de pérdida o *los function*, el *accuracy* y ROC-AUC del conjunto *train* y del conjunto *val* comienzan

a separarse, por lo que un número superior a 30 *epochs* llevaría al *overfitting*. Por ello, la configuración de la rejilla es la que se muestra en la Tabla H.33 (mediante la función RandomizeSearchCV de la librería *Scikit*-Learn). La arquitectura final de la red consta de dos capas ocultas. Además, por cada una de las capas, se probó un número de nodos o neuronas y diferentes funciones de activación.

En la Figura H.60 se muestra los resultados de la rejilla indicial. En la Figura H.60 (a) se observa como con un *learn_rate* de 0.001 y un *drop_*out de 0 se alcanza un ROC-AUC por encima del 90%. En la Figura H.60 (b), se observa como con 6 neuronas por capa (un total de 12 neuronas) se obtuvo un ROC-AUC levemente superior que con 2 y con 4 neuronas por capa. En la Figura H.60 (c) se muestran el número de épocas (*epochs*) agrupadas por el tipo de función de activación (*activation)*. Se observa como con pocas épocas alcanzamos buenos resultados y con la función *tanh* obtenemos siempre mejores resultados que con *relu*. En la Figura H.60 (d) se muestra como con un *batch_size* de 64, el ROC-AUC obtenido es levemente superior que con 32. Finalmente, En la Figura H.60 (e), se muestra la comparación entre los diferentes valores de *learn_rate* agrupados por el tipo de función de activación. Se puede ver como de nuevo con un *learn_rate* de 0.001 y con la función *tanh* se obtienen los mejores resultados.

Según los resultados del estudio anterior, se estableció la configuración para la validación cruzada que se muestra en Tabla H.43.

En la Figura 6.9 , se aprecia como el modelo ANN_3 es el más robusto de los cinco, por lo que fue el modelo ganador. Tras seleccionar el mejor modelo, en la Figura H.61 se muestra la Curva ROC sobre el conjunto de datos *test*. Se observa como la clase con un área AUC más bajo (86%) es la clase 0 (sentimiento negativo) y la clase con un área AUC mayor (94%) es la clase 1 (sentimiento neutro). Finalmente en la Figura H.62 se muestra la matriz de confusión, en la que se observa como la clase con mejor ratio de verdaderos positivos es la clase neutra, seguida con la clase positiva y finalmente la clase negativa.

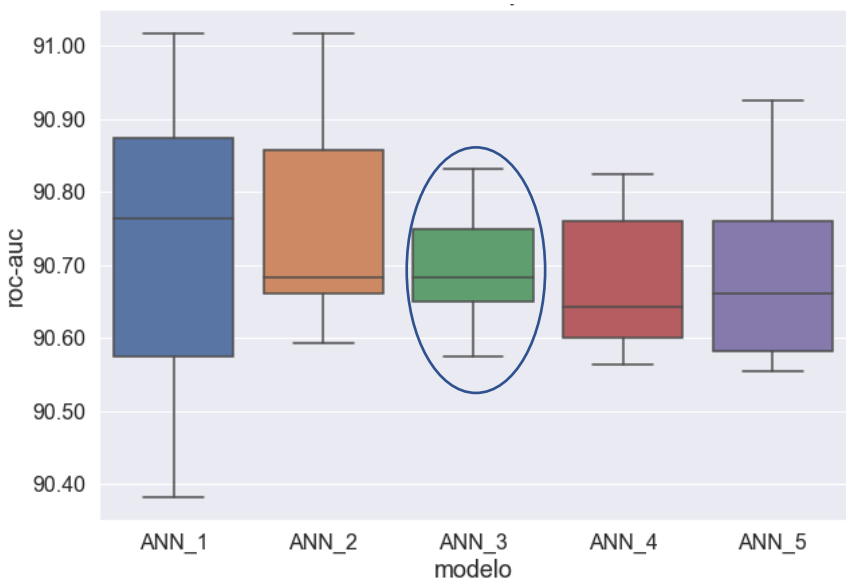


Figura 6.9: Resultados de la validación cruzada con ANN.

6.4.9. Ensamblado de modelos

Por último, se construyeron varios ensamblados con los mejores modelos obtenidos en los apartados anteriores, con la finalidad de intentar mejorar los resultados obtenidos por cada modelo por separado. Para ello, en la Tabla H.44 se muestra la configuración de los ensamblados (no se ha incluido el modelo obtenido en la sección 6.4.3 ya que los resultados obtenidos fueron peores en comparación con el resto de los algoritmos). Por un lado, se han agrupado los modelos basados en árboles, por otro el resto de los modelos y por último todos los modelos juntos. En la Figura 6.10 se muestran los resultados de la validación cruzada con los ensamblados mencionados en la Tabla H.44. Se observa cómo no se obtuvo ninguna mejora en términos de ROC-AUC comparado con el mejor modelo de RL, SVM o ANN por separado. El mejor modelo en este apartado sería el ensamblado *stacking_1*, ya que es el modelo más simple (sin contar el ensamblado de árboles) con el que se alcanza un rendimiento por encima del 92%.

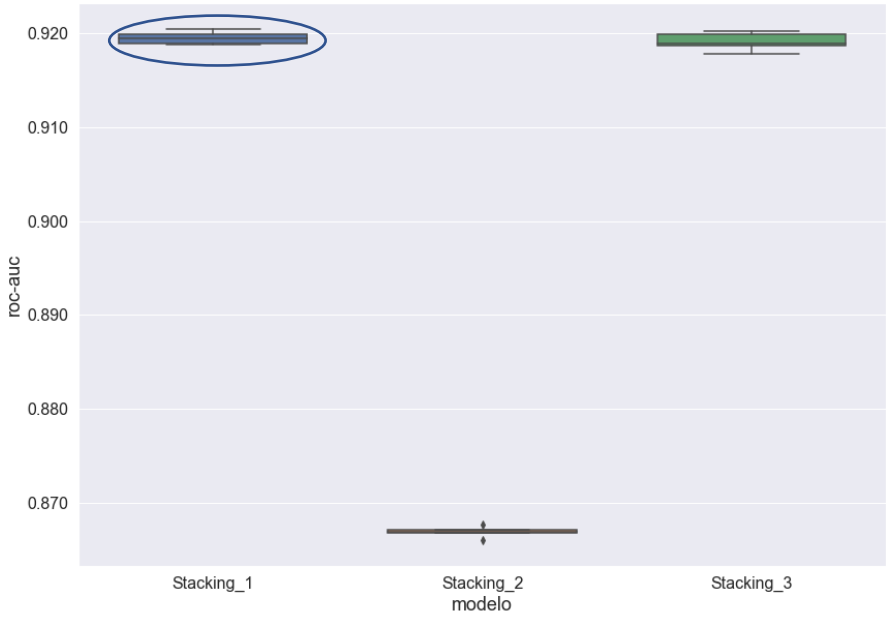


Figura 6.10: Resultados de la validación cruzada con Ensamblados.

6.4.10. Comparación de modelos

Finalmente, en la Figura 6.11 se muestra una comparativa de los modelos obtenidos en las diferentes secciones del modelado. Se peude concluir como el mejor modelo *LinearSVC*, obtenido en la sección 6.4.7, tanto por su rendimiento en términos de ROC-AUC como por su estabilidad.

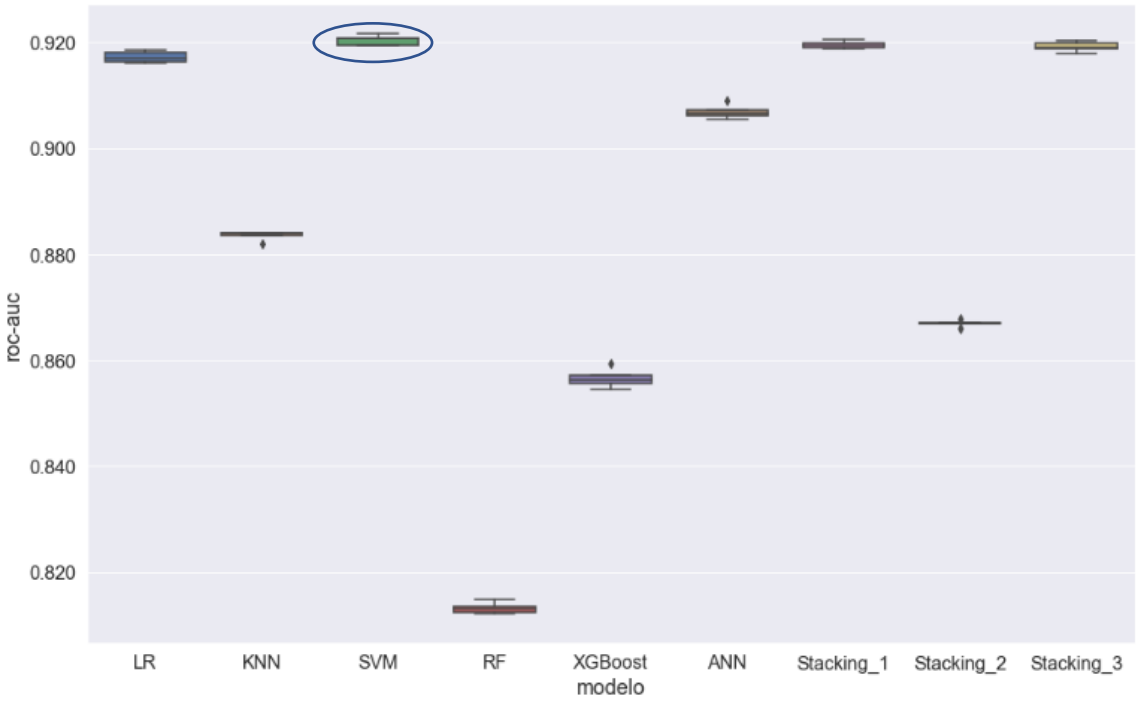


Figura 6.11: Resultados de la validación cruzada con todos los modelos.

A continuación, en la Figura H.63 se muestra el resultado del *test* (45.000 observaciones) sobre el modelo *LinearSVC*. Se aprecia como todas las principales métricas están cerca o incluso superan el 90%. Además, no se observa un descenso entre las métricas obtenidas del conjunto *test* y las que se obtuvieron en la validación cruzada con el conjunto *train* (Tabla H.41)(no parece que exista *overfitting*) por lo que el modelo final es bastante robusto.

6.4.11. Interpretabilidad del modelo

Dado uno de los objetivos principales de este trabajo es construir un clasificador de sentimientos que sea interpretable, no solo para para corregir posibles sesgos (o descubrirlos), sino también para que los modelos construidos puedan explicar sus decisiones. Para ello, en la Figura 6.12, Figura 6.13 y Figura 6.14, se muestran las 20 palabras que más han contribuido (de forma positiva) en el modelo (extraídas mediante el atributo *coef_*, que recogen los pesos asignados a las características). El haber utilizado únicamente los "unigramas" en el conjunto de entrada, limita de forma notable los posibles *insights* que se puedan extraer del análisis de las características más importantes. Por ejemplo, en la Figura 6.12 se observan palabras negativas, posiblemente relacionadas con la vacuna, como "*flu*", "*refuse*", "*allergic*", "*sick*" y otras palabras negativas como "*bad*", "*die*" o "*death*". En la Figura 6.13, donde se muestran las palabras que más contribuyeron a la clase neutra, cabe destacar la presencia de la vacuna de "*jansen*". Finalmente, en la Figura 6.14, se puden ver palabras positivas relacionados con la vacuna, como "*effective*", "*approved*", "*safe*", "*safety*" y otros términos como "*like*", "*good* o "*hope*". Para evitar tener visión demasiado optimista, en la Figura 6.15, se muestran las características más importantes sobre el conjunto de datos *test* calculadas mediante la librería Eli5. El cálculo de las características con el conjunto *train* permite conocer que variables influyen en el modelo para hacer predicciones mientras que el cálculo de las características con el conjunto *test* da una medida de la contribución de cada variable al rendimiento del modelo en datos no vistos. La diferencia de calcular la importancia

de las características sobre el conjunto de datos *train* y *test* podría indicar una tendencia del modelo al sobreajuste.

Para ello, se utilizó la función *Permutation Feature Importante* de Eli5 para medir el aumento del error de predicción del modelo después de permutar los valores de la característica, rompiendo así la relación entre la característica y el resultado real. El primer número de cada fila muestra cuánto disminuyó el rendimiento del modelo con un barajado aleatorio (en este caso, utilizando la ROC-AUC como métrica de rendimiento). La magnitud de aleatoriedad en el cálculo de la importancia de la permutación se mide repitiendo el proceso con múltiples barajados. El número que aparece después del ± mide la variación del rendimiento de una barajada a la siguiente. Ocasionalmente, pueden aparecer valores negativos para la importancia de las permutaciones. En esos casos, las predicciones sobre los datos barajados (o con ruido) resultaron ser más precisas que los datos reales. Esto ocurre cuando la característica no importaba (debería haber tenido una importancia cercana a 0), pero el azar hizo que las predicciones sobre los datos barajados fueran más precisas (esto es más habitual en conjuntos de datos pequeños).

En la sección coloreada de verde de la Figura 6.15, se observa como las características más importantes están relacionadas con las vacunas. Además, muchas de ellas aparecen en la Figura 6.12, Figura 6.13 y Figura 6.14, como por ejemplo, '*like'*, '*flu*', '*good*', '*effective*", '*approved*, '*safe*", "*stop* o *refuse*'. Además, en la parte derecha de la Figura 6.15 se muestran las características menos importantes. Se observa como ninguna de ellas tiene un valor negativo, por lo que ninguna característica que no era relevante en el conjunto *train,* ahora tampoco lo es. Podemos concluir que en nuestro modelo final, *LineraSVM* es bastante robusto.

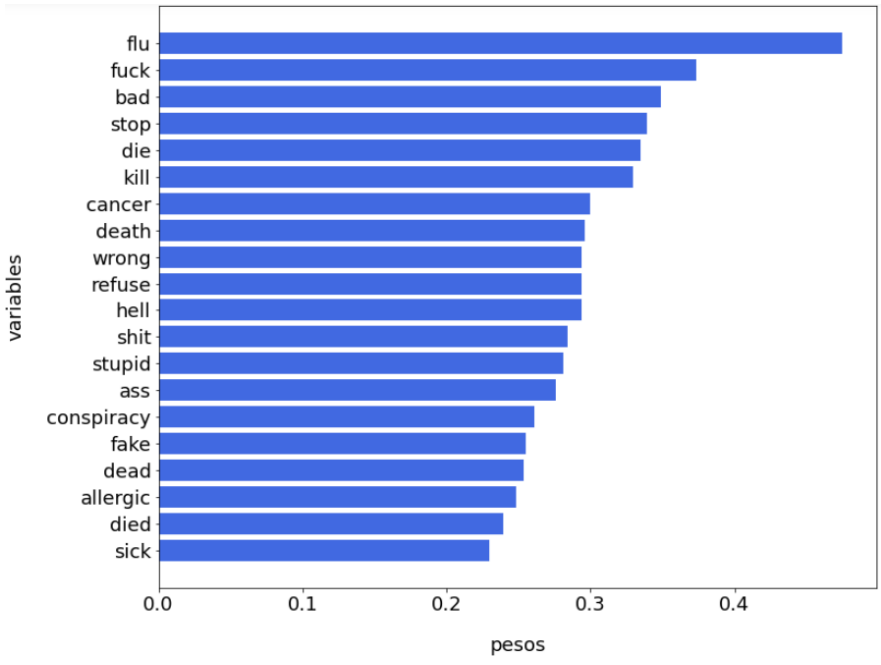


Figura 6.12: Contribución de las 20 principales variables a la clase 0 (sentimiento negativo)

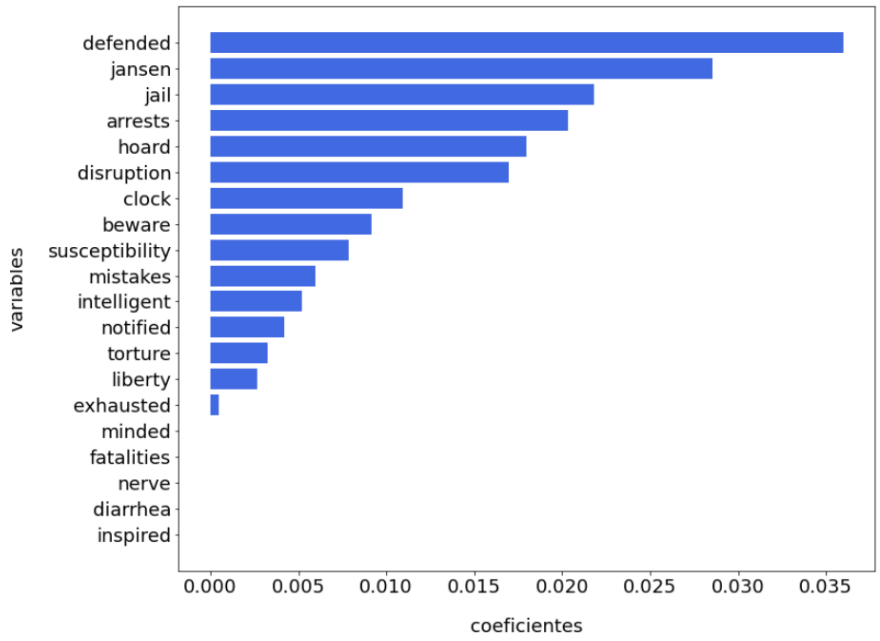


Figura 6.13: Contribución de las 20 principales variables a la clase1 (sentimiento neutro)

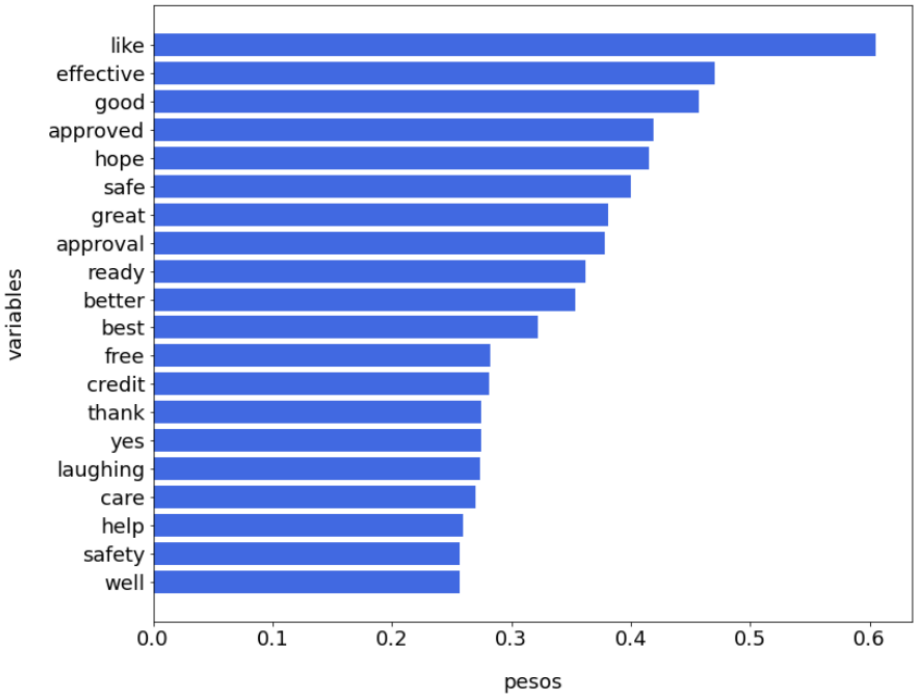
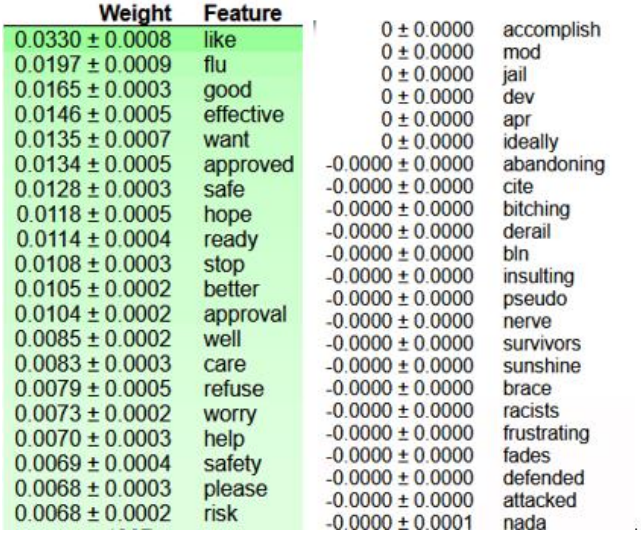


Figura 6.14: Contribución de las 20 principales variables a la clase 2 (sentimiento positivo).



Weight	Feature
0.0330 ± 0.0008	like
0.0197 ± 0.0009	flu
0.0165 ± 0.0003	good
0.0146 ± 0.0005	effective
0.0135 ± 0.0007	want
0.0134 ± 0.0005	approved
0.0128 ± 0.0003	safe
0.0118 ± 0.0005	hope
0.0114 ± 0.0004	ready
0.0108 ± 0.0003	stop
0.0105 ± 0.0002	better
0.0104 ± 0.0002	approval
0.0085 ± 0.0002	well
0.0083 ± 0.0003	care
0.0079 ± 0.0005	refuse
0.0073 ± 0.0002	worry
0.0070 ± 0.0003	help
0.0069 ± 0.0004	safety
0.0068 ± 0.0003	please
0.0068 ± 0.0002	risk
0 ± 0.0000	accomplish
0 ± 0.0000	mod
0 ± 0.0000	jail
0 ± 0.0000	dev
0 ± 0.0000	apr
0 ± 0.0000	ideally
-0.0000 ± 0.0000	abandoning
-0.0000 ± 0.0000	cite
-0.0000 ± 0.0000	bitching
-0.0000 ± 0.0000	derail
-0.0000 ± 0.0000	bln
-0.0000 ± 0.0000	insulting
-0.0000 ± 0.0000	pseudo
-0.0000 ± 0.0000	nerve
-0.0000 ± 0.0000	survivors
-0.0000 ± 0.0000	sunshine
-0.0000 ± 0.0000	brace
-0.0000 ± 0.0000	racists
-0.0000 ± 0.0000	frustrating
-0.0000 ± 0.0000	fades
-0.0000 ± 0.0000	defended
-0.0000 ± 0.0000	attacked
-0.0000 ± 0.0001	nada

Figura 6.15: Resultados de *Permutation feature importance* sobre el *test* con Eli5.

7. Conclusiones, Limitaciones y Trabajo Futuro

En este trabajo se analizaron mediante técnicas de PLN y de ML, un total de 4 millones de *tweets* relacionados con la vacuna contra la COVID-19 (del 15 de noviembre de 2020 al 16 de diciembre de 2020). El uso de tecnología avanzada, como el aprendizaje automático y el procesamiento del lenguaje natural en particular, el análisis de sentimientos y el análisis de textos puede acelerar el proceso de maduración de la comprensión de la opinión pública en el contexto de la propagación de enfermedades infecciosas como la COVID-19. Este trabajo saca a la luz las posibles razones de la desconfianza en las vacunas. Además, el trabajo muestra la eficacia y la utilidad del ML y el análisis de texto para realizar estudios rápidos de grandes conjuntos de datos de datos de las redes sociales para conocer la opinión pública en el área de la salud pública y las enfermedades infecciosas.

A continuación se exponen las principales conclusiones obtenidas de los dos objetivos de este trabajo.

7.1. Conclusiones para el Objetivo I: Percepción de la vacuna contra la COVID-19.

Con respecto al primer objetivo, en este trabajo se pretende obtener una idea general de los sentimientos hacia la vacuna contra la COVID-19. Analizando los resultados globales, el número de *tweets* con sentimiento positivo fue levemente superior al de los negativos. Además, no se apreció una polaridad muy fuerte, ni positiva ni negativa especto a la vacuna, en los *tweets* analizados Durante la duración del análisis, no se apreciaron cambios en los sentimientos de los *tweets* en las tres principales fechas analizadas (aprobación de la vacuna, primera persona vacunada en Reino Unido y EE. UU).

Según los resultados obtenidos en los *tweets* con la geolocalización activada, en su mayoría de EE. UU, hay un porcentaje importante de *tweets* negativos pertenecientes a usuarios de estados del centro y sur de EE. UU, que podrían poner en riesgo la inmunidad de grupo. Si bien es cierto que los sentimientos actuales hacia la vacuna serán diferentes a los analizados en este trabajo (principalmente por el número de personas vacunadas y los pocos efectos adversos), es muy importante analizar los motivos por los que las personas tienen una percepción negativa hacia las vacunas (por ejemplo con el Modelado de Tópicos, análisis de n-gramas) ya que, con la aparición de nuevas variantes más infecciosas como la variante Delta, será necesario subir el porcentaje de personas vacunadas (actualmente en un 70% aproximadamente). Además, como menciona el Doctor Fauci, pronto puede haber 'dos EE.UU.' a medida que se amplía la brecha entre áreas vacunadas y no vacunadas.

Como se observa en la Figura 7.2, los estados con un menor número de casos por cada 100.000 habitantes en los últimos 7 días (actualizado a 6 de julio de 2021) se sitúan en la costa este mientras que los estados con un mayor número de casos son aquellos del centro y sur de EE. UU. Tras analizar los resultados de los mapas y tablas generados, en especial Figura H.10 y Figura 5.8 los *tweets* con un sentimiento positivo hacia la vacuna pertenecían en su mayoría a la costa este (tanto por el análisis de la media del sentimiento como por la diferencia de porcentaje) y los *tweets* con un sentimiento neutro y negativa se sitúan en el interior y en el sur del país. Teniendo en cuanto los estados con una percepción positiva, neutra y negativa hacia la vacuna , como vemos en la Figura H.10 y la Figura 7.2, los estados que presentan un mayor número de casos son en su mayoría los que tienen una percepción negativa hacia la vacuna (en especial con dos dosis) mientras que aquellos estados con una

percepción positiva, al tener un mayor porcentaje de vacunación, han visto disminuido los casos.

Mediante el análisis de los *n*-gramas y el Modelado de Tópicos, se puede concluir como los principales temas y preocupaciones relacionados con la vacuna fueron: la efectividad de la vacuna, la seguridad de la vacuna, los efectos secundarios (en especial como puede influir la vacuna en la fertilidad y las posibles reacciones alérgicas), como será distribuida la vacuna y qué grupo de población será el primero en recibir las primeras dosis. Por todo lo anterior, es necesario concienciar a la población de la importancia de vacunarse contra la COVID-19, pues con la aparición de nuevas variantes, puede ser necesario un mayor porcentaje de personas vacunadas papara alcanzar la inmunidad de rebaño.

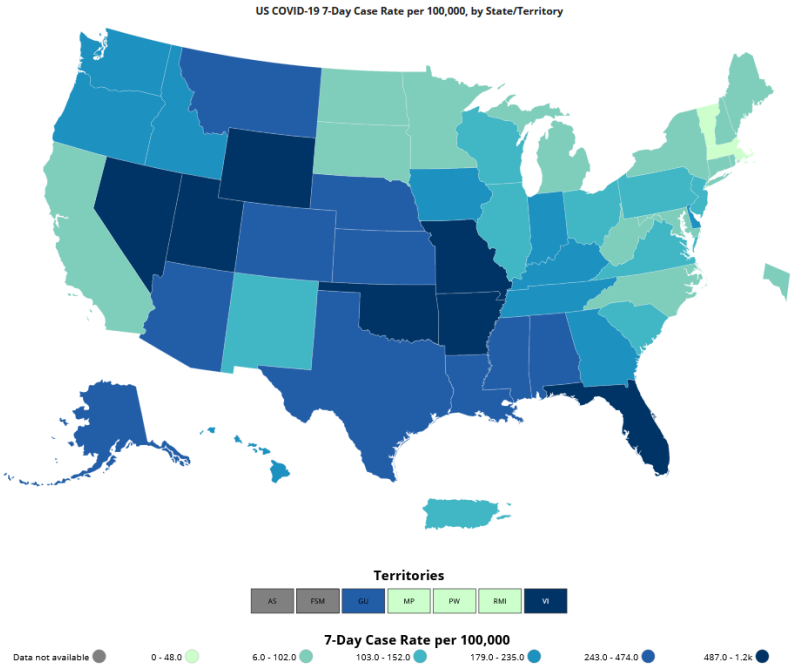


Figura 7.1: Tasa de casos de COVID-19 en 7 días por cada 100.000 por Estado.

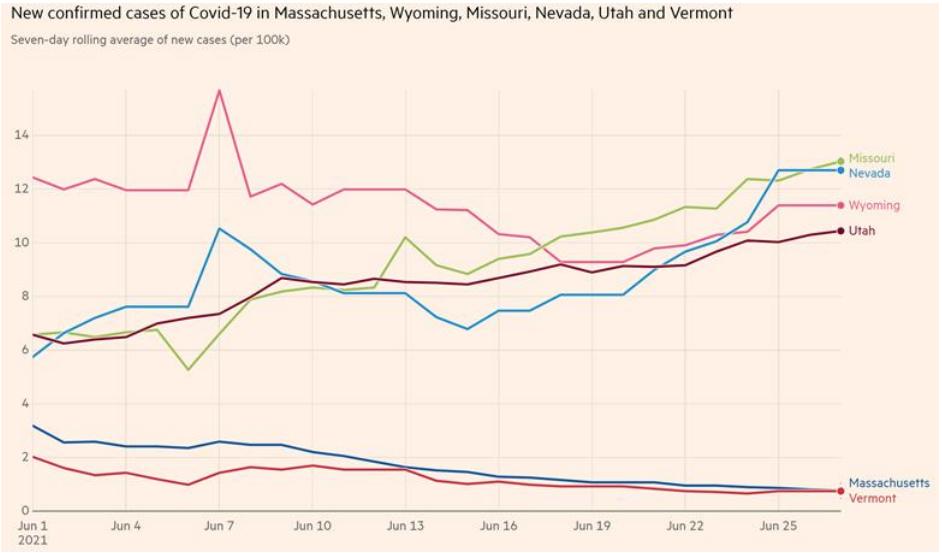


Figura 7.2: Número de nuevos casos de COVID-19 en EE. UU.

7.2. Conclusiones para el Objetivo II: Construcción de un Clasificador de Sentimientos.

En esta segunda línea de investigación, el objetivo fue construir un clasificador de sentimientos interpretable que permita identificar de manera sencilla el sentimiento asociado a los *tweets* de la red social *Twitter*. Se ha demostrado como con un *corpus* mediano, se pueden alcanzar grandes resultados, por lo que si se contará con datos etiquetados (como por ejemplo *reviews* de productos) los clasificadores basados técnicas de ML tendrían un mejor desempeño que los clasificadores basados en reglas, ya que estos clasificadores utilizan una lista de palabras predefinidas para identificar el sentimiento positivo o negativo. Además, el modelo construido da una respuesta simplificada del sentimiento encontrado (negativo, neutro o positivo), comparado a otras librerías basadas en léxicos, como VADER, que puntúan con un score el texto de entrada.

Además, se ha visto como con un preprocesamiento sin *stopwords* customizadas, sin emplear técnicas como la lematización que suponen un coste importante a nivel computacional y sin corrección ortográfica, se obtienen mejores resultados. También se comprobado como con un método de ponderación básico como el *CountVectorizer* con "unigramas" y reduciendo el número de variables con el método RFECV se obtienen resultados iguales o superiores a los obtenidos con métodos un poco más complejos como *TfidfVectorizer*.

Como se ha visto en el apartado de modelado, los mejores modelos han sido la RL, SVM y ANN. Teniendo en cuenta el sesgo, la varianza e interpretabilidad, se seleccionó el modelo *LinearSVC*, a pesar de que sea un algoritmo más "duro" de entrenar, la matriz de confusión obtenida sobre el conjunto de datos *test* mejora a la obtenida con otros modelos más sencillos como puede ser la Regresión Logística.

Con el modelo *LinearSVC* se obtuvoun ROC-AUC del 92% y un f1-score cercano al 90%, con una varianza muy pequeña entre el conjunto de datos *train* y *test*, lo cual indica la gran capacidad de generalización que tiene el modelo seleccionado.

7.3. Limitaciones

Como cualquier otro trabajo, este estudio no está exento de limitaciones. En primer lugar, el análisis de los sentimientos se ha realizado en un periodo de tiempo específico, por lo que el sentimiento del público puede haber cambiado con el paso del tiempo. En segundo lugar, los sentimientos expresados en los *tweets* pueden no ser veraces e introducir elementos de sesgo en los datos. En tercer lugar, la subjetividad introducida en el uso de diversas técnicas como las *stopwords*, la lemtización, etc., puede afectar a los resultados. Por último, debido a la gran cantidad de datos disponibles en las redes sociales, se hace inviable realizar un etiquetado manual por lo que el etiquetado se realizó con la librería VADER , por lo que las métricas de desempeño de estos modelos serán en todos los casos similares a las obtenidas por los modelos basados en reglas. A pesar de ello, en este trabajo se ha demostrado el gran rendimiento de los modelos basados en ML en el campo de análisis de sentimiento si se cuenta con un *corpus* grande y heterogéneo, pudiendo de esta manera servir como base para entrenar otros conjuntos de datos de diferentes dominios.

7.4. Trabajo Futuro

Como trabajo futuro, se podría investigar si existe un periodo óptimo en el que la información pueda presentarse en tiempo real para crear una influencia positiva y mantenerla activa en la memoria de las personas. Además, se podría entrenar el modelo obtenido con un *corpus* etiquetado de forma manual y adaptarlo posteriormente a un domino específico mediante *Transfer Learning*, evitando así la etapa de entrenamiento. En este sentido, un ejemplo del uso del aprendizaje por transferencia sería BERT, que utilizando *Transfer Learning*, podemos configurar el modelo (previamente entrenado) en el área o dominio que necesitado. Por otro lado, se podría usar este tipo de redes neuronales para construir un clasificador de sentimientos sin necesidad de etiquetar los datos previamente. Dichos algoritmos emplean unas redes neuronales llamadas *Transformers* que permiten aprender las relaciones contextuales entre las palabras, mejorando los resultados de obtenidos con métodos clásicos como TF-IDF.

8. Bibliografía y Referencias

[Fahm20] N. Fahmi, “Artificial Intelligence, Machine Learning, and Deep Learning — What the Difference?” 2020.

[KKKS17] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural Language Processing: State of the Art, Current Trends and Challenges,” *arXiv*, no. Figure 1, 2017.

[SSAA19] Y. A. Solangi, Z. A. Solangi, S. Aarain, A. Abro, G. A. Mallah, and A. Shah, “Review on Natural Language Processing (NLP) and Its Toolkits for Opinion Mining and Sentiment Analysis,” *2018 IEEE 5th Int. Conf. Eng. Technol. Appl. Sci. ICETAS 2018*, pp. 1–4, 2019.

[DCLT19] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[Laca14] G. De La Calle Velasco, “Modelo Basado En Técnicas de Procesamiento de Lenguaje Natural Para Extraer y Anotar Información de Publicaciones Científicas,” Tésis Doctoral, Universidad Politécnica de Madrid). Recuperado de http://oa~…, 2014.

[Chom14] N. Chomsky, *Aspects of the Theory of Syntax*, vol. 11. MIT press, 2014.

[Sas20] SAS, “Procesamiento Del Lenguaje Natural.” 2020.

[Orei20] OREILLY, “Chapter 4. Preparing Textual Data for Statistics and Machine Learning.” 2020.

[Pai20] A. Pai, “What Is Tokenization in NLP? Here’s All You Need To Know.” 2020.

[Gane19] K. Ganesan, “All You Need to Know about Text Preprocessing for NLP and Machine Learning.” 2019.

[KCTH20] Y. Kang, Z. Cai, C. W. Tan, Q. Huang, and H. Liu, “Natural Language Processing (NLP) in Management Research: A Literature Review,” *J. Manag. Anal.*, vol. 7, no. 2, pp. 139–172, 2020.

[KGVI15] S. Kannan *et al.*, “Preprocessing Techniques for Text Mining,” *Int. J. Comput. Sci. Commun. Networks*, vol. 5, no. 1, pp. 7–16, 2015.

[BaLl00] V. Balakrishnan and E. Lloyd-yemoh, “Rp030_l3007,” pp. 174–179.

[WDSC19] T. Wolf *et al.*, “Transformers: State-of-the-Art Natural Language Processing,” *arXiv*, pp. 38–45, 2019.

[LiLi12] B. Liu and B. Liu, *Sentiment Analysis and Opinion Mining Opinion Mining*. 2012.

[YaSZ17] A. Yadollahi, A. G. Shahraki, and O. R. Zaiane, “Current State of Text Sentiment Analysis from Opinion to Emotion Mining,” *ACM Comput. Surv.*, vol. 50, no. 2, 2017.

[AgBM09] A. Agarwal, F. Biadsy, and K. R. McKeown, “Contextual Phrase-Level Polarity Analysis Using Lexical Affect Scoring and Syntactic N-Grams,” *EACL 2009 - 12th Conf. Eur. Chapter Assoc. Comput. Linguist. Proc.*, no. April, pp. 24–32, 2009.

[Katr19] A. Katrekar, “An Introduction to Sentiment Analysis.” 2019.

[AZZA20] A. H. Alamoodi *et al.*, “Sentiment Analysis and Its Applications in Fighting COVID-19 and Infectious Diseases: A Systematic Review,” *Expert Syst. Appl.*, p. 114155, 2020.

[ChHZ15] W. Chung, S. He, and D. Zeng, “EMood: Modeling Emotion for Social Media Analytics on Ebola Disease Outbreak,” *2015 Int. Conf. Inf. Syst. Explor. Inf. Front. ICIS 2015*, vol. 1976, no. March 2014, pp. 1–10, 2015.

[Prak19] N. Prakash, “A Comparative Study of Lexicon Based and Machine Learning Based Classifications in Sentiment Analysis,” *Int. J. Data Min. Tech. Appl.*, no. December, pp. 8–13, 2019.

[Lisi69] V. A. Lisichkin, “The Process of Making Forecasts,” *Technol. Forecast.*, vol. 1, no. 1, pp. 97–104, 1969.

[BaES10] S. Baccianella, A. Esuli, and F. Sebastiani, “SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining,” *Proc. 7th Int. Conf. Lang. Resour. Eval. Lr. 2010*, vol. 0, pp. 2200–2204, 2010.

[Gyal20] O. G.Yalçın, “Sentiment Analysis in 10 Minutes with Rule-Based VADER and NLTK.” 2020.

[HuGi14] C. J. Hutto and E. Gilbert, “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text,” *Proc. 8th Int. Conf. Weblogs Soc. Media, ICWSM 2014*, no. January, pp. 216–225, 2014.

[Text19] TextBlobDevelopers, “Textblob Tutorial, Quickstart.” 2019.

[Erem19] R. Eremyan, “Four Pitfalls of Sentiment Analysis Accuracy.” 2019.

[Port20] J. Portela García-Miguel, “Temario de La Asignatura de Machine Learning.” 2020.

[Eiki19] Eiki, “Feature Extraction in Natural Language Processing with Python.” 2019.

[Rodr19] C. Rodríguez Abellán, “Word Embeddings: Cómo La IA Nos Muestra La Evolución de Las Palabras.” 2019.

[PeSM14] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, no. June 2018, pp. 1532–1543, 2014.

[MCCD13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.

[JGBD16] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “FastText.Zip:

Compressing Text Classification Models," pp. 1–13, 2016.

[Kim18] R. Kim, "Another Twitter Sentiment Analysis with Python — Part 4 (Count Vectorizer, Confusion Matrix)." 2018.

[Taco20] tacosdedatos, "De Texto a Vectores." 2020.

[Deve13] S. A. S. Developers, "SAS® Text Miner 12.3 Reference Help." 2013.

[Alon20] J. M. Alonso Revenga, "Temario de La Asignatura de Complementos de Formación En Técnicas de Minería de Datos." 2020.

[Calv20] A. Calviño Martínez, "Temario de La Asignatura de Técnicas de Minería de Datos (SEMMA)." 2020.

[IgEl03] I. Iguyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.*, vol. 3, no. April, pp. 1157–1182, 2003.

[Roma19] V. Roman, "Algoritmos Naive Bayes: Fundamentos e Implementación." 2019.

[Deve00] S.-L. Developers, "Multinomial Naive Bayes." .

[Amat20] J. Amat Rodrigo, "Árboles de Decisión, Random Forest, Gradient Boosting y C5.0." 2020.

[ChLi11] C. C. Chang and C. J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–39, 2011.

[FCHW08] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "LIBLINEAR: A Library for Large Linear Classification," *J. Mach. Learn. Res.*, vol. 9, no. 2008, pp. 1871–1874, 2008.

[Song20] D. Song, "¿Cómo Hidratar Un Conjunto de Tweets?" Jun. 2020.

[Summ20] E. Summers, "Hydrator [Computer Software]." 2020.

[BrVe18] M. A. Breddels and J. Veljanoski, "Vaex: Big Data Exploration in the Era of Gaia," *Astron. Astrophys.*, vol. 618, no. McKinney 2010, Jan. 2018.

[Akul21] R. Akula, "Interpretable Multi-Head Self-Attention Architecture For," 2021.

[KuBJ19] N. Kumaresh, V. Bonta, and N. Janardhan, "A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis," *Asian J. Comput. Sci. Technol.*, vol. 8, no. S2, pp. 1–6, 2019.

[KoTA00] O. Kolchyna, P. C. Treleaven, and T. Aste, "Twitter Sentiment Analysis: Lexicon Method, Machine Learning Method and Their Combination."

[Zarr14] A. A. A. A. I. Zarrad, "The Evaluation of Public Opinion," *Read. public Opin. Commun.*, pp. 664–670, 2014.

[LLSS20] M. O. Lwin *et al.*, "Global Sentiments Surrounding the COVID-19 Pandemic on Twitter: Analysis of Twitter Trends," *JMIR Public Heal. Surveill.*, vol. 6, no. 2, 2020.

[KaMa20] M. Y. Kabir and S. Madria, “CoronaVis: A Real-Time COVID-19 Tweets Data Analyzer and Data Repository,” 2020.

[ONMS20] O. Oyebode *et al.*, “COVID-19 Pandemic: Identifying Key Issues Using Social Media and Natural Language Processing,” *arXiv*, 2020.

[APSN20] R. Awasthi *et al.*, “CovidNLP: A Web Application for Distilling Systemic Implications of COVID-19 Pandemic with Natural Language Processing,” *medRxiv*, 2020.

[RaRR20] V. Raghupathi, J. Ren, and W. Raghupathi, “Studying Public Perception about Vaccination: A Sentiment Analysis of Tweets,” *Int. J. Environ. Res. Public Health*, vol. 17, no. 10, 2020.

[RKAR21] F. Rustam, M. Khalid, W. Aslam, V. Rupapara, A. Mehmood, and G. S. Choi, “A Performance Comparison of Supervised Machine Learning Models for Covid-19 Tweets Sentiment Analysis,” *PLoS One*, vol. 16, no. 2, pp. 1–23, 2021.

[CBBP20] K. Chakraborty, S. Bhatia, S. Bhattacharyya, J. Platos, R. Bag, and A. E. Hassanien, “Sentiment Analysis of COVID-19 Tweets by Deep Learning Classifiers—A Study to Show How Popularity Is Affecting Accuracy in Social Media,” *Appl. Soft Comput. J.*, vol. 97, p. 106754, 2020.

[MüSK20] M. Müller, M. Salathé, and P. E. Kummervold, “Covid-Twitter-Bert: A Natural Language Processing Model to Analyse Covid-19 Content on Twitter,” *arXiv*, 2020.

[Deve20] S.-L. Developers, “Classification Metrics.” 2020.

[Lams20] R. Lamsal, “Design and Analysis of a Large-Scale COVID-19 Tweets Dataset,” *Appl. Intell.*, no. November 2020, pp. 2790–2804, 2020.

[Roha20] P. Rohan, “Term Frequency and Inverse Document Frequency.” 2020.

[Orei21] OREILLY, “Chapter 1. Gaining Early Insights from Textual Data".2020.

[Sye21] S. A. R. Syed Alwi, E. Rafidah, A. Zurraini, O. Juslina, I. B. Brohi and S. Lukas, “A survey on COVID-19 vaccine acceptance and concern among Malaysians", *BMC Public Health,* 2021.

Anexos

A. WordEmbbeding

La premisa de estos modelos [Rodr19] es que una palabra puede ser determinada por otras palabras con las que aparezca acompañada. Puesto que estos vectores representan coordenadas en un espacio vectorial dado, las palabras cercanas o similares pueden calcularse en función de la distancia entre sus vectores. Por ejemplo, la Figura A.1 muestra como cuanto más cerca estén dos vectores de palabras o *word vectors*, más similares serán en términos semánticos que si están más alejados. En este caso, en la Figura A.1 aparecen representados algunos vectores palabra y sus relaciones previamente calculadas con GloVe [PeSM14], el cual es capaz de capturar los conceptos '*men* es a '*woman* como '*king*' es a '*queen*' y relaciones entre verbos y tiempos verbales. El cálculo de la similitud o cercanía entre vectores puede determinarse de diferentes maneras siendo algunas de las más utilizadas la distancia *euclidea* o la similitud del coseno.

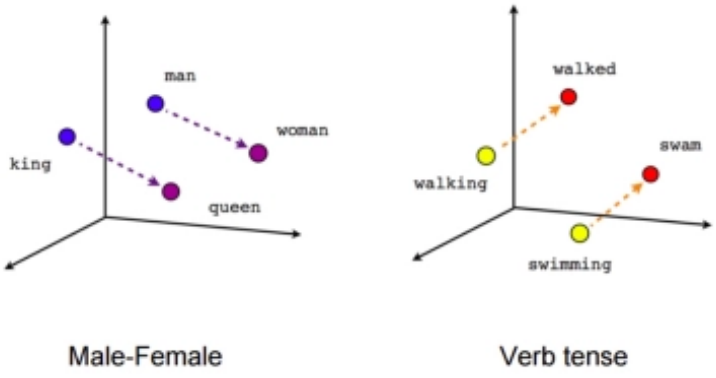


Figura A.1: Visualización del concepto de los *Word Embbedings* capturando información relativa al género (izq.) y tiempos verbales (der.) [26]

Los *Word Embbedings* tiene varias implementaciones diferentes, como BERT de Google [DCLT19], Word2Vec [MCCD13], GloVe [PeSM14] de Stanford, FastText [JGBD16] de Facebook, etc.

Sin embargo, el modelo BERT de Google genera incrustaciones empleando *Transfomers* [WDSC19] permitiendo tener múltiples (más de una) representaciones vectoriales (numéricas) para la misma palabra, en función del contexto en el que se utiliza la palabra. Por lo tanto, las incrustaciones de BERT dependen del contexto.

B. Método Chi-Cuadrado (χ2)

Dados los datos de dos variables, podemos obtener el recuento observado *O* y el recuento esperado *E*. El chi-cuadrado mide cómo se desvían el recuento esperado *E* y el recuento observado *O*.

$$X^2 = n \sum_{i=1}^{k} \sum_{j=1}^{l} \frac{(f_{ij} - f_i f_j)^2}{f_i f_j}$$

donde:

f_{ij}= frecuencia real en la i-ésima fila, j-ésima columna

$f_i f_j$ = frecuencia esperada en la i-ésima fila, j-ésima columna

k = número de filas

l = número de columnas

En la selección de características, el objetivo fue seleccionar las variables que son altamente dependientes de la variableobjetivo.

Cuando dos características son independientes, el recuento observado se acerca al recuento esperado, por lo que el valor de Chi-Cuadrado será menor. El grado de independencia depende del umbral de significación seleccionado y el valor de dicho estadístico.

Pasos en la prueba Chi-Cuadrado:

- Definir la hipótesis.
- Construir una tabla de contingencia.
- Encontrar los valores esperados.
- Calcular el estadístico Chi-Cuadrado.
- Mantener o rechazar la hipótesis nula.

Definimos la hipótesis

- **Hipótesis nula (H0):** Dos variables son independientes
- **Hipótesis alternativa (H1):** Dos variables no son independientes.

Grados de libertad

Los grados de libertad se refiere al número máximo de valores lógicamente independientes, que tienen la libertad de variar. Puede definirse como el número total de observaciones menos el número de restricciones independientes impuestas a las observaciones.

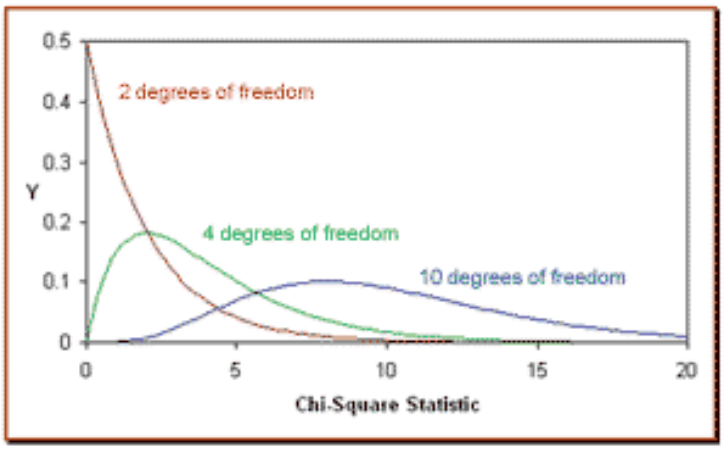


Figura B.1: Estadístico Chi-Cuadrado[27]

En la Figura B.1 se observa como la distribución de Chi-Cuadrado para diferentes grados de libertad. También podemos observar que a medida que aumentan los grados de libertad la distribución Chi-Cuadrado se aproxima a la distribución normal.

Construimos la tabla de contingencia

Esta tabla muestra la distribución de una de las variables a estudiar en filas y de la otra en columnas. Se utiliza para estudiar la relación entre dos variables.

Los grados de libertad para la tabla de contingencia se dan como (r-1) * (c-1) donde r, c son filas y columnas.

Encontrar el valor esperado

Partiendo de la hipótesis nula de que las dos variables son independientes. Se puede decir decir que si A, B son dos eventos independientes:

$$P(AÇB) = P(A) * P(B)$$

Mantener o rechazar la hipótesis nula

Con un 95% de confianza, es decir, alfa = 0,05, el valor de Chi-Cuadrado calculado cae en la región de aceptación o rechazo.

Los valores de Chi-Cuadrado se pueden determinar con la tabla de Chi-Cuadrado.

Rechazaremos la hipótesis nula si el valor de Chi-Cuadrado cae en la región de error (alfa de 0 a 0,05).

C. Eliminación de Características Recursivas con Validación Cruzada

Para configurar el método, es necesario crear previamente una instancia de uno de los algoritmos de ML. Posteriormente, se configuraron los siguientes parámetros:

- **Estimador**: algoritmos de ML creado previamente.
- **Step**: número de características a eliminar en cada iteración.
- **CV**: número de *k-fold* para la validación cruzada.
- **Scoring**: métrica de puntuación.

Una vez finalizado el proceso, podemos graficar (Figura C.1) el número de características que va seleccionadas en cada paso y su puntuación.

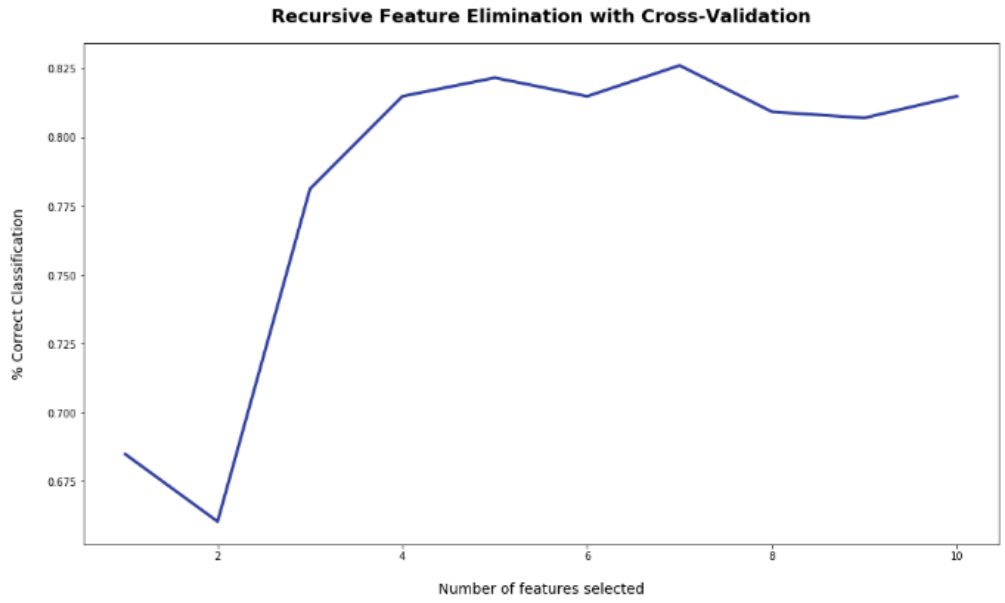


Figura C.1: Resultado del RFECV

D. Análisis de Componentes Principales (PCA)

Normalmente el algoritmo PCA se emplea para reducir las dimensiones del espacio de características original (aunque PCA tiene más aplicaciones). Las nuevas variables se han ordenado en función de "su capacidad para retener la máxima información posible"**.**

Para la selección del número de variables finales podemos seguir 3 métodos:

Método 1: Se seleccionan al azar "las primeras n dimensiones" (las más importantes o la que mayor ratio de varianza explicada tienen). Por ejemplo, si el objetivo es poder graficar en 2 dimensiones, se podría tomar las 2 características nuevas y usarlas como los ejes X e Y.

Método 2: Determinar la proporción *de* varianza explicada para cada característica extraer las dimensiones hasta alcanzar un mínimo que nos propongamos, por ejemplo, hasta llegar a explicar al 90% de la variabilidad total explicada.

Método 3: Crear una gráfica (Figura D.1) especial llamada *scree plot* -a partir del Método 2- y seleccionar cuántas dimensiones se utilizaron, por el método "del codo", en el que identificaremos visualmente el punto en el que se produce una caída significativa de la varianza explicada en relación con la característica anterior.

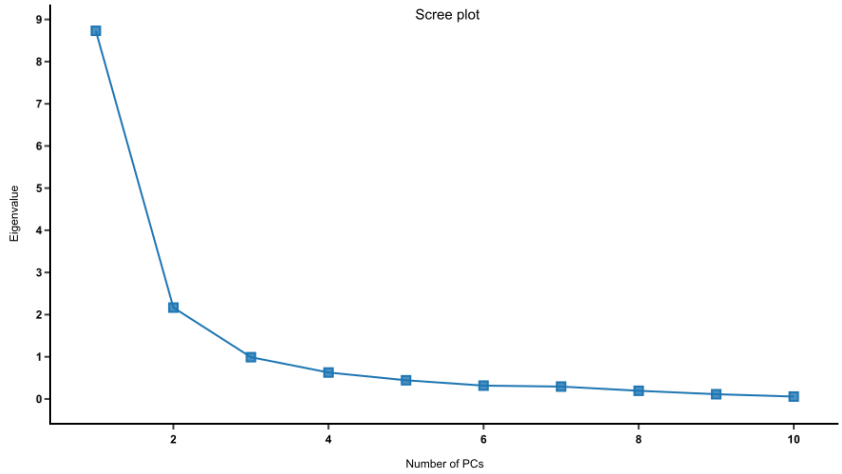


Figura D.1: *Scree plot*

Suponiendo que las variables de entrada estén estandarizadas como la matriz Z y Z^T su transpuesta, cuando creamos la matriz de covarianza Z^TZ, los elementos de dicha matriz miden cómo cada variable de Z se relaciona con cada otra variable de Z.

Los *autovectores* representan dirección. Los autovalores representan magnitud.

Por último, se asume que cuanto mayor sea la variabilidad en una dirección determinad, mayor será la correlación con la variable dependiente. Mucha variabilidad usualmente indica "más información" mientras que poca variabilidad indica "ruido".

E. Otras métricas para evaluación

En un modelo multiclase, podemos dibujar *N* números de curvas AUC ROC para *N* clases utilizando la metodología de uno contra todos (*one* vs *all*). Así, por ejemplo, si existen tres clases denominadas *X*, *Y* y *Z*, tendremos un ROC para *X* clasificado contra *Y* y *Z*, otro ROC para *Y* clasificado contra *X* y *Z*, y el tercero de *Z* clasificado contra *X e Y*. A continuación se

definen de forma matemática los términos necesarios usados para el cálculo de la curva ROC-AUC:

- TPR/Recall/Sensibilidad = $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- Specifity/Especifidad = $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$
- FPR/Tasa de Falsos Positivos = $\mathrm{FPR} = 1 - Specifity = \frac{\mathrm{FP}}{\mathrm{TN+FP}}$

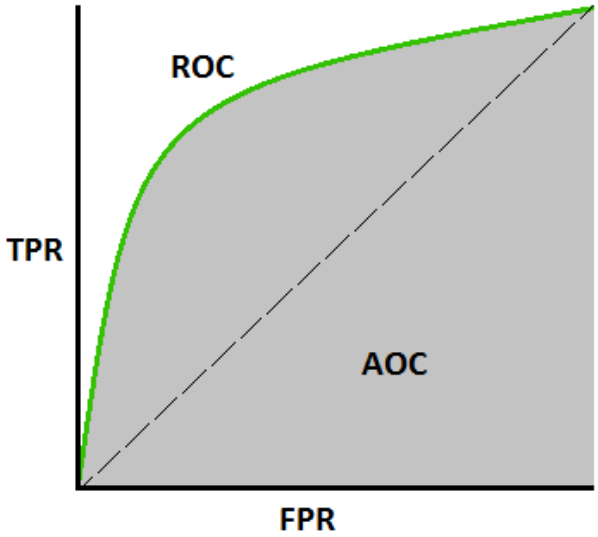


Figura E.1: Curva ROC-AUC[28]

- **Matriz de Confusión**: La matriz de confusión (Figura E.2) evalúa la precisión de la clasificación calculando los valores de la matriz por cada fila correspondiente a la clase verdadera. Los elementos diagonales representan el número de observaciones para los que la etiqueta predicha es igual a la verdadera, mientras que los elementos no diagonales son los que el clasificador ha etiquetado erróneamente. Cuanto más altos sean los valores diagonales de la matriz de confusión mayor será el número de predicciones correctas.

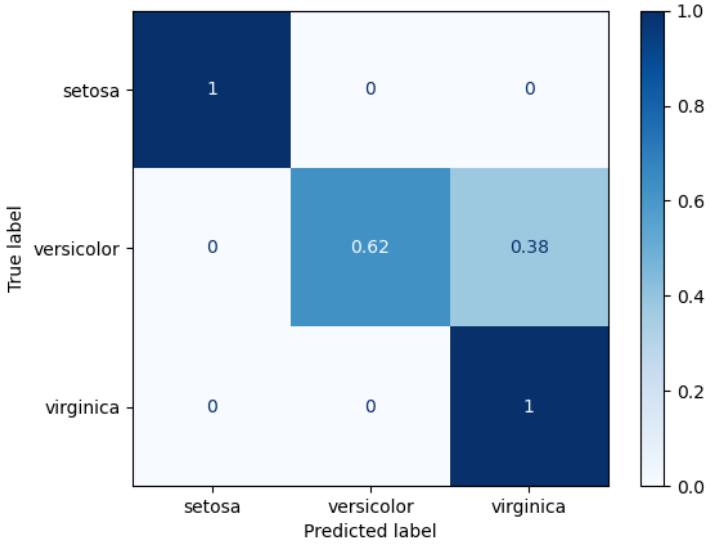


Figura E.2: Matriz de Confusión Multiclase

- ***Accuracy***: Número de elementos correctamente clasificados como verdaderamente positivos (TP) o verdaderamente negativos (TN) del número total de elementos.

$$\textbf{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Finalmente, también nos fijaremos en las siguientes dos métricas:

- ***Precision*** = $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
- **F1-Score** = $\frac{(2*\mathrm{Precision}*\mathrm{Recall})}{(Precision+Recall)}$

F. Análisis de hashtags

Por último, se realizó un análisis de *hashstags* más frecuentes y hashtags más frecuentes asociados a sentimientos negativos y positivos sobre la vacuna. Este análisis tiene como objetivo ver si es posible extraer cierto conocimiento sobre la percepción de la gente sobre la pandemia a nivel global, no solo sobre las vacunas. Como el conjunto de datos de este apartado se ha obtenido tras realizar el filtrado de la sección 5.2, se espera que las palabras de los *hashtags* más frecuentes contengan la palabra vacuna (*vaccine)*. En la Figura F.1, vemos como seis de los 15 hashtags más frecuentes están relacionados con la vacuna. Cabe destacar la presencia de las dos principales vacunas existentes en el mes de noviembre de 2020, como Pfizer y Moderna.

En la Figura F.2, se muestran los *hashtags* más frecuentes asociados a sentimientos positivos. Se observan *hashtags* como "*patents*" o "vaccineswork" asociados a sentimientos positivos.

En la Figura F.3 se muestran los *hashtags* más frecuentes asociados a sentimientos negativos. Vemos como aparecen *hashtags* como por ejemplo "trumpvirus" o "trump" asociados a sentimientos negativos.

Finalmente, se observa como el número de *hashtags* relacionados con la vacuna es casi el doble en la Figura F.2 que en la Figura F.3, lo cual indica que percepción positiva hacia la vacuna es mayor que la negativa.

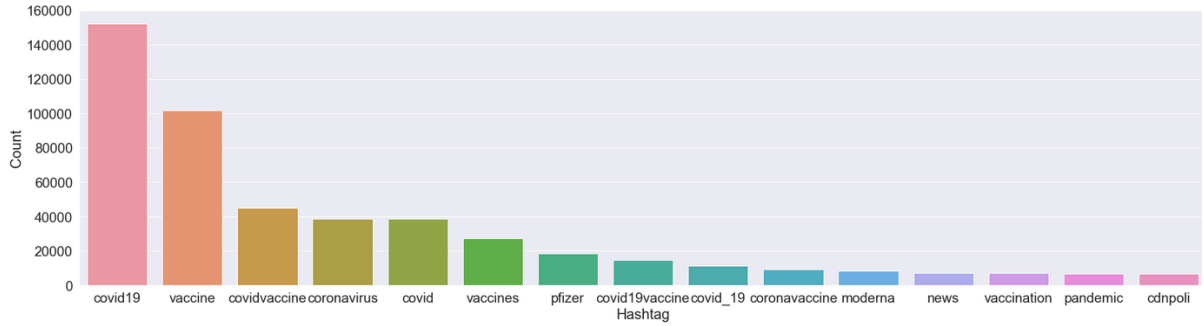


Figura F.1: Hashtags más frecuentes (Top-15)

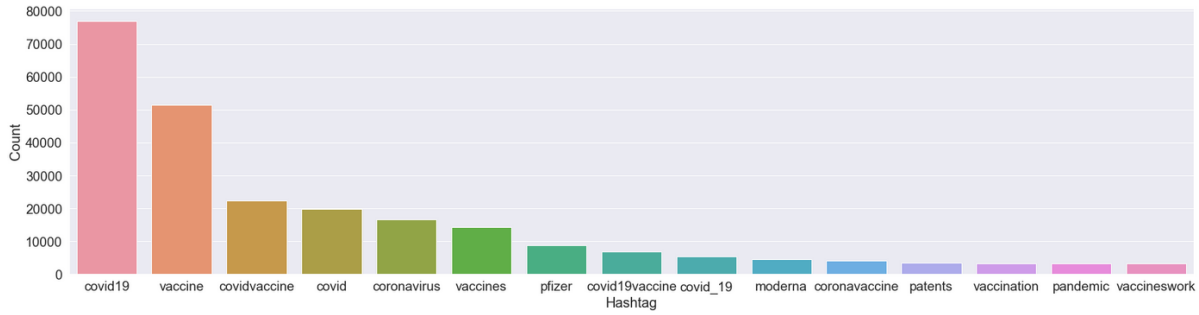


Figura F.2: Hashtags asociados a sentimientos positivos más frecuentes (Top-15)

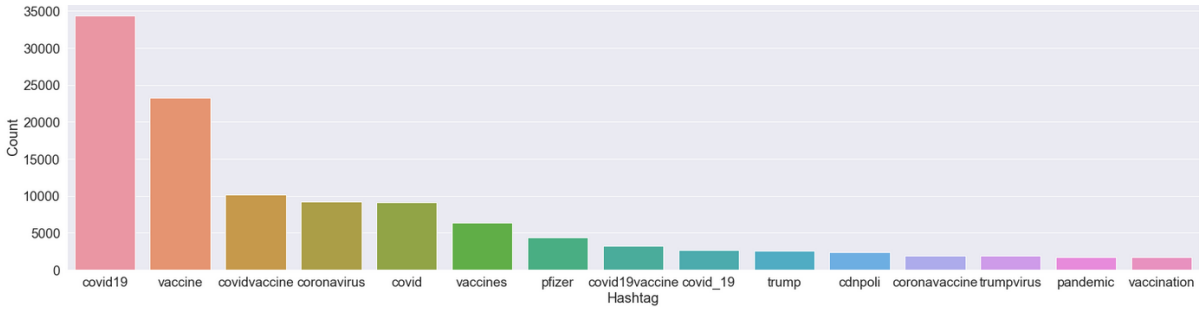


Figura F.3: Hashtags asociados a sentimientos negativos más frecuentes (Top-15)

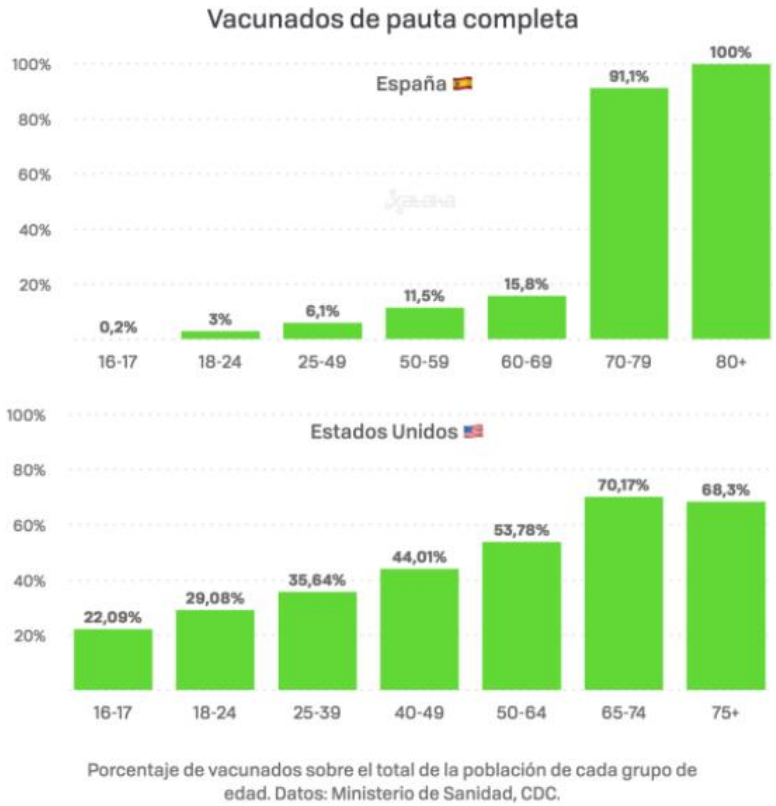


Figura F.4: Comparación del ritmo de vacunación entre EE. UU y España.

G. Feature engineering

Con el objetivo de crear nuevas variables que puedan ayudar a predecir los sentimientos de los *tweets*, en la Figura H.14 del Anexo se muestra la matriz triangular *inferior* de correlación (correlación de Pearson) entre nuevas variables, como por ejemplo longitud del *tweet*, número de mayúsculas, numero de caracteres etc., y la variable objetivo, También, se han añadido transformaciones en la variable objetivo, siendo *compound* la variable objetivo original, *target* la variable transformada con codificación ordinal (negativo: 0, neutro:1 y positivo:2) y tres nuevas variables fruto de la codificación *one-hot* (crear una nueva variable binaria (también llamada *dummy*) por cada categoría existente en la variable a codificar). Se observa como desafortunadamente no existe una correlación, ni negativa ni positiva, entre las nuevas variables y las diferentes transformaciones de la variable objetivo. Por ejemplo, cuanta menor sea el número de palabras (*num_words*) menor es la correlación con la variable "neutro" y por el contrario cuanto mayor sea el número de palabras del *tweet*, mayor es la correlación con la variable "positivo" y "negativo". Lo mismo pasa con las variables "num_unique_words" (número de palabras únicas del *tweet)*, "sentence_count" (número de frases en el *tweet*), "char_count" (número de caracteres del *tweet*), y "lenght" (longitud del *tweet*).

H. Tablas y Figuras

```
Wall time: 57.2 s
```

	tweet_id	sentiment_score
0	1333628333574254595	0.15
1	1333628333834330112	-0.0625
2	1333628333968420869	0
3	1333628334035517441	0.17045454545454544
4	1333628334345908225	0
...	...	...
97841739	1333628206683975681	0
97841740	1333628206549594112	0.1619047619047619
97841741	1333628206893723648	0.65
97841742	1333628207191400452	0.36363636363636365
97841743	1333628207065489413	-0.024999999999999994

Figura H.1: Conjunto de datos original sin "hidratar" cargado con Pandas.

```
Wall time: 3.99 ms
```

#	tweet_id	sentiment_score
0	1333628333574254595	0.15
1	1333628333834330112	-0.0625
2	1333628333968420869	0.0
3	1333628334035517441	0.17045454545454544
4	1333628334345908225	0.0
...	...	...
97,841,739	1333628206683975681	0.0
97,841,740	1333628206549594112	0.1619047619047619
97,841,741	1333628206893723648	0.65
97,841,742	1333628207191400452	0.3636363636363637
97,841,743	1333628207065489413	-0.024999999999999998

Figura H.2: Conjunto de datos original sin "hidratar" cargado con la librería VAEX.

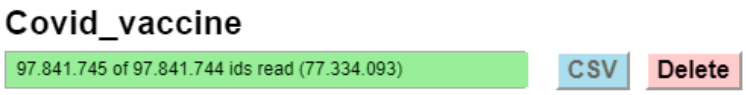


Figura H.3: Conjunto final de *tweets* "hidratados" mediante *Hydrator App*.

Tabla H.1: Campos “hidratados”.

Campo (Tipo)	Descripción
Created_at (String)	Hora y fecha en formato UTC en que se creó este *tweet.*
Id_str (String)	La representación en forma de cadena de texto del identificador único para este usuario.
Text (String)	El texto UTF-8 real del *tweet*.
User_screen_name (String)	El alias con el que se identifica este usuario. Los *screen_names* son únicos pero están sujetos a cambios. Es mejor utilizar el campo *id_str* como identificador de usuario siempre que sea posible. Normalmente tiene un máximo de 15 caracteres, pero pueden existir cuentas históricas con nombres más largos.
User_location (String)	La ubicación definida por el usuario para el perfil de su cuenta. No es necesariamente una ubicación, ni es analizable por la máquina. Este campo podría ser interpretado ocasionalmente de forma imprecisa por el servicio de búsqueda.
User_followers_count (Int)	El número de seguidores que tiene actualmente el usuario. En determinadas condiciones de restricción, este campo indicará temporalmente "0".
User_favouries_count (int)	El número de favores que tiene actualmente el usuario.
Lang (String)	Cuando está presente, indica un identificador de idioma correspondiente al idioma detectado por la máquina del texto del *tweet*.
User_verified (Boolean)	Cuando es verdadero, indica que el usuario tiene una cuenta verificada.
Coordinates_coordinates_0 (Float)	La longitud de la ubicación del *tweet*. *
Coordinates_coordinates_1 (Float)	La latitud de la ubicación del *tweet*. *
Place_country_code (String)	Código de país abreviado *.
Place_country (String)	Nombre del país *.
Place_name (String)	Representación breve legible del nombre del lugar (normalmente la ciudad/pueblo/condado, etc.). *
Source (String)	Aplicación utilizada para publicar el *tweet*, como una cadena con formato HTML. Los *tweets* del sitio web de *Twitter* tienen un valor de origen de web.
Retweeted_status_id_str (String)	Los usuarios pueden amplificar la transmisión de *tweets* creados por otros usuarios mediante el *Retweeting*. Los *retweets* se pueden distinguir de los *tweets* típicos por la existencia del atributo retweeted_status. Este atributo contiene el “id_str” del *tweet* original que se “retuiteó”. Hay que tener en cuenta que los *retweets* de

	retweets no muestran representaciones del *retweet* intermediario, sino solo el t*weet* original. (Los usuarios también pueden cancelar un *retweet* que crearon eliminando su *retweet*).

* Cuando se utiliza la API de Twitter para publicar un *tweet*, se puede adjuntar un *Twitter Place* especificando un *place_id* al publicar el *tweet*. Los *tweet*s asociados a "lugares" no se emiten necesariamente desde ese lugar, pero también podrían ser potencialmente sobre ese lugar.

Tabla H.2: Resumen estadístico de las variables (I).

	Created_at	**Id_str**	**Text**	**Screen_name**	**Location**	**Followers and favourites**
Count	77.332.093	77.332.093	77.332.093	77.334.082	52.106.859	77.332.093
Missings	0	0	0	11	25.227.234	0

Tabla H.3: Resumen estadístico de las variables (II).

	Verified	**Coordinates_0 y Coordinates_1**	**Place_name**	**Place_country_ code**	**Lang**	**Source**
Count	77.332.093	17.406	807.160	806.123	77.334.093	77.1966.72
Missings	0	77.316.687	76.526.933	76.527.970	0	137.421

Tabla H.4: Resumen estadístico de las variables del *corpus* final (I).

	Created_at	**Id_str**	**Text**	**Screen_name**	**Location**	**Followers and favourites**
Count	10.523.999	10.523.999	10.523.999	10.523.999	7.029.451	10.523.999
Missings	0	0	0	0	3.494.548	0

Tabla H.5: Resumen estadístico de las variables del corpus final (II).

	Verified	**Coordinates_0 and Coordinates_1**	**Place_name**	**Place_country_code**	**Lang**	**Source**
Count	10.523.999	1144	86.307	86.202	10.523.999	10.512.756
Missings	0	10.522.855	10.437.692	10.437.797	0	11.243

Tabla H.6: Resumen descriptivo conjunto final (I).

	Created_at	**Id_str**	**Text**	**Screen_name**	**Location**	**Followers and favourites**
Count	4026416	4026416	4026416	4026416	2735125	4026416
Missings	0	0	0	0	1291291	0

Tabla H.7: Resumen descriptivo conjunto final (II).

	Coordinates_0 and Coordinates_1	Place_name	Place_country	Place_country_ code	Verified	Source
Count	1144	4998	86197	86175	4026416	4022718
Missings	4025272	4021418	3940219	3940241	0	3698

Tabla H.8: Resumen descriptivo conjunto final (III).

	Compound (VADER)	Sentiment_score (TextBlob)
Count	4026416	4026416
Missings	0	0
Mean	0.0627	0.0821
Std	0.47312	0.251952
Min	-0.9962	-1.0
Max	0.997	1.0

Tabla H.9: Resumen estadístico del *DataFrame* agrupado por usuarios (I).

	User_screen_name	Tweets	Sentiment_score
Count	1.457.046	1.457.046	1.457.046
Missings	0	0	0
Mean	-	2.79838	0.0820
Std	-	10.795691	0.23
Min	-	1	-1.0
Max	-	3208	1.0

Tabla H.10: Resumen estadístico agrupado por usuarios (II).

	User_screen_name	Tweets	Sentiment_score	Mean_tweets_by_day
Count	550.544	550.544	550.544	550.544
Missings	0	0	0	0
Mean	-	5.7595	0.0813	0.17998
Std	-	17.1567	0.1609	0.53614
Min	-	2	-1.0	0.0625
Max	-	3208	1.0	100.25

Tabla H.11:Resumen estadístico agrupado por usuarios (III).

	User_screen_name	Tweets	Sentiment_score	Mean_tweets_by_day
Usuarios	6984	6984	6984	6984
Missings	0	0	0	0
Mean	-	94.13044	0.0829	2.941576
Std	-	115.9147	0.0765	3.622334
Min	-	41	-0.7374	1.28125
Max	-	3208	0.9503	100.25

text	processed_text_vader	compound	sentiment_score
'Dr. Slaoui: Operation Warp Speed will meet coro...	'Dr. Slaoui: Operation Warp Speed will meet coro...	0	0
'This white woman in the zoom meeting just said ...	'This white woman in the zoom meeting just said ...	-0.4019	0
'@cjtruth Why was there a few clips showing POTU...	'Why was there few clips showing POTUS affirming...	0	-0.225
'The vaccine was first approved by the U.S. Food...	'The vaccine was first approved by the U.S. Food...	0.4215	0.25
'@realDonaldTrump Thank you for your great Leade...	'Thank you for your great Leadership on moving t...	0.9217	0.8
'@drninashapiro There are plenty of vaccines in ...	'There are plenty of vaccines in the pipeline th...	0.4404	0.2
'@CNN really? Nonstop coverage of this vaccine m...	'really? Nonstop coverage of this vaccine moving...	-0.1027	-0.0833333
'We have a lot of evidence around current vaccin...	'We have lot of evidence around current vaccines...	-0.5093	0
'The State team has worked closely with our part...	'The State team has worked closely with our part...	0.4939	0.25
'@AheadoftheNews State and local government aid,...	'State and local government aid, REGARDLESS OF S...	-0.6249	-0.125

Figura H.4: Preprocesamiento 1.

text	processed_text_vader	compound	sentiment_score
'Dr. Slaoui: Operation Warp Speed will meet coro...	'Dr Slaoui Operation Warp Speed will meet corona...	0	0
'This white woman in the zoom meeting just said ...	'This white woman in the zoom meeting just said ...	-0.4019	0
'@cjtruth Why was there a few clips showing POTU...	'Why was there few clips showing POTUS affirming...	0	-0.225
'The vaccine was first approved by the U.S. Food...	'The vaccine was first approved by the US Food a...	0.4215	0.25
@realDonaldTrump Thank you for your great Leade...	'Thank you for your great Leadership on moving t...	0.9217	0.8
'@drninashapiro There are plenty of vaccines in ...	'There are plenty of vaccines in the pipeline th...	0.4404	0.2
'@CNN really? Nonstop coverage of this vaccine m...	'really Nonstop coverage of this vaccine moving ...	-0.1027	-0.0833333
'We have a lot of evidence around current vaccin...	'We have lot of evidence around current vaccines...	-0.1511	0
'The State team has worked closely with our part...	'The State team has worked closely with our part...	0.4939	0.25
'@AheadoftheNews State and local government aid,...	'State and local government aid REGARDLESS OF ST...	-0.6249	-0.125

Figura H.5: Preprocesamiento 2.

text	processed_text_vader	compound	sentiment_score
'Dr. Slaoui: Operation Warp Speed will meet coro...	'Dr. Slaoui: Operation Warp Speed will meet coro...	0	0
'This white woman in the zoom meeting just said ...	'This white woman in the zoom meeting just said ...	-0.4019	0
'@cjtruth Why was there a few clips showing POTU...	'@cjtruth Why was there a few clips showing POTU...	0	-0.225
'The vaccine was first approved by the U.S. Food...	'The vaccine was first approved by the U.S. Food...	0.4215	0.25
'@realDonaldTrump Thank you for your great Leade...	'@realDonaldTrump Thank you for your great Leade...	0.9217	0.8
'@drninashapiro There are plenty of vaccines in ...	'@drninashapiro There are plenty of vaccines in ...	0.4404	0.2
'@CNN really? Nonstop coverage of this vaccine m...	'@CNN really? Nonstop coverage of this vaccine m...	-0.1027	-0.0833333
'We have a lot of evidence around current vaccin...	'We have a lot of evidence around current vaccin...	-0.5093	0
'The State team has worked closely with our part...	'The State team has worked closely with our part...	0.4939	0.25
'@AheadoftheNews State and local government aid,...	'@AheadoftheNews State and local government aid,...	-0.6249	-0.125

Figura H.6: Preprocesamiento 3.

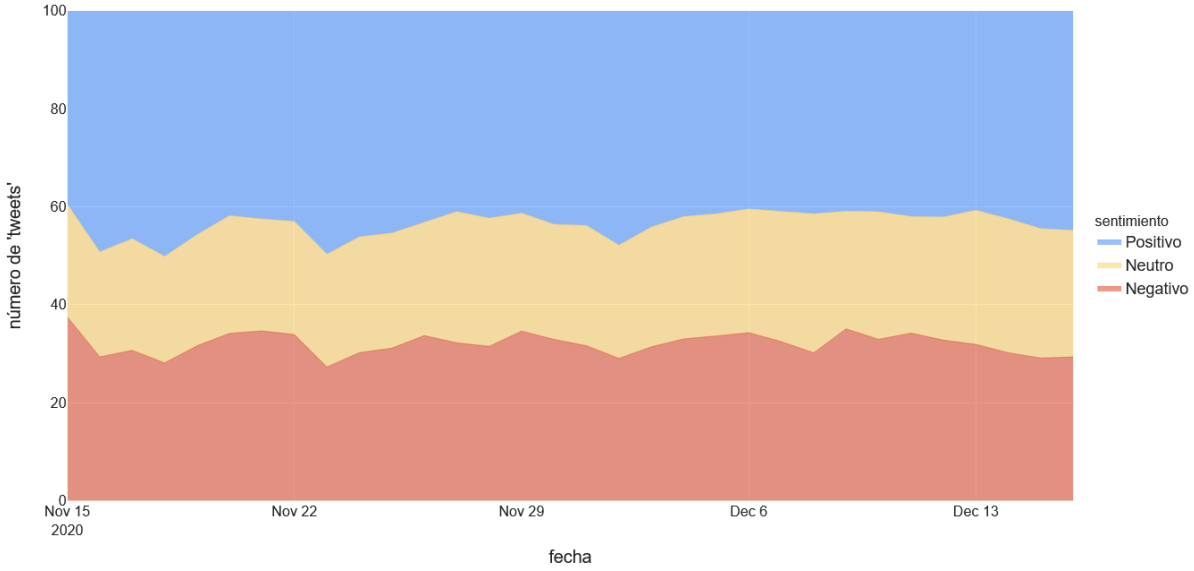


Figura H.7: Distribución de sentimientos por fecha (III).

Tabla H.12: Resumen estadístico: Top-3 días con mayor tráfico de *tweets*.

Fecha	Tweets	Tweets negativos	Tweets neutros	Tweets positivos	Negativos	Neutros	Positivos
14-12-2020	237.226	71.685	64.841	100.700	30.218 %	27.333 %	42.449 %
08-12-2020	230.989	69.796	65.451	95.742	30.216 %	28.335 %	41.4487 %
02-12-2020	235.014	68.317	54.249	112.448	29.069 %	23.083%	47.847 %

Tabla H.13: Fechas y porcentaje de *tweets* positivos y negativos ordenados de mayor a menor (Top-5).

Positivos		Negativos	
Fecha	Porcentaje	Fecha	Porcentaje
18-11-2020	50.1211	15-11-2020	37.5211
23-11-2020	49.6718	09-12-2020	35.0932
16-11-2020	49.206	21-11-2020	34.6646
02-12-2020	47.8474	29-11-2020	34.6269
17-11-2020	46.5149	06-12-2020	34.3140

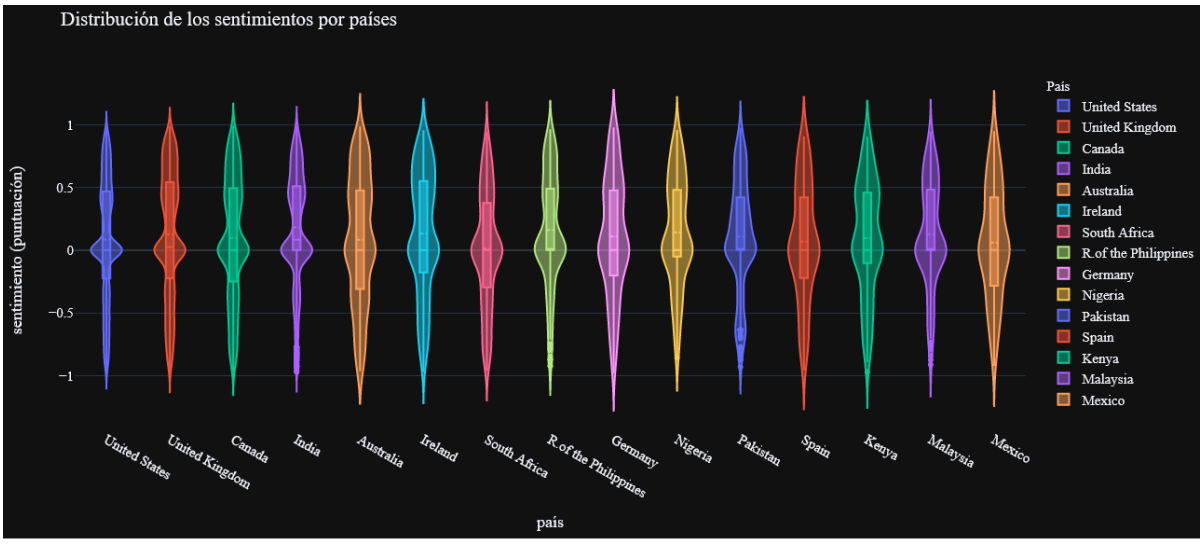


Figura H.8: Distribución de los sentimientos por países con *Plotly*

Tabla H.14: Resumen estadístico del sentimiento por estado

Abreviatura	Estado	Tweets	Compound	Negativos	Neutros	Positivos
AK	Alaska	1165	0.050	388	282	495
AL	Alabama	2709	0.027	911	713	1085
AR	Arkansas	1466	0.037	510	333	623
AZ	Arizona	6672	0.038	2357	1466	2849
CA	California	28734	0.045	9685	6528	12521
CO	Colorado	8555	0.008	3160	1916	3479
CT	Connecticut	4328	0.087	1263	1117	1948
DC	District of Columbia	530	0.106050	159	113	258
DE	Delaware	1454	0.054522	462	350	642
FL	Florida	21198	0.032590	7345	5060	8793
GA	Georgia	6169	0.038718	2050	1508	2611
HI	Hawaii	1691	0.075233	539	396	756
IA	Iowa	2643	0.083577	851	582	1210
ID	Idaho	1088	0.023467	398	255	435
IL	Illinois	5377	0.033085	1867	1330	2180
IN	Indiana	7734	0.070844	2395	1946	3393
KS	Kansas	5898	0.095548	1706	1510	2682
KY	Kentucky	3126	0.032152	1102	720	1304
LA	Louisiana	2103	0.031780	722	530	851
MA	Massachusetts	6547	0.065182	2163	1430	2954
MD	Maryland	6505	0.070309	2098	1506	2901

ME	Maine	2320	0.062766	776	516	1028
MI	Michigan	9202	0.040098	3148	2178	3876
MN	Minnesota	5516	0.049229	1882	1236	2398
MO	Missouri	3003	0.044453	1045	635	1323
MS	Mississippi	1364	0.058800	420	348	596
MT	Montana	1332	0.039353	450	322	560
NC	North Carolina	7151	0.039854	2470	1610	3071
ND	North Dakota	324	0.017709	113	76	135
NE	Nebraska	1052	0.034869	374	220	458
NH	New Hampshire	1722	0.071941	532	421	769
NJ	New Jersey	10230	0.054557	3392	2344	4494
NM	New Mexico	1724	0.033422	618	372	734
NV	Nevada	1292	0.053409	441	292	559
NY	New York	62070	0.095069	17420	16490	28160
OH	Ohio	9372	0.027885	3372	2074	3926
OK	Oklahoma	3887	0.072960	1166	1044	1677
OR	Oregon	5273	0.036509	1828	1239	2206
PA	Pennsylvania	7575	0.039553	2633	1666	3276
RI	Rhode Island	1246	0.046528	435	291	520
SC	South Carolina	3009	0.034918	1059	684	1266
SD	South Dakota	563	0.106299	158	129	276
TN	Tennessee	3747	0.028654	1377	787	1583
TX	Texas	22498	0.044461	7459	5618	9421
UT	Utah	2178	0.119533	622	478	1078
VA	Virginia	7856	0.081891	2447	1816	3593
VT	Vermont	1400	0.081043	434	327	639
WA	Washington	40214	0.126290	10300	10640	19274
WI	Wisconsin	4205	0.064662	1341	1000	1864
WV	West Virginia	1090	0.078111	336	256	498
WY	Wyoming	540	0.035394	202	110	228

Tabla H.15: Resumen estadístico de los *tweets* de EE. UU agrupados por sentimientos

	Nº de *tweets*	**Porcentaje**
Negativo	1828	29.77%
Neutro	1515	24.67%

Positivo	2798	45.56%

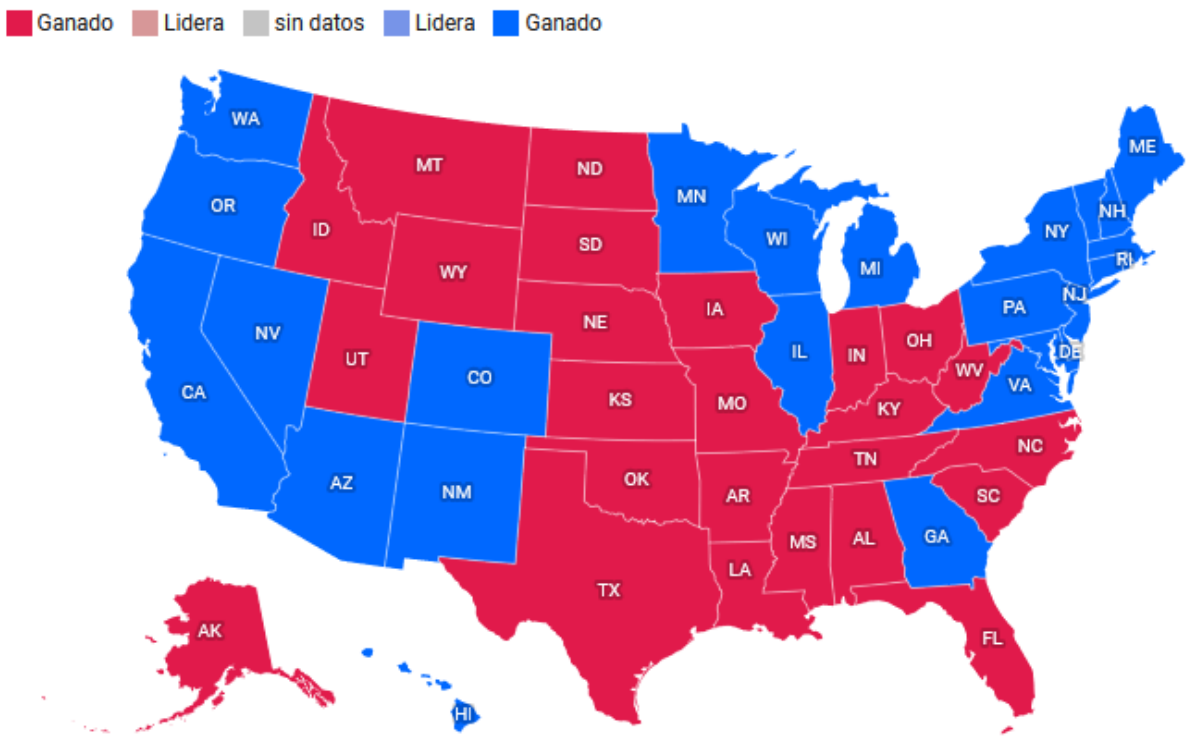


Figura H.9: Resultados de las elecciones presidenciales de EEUU del 3 de noviembre de 2020

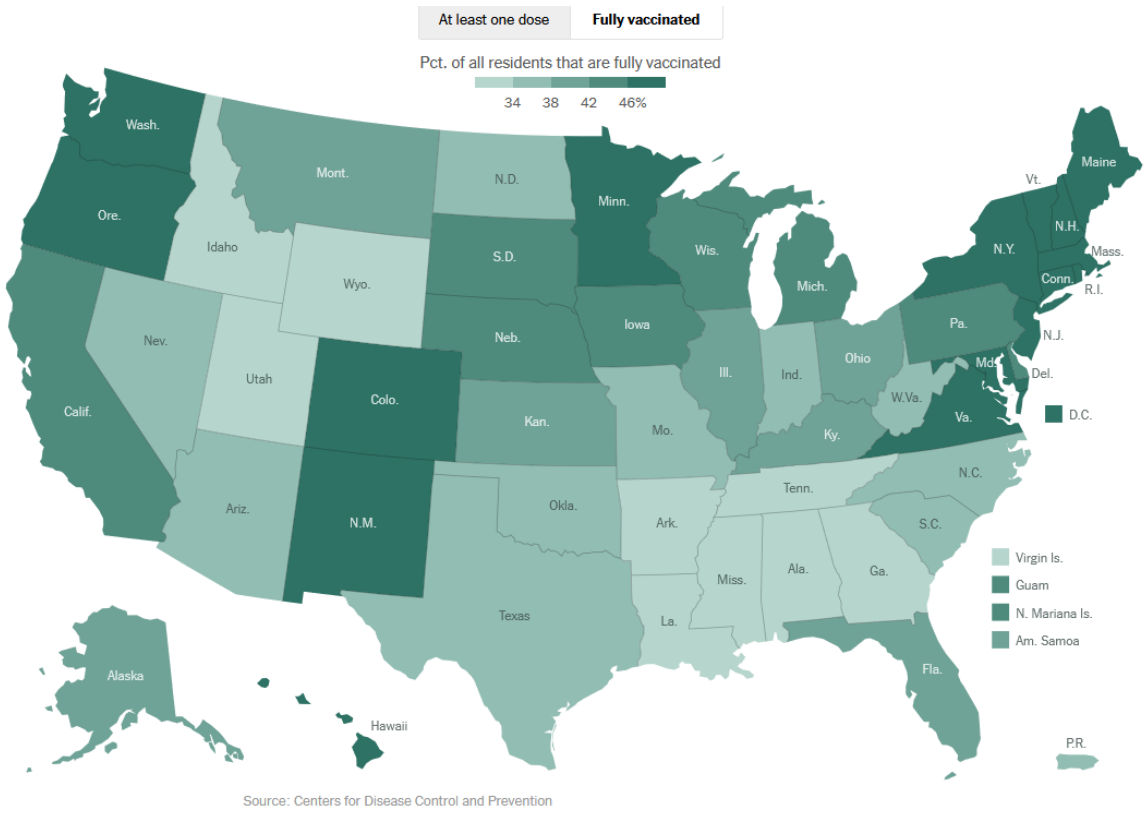


Figura H.10: Ritmo de vacunación (pauta completa) en EE. UU a 7 de junio 2021[29].

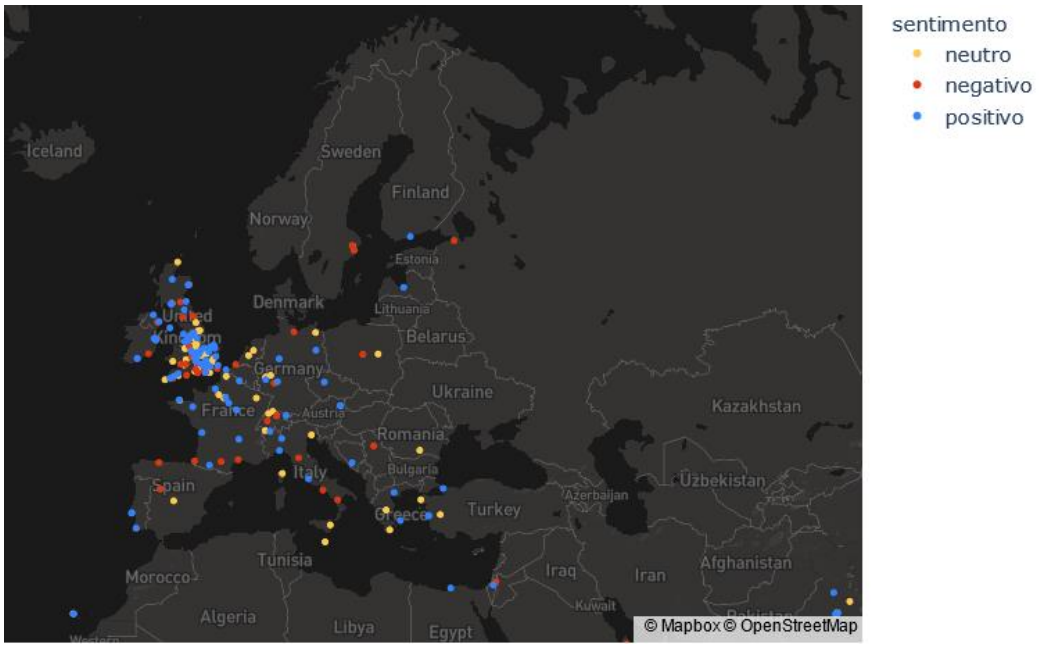


Figura H.11: Mapa de sentimientos de Europa

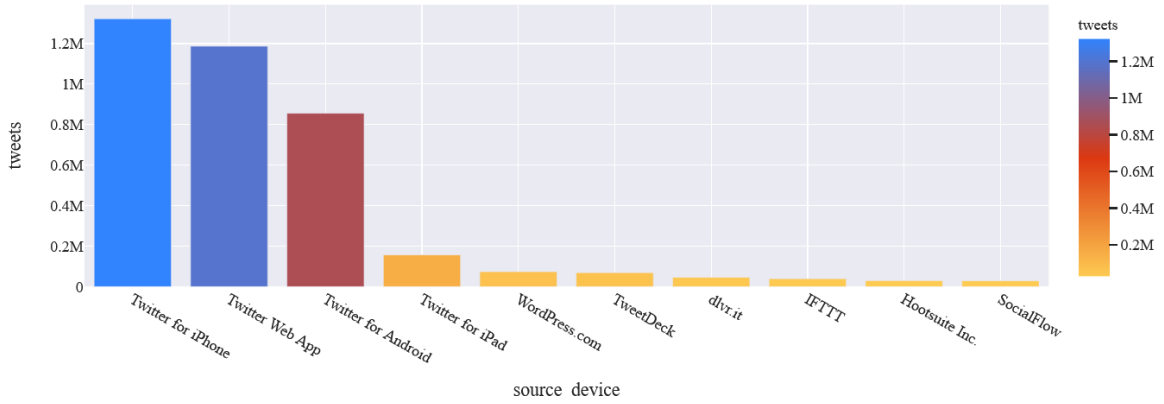


Figura H.12: Clientes de Twitter más usados

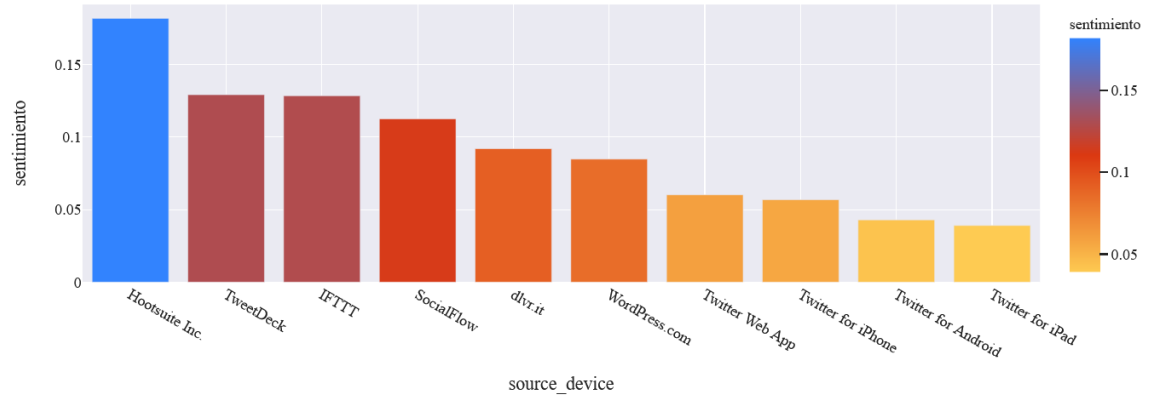


Figura H.13: Sentimientos de los usuarios agrupados por clientes de Twitter

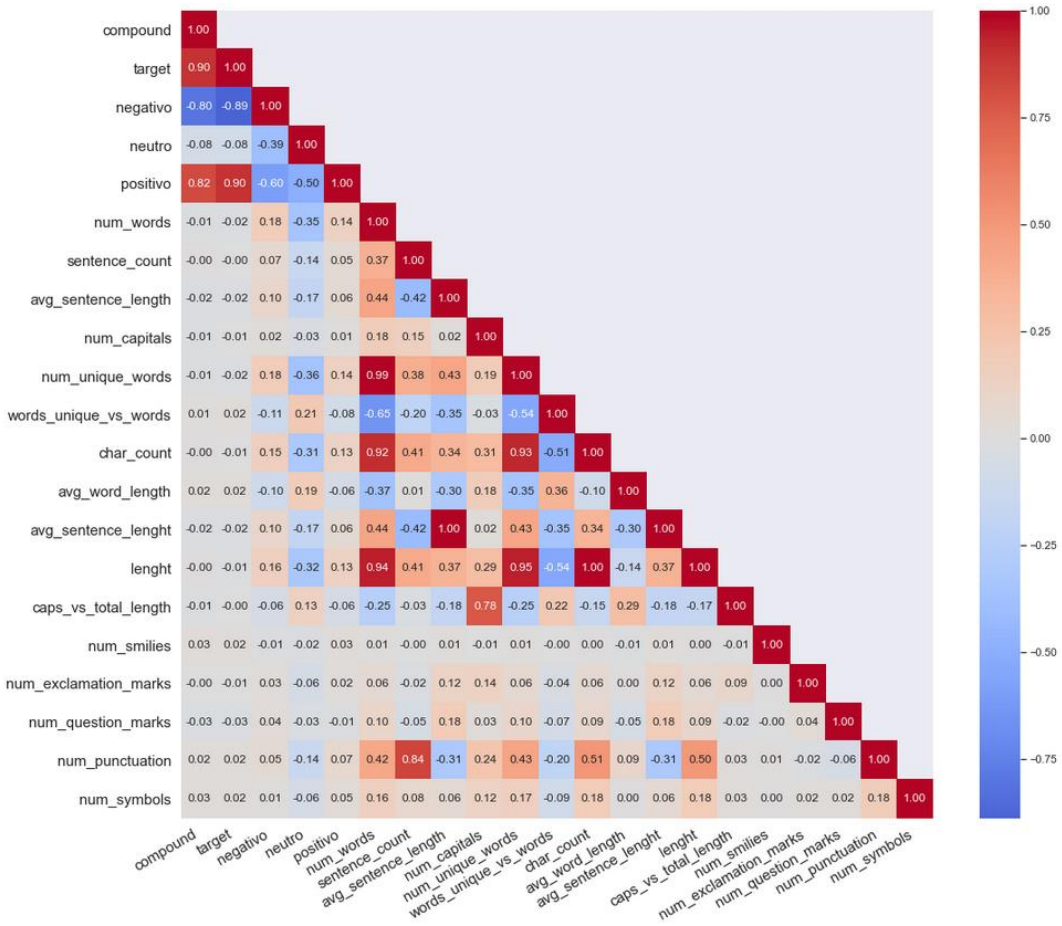


Figura H.14: Matriz de correlación

```python
def count_words(df, column='tokens', min_freq=2):
    # process tokens and update counter
    def update(doc):
        tokens = doc
        counter.update(tokens)

    # create counter and run through all data
    counter = Counter()
    df[column].map(update)

    # transform counter into a DataFrame
    freq_df = pd.DataFrame.from_dict(counter, orient='index', columns=['freq'])
    freq_df = freq_df.query('freq >= @min_freq')
    freq_df.index.name = 'token'

    return freq_df.sort_values('freq', ascending=False)
```

```python
def compute_idf(df, column='tokens', min_df=2):

    def update(doc):
        tokens = doc
        counter.update(set(tokens))

    # count tokens
    counter = Counter()
    df[column].map(update)
    # create DataFrame and compute idf
    idf_df = pd.DataFrame.from_dict(counter, orient='index', columns=['df'])
    idf_df = idf_df.query('df >= @min_df')
    idf_df['idf'] = np.log(len(df)/idf_df['df'])+0.1
    idf_df.index.name = 'token'
    return idf_df
```

Figura H.15: Funciones para el cálculo de TF e IDF.

Tabla H.16: Expresiones Regulares para la normalización del texto (*normalization)*

Tarea	Expresión Regular o Función
“Parsear” código HTML y reemplazarlo por espacios en blanco mediante textacy.	BeautifulSoup(text,'html.parser').get_text() replace_urls(text)
Normalizar los guiones, las comillas, texto en formato” Unicode” y eliminar los acentos con textacy.	tprep.normalize_hyphenated_words(text) tprep.normalize_quotation_marks(text) tprep.normalize_unicode(text) tprep.remove_accents(text)
Otras operaciones de normalización como: expandir contracciones, conversión de abreviaciones, eliminar números, eliminar signos de puntuación	contractions(text) short_to_original(text) _remove_numbers(text)
Lematización con SpaCy.	lemmatizer(text)
Eliminar letras repetidas al menos 4 veces (por seguridad, ya que si eliminamos las letras repetidas al menos una vez, podrían surgir nuevas palabras no existentes en el diccionario o cambiar la palabra por completo, por ejemplo “book” a “bok” y podría introducir ruido en nuestro análisis). En la misma palabra ya que es muy común en las redes sociales repetir la misma letra para dar un mayor enfusíais a la palabra y por tanto es necesario eliminar.	re.sub(pattern= r"(\w)\1\1\1\1", repl=r'\1', string=text)
Eliminar palabras cuya longitud sea menor que 2 (para eliminar aquellas palabras que se hayan escrito por error o palabras que tras los métodos de preprocesamiento anteriores hayan dado lugar a palabras no existentes)	re.sub(pattern=r'\W*\b\w{1,2}\b', repl=" ", string=text)
Eliminar las secuencias de espacios en blanco que hayan podido quedar tras aplicar los pasos anteriores.	re.sub(r'\s+', ' ', text)
Eliminar los espacios al principio y al final de la cadena.	text.strip()

Tabla H.17: Ejemplo de preprocesamiento y “tokenizacion”

***Tweet* original**	***Tweet preprocesado***	***Tokens* (lista de palabras)**	**Puntuación (*Compound*)**
“@realDonaldTrump Thank you for your great Leadership on moving this vaccine forward, history will thank your administration for a great 4 years of accomplishments” ("@realDonaldTrump Gracias por su gran Liderazgo para sacar adelante esta vacuna, la historia agradecerá a su administración por unos grandes 4 años de logros”)	“thank you for your great leadership move this vaccine forward history will thank your administration for great year accomplishment”	list(['thank', 'you', 'for', 'your', 'great', 'leadership', 'move', 'this', 'vaccine', 'forward', 'history', 'will', 'thank', 'your', 'administration', 'for', 'great', 'year', 'accomplishment']	0.9217
“We have a lot of evidence around current vaccines to see that "herd immunity" and mandates don't	“have lot evidence around current vaccine see that herd immunity and mandate	list(['have', 'lot', 'evidence', 'around', 'current', 'vaccine', 'see',	-0.5093

automatically solve the problem of transmission.” ("Tenemos muchas pruebas en torno a las vacunas actuales para ver que la "inmunidad de rebaño" y los mandatos no resuelven automáticamente el problema de la transmisión".)	not automatically solve the problem transmission”	'that', 'herd', 'immunity', 'and', 'mandate', 'not', 'automatically', 'solve', 'the', 'problem', 'transmission']	

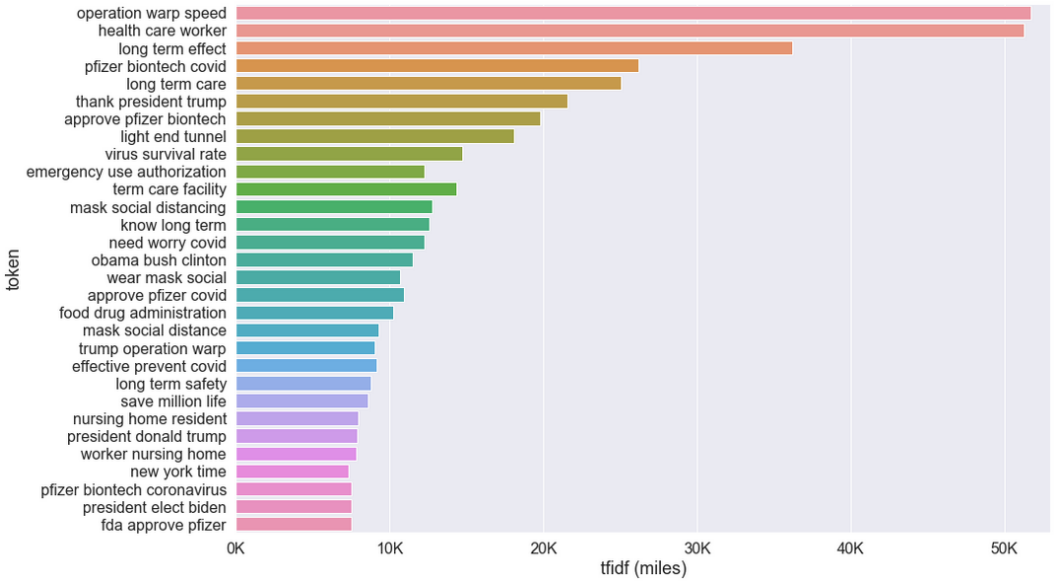


Figura H.16: Gráfico de barras de los 30 “trigramas” positivos más frecuentes calculados con TF-IDF

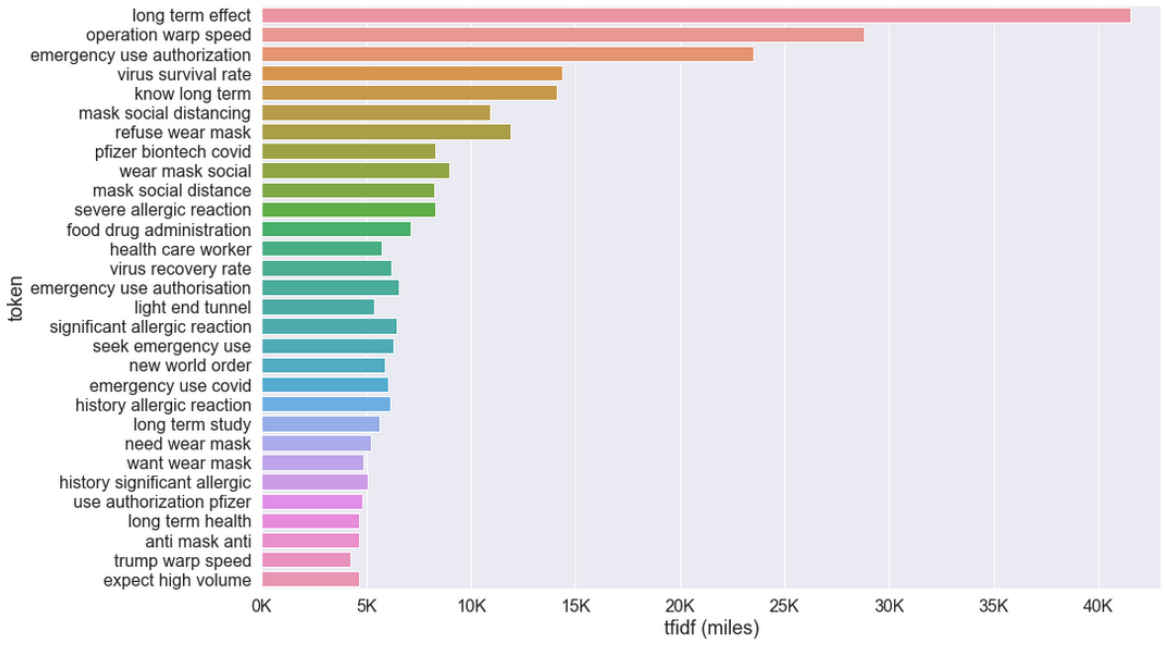


Figura H.17: Gráfico de barras de los 30 “trigramas” negativos más frecuentes calculados con TF-IDF

General	
ID de nodo	TextParsing
Datos importados	
Datos exportados	
Notas	
Entrenamiento	
Variables	
Parse	
Variable parse	clean_text_w2vec_2
Idioma	Inglés
Detectar	
Diferentes partes del discurso	Sí
Grupos de nombre	Sí
Términos multipalabra	SASHELP.ENG_MULTI
Buscar entidades	Estándar
Personalizar entidades	
Ignorar	
Ignorar partes del discurso	'Aux' 'Conj' 'Det' 'Interj' 'Prep'
Ignorar tipos de entidades	
Ignorar tipos de atributos	'Núm' 'Punct'
Sinónimos	
Términos tema	Sí
Sinónimos	SASHELP.ENGSYNMS
Filtro	
Lista de términos que se van a incluir	
Lista de palabras vacías	SASHELP.ENGSTOP
Seleccionar idiomas	

Figura H.18: Configuración nodo "*Parsing* del texto"

General	
ID de nodo	TextFilter2
Datos importados	
Datos exportados	
Notas	
Entrenamiento	
Variables	
Ortografía	
Revisar ortografía	No
Diccionario	
Pesos	
Peso de frecuencia	Ninguno
Peso de término	Frecuencia de documento inverso
Filtros del término	
Número mínimo de documentos	4
Número máximo de términos	.
Importar sinónimos	
Filtros del documento	
Expresión de búsqueda	
Extraer documentos	
Resultados	
Visor del filtro	
Resultado de la revisión ortográfica	
Sinónimos exportados	
Informe	
Términos que se van a ver	Todo
Número de términos que van a aparecer	20000

Figura H.19: Configuración nodo "filtrado del texto"

General	
ID de nodo	TextTopic
Datos importados	
Datos exportados	
Notas	
Entrenamiento	
Variables	
Temas del usuario	
Temas del término	
Número de temas de un solo término	0
Temas aprendidos	
Número de temas multitérminos	10
Temas correlacionados	No
Resultados	
Visor de temas	

Figura H.20: Configuración nodo “Tema del texto”

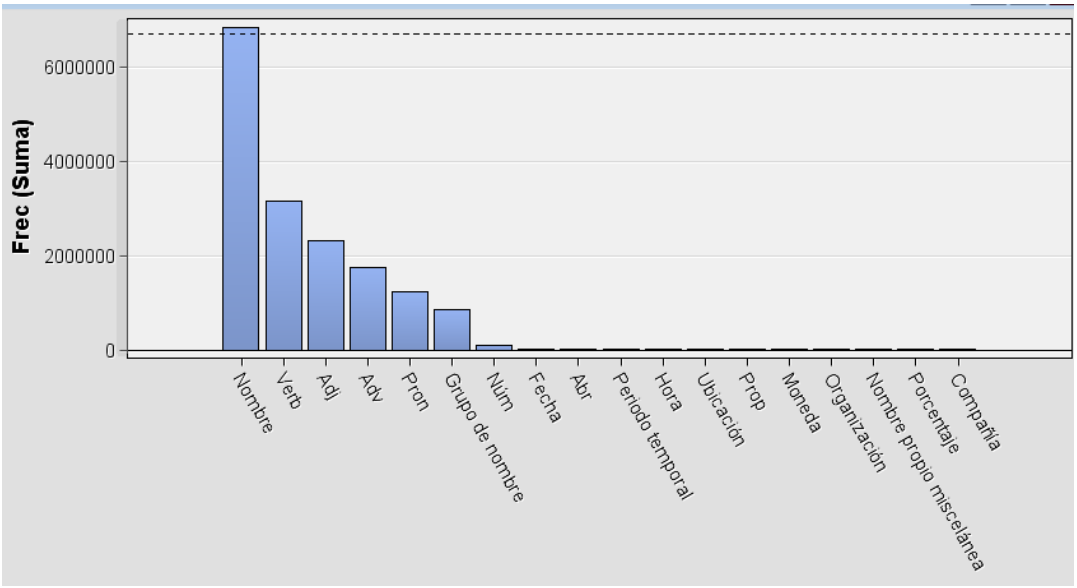
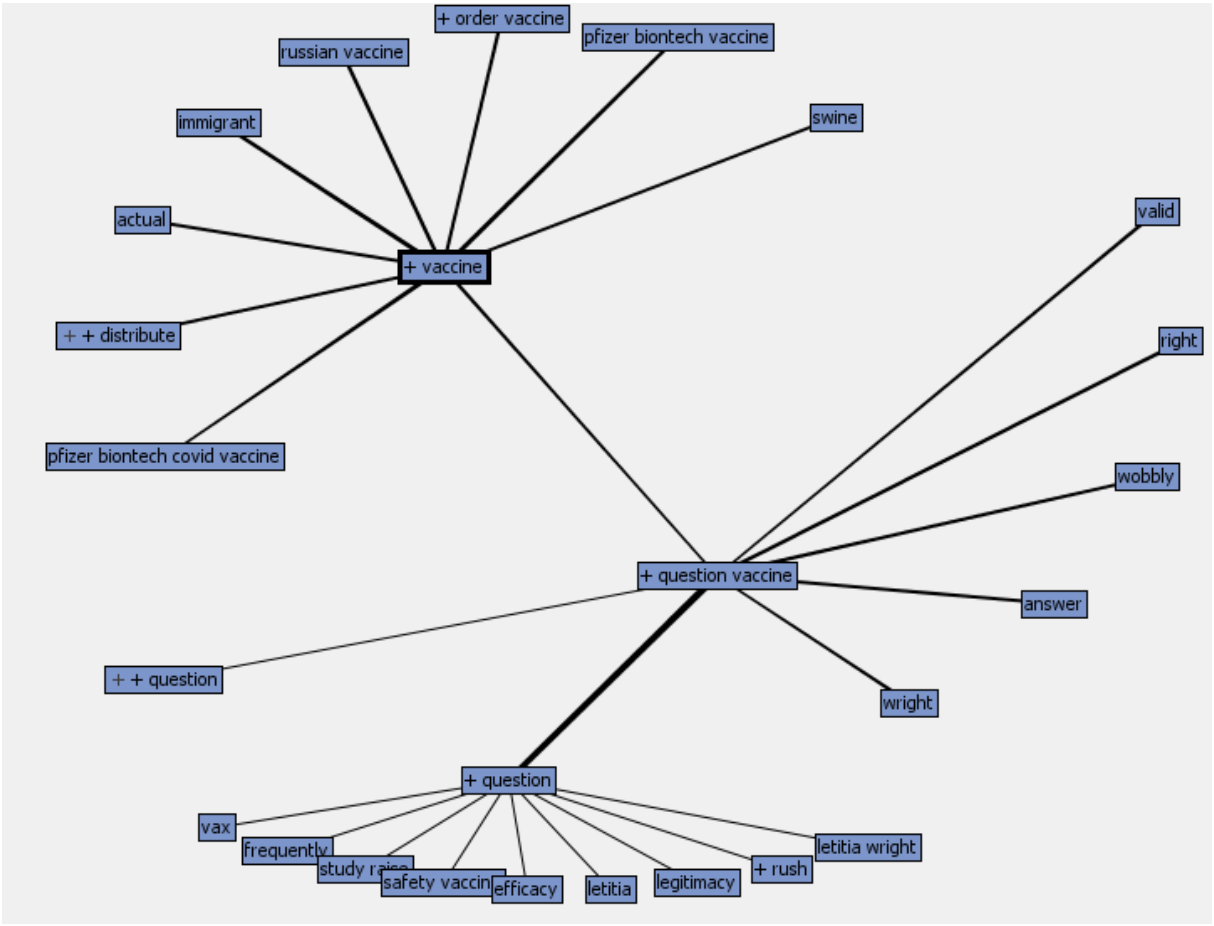


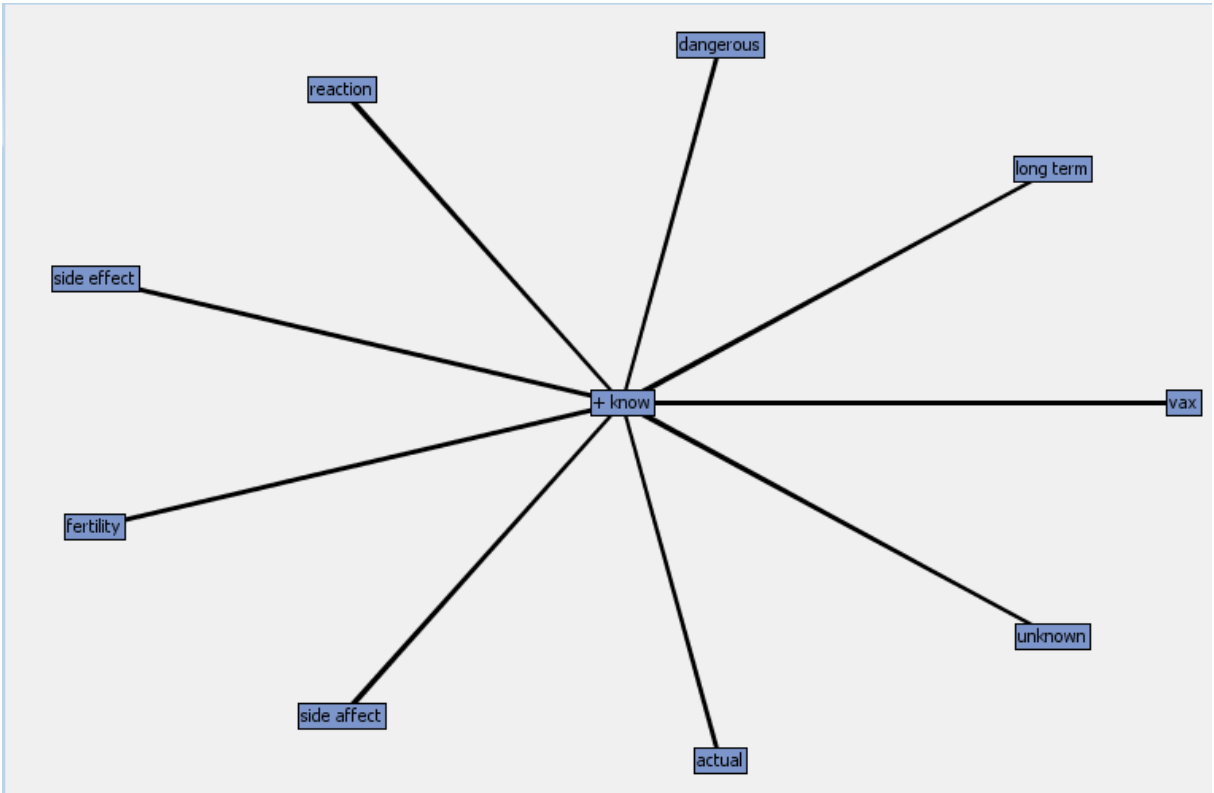
Figura H.21: Resultados del nodo *parsing* de SAS.

Tabla H.18: Descripción de las variables del nodo “Tema del texto".

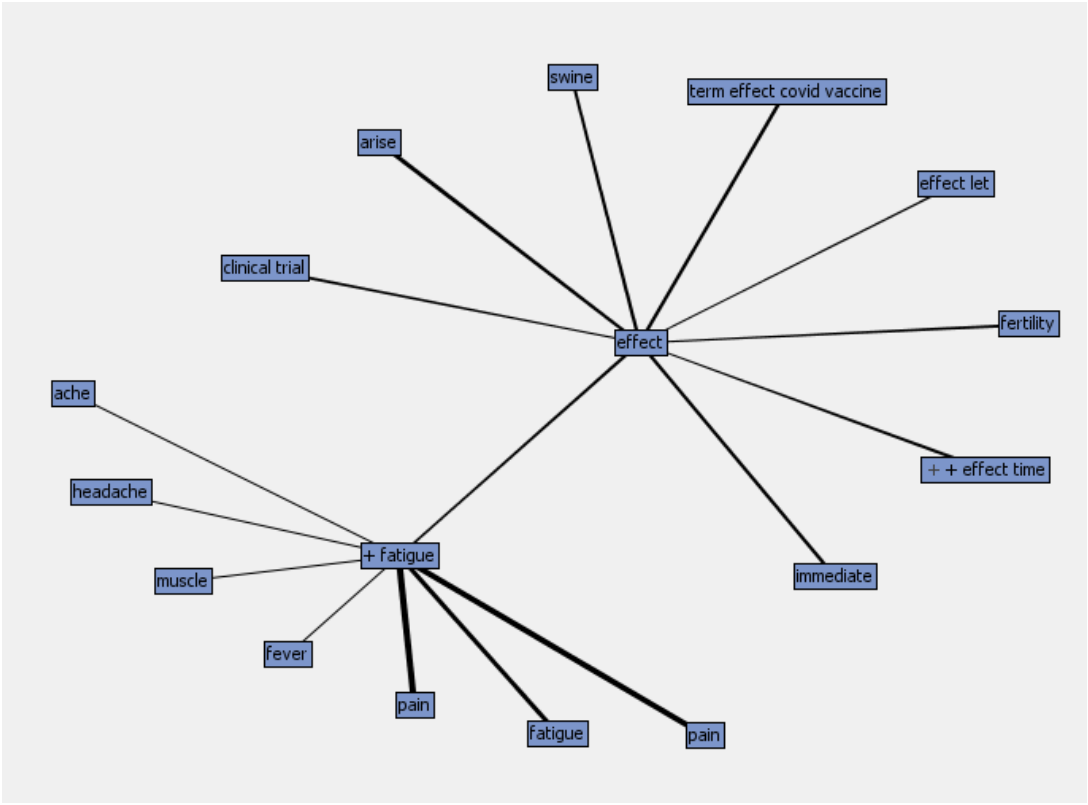
Variable	Descripción
Topic ID	Valor clave del tema.
Document Cutoff	Peso mínimo del tema que debe tener un documento para ser incluido en este tema.
Term cutoff	Peso mínimo del tema que debe tener un término para ser utilizado como término en este tema. El peso de cualquier término que tenga un valor absoluto de peso inferior a éste se pone a cero.
Topic	Términos que describen el tema.
Number of Terms	Número de términos en el tema.
# Docs	Número de documentos que contienen el tema.
Category	Usuario si el tema es definido por el usuario, simple si el tema es de un solo término, y múltiple si el tema es de varios términos.



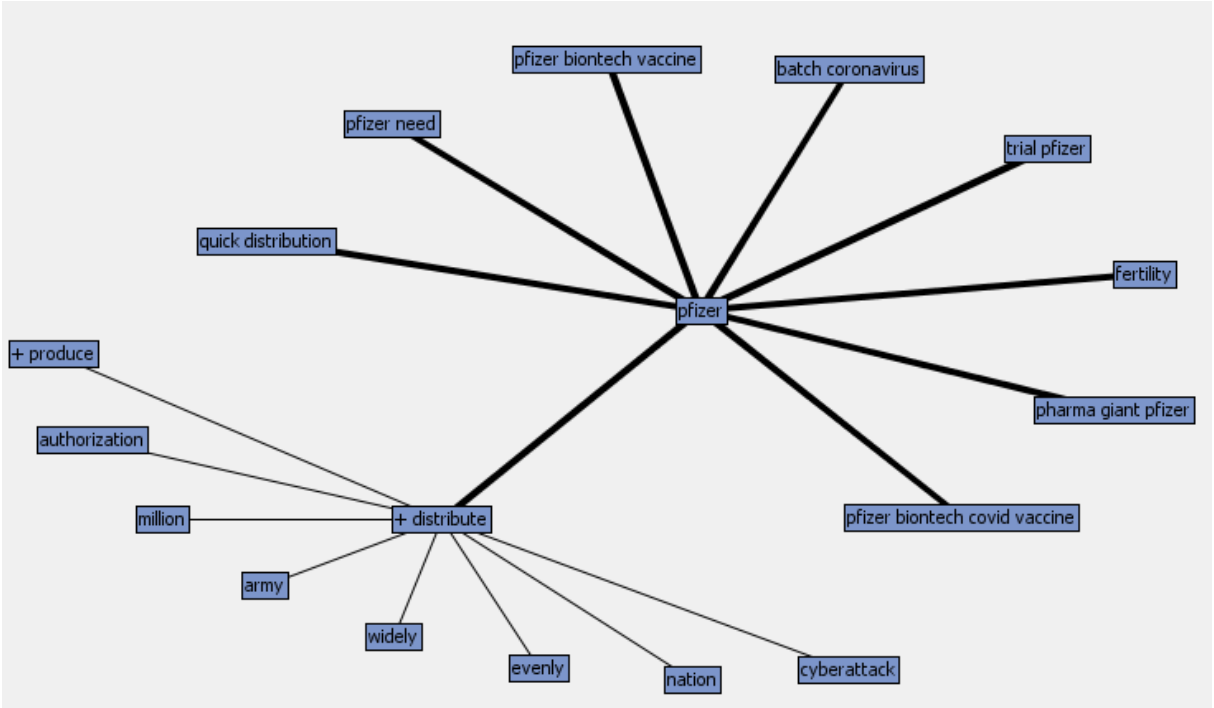
a) Enlaces de concepto correspondiente a *vaccine*.



b) Enlaces de concepto correspondiente a *know*.



c) Enlaces de concepto correspondiente a *effect* (efectos de la vacuna).



d) Enlaces de concepto correspondiente a *pfizer.*

Figura H.22: Mapa de enlaces conceptuales.

Términos

Peso del tema	+	Término	Rol	Nº docs	Frec
0.572	+	know	Verb	62680	68145
0.466		effect	Nombre	23428	25488
0.302	+	long	Adj	20971	22349
0.288		side	Adj	13409	14131
0.266		term	Nombre	13528	14763
0.202		side effect	Grupo de nombre	6856	7110
0.125		year	Nombre	50586	56341
0.113	+	vaccine	Nombre	654389	715142
0.109		term effect	Grupo de nombre	2498	2609
0.1		side	Nombre	5301	5436
0.08		vaccine	Adj	362572	383966
0.062	+	month	Nombre	30829	32711
0.06	+	test	Verb	16683	17783
0.052		test	Nombre	19824	21315

Documentos

Peso del tema	clean_text_w2vec
0.769	still can not get over people say they will not get the vaccine because they not
0.71	have see lot people say but what about the long term side effect can not know
0.7	you will not know the long term effect until use long term the side effect not
0.692	course unknown have only know about the virus for less than year obviously not
0.685	feel that the time offer will have enough information know none know the long
0.684	cocaine around hundred year know risk know effect know long term implication
0.676	you reread what say before scientist not know enough about the virus long term
0.672	know vaccine come but wary this long term side effect thing how can you know
0.67	well will not take the vaccine till know about long term effect you can not know the
0.67	will right behind you love the people that worried about long term effect when
0.669	the process could base that which also save some time and for people worry
0.666	hey isabelle you know what the vaccine and what the side effect not think psst
0.659	have you see the side effect you vaccine also have not enough testing and could
0.655	you realize covid have unknown long term side effect right like not get the vaccine
0.653	the vaccine have never research for safety one know the side effect long term

Figura H.23: Visor de Temas de Texto (Tópico 7)

Tabla H.19: Tipos de preprocesamiento para la etapa de modelado.

	Preprocesamiento 1	**Preprocesamiento 2**
Eliminación de ruido (noise removal)	SI	SI
Normalización del texto	SI	SI (Sin lematización)
Eliminación de *stopwords*	NO	NO

Tabla H.20: *Tweets* únicos por procesamiento

	Nº de *tweets*	**Nº de *tweets* únicos**
Preprocesamiento 1	4.026.416	3.401.961
Preprocesamiento 2	4.026.416	3.410.790

“join” entre Preprocesamiento 1 y Preprocesamiento 2	3.401.864	3.401.864

	id_str	compound	vader_sentiment	clean_text_w2vec	target
0	1327830320595529728	0.4404	positivo	hope rise covid vaccine the stage validation	2
1	1327830491559567360	0.7351	positivo	dissemination the vaccine take place the twili...	2
2	1327830738385985537	-0.2960	negativo	you have not produce anything especially not v...	0
3	1327830753380622342	-0.2960	negativo	reasonable person would submit inject with pol...	0
4	1327830801862565888	0.0000	neutro	trump new york will not receive covid vaccine ...	1
...	...	...	...	...	...
149995	1339063722887835650	0.5574	positivo	gold ease vaccine rollout expand feed meet eyed	2
149996	1339063883621777408	-0.2960	negativo	politician should get the vaccine until the en...	0
149997	1339063885559689216	0.6249	positivo	president trump tireless work give the vaccine...	2
149998	1339063893054738433	0.4404	positivo	every other vaccine those people take first de...	2
149999	1339063913653014529	0.0000	neutro	they have just roll out but take your point wi...	1

150000 rows × 5 columns

Figura H.24: *DataFrame* seleccionado de forma aleatoria del preprocesamiento 1.

	id_str	compound	vader_sentiment	clean_text_w2vec_2	target
0	1327830320595529728	0.4404	positivo	hope rises covid vaccines are the stages valid...	2
1	1327830491559567360	0.7351	positivo	dissemination the vaccine takes place the twil...	2
2	1327830738385985537	-0.2960	negativo	you have not produced anything especially not ...	0
3	1327830753380622342	-0.2960	negativo	reasonable person would submit injected with p...	0
4	1327830801862565888	0.0000	neutro	trump new york will not receive covid vaccine ...	1
...	...	...	...	...	...
149995	1339063722887835650	0.5574	positivo	gold eases vaccine rollouts expand fed meet eyed	2
149996	1339063883621777408	-0.2960	negativo	politician should get the vaccine until the en...	0
149997	1339063885559689216	0.6249	positivo	president trumps tireless work gave the vaccin...	2
149998	1339063893054738433	0.4404	positivo	every other vaccine those people take first de...	2
149999	1339063913653014529	0.0000	neutro	they have just rolled out but take your point ...	1

150000 rows × 5 columns

Figura H.25: *DataFrame* seleccionado de forma aleatoria del preprocesamiento 2.

Tabla H.21: Estados con mayor porcentaje de tweets negativos

Estado	Tweets	Compound (media)	Negatividad (% sobre el total de tweets por estado)
Wyoming	540	0.035394	37.4074
Colorado	8555	0.008462	36.9374
Tennessee	3747	0.028654	36.7494
Idaho	1088	0.023467	36.5808

Ohio	9372	0.027885	35.9795
New Mexico	1724	0.033422	35.8468
Nebraska	1052	0.034869	35.5513
Arizona	6672	0.038181	35.3267
Kentucky	3126	0.032152	35.2527
South Carolina	3009	0.034918	35.1944
Rhode Island	1246	0.046528	34.9117
North Dakota	324	0.017709	34.8765
Missouri	3003	0.044453	34.7985
Arkansas	1466	0.037457	34.7885
Pennsylvania	7575	0.039553	34.7590

Tabla H.22: Estados con mayor porcentaje de tweets positivos

Estado	Tweets	Compound (media)	Positividad (% sobre el total de tweets por estado)
Utah	2178	21.946740	49.4949
South Dakota	563	22.912966	49.0230
District of Columbia	530	21.320755	48.6792
Washington	40214	26.458447	47.9285
Iowa	2643	22.020431	45.7813
Virginia	7856	23.116090	45.7357
West Virginia	1090	23.486239	45.6880
Vermont	1400	23.357143	45.6428
Kansas	5898	25.601899	45.4730
New York	62070	26.566779	45.3681
Massachusetts	6547	0.065182	45.1199
Connecticut	4328	0.087190	45.0092
Hawaii	1691	0.075233	44.7072
New Hampshire	1722	0.071941	44.6573
Maryland	6505	0.070309	44.5964

Tabla H.23: Estados con mayor diferencia entre sentimientos positivos y negativos

Estado	Tweets	Compound (media)	Negativos	Neutros	Positivos	Diferencia (valor absoluto)
Washington	40214	0.126290	25.6129	26.4584	47.9285	22.3156
South Dakota	563	0.106299	28.0639	22.9129	49.0230	20.9591
Utah	2178	0.119533	28.5583	21.9467	49.4949	20.9366
District of Columbia	530	0.106050	30	21.3207	48.6792	18.6792
New York	62070	0.095069	28.0650	26.5667	45.3681	17.3030
Kansas	5898	0.095548	28.9250	25.6018	45.4730	16.5479
Connecticut	4328	0.087190	29.1820	25.8086	45.0092	15.8271
West Virginia	1090	0.078111	30.8256	23.4862	45.6880	14.8623
Vermont	1400	0.081043	31	23.3571	45.6428	14.6428

Virginia	7856	0.081891	31.1481	23.1160	45.7357	14.5875

Tabla H.24: Estados con menor diferencia entre sentimientos positivos y negativos

Estado	Tweets	Compound (media)	Negativos	Neutros	Positivos	Diferencia (valor absoluto)
Idaho	1088	0.023467	36.580882	23.4375	39.9816	3.4007
Colorado	8555	0.008462	36.937463	22.3962	40.66627	3.7288
Wyoming	540	0.035394	37.407407	20.3703	42.22222	4.8148
Tennessee	3747	0.028654	36.749400	21.0034	42.24713	5.4977
Illinois	5377	0.033085	34.721964	24.7349	40.54305	5.8210
Ohio	9372	0.027885	35.979513	22.1297	41.89073	5.9112
Louisiana	2103	0.031780	34.331907	25.2020	40.46600	6.1340
Alabama	2709	0.027870	33.628645	26.3196	40.05168	6.4230
Kentucky	3126	0.032152	35.252719	23.0326	41.71465	6.4619
New Mexico	1724	0.033422	35.846868	21.5777	42.5754	6.7285

Tabla H.25: Lista de *stopwords* “customizadas”

Preprocesamiento	Custom *stopwords*
df_preprocesado_1	'about', 'all', 'and', 'can', 'covid', 'for', 'get', 'have', 'not', 'people', 'take', 'that', 'the', 'they', 'this', 'vaccine', 'will', 'with', 'you'
df_preprocesado_2	'about', 'and', 'are', 'can', 'covid', 'for',

	'get', 'have', 'not', 'people', 'that', 'the', 'they', 'this', 'vaccine', 'vaccines', 'will', 'with', 'you'

Tabla H.26: Configuración del experimento TF y TF-IDF y RL.

Modelo	Preprocesamiento	Tipo de vectorización	*n*-gramas	Stopwords	Ganador
1	df_preproceso_1	CountVectorizer (TF) (1000,10000, 1000)	Unigramas	• Sin custom stopwords • Sin stopwords • Con stopwords	• Sin custom stopwords
2	df_preproceso_2	CountVectorizer (TF) (1000,10000, 1000)	Unigramas	• Sin custom stopwords • Sin stopwords • Con stopwords	• Sin custom stopwords
3	df_preproceso_1	TfidfVectorizer (TFIDF) (1000,10000, 1000)	Unigramas	• Sin custom stopwords • Sin stopwords • Con stopwords	• Sin custom stopwords
4	df_preproceso_2	TfidfVectorizer (TFIDF) (1000,10000, 1000)	Unigramas	• Sin custom stopwords • Sin stopwords • Con stopwords	• Sin custom stopwords

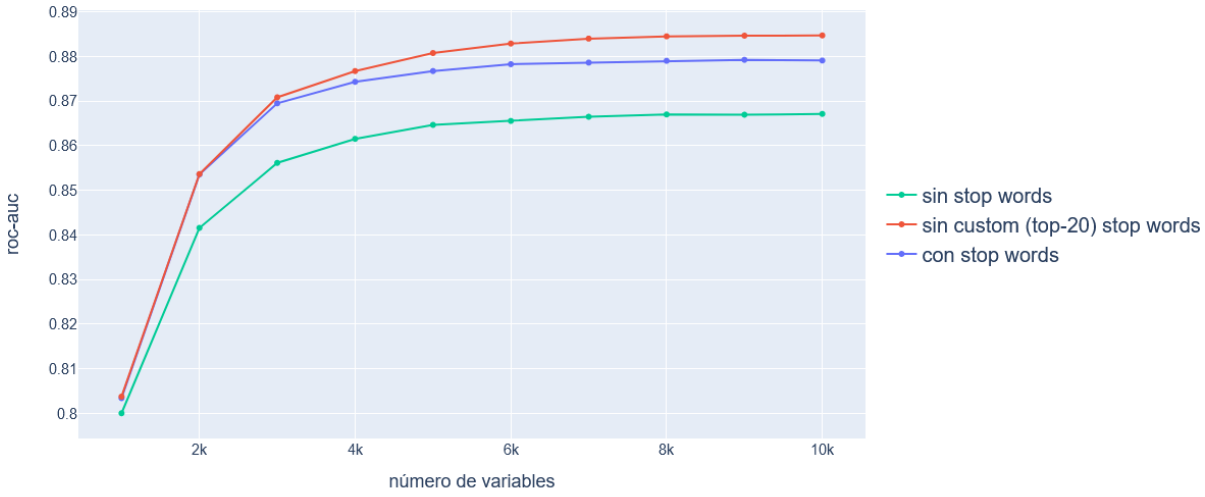


Figura H.26: *CountVectorizer* (“Unigramas”): Sin *stopwords* y con lematización vs con *stopwords* y con lematización.

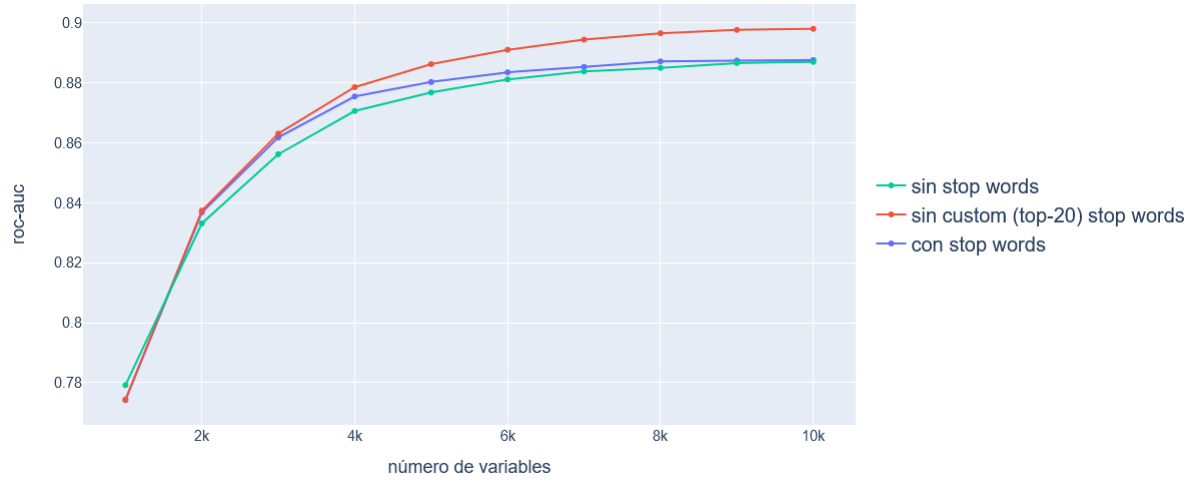


Figura H.27: *CountVectorizer* (“Unigramas”): Sin *stopwords* y sin lematización vs con *stopwords* y sin lematización.

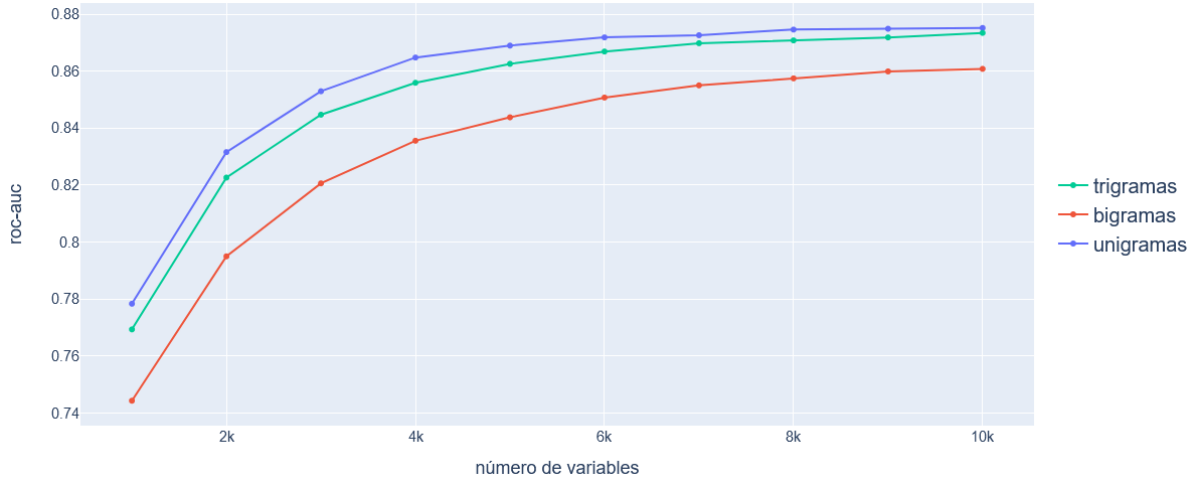


Figura H.28: Resultados (*Roc-Auc*) *n*-gramas (1-3) con *CountVectorizer.*

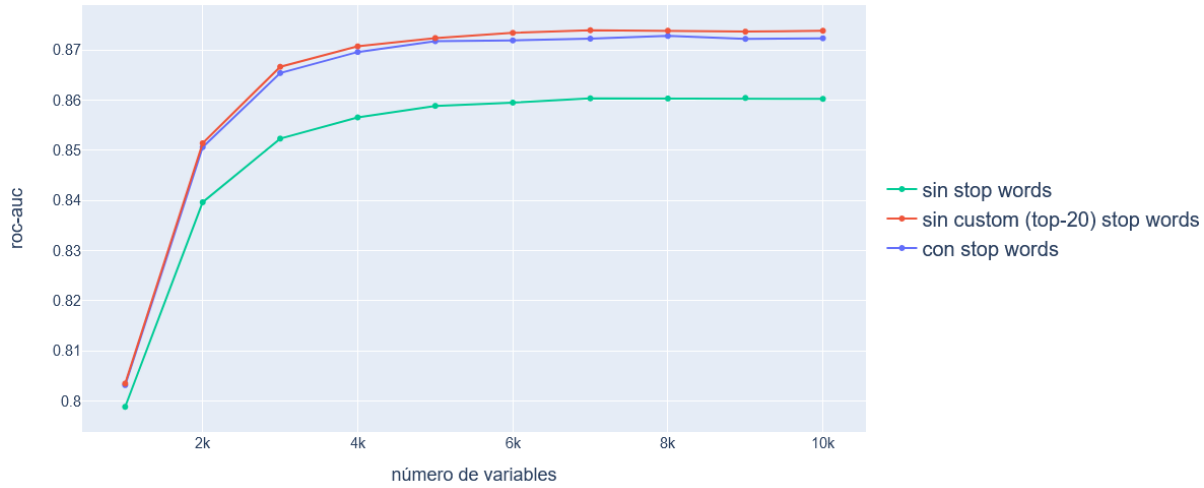


Figura H.29: TF-IDF: *TfidfVectorizer* (“Unigramas”): Sin *stopwords* y con lematización vs con *stopwords* y con lematización.

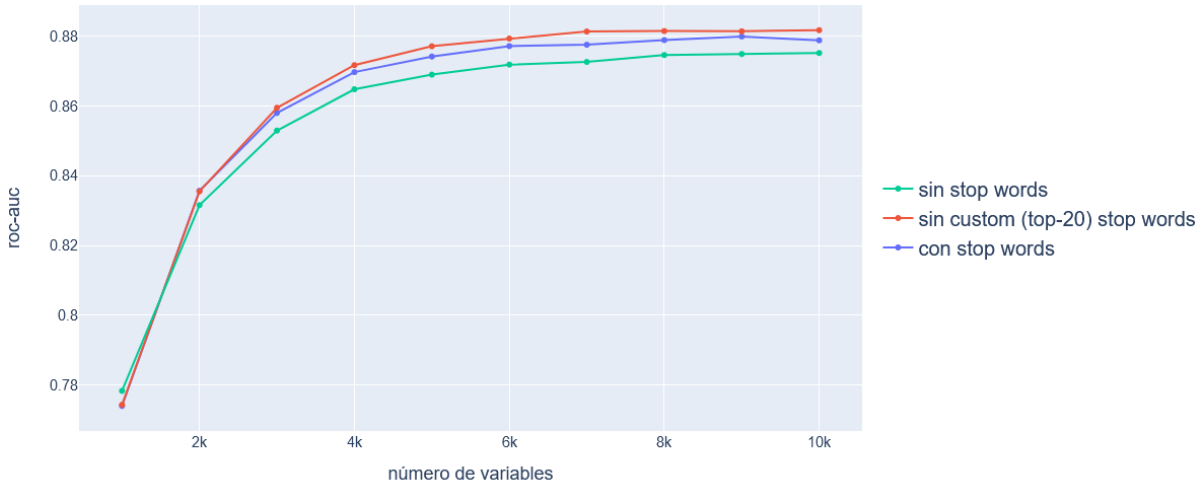


Figura H.30: TF-IDF: *TfidfVectorizer* (“Unigramas”): Sin *stopwords* y sin lematización vs con *stopwords* y sin lematización.

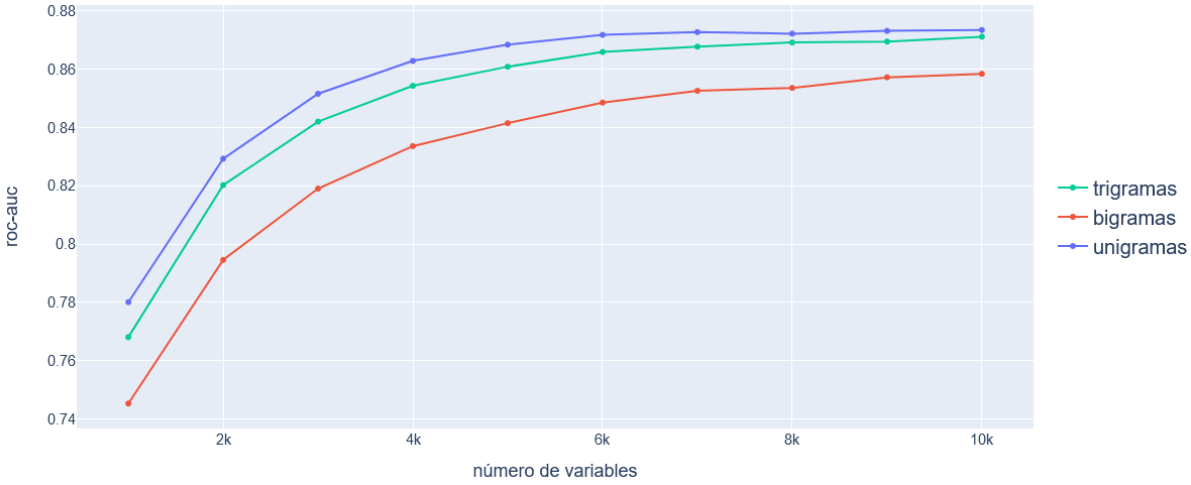


Figura H.31: Resultados (*Roc-Auc*) *n*-gramas (1-3) con *TfidfVectorizer.*

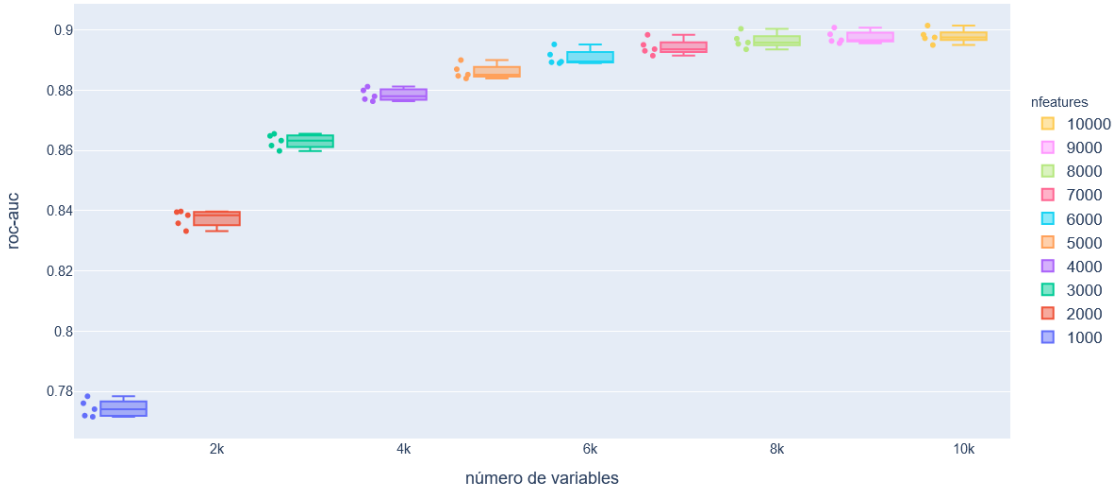


Figura H.32: Resultados de la validación cruzada sobre el mejor modelo con *CountVectorizer* (modelo 2).

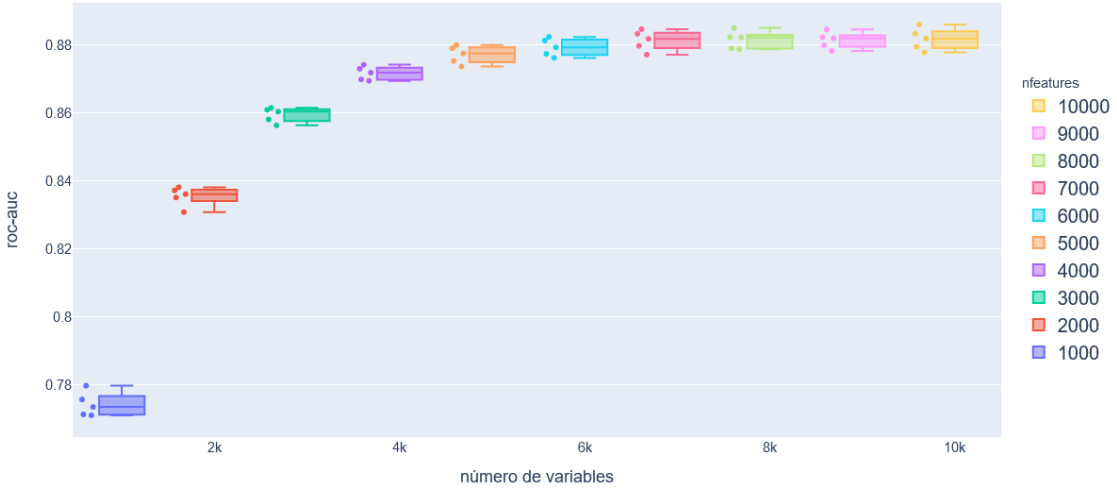


Figura H.33: Resultados de la validación cruzada sobre el mejor modelo con *TfidfVectorizer* (modelo 4).

Tabla H.27: Resultado validación cruzada con Chi2 utilizando los *p*-valores.

Validación cruzada	Roc-Auc	nfeatures	Modelo
0	0.908321	3912	chi_pvalue_cv
1	0.907250	3912	chi_pvalue_cv
2	0.907393	3912	chi_pvalue_cv
3	0.906036	3912	chi_pvalue_cv
4	0.908679	3912	chi_pvalue_cv
5	0.883964	1475	chi_pvalue_tf_idf
6	0.883643	1475	chi_pvalue_tf_idf
7	0.883964	1475	chi_pvalue_tf_idf
8	0.882500	1475	chi_pvalue_tf_idf
9	0.885536	1475	chi_pvalue_tf_idf

Tabla H.28: Resultado validación cruzada con Chi2 utilizando los *n scores*.

Validación cruzada	Roc-Auc	nfeatures	Modelo
0	0.886357	1000	chi_score_cv_1K
1	0.887536	1000	chi_score_cv_1K
2	0.887750	1000	chi_score_cv_1K
3	0.888107	1000	chi_score_cv_1K
4	0.888679	1000	chi_score_cv_1K
0	0.906714	2000	chi_score_cv_2K
1	0.906143	2000	chi_score_cv_2K
2	0.905929	2000	chi_score_cv_2K
3	0.904643	2000	chi_score_cv_2K
4	0.906643	2000	chi_score_cv_2K
0	0.908321	3000	chi_score_cv_3K
1	0.907964	3000	chi_score_cv_3K
2	0.907714	3000	chi_score_cv_3K
3	0.906821	3000	chi_score_cv_3K
4	0.910000	3000	chi_score_cv_3K
0	0.908393	4000	chi_score_cv _4K
1	0.907679	4000	chi_score_cv _4K
2	0.906929	4000	chi_score_cv _4K

3	0.905714	4000	chi_score_cv _4K
4	0.908357	4000	chi_score_cv _4K
0	0.907036	5000	chi_score_cv_5K
1	0.905214	5000	chi_score_cv_5K
2	0.906786	5000	chi_score_cv_5K
3	0.906214	5000	chi_score_cv_5K
4	0.907571	5000	chi_score_cv_5K
0	0.878929	1000	chi_score_tf_1K
1	0.877107	1000	chi_score_tf_1K
2	0.879286	1000	chi_score_tf_1K
3	0.880071	1000	chi_score_tf_1K
4	0.879464	1000	chi_score_tf_1K
0	0.883964	2000	chi_score_tf_2K
1	0.885679	2000	chi_score_tf_2K
2	0.884429	2000	chi_score_tf_2K
3	0.883893	2000	chi_score_tf_2K
4	0.885286	2000	chi_score_tf_2K
0	0.887964	3000	chi_score_tf_3K
1	0.886714	3000	chi_score_tf_3K
2	0.886786	3000	chi_score_tf_3K
3	0.885714	3000	chi_score_tf_3K
4	0.887429	3000	chi_score_tf_3K
0	0.888929	4000	chi_score_tf_4K
1	0.886357	4000	chi_score_tf_4K
2	0.887464	4000	chi_score_tf_4K
3	0.885179	4000	chi_score_tf_4K
4	0.888536	4000	chi_score_tf_4K
0	0.887571	5000	chi_score_tf_5K
1	0.884107	5000	chi_score_tf_5K
2	0.884821	5000	chi_score_tf_5K
3	0.883357	5000	chi_score_tf_5K
4	0.888679	5000	chi_score_tf_5K

Tabla H.29: Resultado validación cruzada con RFECV.

Validación cruzada	Roc-Auc	nfeatures	Modelo
0	0.915321	1327	RFECV_cv
1	0.915000	1327	RFECV_cv
2	0.914179	1327	RFECV_cv
3	0.912679	1327	RFECV_cv
4	0.914500	1327	RFECV_cv
0	0.886250	1208	RFECV_tfidf
1	0.885786	1208	RFECV_tfidf
2	0.888643	1208	RFECV_tfidf
3	0.886250	1208	RFECV_tfidf
4	0.888321	1208	RFECV_tfidf

Tabla H.30: Resultado validación cruzada con PCA.

Validación cruzada	Roc-Auc	nfeatures	Modelo
0	0. 862667	2878	PCA_cv
1	0. 857778	2878	PCA_cv
2	0. 861444	2878	PCA_cv
3	0. 8555	2878	PCA_cv
4	0. 857676	2878	PCA_cv
5	0. 87589	4723	PCA_tfidf
6	0. 874056	4723	PCA_tfidf
7	0. 876111	4723	PCA_tfidf
8	0. 871611	4723	PCA_tfidf
9	0. 874	4723	PCA_tfidf

Tabla H.31: Configuración de los parámetros RFECV.

Estimador	Nº Mínimo de características	*Step*	Validación cruzada	*Scoring*
Regresión Logística	1000	*1*	5 *k*-fold	ROC-AUC

Tabla H.32: Resultados validación cruzada de los mejores modelos seleccionados por cada método de selección de características.

Método de selección	Nombre modelo	Método de vectorización	Nº de características	ROC-AUC (media)
Chi2 Pvalor	1	*CountVectorizer* (modelo 2)	3912	0.907535714
	2	*TfidfVectorizer* (modelo 4)	1475	0.88392143
Chi2 Score	3	*CountVectorizer* (modelo 2)	2000	0.9060144
	4	*TfidfVectorizer* (modelo 4)	2000	0.8846502
RFECV	5	*CountVectorizer* (modelo 2)	1327	0.9143358
	6	*TfidfVectorizer* (modelo 4)	1208	0.88705
PCA	7	*CountVectorizer* (modelo 2)	2878	0.8712858
	8	*TfidfVectorizer* (modelo 4)	4723	0.8585642
Baseline CV	9	*CountVectorizer* (modelo 2)	10.000	0.8979714
Baseline TF-IDF	10	*TfidfVectorizer* (modelo 4)	10.000	0.8816856

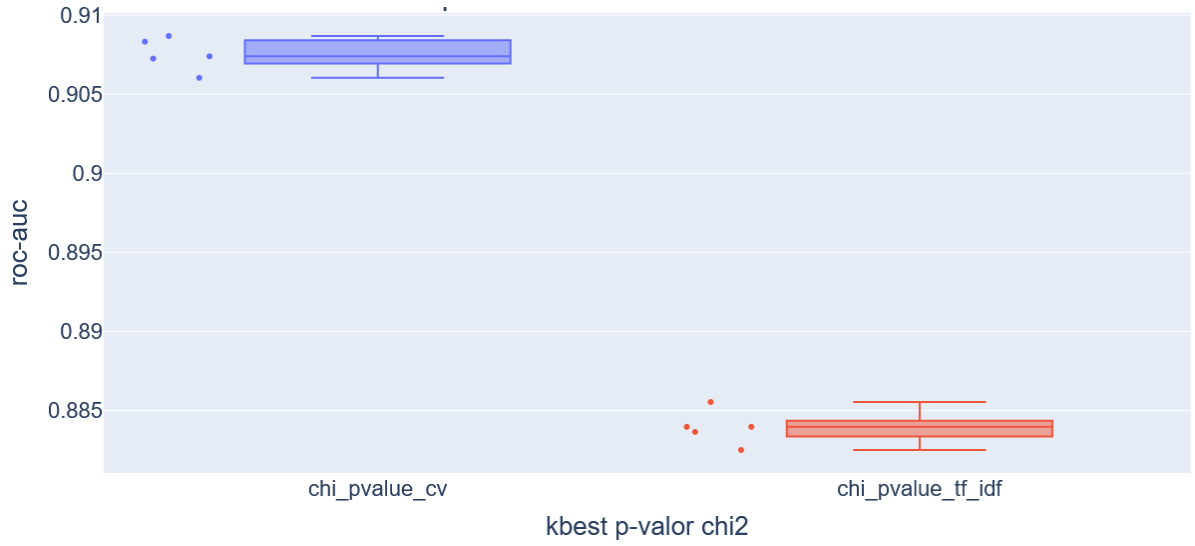
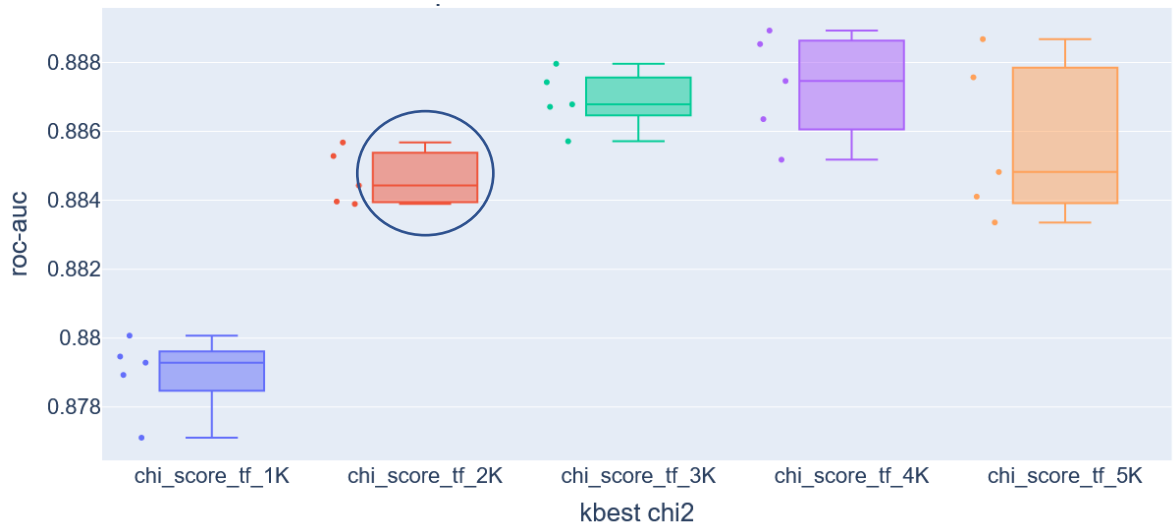


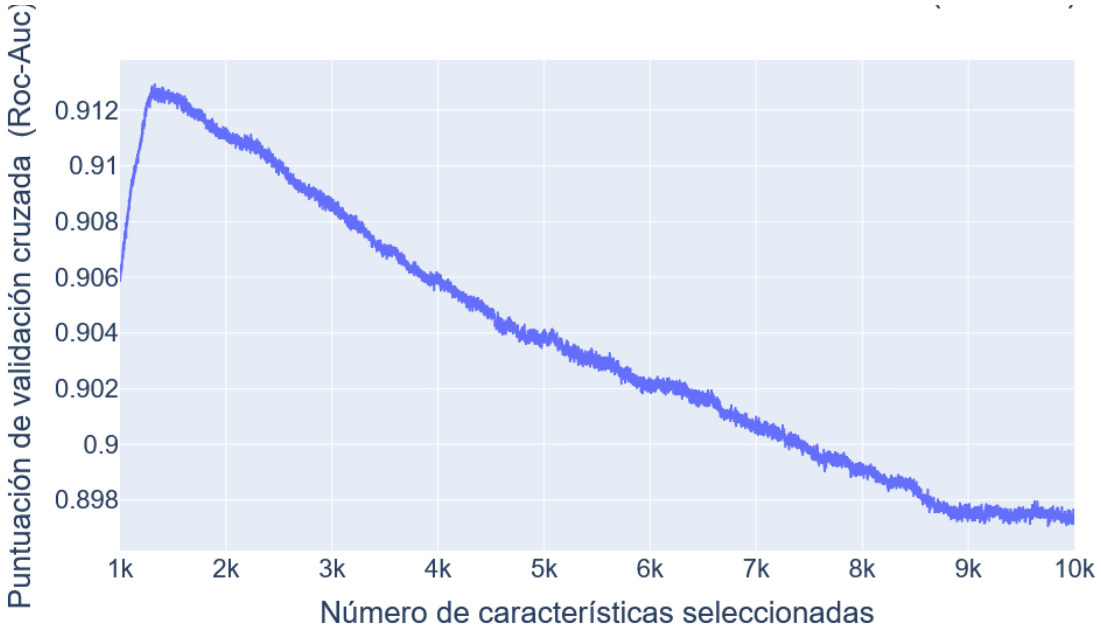
Figura H.34: Resultado validación cruzada con Chi2 (*p*-valores).

a) CountVectorizer (modelo 2)

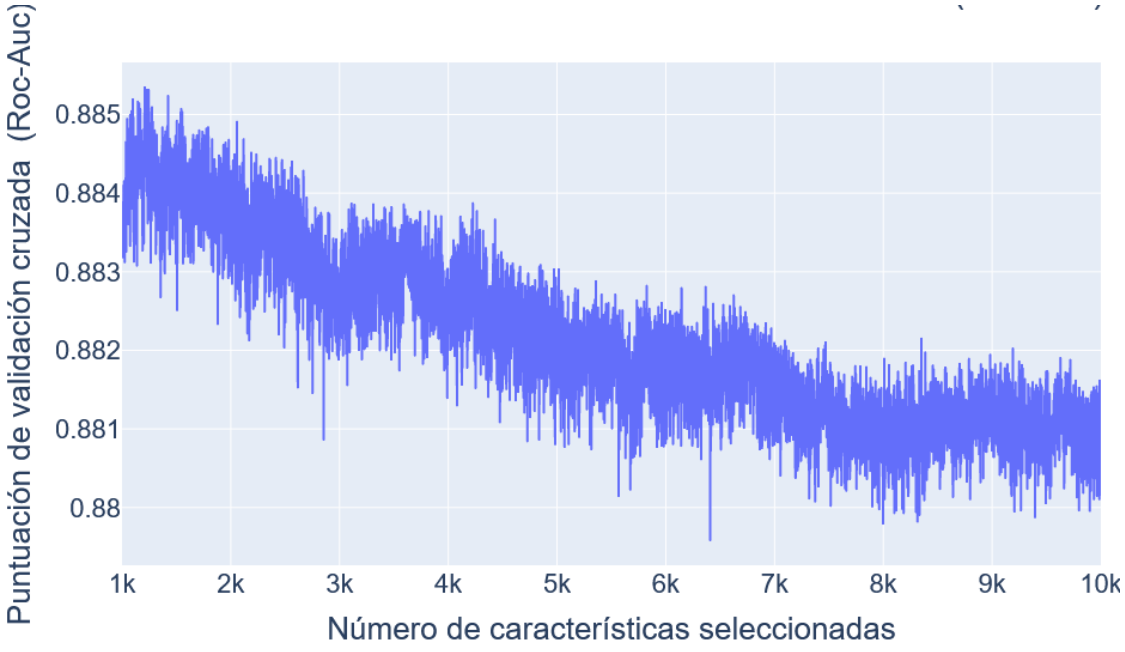


b) TfidftVectorizer (modelo 4)

Figura H.35: Resultado validación cruzada con Chi2 (n-scores).

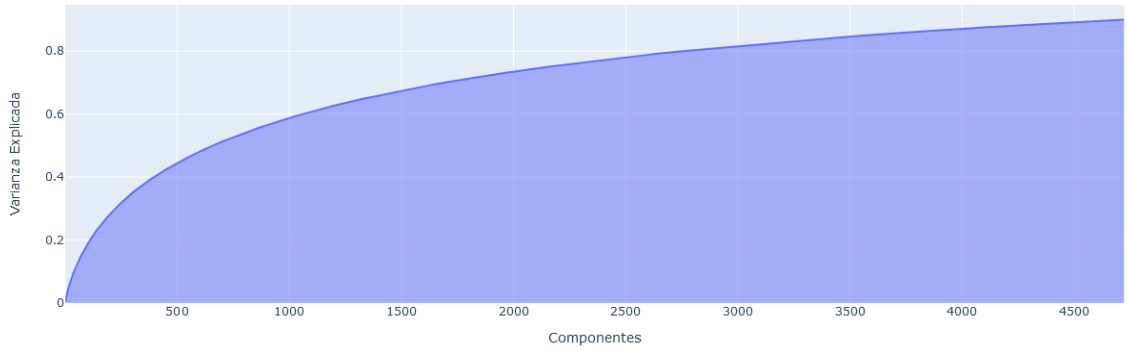


a) CountVectorizer (modelo 2)

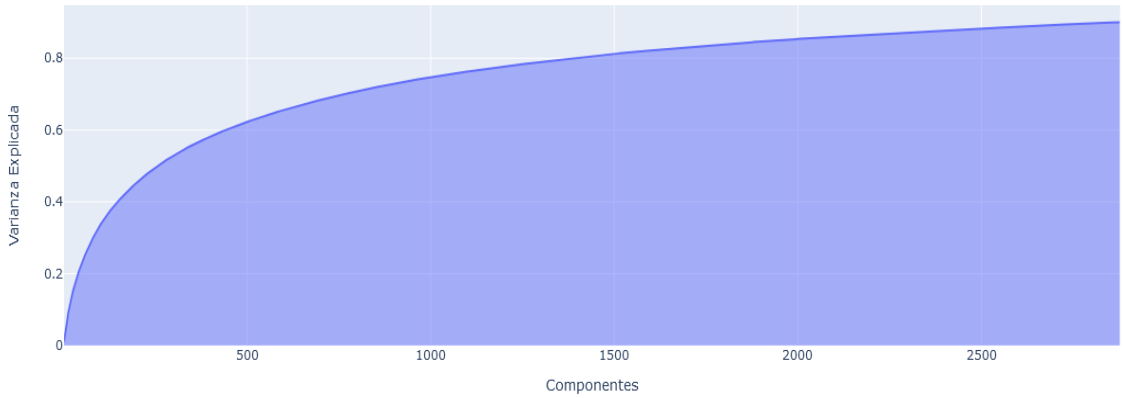


b) TfidftVectorizer (modelo 4)

Figura H.36: Resultado validación cruzada con RFECV.



a) CountVectorizer (modelo 2)



b) TfidftVectorizer (modelo 4)

Figura H.37: Varianza explicada con PCA.

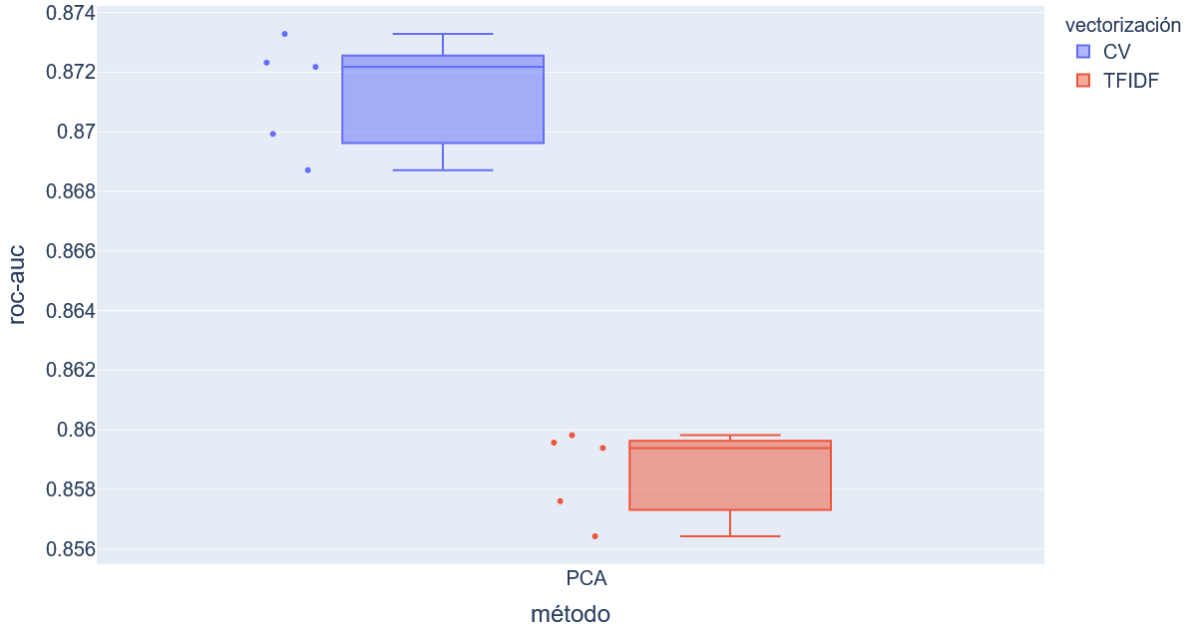


Figura H.38: Resultado validación cruzada con PCA.

Tabla H.33: Configuración de la rejilla inicial.

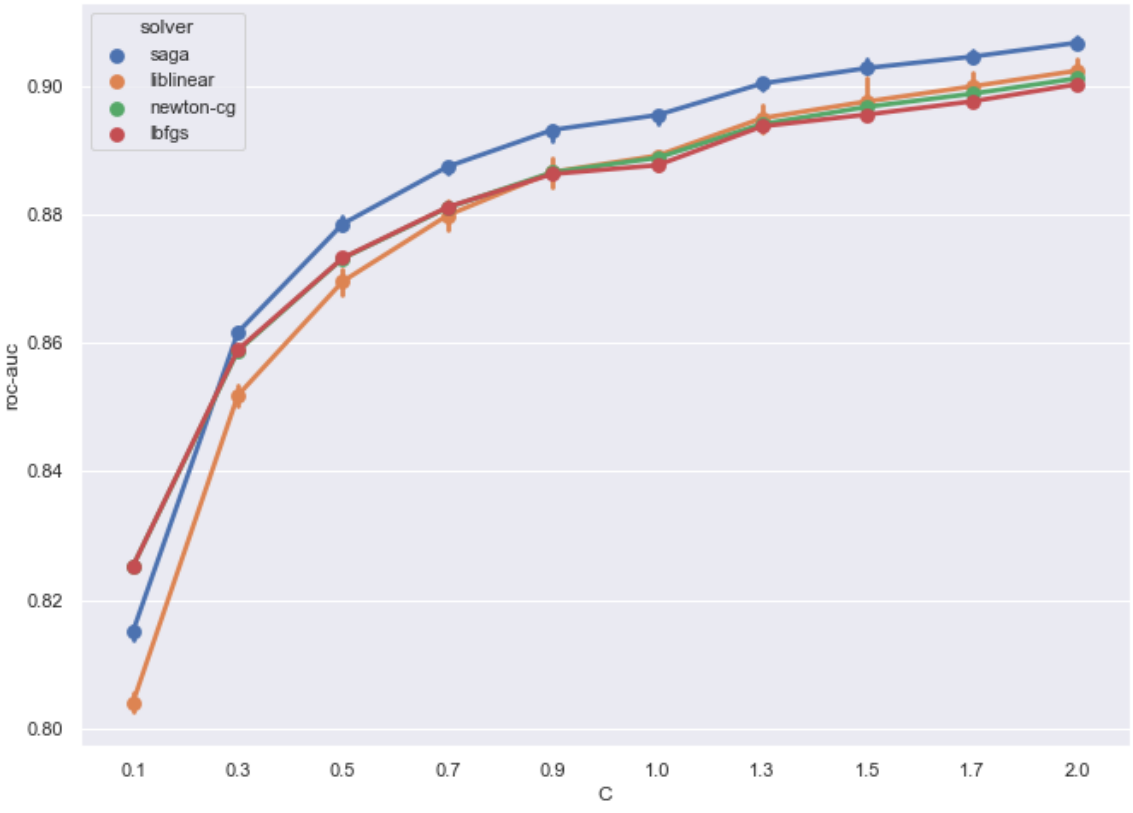
Modelo	Función	Rejilla
Regresión Logística	*GridSearchCV*	'C': [0.1, 0.3, 0.5, 0.7, 0.9, 1.0, 1.3, 1.5, 1.7, 2.0], 'penalty': ['l2', 'l1','elasticnet'], 'solver': ['liblinear','lbfgs','newton-cg','saga'],' l1_ratio': [0.1, 0.3, 0.5, 0.7,0.9]
KNN	*GridSearchCV*	'n_neighbors': [1,2,3,4,5,6,7,8], 'weights': ['uniform'] (por defecto),, 'metric': ['euclidean','minkowski', 'cosine']
MultinomialNB	*GridSearchCV*	'alpha': [10,5,1, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0]
Random Forest	*RandomizedSearchCV*	'bootstrap': True, 'max_depth': [10, 20, 40, 60, 80, 100] 'max_features': ['auto', 'log2'], 'min_samples_leaf': [10, 20, 40, 60] 'min_samples_split': [2, 5, 10,15,100], 'n_estimators': [100,200, 400, 600]
XGBoost	*RandomizedSearchCV*	'learning_rate': [0.1,0.05,0.03], 'min_child_weight': [5,10,20], 'gamma': [0], 'subsample': [1.0] (por defecto), 'colsample_bytree': [1.0] (por defecto), 'max_depth': [6] (por defecto), 'n_estimators': [100,300,500,1000]
SVM (LIBLINEAR)	*GridSearchCV* (nº iteraciones máximas = 10000)	'C': [1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100], 'loss': ['squared_hinge',' hinge'] 'penalty': ['l2']

SVM (LIBSVM)	*GridSearchCV* (nº iteraciones máximas = 10000)	params_svm_lineal = {
		'kernel': ['linear'],
		'C': [1e-4, 1e-3, 1e-2, 1e-1, 1]
		}
		params_svm_poly = {
		'kernel': ['poly'],
		'C': [1e-4, 1e-3, 1e-2, 1e-1, 1],
		'gamma': [1, 0.1, 0.01],
		'degree': [2]
		}
		params_svm_rbf {
		'kernel': ['rbf'],
		'gamma': [1, 0.1, 0.01]
		}

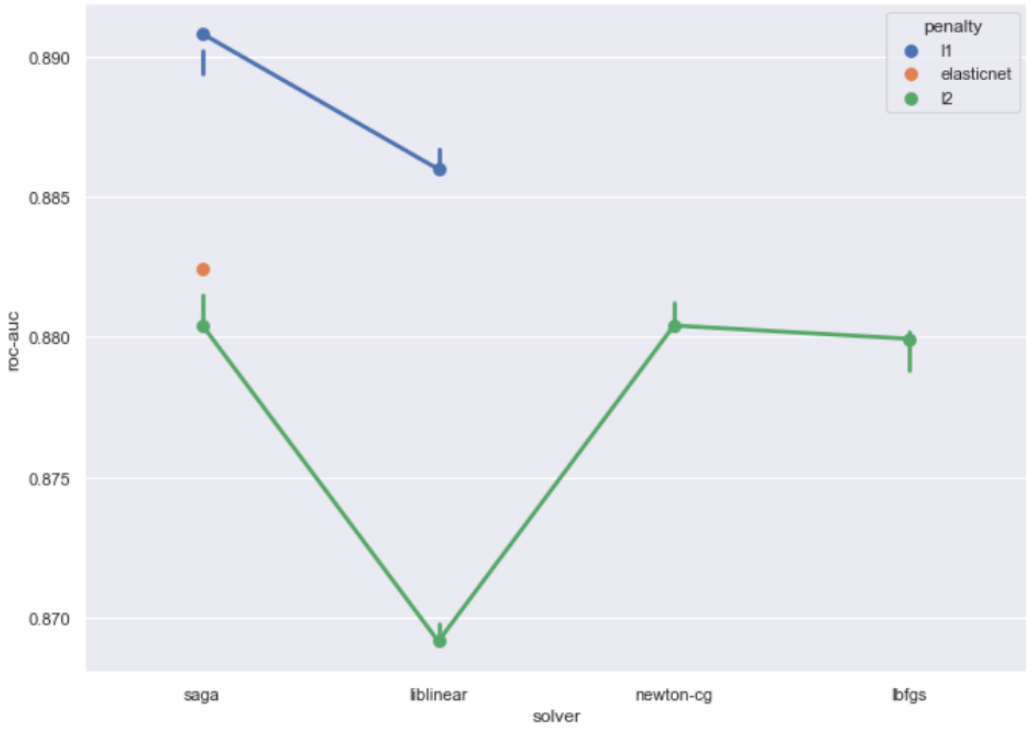
ANN	*RandomizedSearchCV*	activation': ['relu', 'tanh'], 'epochs': [10,20,30], 'batch_size': [32,64], 'optimizer: ['adam'], 'neurons': [2,4,6], 'weight_constraint': [1], 'dropout_rate': [0.0, 0.1], 'learn_rate': [0.001, 0.01, 0.1], 'init_mode': ['uniform']

Tabla H.34: Resultados de la rejilla con RL y validación cruzada (top-10).

Modelo	*C*	*penalty*	*solver*	*l1_ratio*	ROC-AUC (media validación)
LR_1	2.0	l1	saga	0.7	0.913150
LR_2	2.0	l1	liblinear	0.9	0.911964
LR_3	2.0	l1	liblinear	0.7	0.911957
LR_4	1.7	l1	saga	0.5	0.911343
LR_5	1.7	l1	saga	0.1	0.911336
LR_6	2.0	elasticnet	saga	0.9	0.910607
LR_7	1.5	l1	saga	0.5	0.910114
LR_8	1.7	l1	liblinear	0.1	0.910050
LR_9	1.7	l1	liblinear	0.5	0.910043
LR_10	1.7	elasticnet	saga	0.9	0.908771



a) Comparativa entre el parámetro *solver* y *C*.



b) Comparativa entre el parámetro *penalty* y *solver*.

Figura H.39: Estudio de los principales parámetros después de ejecutar la rejilla inicial con *GridSearchCV* (RL).

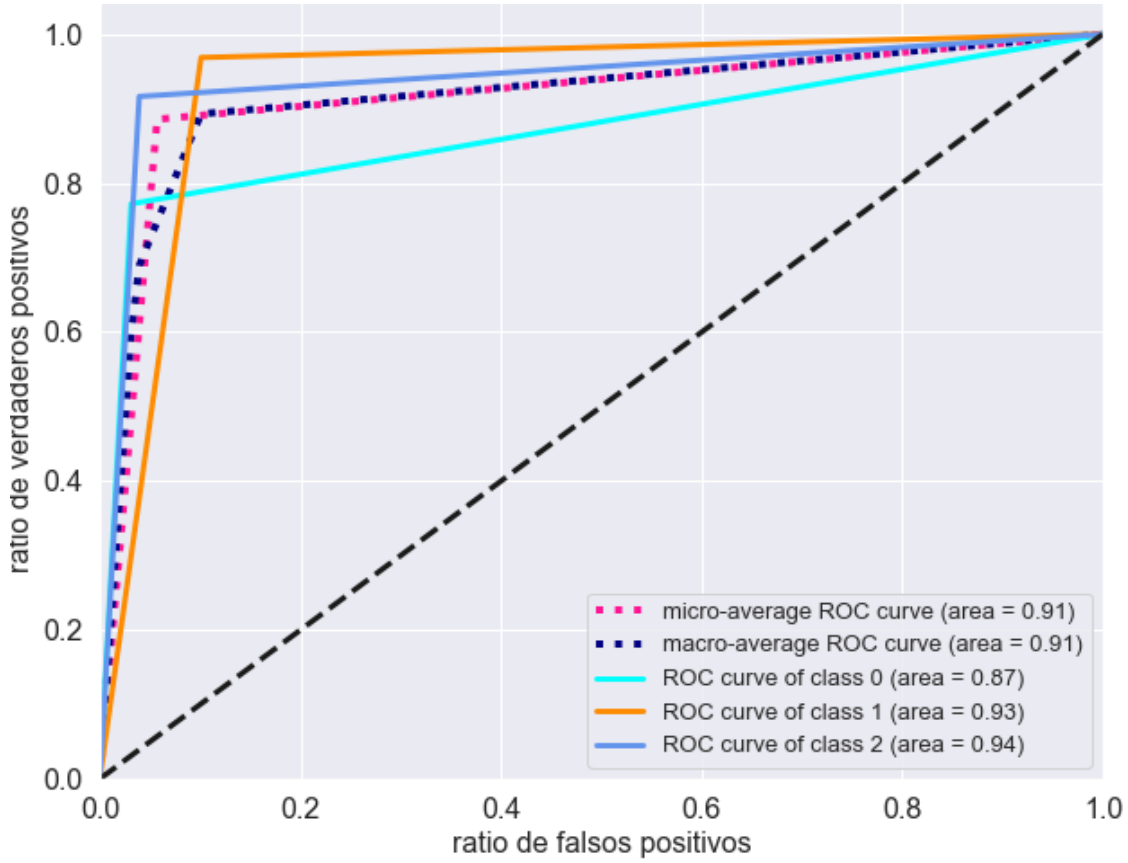


Figura H.40: Curva ROC-AUC del mejor modelo (RL).

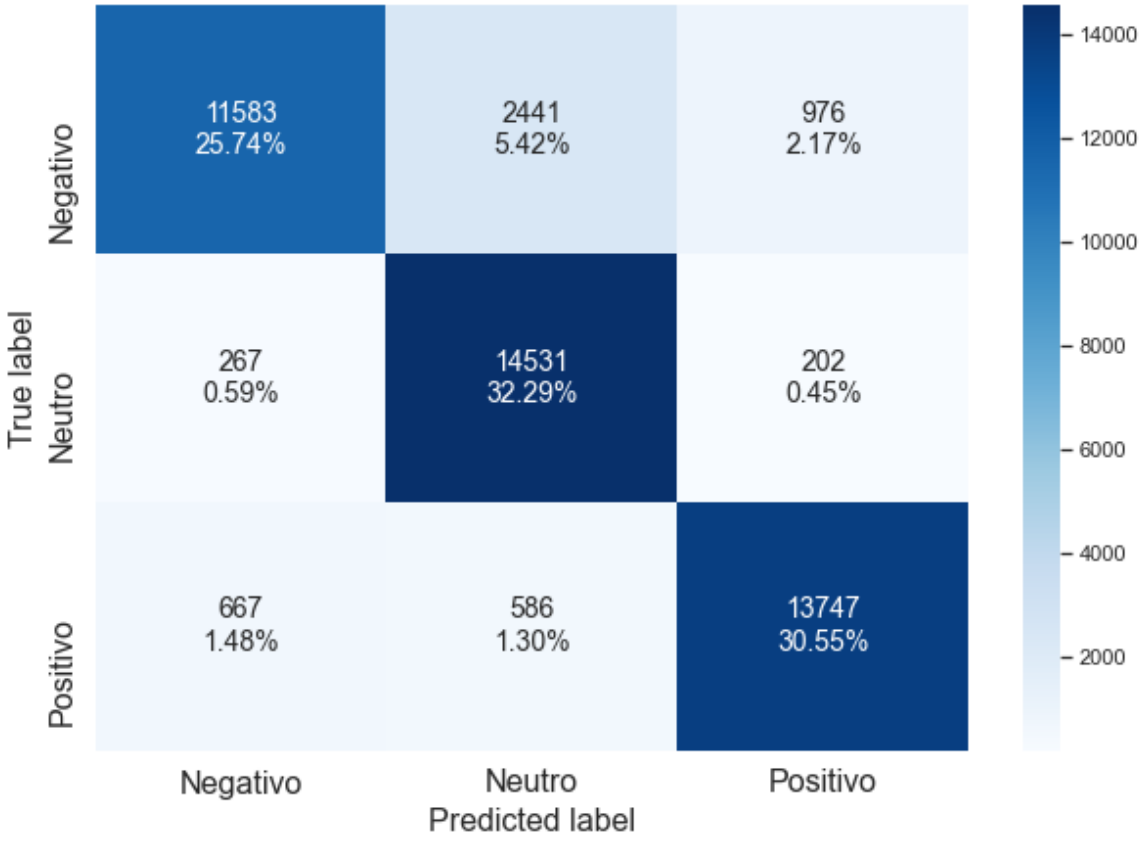


Figura H.41: Matriz de confusión test del mejor modelo (RL).

Tabla H.35: Resultados de la rejilla con MultinomialNB y validación cruzada (top-5).

Modelo	***alpha***	**ROC-AUC (media validación)**
NB_1	10	0.661964
NB _2	5	0. 661943
NB _3	0.01	0. 654093
NB _4	0.1	0. 654700
NB _5	0.0001	0. 653586

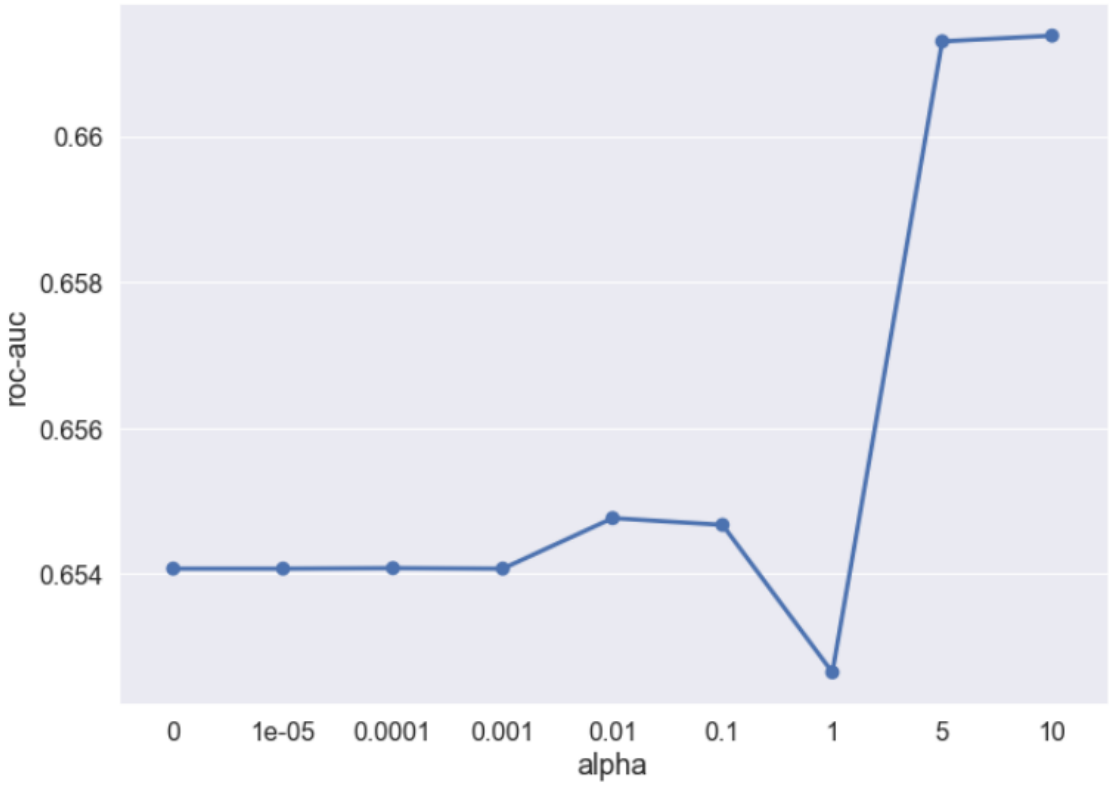


Figura H.42: Estudio del parámetro *alpha* del mejor modelo (NB).

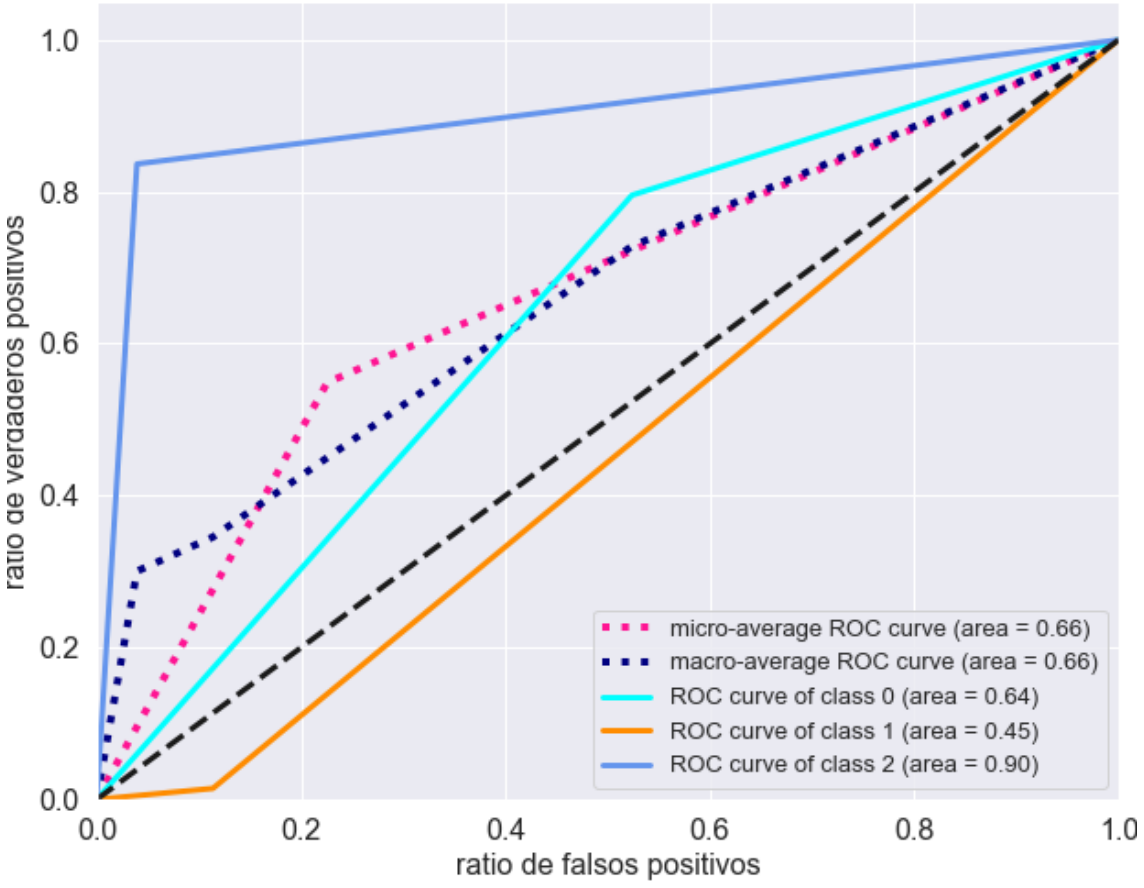


Figura H.43: Curva ROC-AUC del mejor modelo (NB).

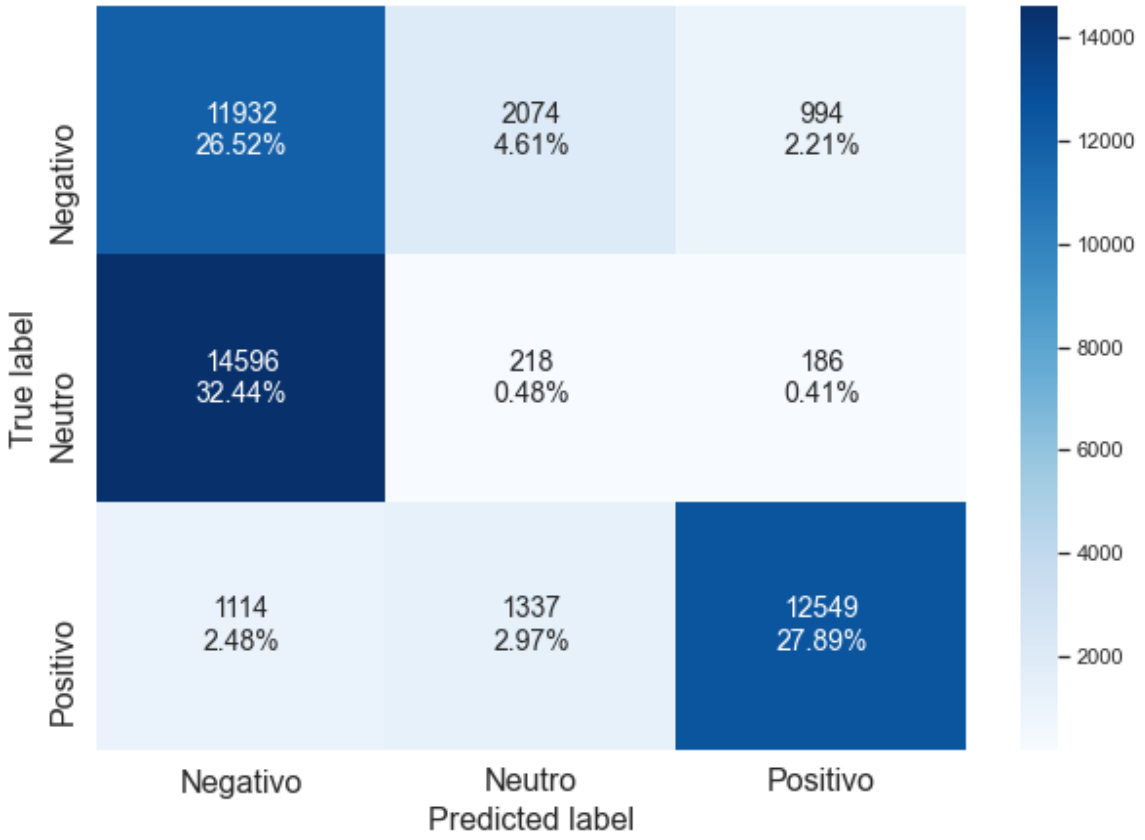


Figura H.44: Matriz de confusión del test del mejor modelo (NB).

Tabla H.36: Resultados de la rejilla con KNN y validación cruzada (top-4).

Modelo	Nº de vecinos	Métrica (distancia)	ROC-AUC (media validación)
KNN_1	4	cosine	0.843086
KNN_2	5	cosine	0.884436
KNN_3	4	minkowski	0.881093
KNN_4	5	minkowski	0.883057

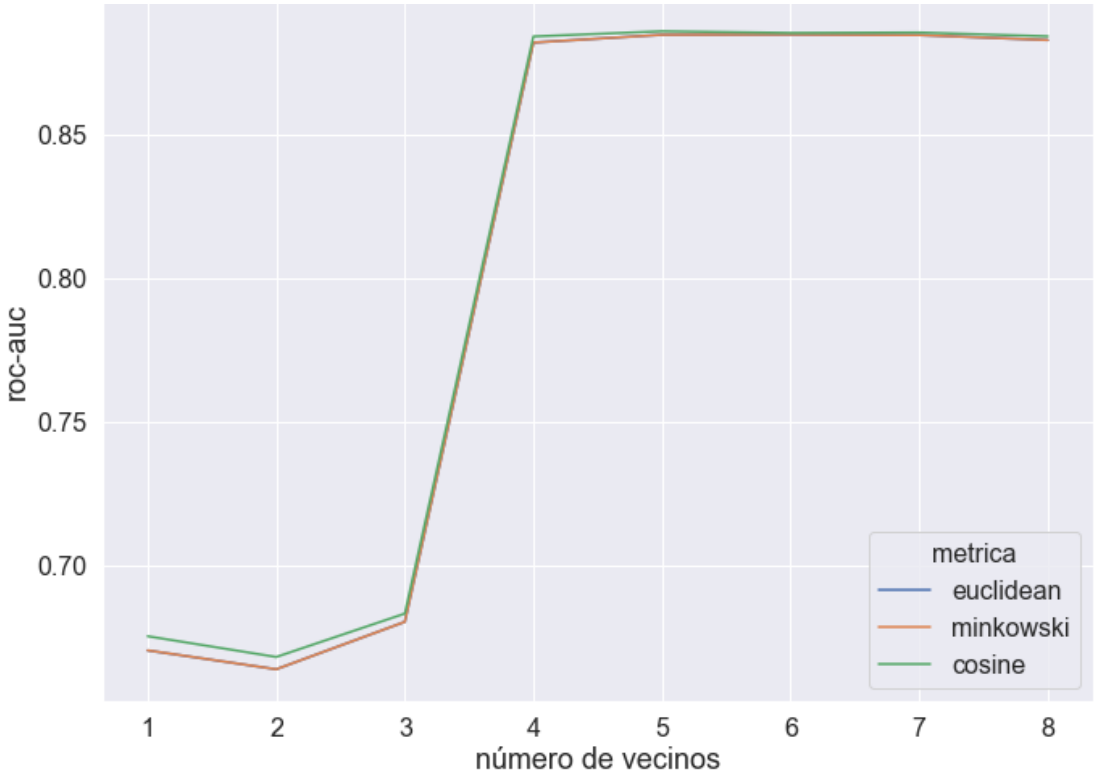


Figura H.45: Estudio del parámetro *k* (número de vecinos) del mejor modelo (KNN).

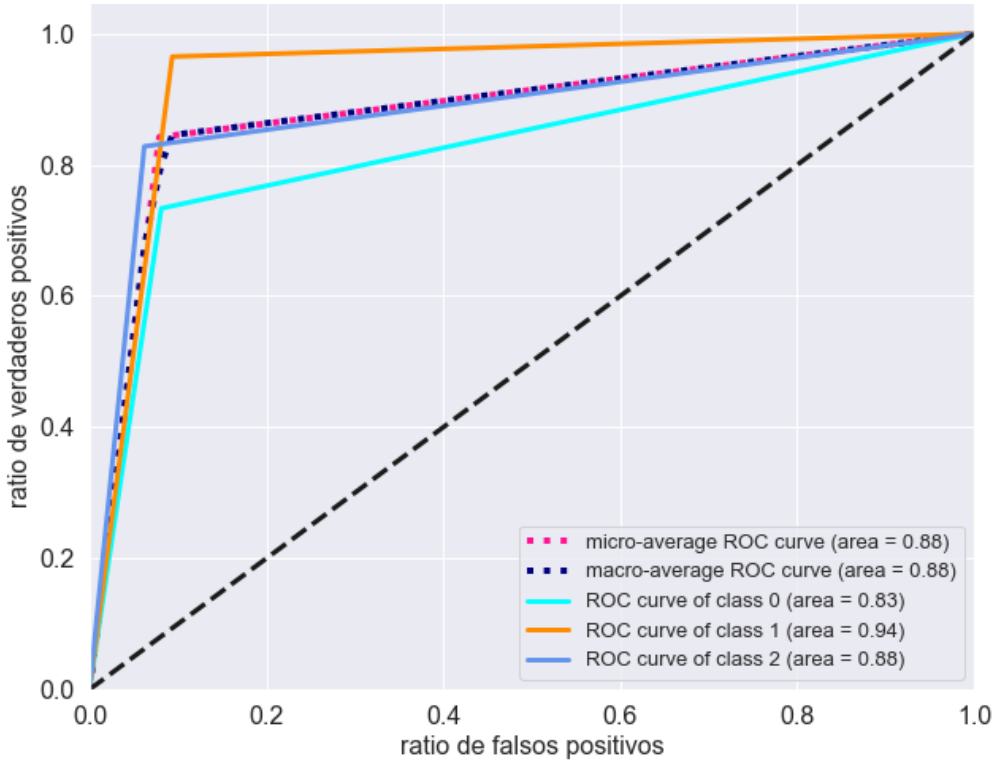


Figura H.46: Curva ROC-AUC del mejor modelo (KNN).

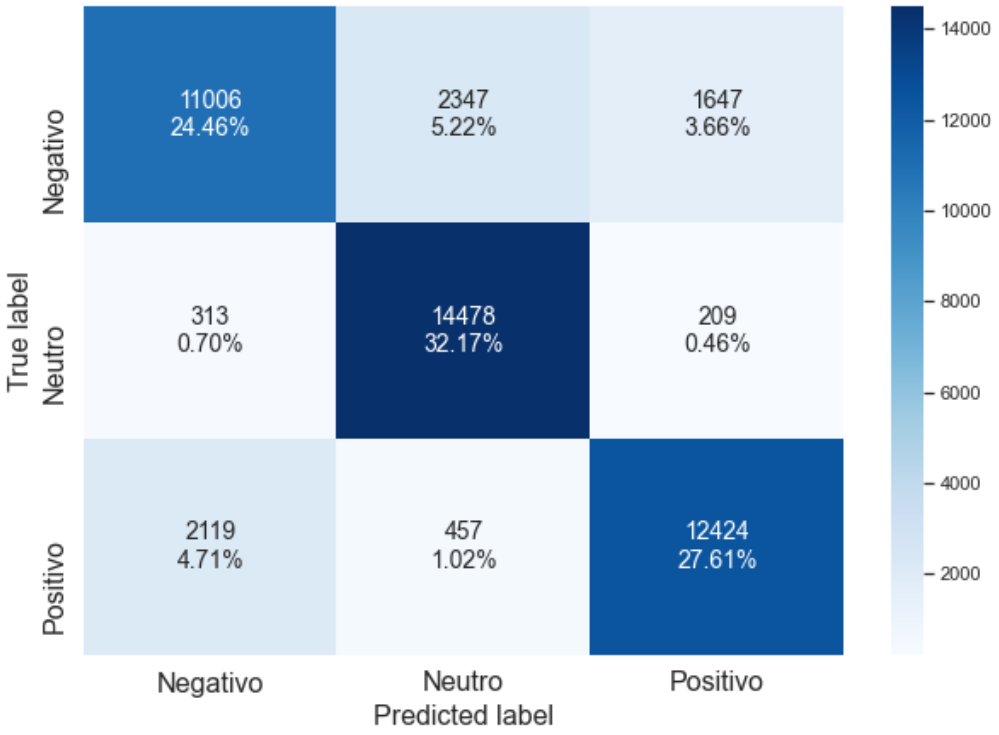
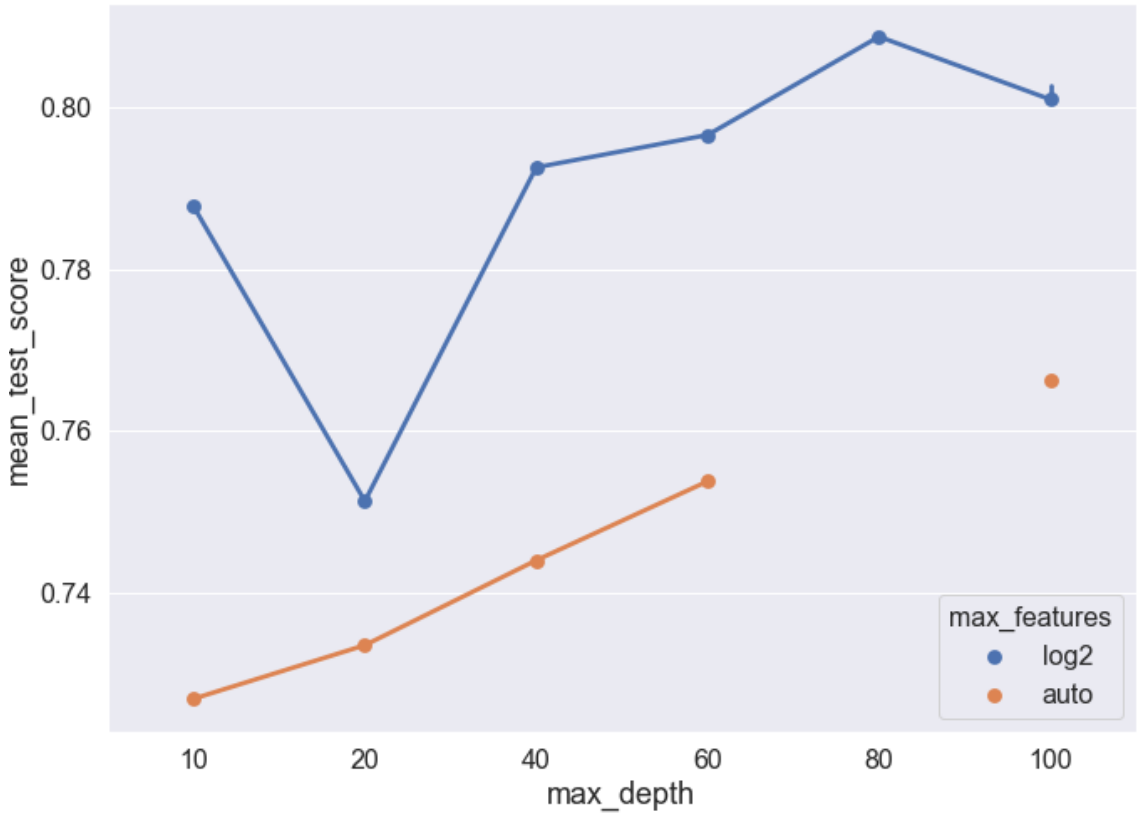
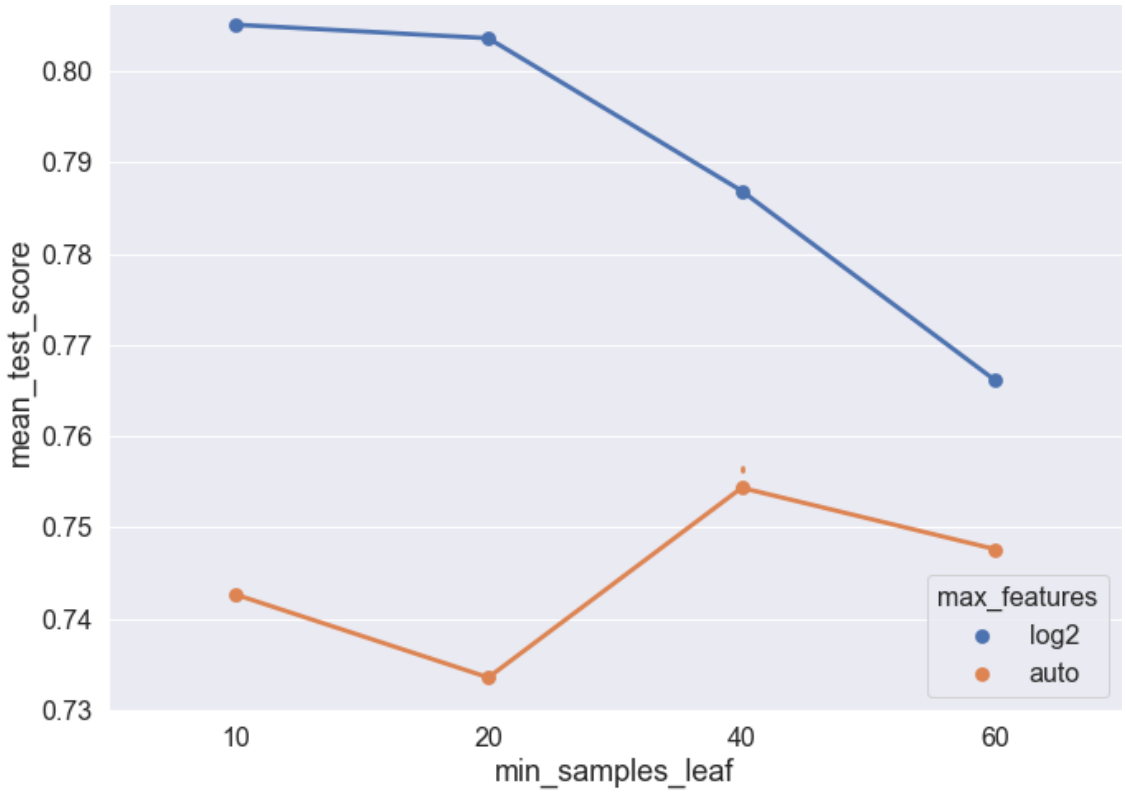


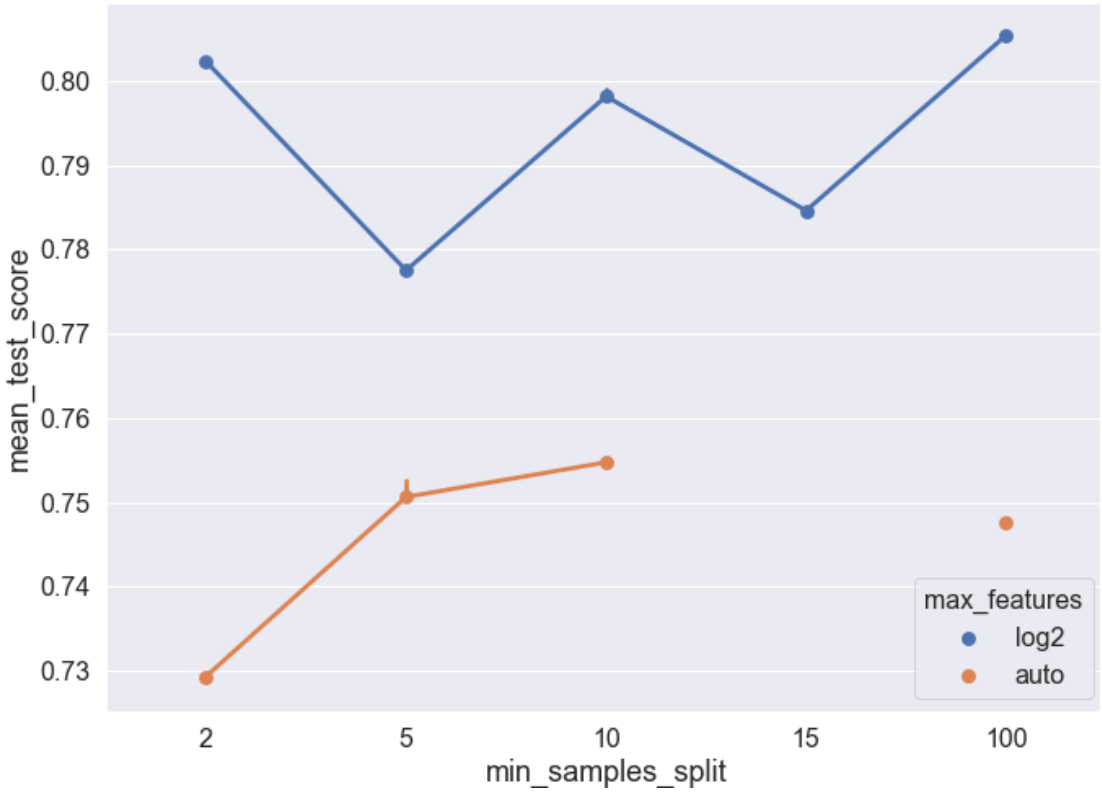
Figura H.47: Matriz de confusión del test del mejor modelo (KNN).



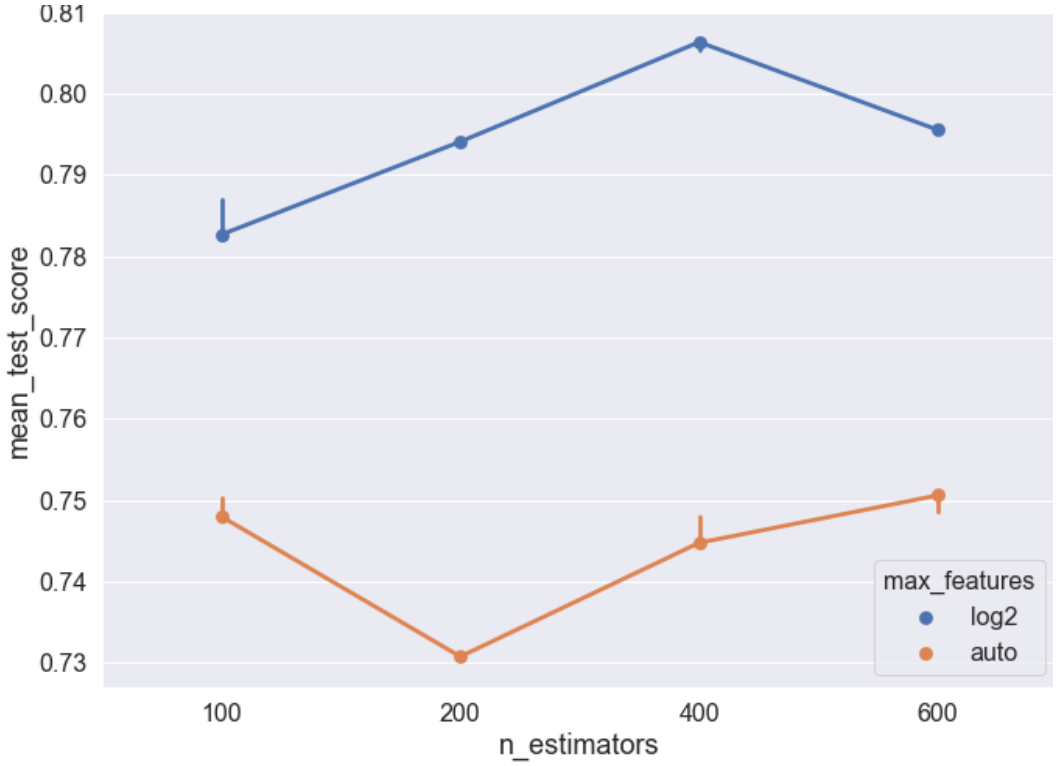
a) Estudio del parámetro *max_depth* agrupado por el parámetro *max_features* (RF).



b) Estudio del parámetro *min_samples_leaf* agrupado por el parámetro *max_features* (RF).



c) Estudio del parámetro *min_samples_split* agrupado por el parámetro *max_features* (RF).



d) Estudio del parámetro *n_estimators agrupado* por el parámetro *max_features* (RF).

Figura H.48: Estudio de los principales parámetros después de ejecutar la rejilla inicial con *RandomizeSearchCV* (RF).

Tabla H.37: Configuración segunda rejilla para RF.

Modelo	Librería	Rejilla
Random Forest	*GridSearchCV*	'bootstrap': True, 'max_depth': [80, 100, None] 'max_features': ['log2'], 'min_samples_leaf': [10, 20] 'min_samples_split': [2, 100], 'n_estimators': [400], 'criterion': ['gini', 'entropy']

Tabla H.38: Resultados de la segunda rejilla con RF y validación cruzada (top-5).

Modelo	*criterion*	*max_depth*	*min_samples_split*	ROC-AUC (media validación)
RF_1	*entropy*	100	100	0.812986
RF_2	*gini*	100	100	0.813886
RF_3	*gini*	100	2	0.813800
RF_4	*entropy*	100	2	0.813736
RF_5	*gini*	80	2	0.809657

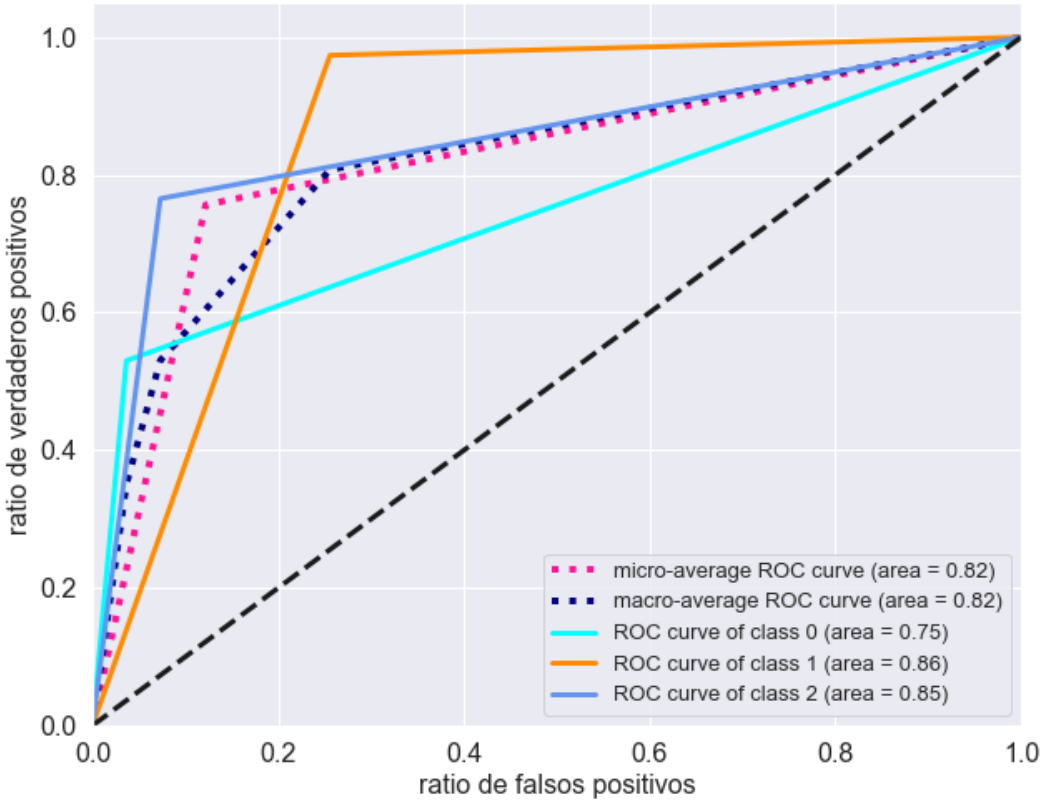


Figura H.49: Curva ROC-AUC del mejor modelo (RF).

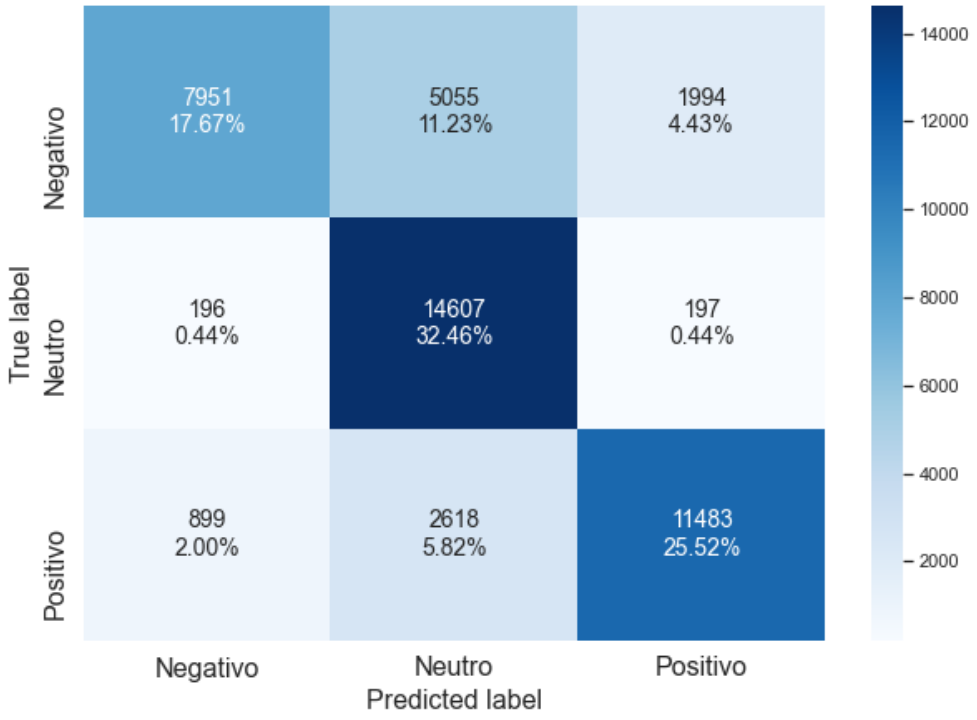
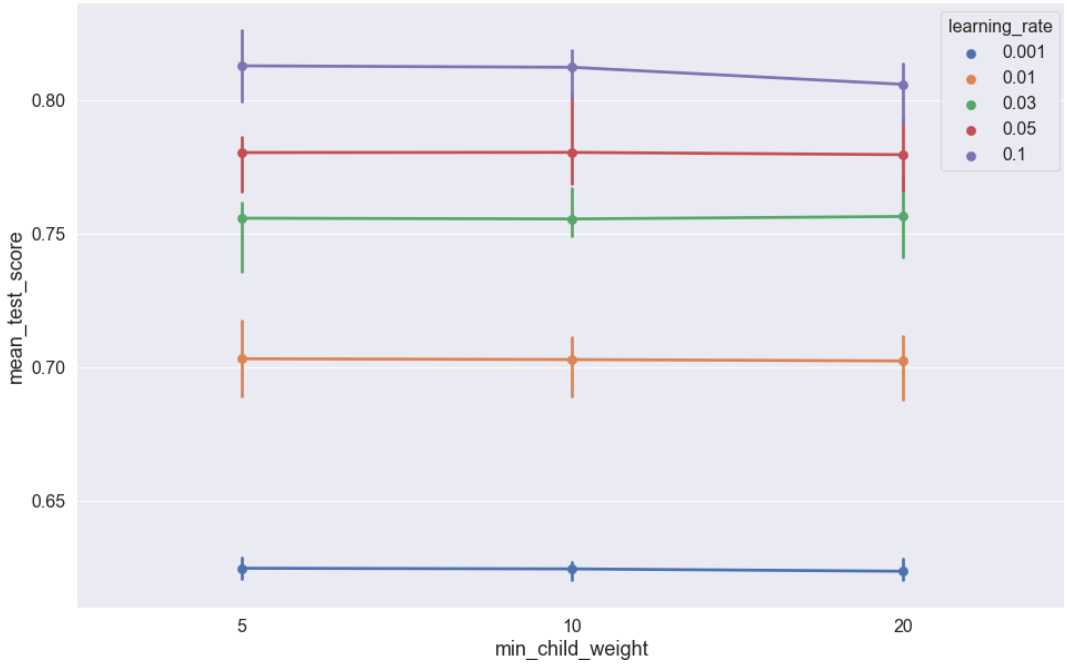
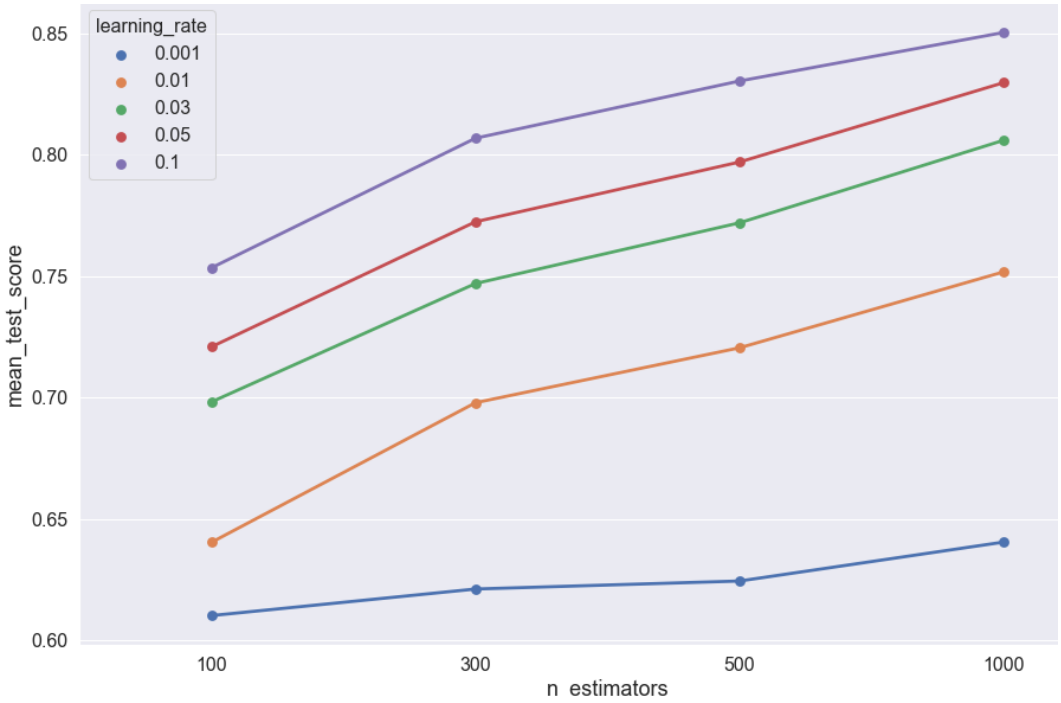


Figura H.50: Matriz de confusión del test del mejor modelo (RF).



a) Estudio del parámetro *min_child_weight* agrupado por el parámetro *learning_rate* (XGBoost).



b) Estudio del parámetro *n_estimators* agrupado por el parámetro *learning_rate* (XGBoost).

Figura H.51: Estudio de los principales parámetros después de ejecutar la rejilla inicial con *GridSearchCV* (XGBoost).

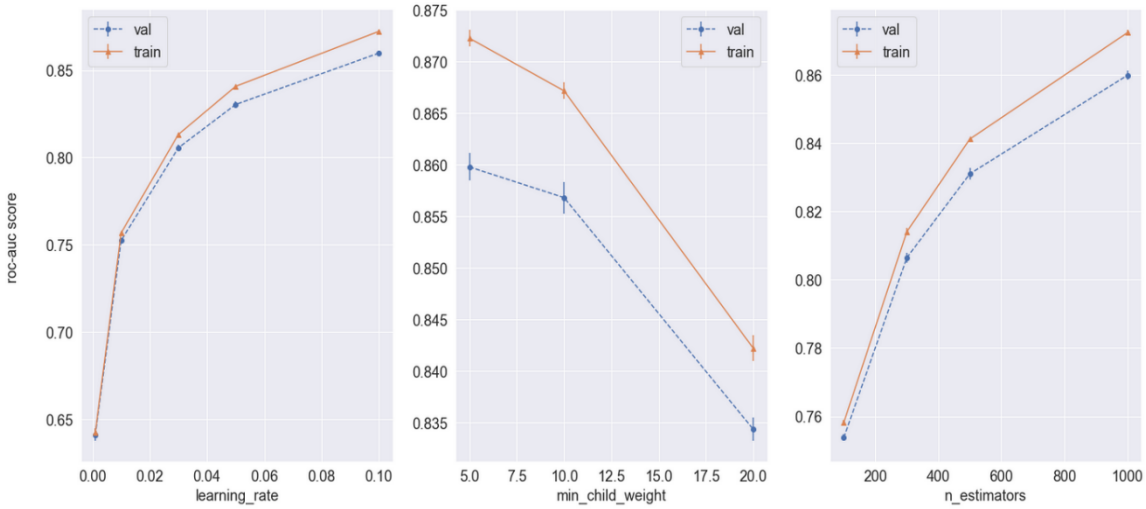


Figura H.52: Comparación *train* vs *val* entre los principales parámetros (XGBoost).

Tabla H.39: Configuración segunda rejilla para XGBoost.

Modelo	Librería	Rejilla
XGBoost	*GridSearchCV*	'learning_rate': [0.1], 'min_child_weight': [5] 'gamma': [0, 0.001, 0.01, 0.05, 0.1, 0.5, 1], 'subsample': [1.0], 'colsample_bytree': [1.0], 'max_depth': [6], 'n_estimators': [1000]

Tabla H.40: Resultados de la segunda rejilla con XGBoost y validación cruzada (top-5).

Modelo	*learning_rate*	*min_child_weight*	n_estimators	ROC-AUC (media validación)
XGB_1	0.1	5	1000	0.857786
XGB _2	0.1	20	1000	0.835186
XGB _3	0.1	10	500	0.833614
XGB _4	0.05	20	1000	0.829693
XGB _5	0.05	10	500	0.798193
XGB_6	0.03	20	300	0.747071

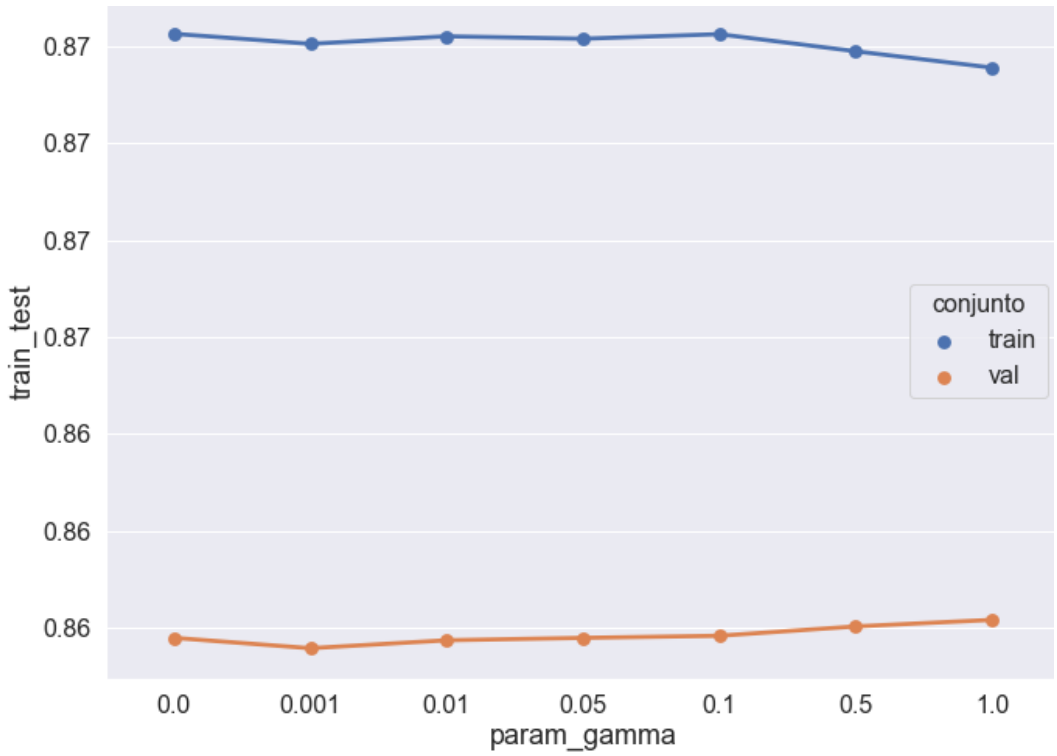


Figura H.53: Estudio del parámetro gamma (XGBoost).

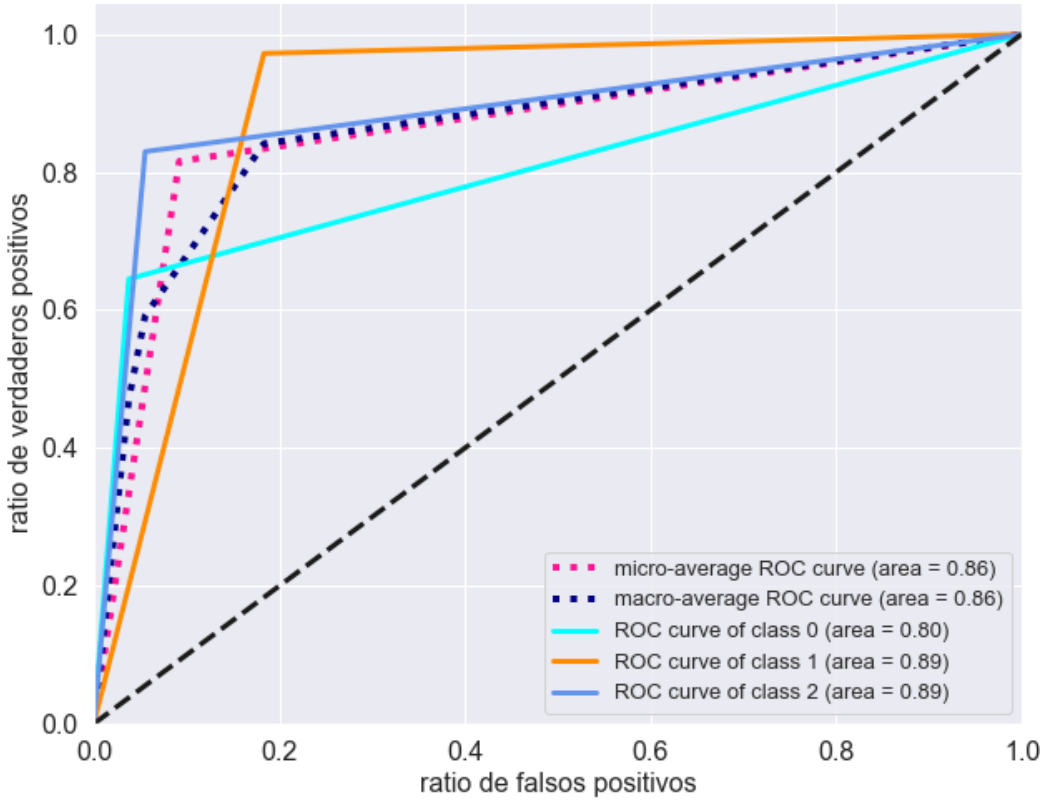


Figura H.54: Curva ROC-AUC del mejor modelo (XGBoost).

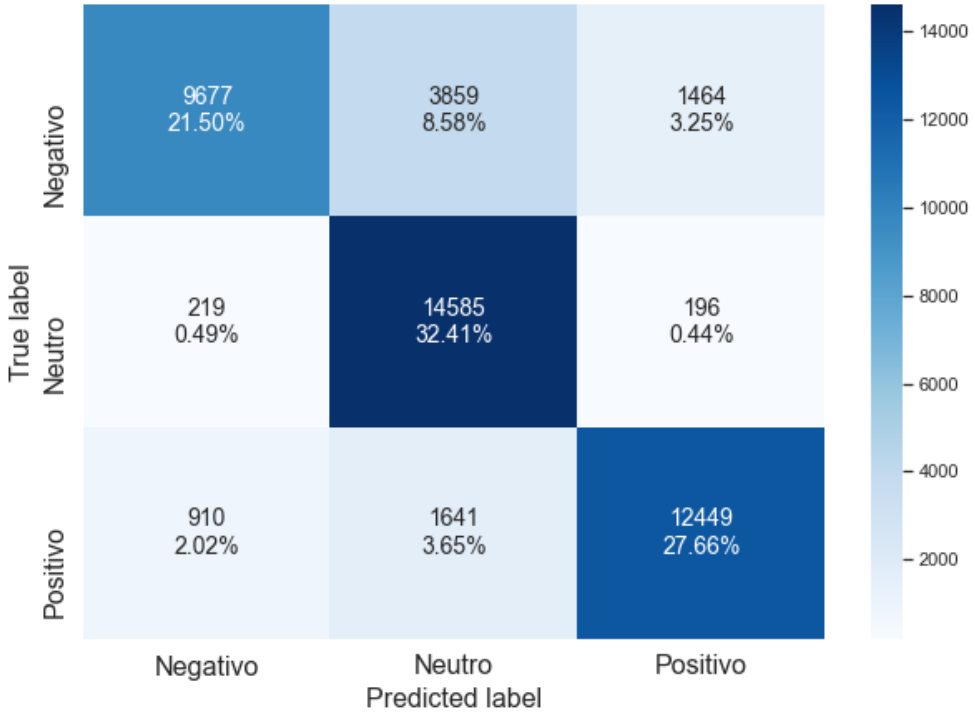
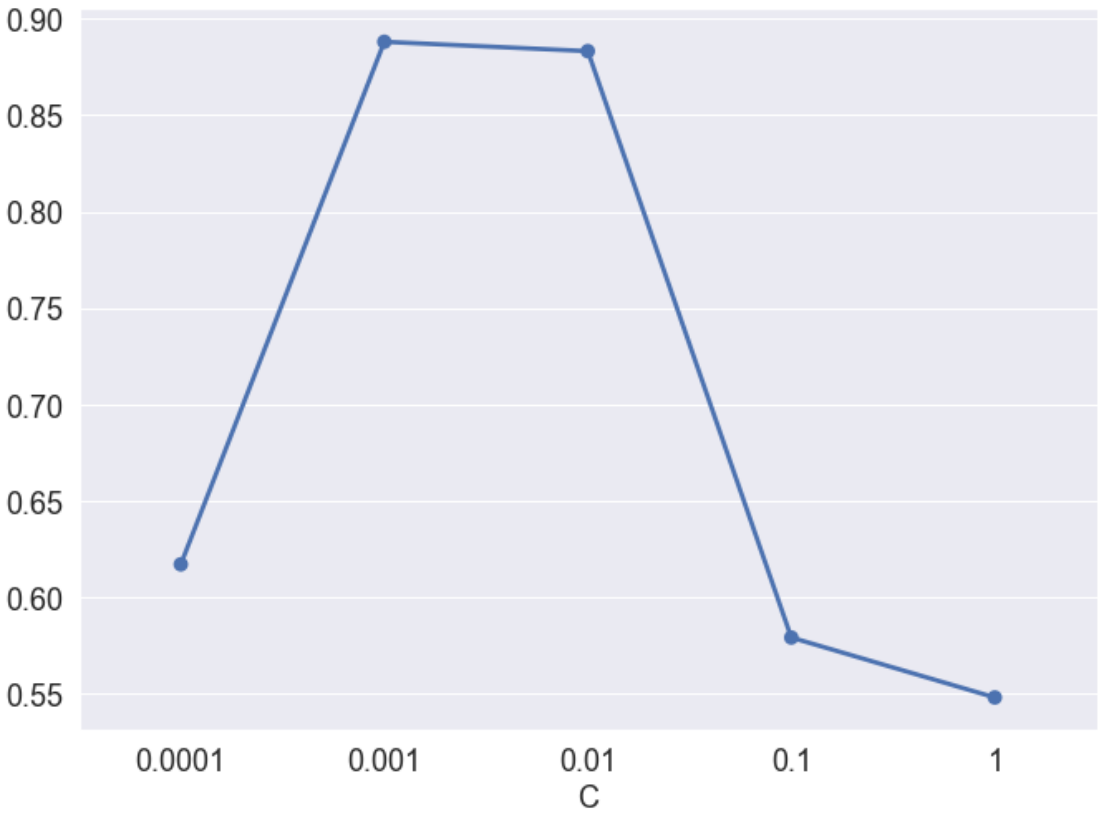
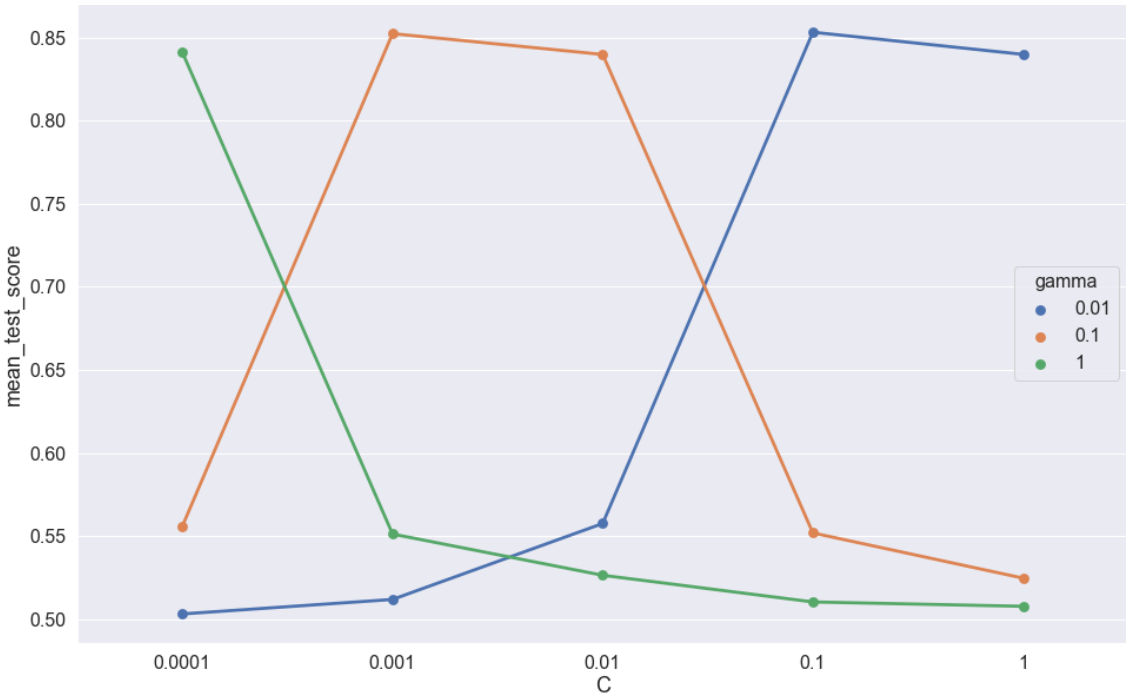


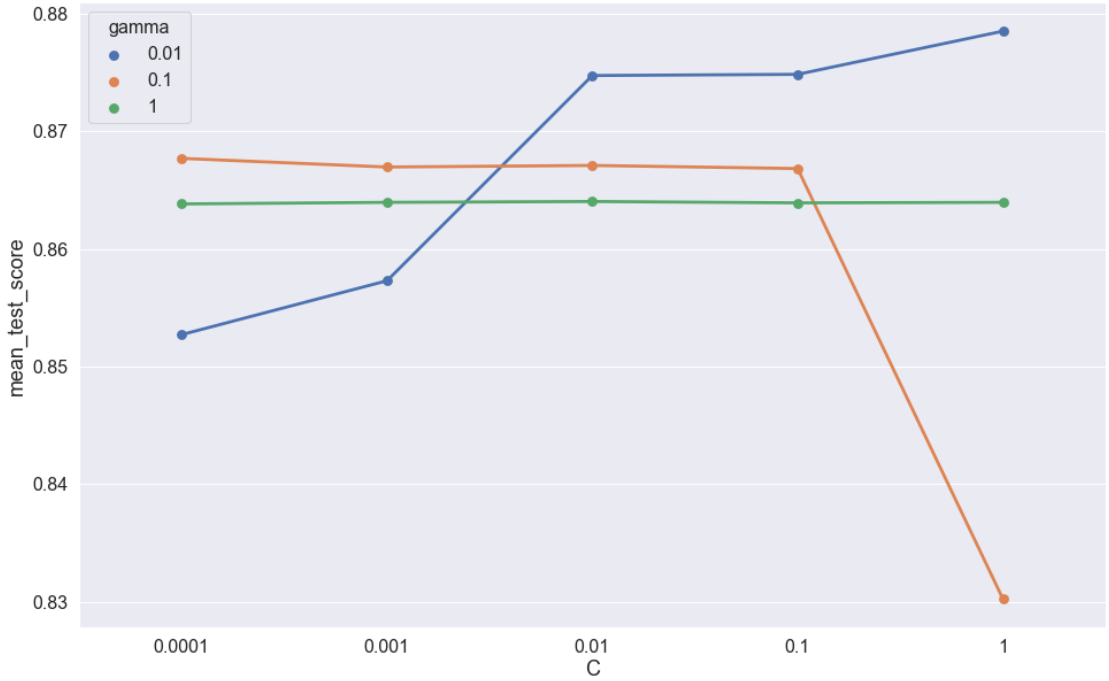
Figura H.55: Matriz de confusión del test del mejor modelo (XGBoost).



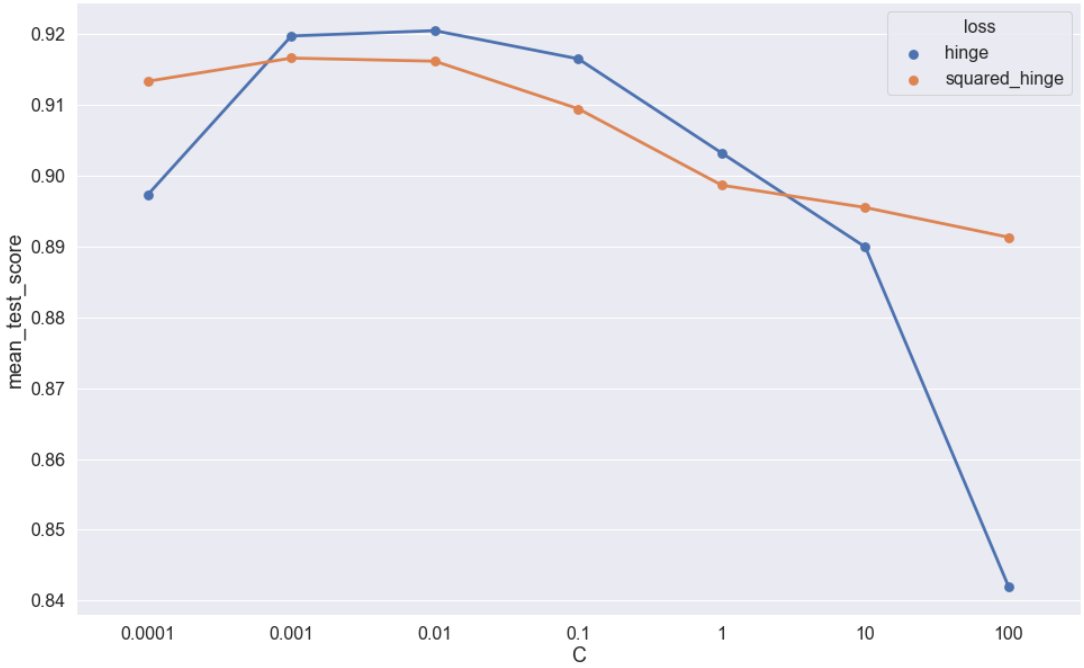
a) Estudio del parámetro *C* con *kernel* lineal (SVM LIBSVM).



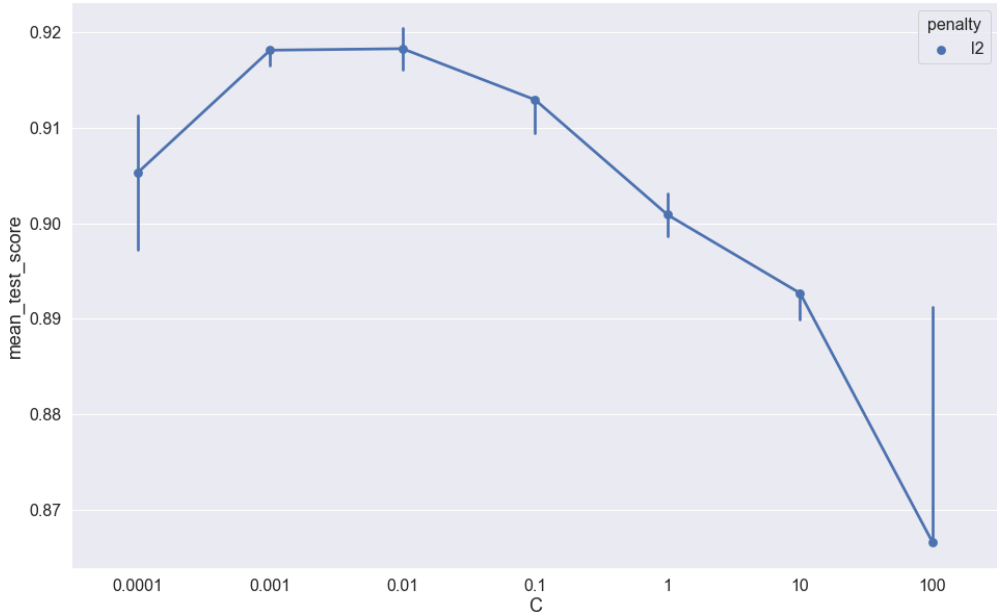
b) Estudio del parámetro *C* y el parámetro *gamma* con *kernel poly* (SVM LIBSVM).



c) Estudio del parámetro *C* y el parámetro *gamma* con *kernel rbf* (SVM LIBSVM).



d) Estudio del parámetro *C* y el parámetro *loss* con (SVM LIBLINEAR).



e) Estudio del parámetro *C* y el parámetro *penalty* con (SVM LIBLINEAR).

Figura H.56: Estudio de los principales parámetros después de ejecutar la rejilla inicial con *GridSearchCV* (SVM).

Tabla H.41: Resultados de la segunda rejilla con SVM y validación cruzada (top-5).

Modelo	***C***	***kernel***	***gamma***	***degree***	***loss***	***penalty***	**ROC-AUC (media validación)**
SVM_Linear	0.001	linear	-	-	-	-	0.887593
SVM_Poly	0.1	poly	0.01	2	-	-	0.860164
SVM_RBF	0.0001	rbf	1	-	-	-	0.874550
LinearSVC	0.001	-	-	-	*squared_hinge*	*l2*	0.920336

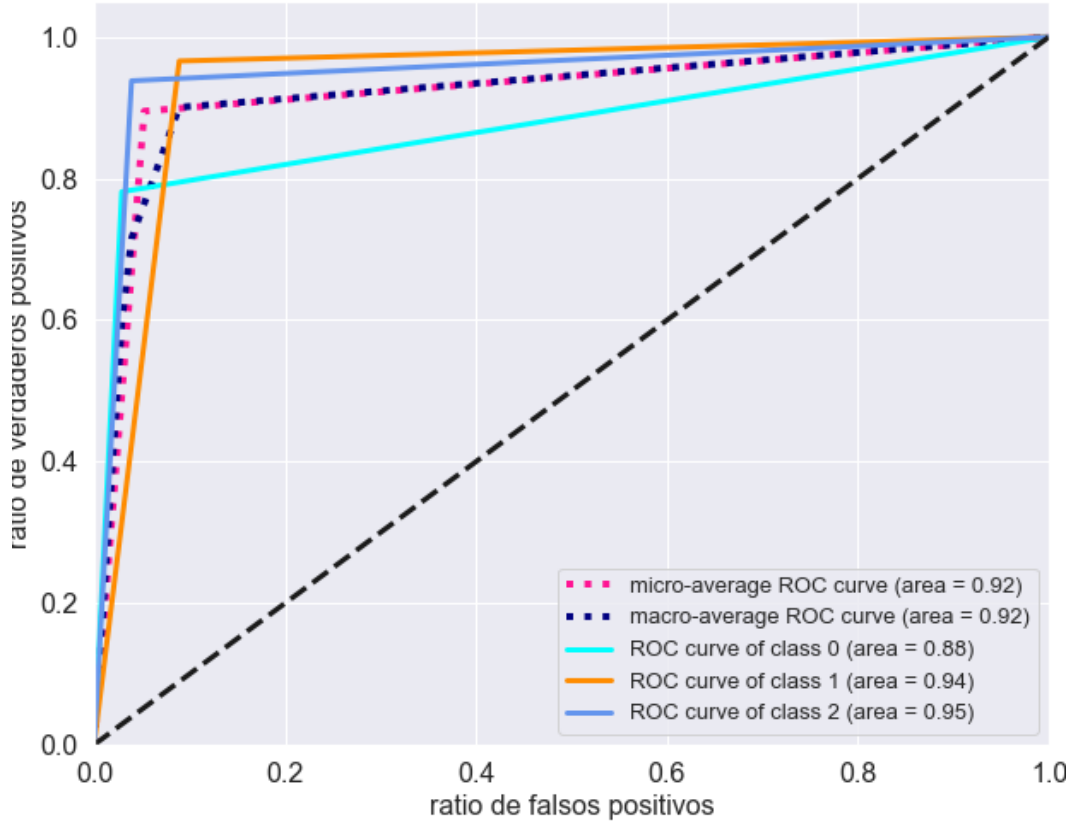


Figura H.57: Curva ROC-AUC del mejor modelo (SVM).

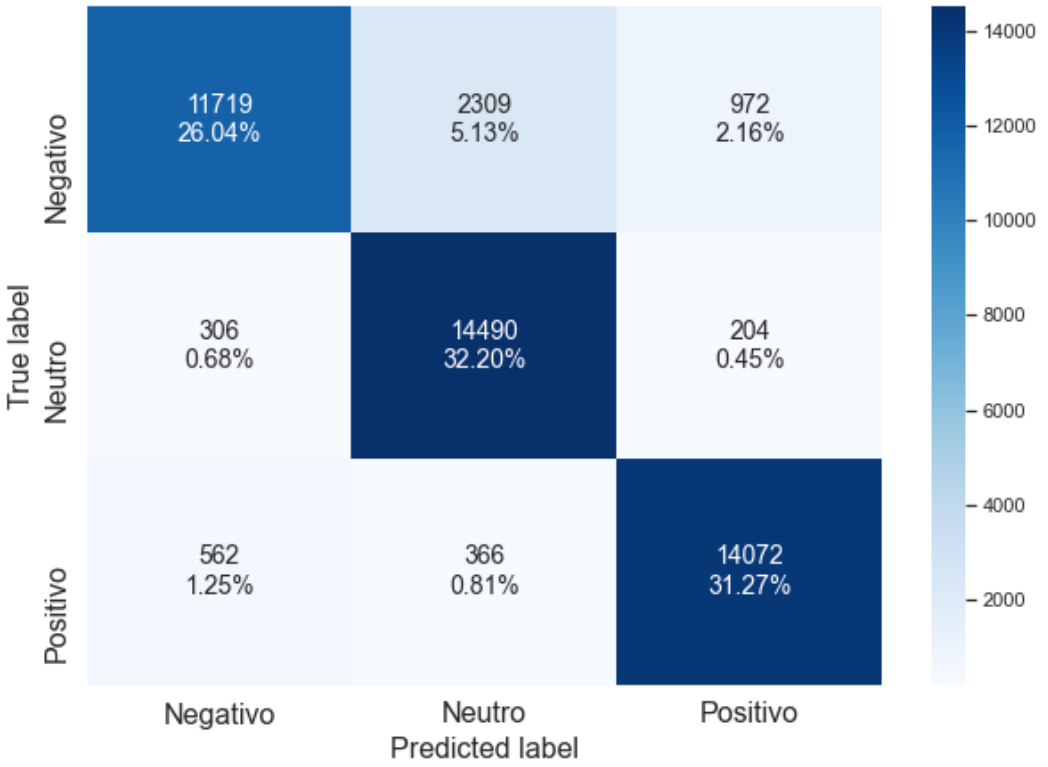
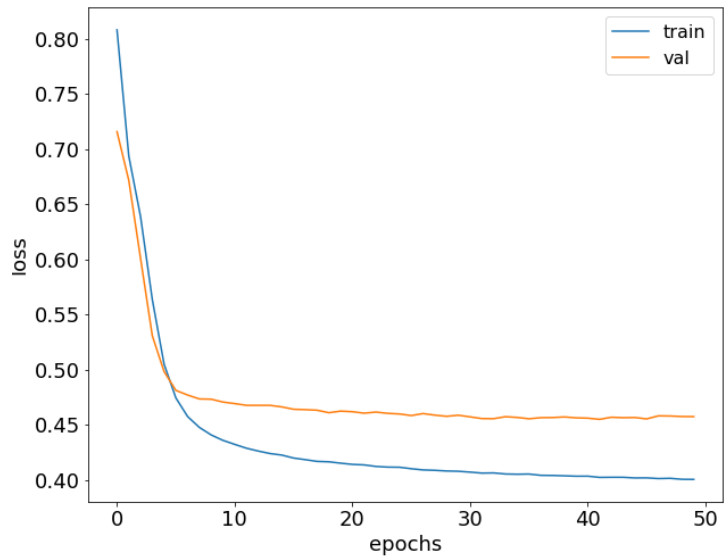


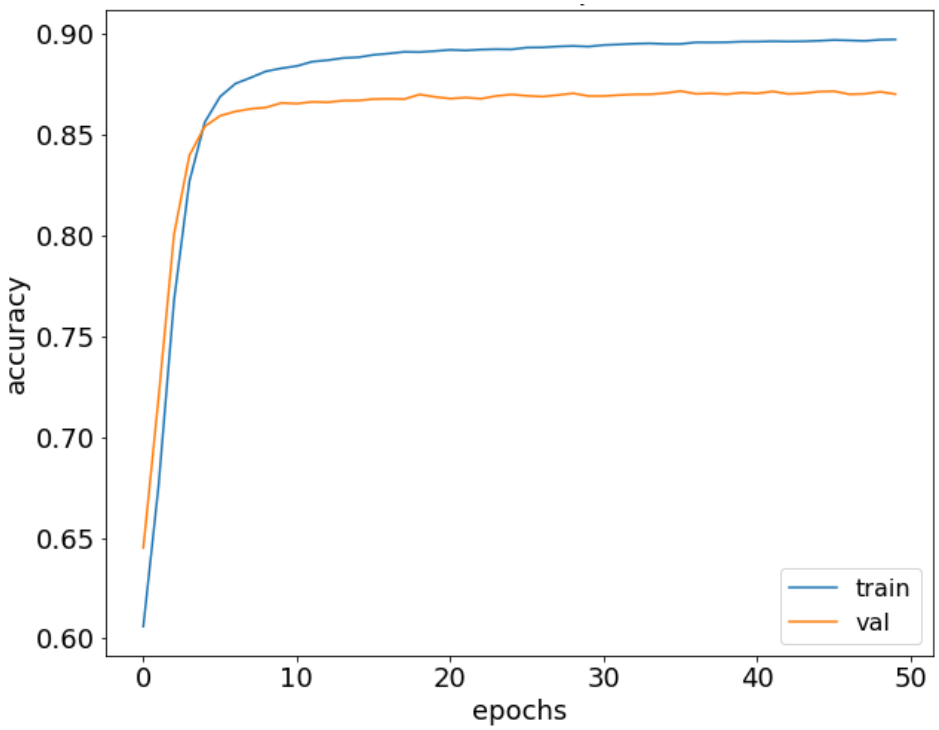
Figura H.58: Matriz de confusión del test del mejor modelo (SVM).

Tabla H.42: Rejilla para el estudio del número de *epochs* (ANN).

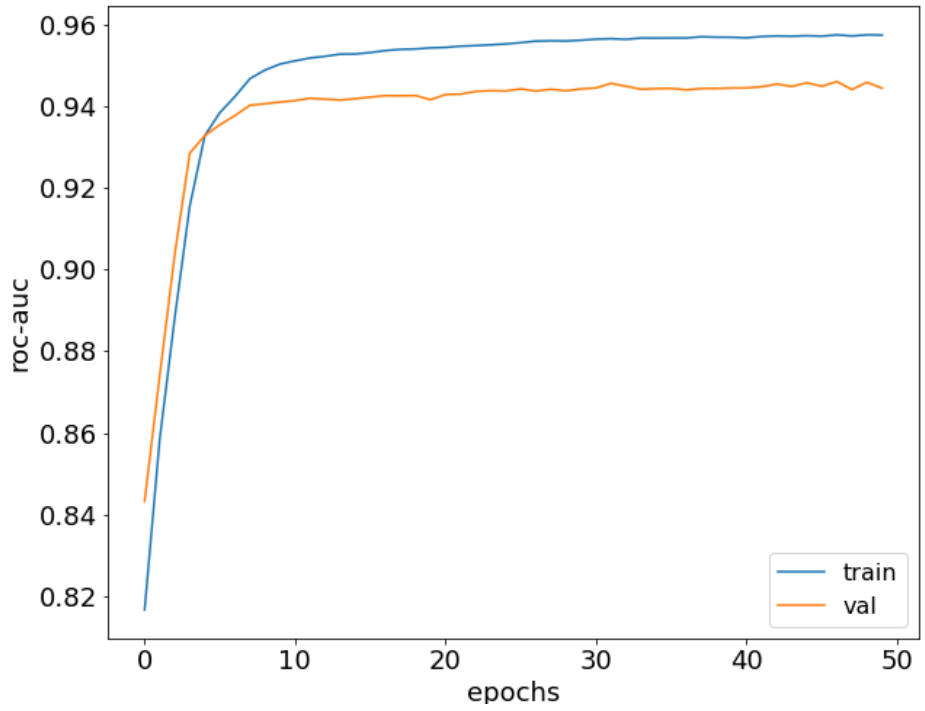
Librería	**epochs**	***loss***	**Función de activación (*activation*)**		***learning rate***
			Capasoc ultas (2)	***Capa de salida***	
Keras	50	*categorical_crossentropy*	tanh	softmax	0.001



a) Resultados del estudio del número de *epochs* (*loss function*) (ANN).

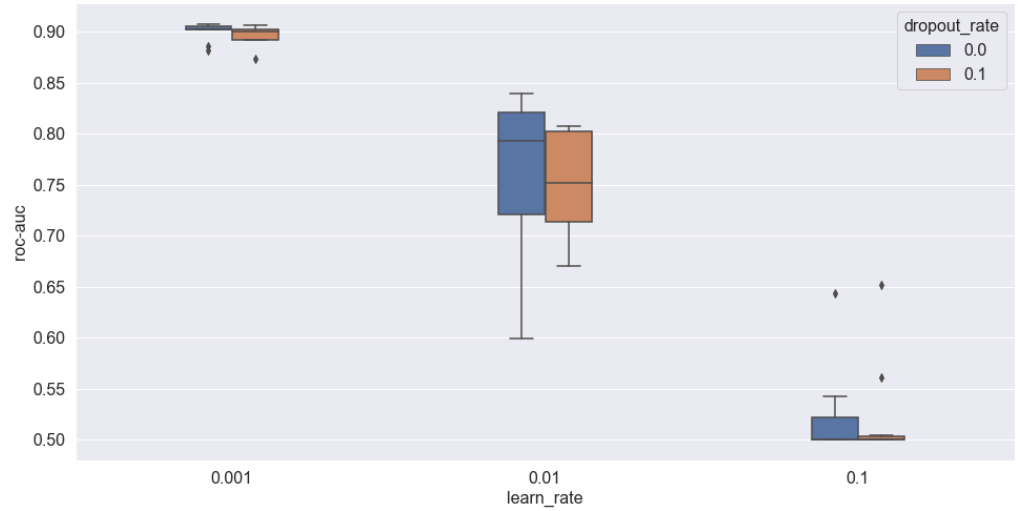


b) Resultados del estudio del número de *epochs* (*accuracy*) (ANN).

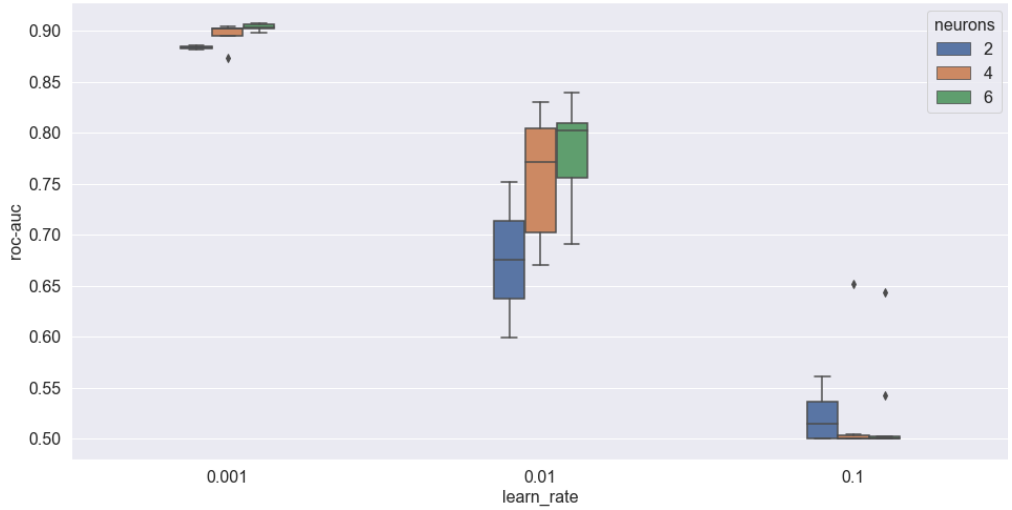


c) Resultados del estudio del número de *epochs (*ROC-AUC) (ANN).

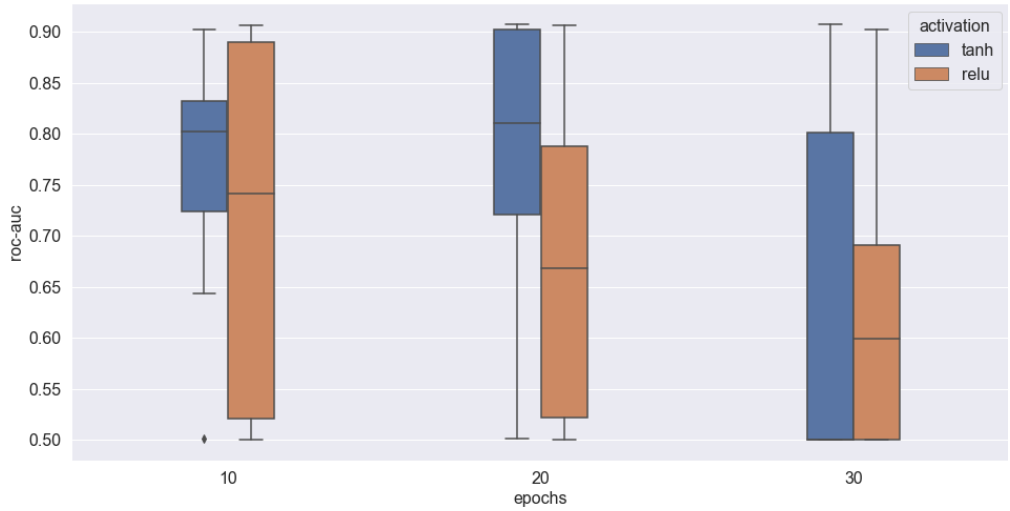
Figura H.59: Estudio del parámetro *epoch*(ANN)



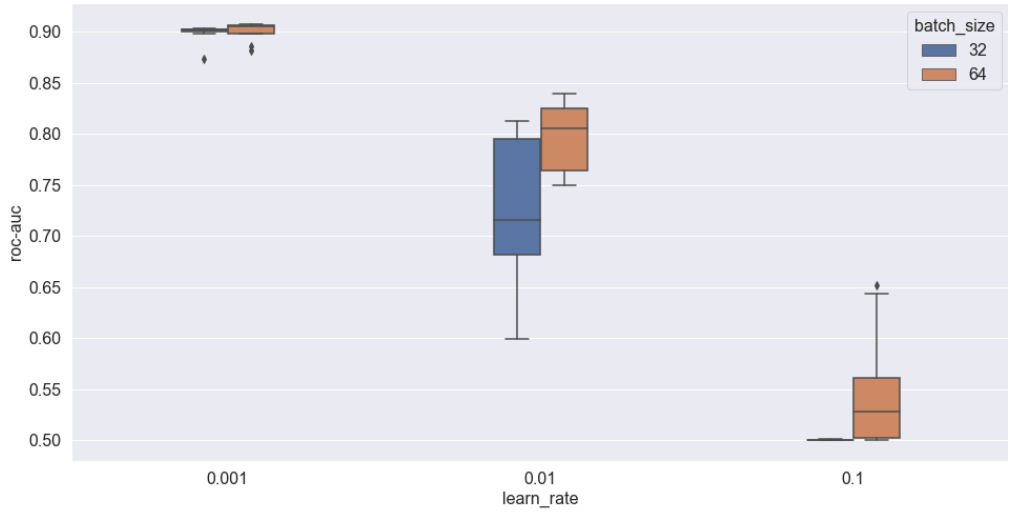
a) Resultados del estudio del *learn_rate* agrupado por *drop_out_rate* (ANN).



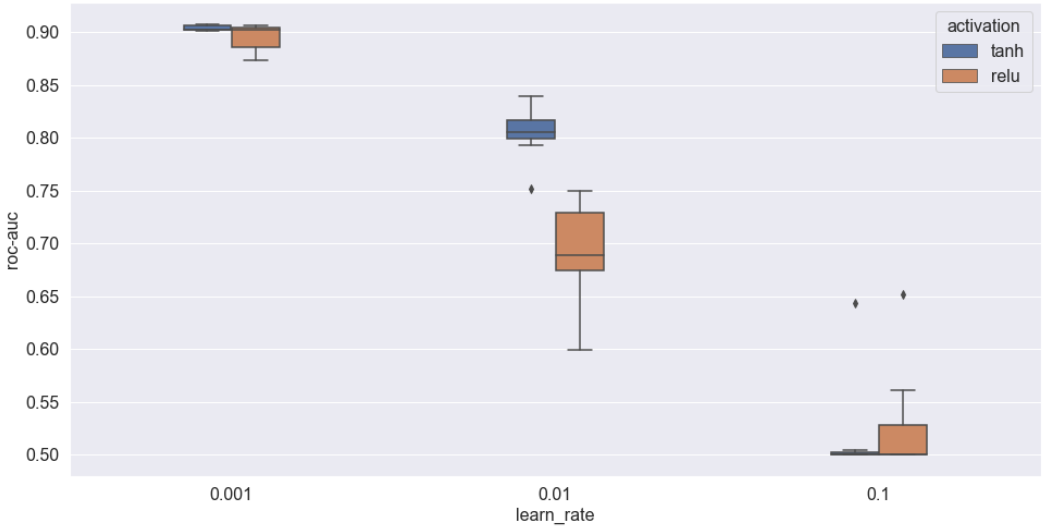
b) Resultados del estudio del parámetro *epochs* agrupado por el número de neuronas (*neurons*) (ANN).



c) Resultados del estudio del parámetro *epochs* agrupado por la función de activación (activation) (ANN).



d) Resultados del estudio del parámetro *learn_rate* agrupado por el parámetro *batch_size* (ANN).



e) Resultados del estudio del parámetro *learn_rate* agrupado por el parámetro *la función de activación (*activation) (ANN).

Figura H.60: Estudio de los principales parámetros después de ejecutar la rejilla inicial con *GridSearchCV* (ANN).

Tabla H.43: Resultados de la validación cruzada validación cruzada con ANN (top-5).

Modelo	***activation***	**dropout_rate**	***epochs***	***neurons***	**ROC-AUC (media validación)**
ANN_1	*tanh*	0.1	20	6	90.722857
ANN_2	*tanh*	0.1	10	6	90.762143
ANN_3	*tanh*	0	10	4	90.697857
ANN_4	*tanh*	0.1	20	4	90.678571
ANN_5	*relu*	0	20	6	90.696429

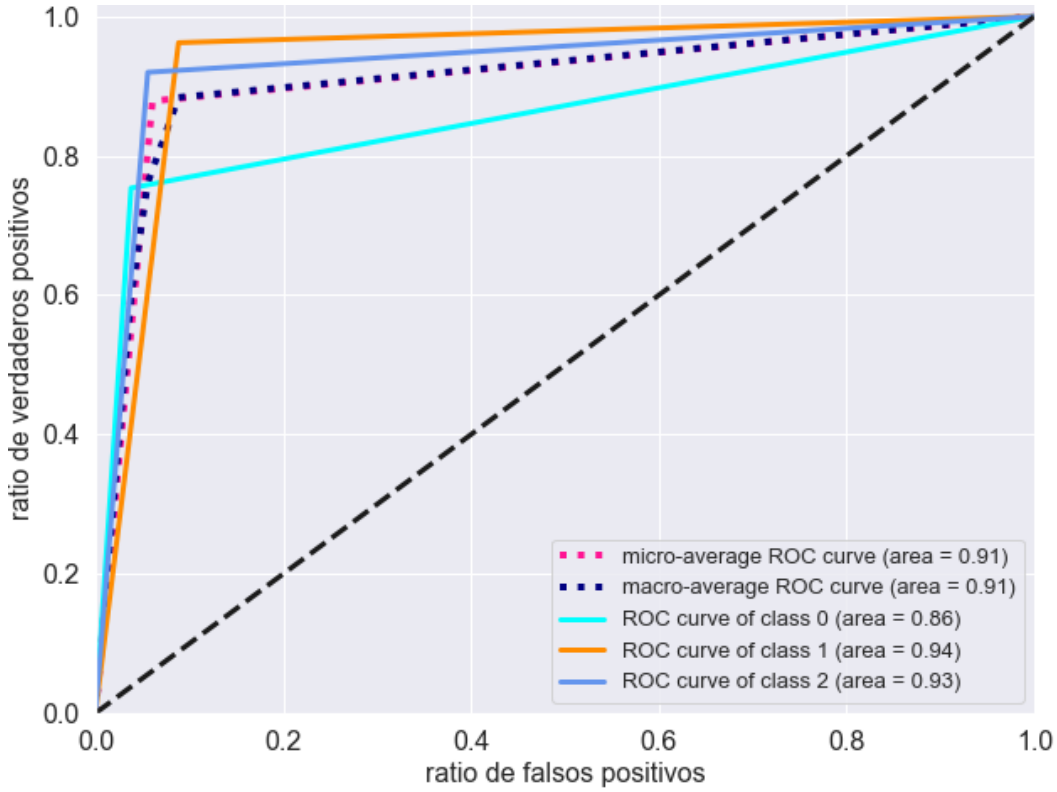


Figura H.61: Curva ROC-AUC del mejor modelo (ANN).

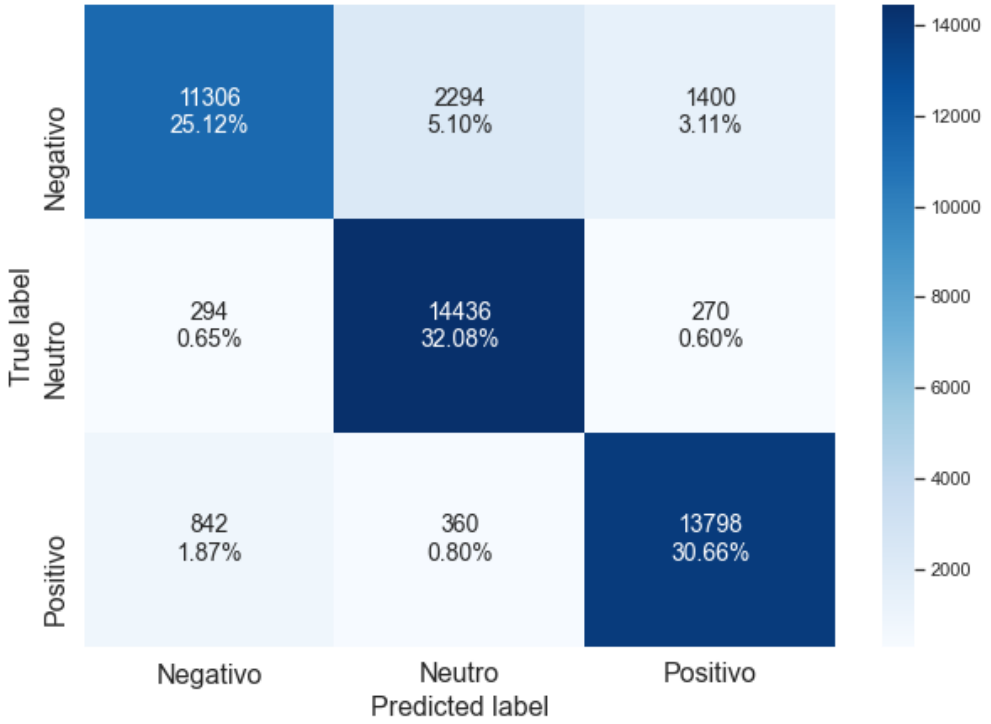


Figura H.62: Matriz de confusión del test del mejor modelo (ANN).

Tabla H.44: Grupos de ensamblados

Grupo	Modelos (mejor modelo)
Stacking_1	RL(7), KNN(3), SVM(SVCLinear), ANN(3)
Stacking_2	RF(3) y *XGBoost*(1)
Stacking_3	*Todos*

```
Accuracy of LinearSVC classifier on training set: 0.900
Accuracy of LinearSVC classifier on test set: 0.895
              precision    recall  f1-score   support

           0       0.93      0.78      0.85     15000
           1       0.84      0.97      0.90     15000
           2       0.92      0.94      0.93     15000

    accuracy                           0.90     45000
   macro avg       0.90      0.90      0.89     45000
weighted avg       0.90      0.90      0.89     45000

ROC_AUC test:  0.9214166666666667
precision test: 0.8994401667068547
recall test: 0.8952222222222223
fscore test: 0.8937597190585352
support test: 15000.0
Wall time: 45.1 s
```

Figura H.63: Resultados del *test* del mejor modelo (*SVC_Linear*).

[1] Fuente: https://www.worldometers.info/coronavirus/?utm_campaign=homeAdvegas1?

[2] Fuente: https://www.washingtonpost.com/opinions/global-opinions/the-delta-variant-spells-trouble-the-best-way-to-counter-it-is-vaccines/2021/06/23/a1c19286-d38b-11eb-9f29-e9e6c9e843c6_story.html

[3] Fuente: https://towardsdatascience.com/artificial-intelligence-machine-learning-and-deep-learning-what-the-difference-8b6367dad790

[4] Fuente: https://blogs.iadb.org/conocimiento-abierto/es/aplicando-el-procesamiento-del-lenguaje-natural-para-clasificar-articulos-del-coronavirus/

[5] Fuente: https://stanfordnlp.github.io/CoreNLP/pipeline.html

[6] Fuente: https://pubmed.ncbi.nlm.nih.gov/33139966/

[7] Fuente: https://www.researchgate.net/publication/283954600_Sentiment_Analysis_An_Overview_from_Linguistics

[8] Fuente: https://es.wikipedia.org/wiki/Modelo_de_espacio_vectorial

[9] Fuente: https://tacosdedatos.com/texto-vectores

[10] Fuente: https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-4-count-vectorizer-b3f4944e51b5

[11] Fuente: https://tacosdedatos.com/texto-vectores

[12] Fuente: https://tacosdedatos.com/texto-vectores

[13] Fuente: https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analysis/

[14] Fuente: https://towardsdatascience.com/cross-validation-430d9a5fee22

[15] Fuente: https://www.theguardian.com/society/2020/dec/02/pfizer-biontech-covid-vaccine-wins-licence-for-use-in-the-uk
[16] Fuente: https://www.bbc.com/news/uk-55227325
[17] Fuente: https://edition.cnn.com/world/live-news/coronavirus-pandemic-vaccine-updates-12-14-20/h_e4e95ac5f7e6100a2d3e4ba4c75a20d5
[18] Fuente: https://edition.cnn.com/world/live-news/coronavirus-pandemic-11-16-20-intl/h_3c91c6f3b13c0b6d96bd126d9cda9018
[19] Fuente: https://www.bbc.com/news/live/world-54984108
[20] Fuente: https://edition.cnn.com/world/live-news/coronavirus-pandemic-11-23-20-intl/h_50e0c473cbe0566941719b1d98d75aba
[21] Fuente: https://www.canada.ca/en/health-canada/news/2020/12/health-canada-authorizes-first-covid-19-vaccine.html
[22] Fuente: https://learning.oreilly.com/library/view/blueprints-for-text/9781492074076/ch01.html#ch-exploration
[23] Fuente: https://towardsdatascience.com/sentiment-analysis-comparing-3-common-approaches-naive-bayes-lstm-and-vader-ab561f834f89
[24] Fuente: https://es.wikipedia.org/wiki/Operaci%C3%B3n_Warp_Speed
[25] Fuente: https://support.sas.com/documentation/onlinedoc/txtminer/12.3/tmref.pdf
[26] Fuente: https://empresas.blogthinkbig.com/word-embedings-como-la-ia-nos-muestra-la-evolucion-de-las-palabras/
[27] Fuente: https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223

[28] Fuente: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
[29] Fuente: https://www.nytimes.com/interactive/2020/us/covid-19-vaccine-doses.html