UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN

BIG DATA ANALYTICS - MBI

TRABAJO FIN DE MÁSTER

DESARROLLO DE UN SISTEMA DE ANÁLISIS TEXTUAL DE SENTIMIENTOS CON VISUALIZACIÓN APLICADA

NOMBRE:

ÁNGEL MANUEL GARCÍA CARMONA

CURSO 2020-2021

TÍTULO: DESARROLLO DE UN SISTEMA DE ANÁLISIS TEXTUAL DE SENTIMIENTOS CON VISUALIZACIÓN APLICADA

AUTOR: ÁNGEL MANUEL GARCÍA CARMONA

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE GRANDES CANTIDADES DE DATOS

DIRECTOR DEL PROYECTO: JESÚS CARRETERO PÉREZ

FECHA: SEPTIEMBRE de 2021

RESUMEN

El análisis de cantidades de datos cada vez más voluminosas, veloces y variadas, aunque sea a distintas velocidades, no es solo una realidad patente y permanente. También trasciende meras aplicaciones de índice matemático, estadístico y contable-financiero, aunque nada esté absolutamente disperso.

La digitalización de la sociedad y la mayo dependencia de los individuos de la "red de redes", tanto para bien como para mal nos permiten extraer información que permita elaborar un perfil de cierta envergadura sobre estos, sin necesidad de someterle a un complejo interrogatorio (por ejemplo).

Esto mismo nos motiva a elaborar un prototipado que, de manera desinteresada y plenamente constructiva, contribuya al desarrollo de la telemedicina y otras aplicaciones tecnológicas relacionadas con las distintas especialidades de la misma (*e-Health,* obedeciendo al anglicismo). En nuestro caso, Psiquiatría y Psicología serán las más beneficiadas.

Concretamente, implementamos un sistema de análisis de sentimientos mediante aprendizaje supervisado, a fin de clasificar por medio de la predicción, partiendo de un conjunto de datos entrenado. Queremos que se pueda clasificar, con la mayor exactitud posible, unos mensajes, según la positividad o la negatividad sentimental que quede reflejada en estos.

Al mismo tiempo, se aplica una serie de técnicas de visualización que facilitan al interesado, en principio un facultativo sanitario, la comprensión de lo que no dejan de ser indicadores que le ayuden a determinar el diagnóstico de un posible paciente con problemas psiquiátricos, si es que, simplemente, no se trata de prevenir otros males mayores.

Palabras clave: aprendizaje supervisado, visualización, telemedicina, psiquiatría, análisis de sentimientos

ABSTRACT

The data to be analysed is every time more bulky, fast and variated, despite the fact that there are different velocities. That is not only a patent and permanent reality, but something that goes beyond Mathematical, Statistical and Financial applications.

The digitalization of the society and the higher dependence of individuals on the "net of networks", for good and bad purposes, let us extract information that allows us to elaborate a relevant profile without any need for a complex interrogatory.

This leads us to develop a prototype in a disinterested and totally constructive way, which may contribute to the development of e-Health and other technological applications that may be related to its different specialities. Precisely, we will focus on Psychiatry and Psychology.

Concretely, we will implement a sentiment analysis system based on supervised learning, in order to make classifications through prediction and a trained dataset. We want to classify, with the highest possible accuracy, some messages, according to their positivity or negativity.

At the same time, some visualization techniques will be applied. Those will help the doctor and other professionals to understand which are mental health indicators for a psychiatric diagnosis or the mere prevention of worse disgraces.

Key words: supervised learning, visualization, e-Health, psychiatry, sentiment analysis

AGRADECIMIENTOS

Mi agradecimiento y recuerdo a todos aquellos que, desde distintos ámbitos profesionales, distintas relaciones con mi persona y distintos entornos (dicotomía terrenal y trascendental-espiritual) han permitido ejercer este acto de libertad de oportunidades que supone formarme, en sentido práctico también, en este ámbito tecnológico, que supone una tendencia tecnológica, que refuerza mi perfil multidisciplinar como Ingeniero de Software.

Índice

Índice de Figuras

Para facilitar la comprensión de lo abordado en la presente memoria, hemos utilizado una serie de figuras, a indexar tal y como sigue:

1. INTRODUCCIÓN

1.1 Hipótesis

La telemedicina, como tendencia y como concepto de avance científico-tecnológico en la atención sanitaria y otras cuestiones relacionadas con la medicina, no es algo del futuro, sino algo ya existente.

Lo que también puede conocerse como *e-Health* no implica solo, siquiera, la digitalización de la atención clínica, lo cual se ha acelerado en el último bienio a raíz de las medidas de distanciamiento social impuestas con motivo de la pandemia del SARS-CoV-2 o COVID-19.

Podemos hablar, *lato sensu*, de un refuerzo de los medios utilizados para el diagnóstico clínico, sin necesidad de desembocar en el reemplazo literal del facultativo sanitario humano (no es lo mismo amenazar con sustituir que dotar de nuevos medios).

Se tiene confianza en la combinación conjunta de la Inteligencia Artificial, el *Big Data* y el *Machine Learning* a la hora de diseñar, desarrollar e implementar soluciones que no solo automaticen sino que optimicen los procesos de prevención y detección de enfermedades.

Así, se pretende prestar atención a los nichos que se presentan, no necesariamente en el sentido más estrictamente informático, en lo relacionado con la salud mental (ámbito al cual responde la Psiquiatría, por cuanto y en tanto que estudia la mente).

Existe cierta estigmatización a la hora de abordar patologías de rango psiquiátrico, lo cual fomenta que muchas personas afectadas por problemas de esta índole estén desamparadas, sin el tratamiento sanitario y psicosocial adecuado.

De hecho, no conviene eludir que, en las actuales circunstancias, a consecuencia de la convulsión causada por el COVID-19, se estaría desarrollando otra “pandemia paralela”, a interpretar como un aumento en la tasa de incidencia de enfermedades mentales.

De acuerdo con el psicólogo clínico Rafael Santandreu , casi un cuarenta por ciento de españoles presenta ya cuadros relacionados con ataques de pánico, hipocondría y trastorno obsesivo compulsivo, lo cual puede ir perfectamente ligado a factores de ansiedad.

Este mismo experto advierte de que ya en España, alrededor de un doce por ciento de la población toma ansiolíticos, advirtiendo, a su vez, de los riesgos adictivos de estas sustancias, lo cual aminora el efecto farmacológico esperado.

A su vez, cabe no olvidar que han aumentado los casos de depresión y de intentos de suicidio. Por lo tanto, conviene disponer de herramientas que ayuden, al menos, al facultativo, a detectar determinados cuadros psiquiátricos que puedan desembocar en males mayores.

1.2 Objetivos del proyecto

El principal objetivo del presente proyecto es desarrollar un sistema de predicción analítica que trabaje en el análisis de sentimientos, llevando a cabo una técnica de clasificación (previamente entrenada) de una serie de mensajes.

No obstante, haremos, a continuación, un pequeño desglose que, en otros términos más sencillos y quizá menos abstractos, pueda interpretarse como una serie de objetivos más específicos:

a. Disponer de un conjunto de mensajes clasificados lo más extenso y equilibrado posible, que sirva para un entrenamiento que fomente una mayor precisión.

b. Clasificar de la manera más precisa posible un conjunto de mensajes que remitamos a la plataforma, en un orden cronológico.

c. Asegurar medios de visualización lo suficientemente intuitivos y clarificadores.

d. Facilitar al facultativo el diagnóstico de enfermedades mentales y el seguimiento clínico de estas personas, que también puede ser meramente peventivo.

Dicho esto, cabe indicar que el proyecto recibirá el nombre *Sientmasistes*, que será una combinación terminológica selectiva del eslogan *Yo siento, tú me asistes*, orientado a la finalidad profesional y social del mismo.

1.3 Estructura del proyecto

Dejando al margen la sección introductoria y el área de las referencias bibliográficas, conviene indicar que nuestro documento tiene una serie de bloques bien diferenciados en relación al estado del arte, el desarrollo de la solución y los resultados de la misma, más unos anexos.

En el bloque relacionado con el estado del arte, se aborda una contextualización y puesta conceptual relacionada con el *e-Health* o telemedicina y el *machine learning* en el ámbito sanitaria, tras lo cual, con casos de uso adjuntos, se habla sobre el análisis de sentimientos.

A continuación, en la parte de diseño y modelado, se explica cómo se han tratado para su procesamiento (contando con las adecuaciones de preprocesamiento) los datos con los que posteriormente haremos la predicción y la clasificación, y definimos la visualización.

En un capítulo posterior justificamos la escritura (para compilación posterior) de los distintos fragmentos de código que nos han permitido implementar y poner en funcionamiento esta solución.

Donde nos referimos a los resultados, exponemos demostraciones que resultan del procesamiento de determinados conjuntos de mensajes, incluyendo la evaluación métrica de las técnicas de clasificación que se han aplicado.

Al finalizar con ello, incorporamos un apartado relacionado con la propuesta de caso de negocio, aunque sin detallar en la misma la parte presupuestaria, que va en un apartado posterior a la sección de anexos, en la que se hablan detalles de planificación temporal y organizativa.

Por último, incorporamos un apartado de anexos que hace referencia a información que puede suscitar interés relativo y relacional para quien consulte este trabajo, pudiendo servir para ayudar a "profundizar" determinadas cuestiones.

Eso sí, cabe no olvidar que, previamente, se desarrolla, convenientemente en un apartado independiente, lo relacionado con las conclusiones del trabajo así como con posibles consideraciones futuras.

2. ESTADO DEL ARTE

2.1. El *Machine Learning* a día de hoy

A día de hoy, se puede considerar la Inteligencia Artificial (IA) como una de las tendencias tecnológicas más prometedoras, con buenas expectativas de futuro, con independencia de que no todos sus usos sean estrictamente loables –lo cual no ha de ser, bajo ningún concepto, razón para adoptar una postura acrítica absolutamente escéptica y neoludita.

De acuerdo con el portal *Statista*, se estima que, a finales de 2021, el volumen global de la Inteligencia Artificial en el mercado del *software* suponga alrededor de 34’84 millardos de dólares norteamericanos ($) . Al mismo tiempo, se prevé que a lo largo de 2022, se vaya desarrollando una variación porcentual de un 47’03%.

Dicho esto, conviene tener en cuenta que una de las “subdivisión” de la IA sería el “aprendizaje máquina”, más conocido por su traducción terminológica al inglés, esto es, el término *Machine Learning* (ML), que es definido por la entidad eléctrica *Iberdrola* de la siguiente forma:

> «El *Machine Learning* es una disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, dota a los ordenadores de la capacidad de identificar patrones en datos masivos para hacer predicciones. Este aprendizaje permite a los computadores realizar tareas específicas de forma autónoma, es decir, sin necesidad de ser programados.»

La Medicina es uno de los ámbitos donde este concepto tiene mayor margen de maniobra, actuación y proyección. Podemos decir así que entre sus principales aplicaciones en el área sanitaria , dentro de la cual figura la telemedicina, a la que haremos mención más adelante, están los siguientes puntos:

- → Asistentes virtuales de conversación con finalidades de detección de síntomas y otros patrones patológicos (comúnmente son conocidos como *chatbots*).
- → Algoritmos en entrenamiento para detectar tumores y otros tejidos cancerígenos, mediante el llamado *deep learning*.
- → Diagnóstico de enfermedades por medio de resultados de analíticas de laboratorio (por ejemplo, la sangre y la orina).
- → Detección facial para aminorar la improbabilidad de detección diagnóstica de enfermedades raras.

2.2. Impacto social y económico del *e-Health*

Desde hace unos años, ya se venía hablando sobre el *e-Health* (entiéndase como la concepción digital de las prestaciones para el cuidado de la salud) como algo que, en su momento, se implementaría en la práctica totalidad de la red de entidades sanitarias, más allá de los centros hospitalarios.

Se venía hablando de una digitalización respaldada por la computación en la nube, las aplicaciones de Inteligencia Artificial o *apps* móviles que pudiesen responder perfectamente al devenir cotidiano para mantenerse sano (incluso el factor de la dependencia de discapacitados y personas de avanzada edad).

Pero como ha ocurrido con el fenómeno del teletrabajo, la situación excepcional ocasionada por el coronavirus codificado como SARS-CoV-2 o COVID-19, que dio lugar a que muchas empresas acelerasen su integración en la era digital para que las restricciones impuestas les permitiesen desarrollar cierta actividad económica, también influyó en la atención sanitaria.

No obstante, sería absurdo limitar la aplicación de esta combinación de soluciones tecnológicas a una dicotomía entre "ir presencialmente a consulta" o requerir atención médica de carácter telemático. Por ello, cobra más interés resaltar algunas de las aplicaciones más relevantes, sobre todo aquellas que "afectan" a este trabajo .

Podemos concebir dispositivos que no solo sean más amigables en lo concerniente a interfaz persona-dispositivo, sino que estén más centrados en el usuario, así como sensores para medir distintas variables y constantes vitales: presión arterial, agudeza visual, concentración de determinadas sustancias químicas, etc.

Pero la cuestión es que de una u otra forma va a entrar de lleno, junto al Internet de las Cosas y el Internet del Comportamiento, el *Big Data*. Los datos serán esenciales como fuentes para los algoritmos que resolverán nuestros problemas. Del mismo modo, habrá simulaciones que trascenderán el diagnóstico por imagen.

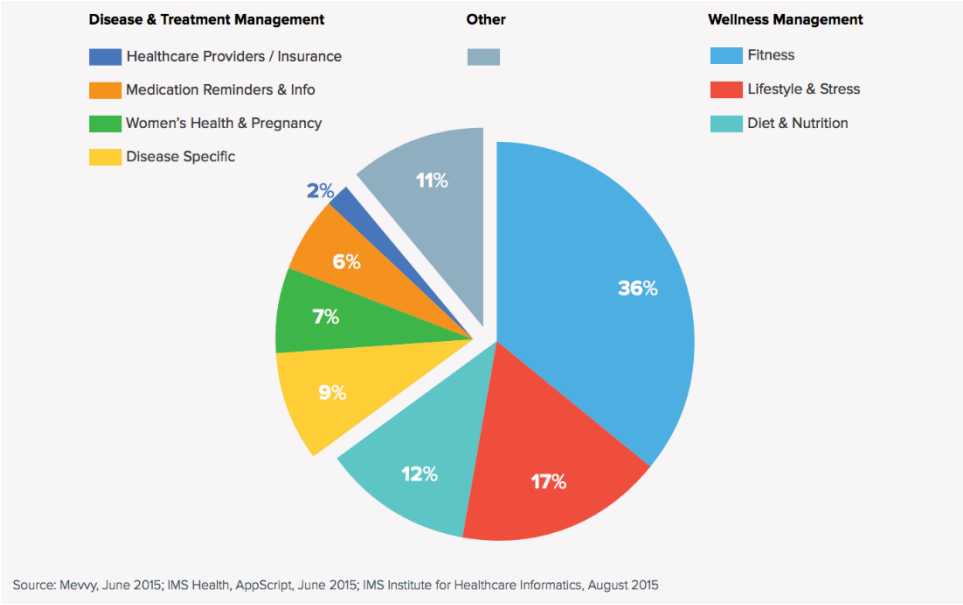


Figura 1 - Proporción de casos de uso relacionados con el *e-Health*

De hecho, como afirman Jegen y Byers , podríamos llegar a hablar de una "liberación" de datos, la cual llegan a equiparar con la facilidad que tiene a día de hoy el usuario para llevar a cabo varias operaciones financieras y bancarias desde su aplicación móvil o un portal web (estos llegaran a acuñar el término "liquidez de datos").

Así pues, puede obviarse que varias empresas de distintas dimensiones estén apostando por atender a los distintos nichos de mercados relacionados con el área tecnológica y las respectivas

aplicaciones tecnológicas. De hecho, en países como España hay una eclosión muy heterogénea en materia de *start-ups* relacionadas con el sector .

Esto, a su vez, permite evidenciar, en cierto modo, que la rama sanitaria sea otro de los motivos del cada vez mayor peso proporcional de la ingeniería y el sector de las Tecnologías de la Información en el mercado, repercutiendo esto en el tipo de puestos de trabajo que comiencen a demandarse con mayor urgencia y prioridad.

2.3. Concepto del análisis de sentimientos

Un diagnóstico clínico no es ajeno a las cuestiones y problemáticas de salud mental, de las cuales se encarga la Psiquiatría. En términos más sencillos a la hora de la abstracción y la comprensión, no todos los problemas patológicos que nos pueden afectar son ajenos a nuestra mente.

Es por ello, aunque pueda ser para situaciones que simplemente sean una mera competencia de la Psicología, por lo que una de las aplicaciones del *e-Health* más relacionadas con las grandes cantidades de datos pueden incorporar y llevar implementadas funciones relacionadas con el análisis de sentimientos .

Este se puede definir como un área vinculada al procesamiento del lenguaje natural que nos permite alcanzar conclusiones por medio de la información subjetiva que se refleja en diversos datos. De este modo se puede intentar determinar si el ánimo de uno tiene un estado positivo o negativo.

Ahora bien, no es ese el único propósito posible de estas técnicas. Otras aplicaciones son la valoración de productos y servicios, la clasificación puntuada de la opinión sobre algo concreto, el posicionamiento SEO y de publicidad online, el análisis y estudio de mercados, y la demoscopia electoral y sociológica.

2.4. Casos de uso reales

A continuación, enumeraremos algunos casos reales de uso de soluciones informáticas basadas en el análisis de sentimientos, que estamos entendiendo como un análisis del lenguaje natural (es posible procesar datos no estructurados en formato de audio también), de estructuras semánticas. La relación será la siguiente:

- Un banco con sede en Johannesburgo (Sudáfrica) recurrió a una interfaz de programación de aplicaciones (por su acrónimo en lengua inglesa, nos referiremos a la misma como API) para analizar 2 millones de textos publicados en varios servicios de *social media* (redes sociales). Empleó el reconocimiento de entidades nombradas y reflejó los resultados en un panel de visualización. Pretendían averiguar si muchos usuarios estaban conformes o no con sus servicios.

- Una operadora telefónica europea pretendía analizar en qué punto se ocasionaba atrición del consumidor a raíz de interacciones negativas. Se hizo una conversión del audio de las llamadas de voz en texto que fue granularizado para comprobar si había

promedios negativos en torno a una puntuación determinada. Esto ayudó a captar nuevos clientes.

- Un hospital estadounidense trató de adquirir referencias que le permitiesen saber si los pacientes estaban conformes con el servicio asistencial prestado desde el momento de la primera entrada hasta que este finalizase, independientemente de que la visita fuera única o no.

- La multinacional de comida rápida KFC monitorizó una campaña basada en memes e iconografía propia de la cultura pop. Se combinó el análisis de sentimientos con la monitorización del rendimiento y la minería de temas para extraer nuevas ideas y variaciones, lo cual ayudó a robustecer las campañas publicitarias.

3. DISEÑO Y MODELADO DEL SISTEMA

3.1. Introducción

En base a los objetivos y la finalidad del análisis de sentimientos, queremos desarrollar una solución que permita evaluar el estado de ánimo de un usuario por medio de las conversaciones textuales que se remitan y registren para su procesamiento.

La metodología se basará en el aprendizaje supervisado, con ciertas técnicas de refinado posteriores a las fases de ETL de los datos que nos ayuden a optimizar en cierta medida los resultados de la clasificación.

Una vez que se clasifican los mensajes textuales del usuario, aplicamos estrategias de visualización que faciliten el análisis y la observación de los datos, más allá de meros cálculos matemáticos y estadísticos.

3.2. Obtención y preprocesamiento de datos

Para definir nuestro conjunto de datos de entrenamiento y evaluación, hemos hecho una combinación de registros tomados de distintos *datasets* abiertos y publicados en internet, que enumeraremos a continuación:

- *TwitterSentimentDataset* – Github
- *Stopword Lists for 19 Languages* – Kaggle
- *IMDB Dataset of 50K Movie Reviews (Spanish)* – Kaggle
- *Spanish tweets suggesting depression* – Kaggle
- *Spanish Arilines Tweets Sentiment Analysis* – Kaggle

Ahora bien, no todos tienen estrictamente la misma utilidad, *lato sensu*. Por ejemplo, los términos de parada (*stopwords*) en inglés son una referencia de términos a eliminar, en la fase de transformación, por entender que no aportan nada a las técnicas predictivas.

Aparte de ello, conviene indicar que el lexicón de términos positivos y negativos ha sido tomado del *dataset* iSOL, un lexicón que resulta de traducciones automáticas, corregidas manualmente con posteridad, de la lista de palabras del profesor Bing Liu.

La fase de transformación puede contemplarse dentro de las estrategias de preprocesamiento de los datos, que por un lado buscan alcanzar un grado de limpieza considerable en el *dataset* y, por otro, reducir la probabilidad de que no sea óptimo.

Así pues se procede, en materia de limpieza, a:

I. Eliminar las columnas anómalas, sin datos
II. Eliminado de los siguientes elementos:
    a. Marcadores de tabulación y retorno
    b. Nicks de usuarios de Twitter y de *hashtags*
    c. Enlaces externos (URLs)
    d. Signos de puntuación

e. Indicadores de retuiteo (RT)
f. Emojis
g. *Stopwords* (se excluyen los términos "no", "nada", "nunca", "ni", "jamás", "tampoco" y "siquiera", dado que pueden ser determinantes en ciertas cláusulas de negación)

III. Conversión del texto, en su integridad, de modo que todas las letras estén en minúsculas (esto se hace antes del paso II.g).

Posteriormente, a fin de establecer una clasificación más sofisticada, tratamos de ampliar en dos categorías (clases) adicionales la categorización de los mensajes, en base a los sentimientos ("muy negativo" y "muy positivo").

Se entiende que hay sentimientos de una intensidad mayor, más extrema. Esto depende también del tipo de términos que se utilicen (por ejemplo, no es lo mismo estar sin inspiración que con una depresión).

Para ello se toma como referencia un lexicón con subclasificaciones de términos según su utilidad para transmitir algo positivo o algo negativo y se hace un recuento de términos positivos y negativos en las distintas frases.

Además, como se considera que no todos los términos tienen la misma importancia, se establecen unos patrones de expresiones regulares (*regex*) que nos permitan considerar como "términos relevantes" aquellos que cumplan con alguno de los siguientes patrones:

Concepto	Patrón *regex*	Categoría
Emoción	(emoci|emotiv)[a-zA-Z]*	Positiva
Alegría	(alegr)[a-zA-Z]*	Positiva
Felicidad	(felic|feliz)[a-zA-Z]*	Positiva
Maravilla	(maravill)[a-zA-Z]*	Positiva
Encanto	(encant)[a-zA-Z]*	Positiva
Pasión	[a]*(pasion)[a-zA-Z]*	Positiva
Amor	(enamor|am(a|e|o)|amab|amar[^g])[a-zA-Z]*	Positiva
Gustar	(gust)[a-zA-Z]*	Positiva
Agrado (paladar)	(sabr|delici)[a-zA-Z]*	Positiva
Excitación	(excit)[a-zA-Z]*	Positiva
Impresionar	(impresi[o-ó]n)[a-zA-Z]*	Positiva
Agradecido	(agradez|agradec)[a-zA-Z]*	Positiva
Querer	(quiero|querem|querre)[a-zA-Z]*	Positiva
Adorar	(adoro|adora|adore)[a-zA-Z]*	Positiva
Diversión	(diverti|diviert|divertid)[a-zA-Z]*	Positiva
Depresión	(depri|deprim|depresi)[a-zA-Z]*	Negativa
Angustia	(angust)[a-zA-Z]*	Negativa
Desánimo	(desanim)[a-zA-Z]*	Negativa
Decepción	(descont)[a-zA-Z]*	Negativa
Tristeza	(trist)[a-zA-Z]*	Negativa
Desesperación	(desespe)[a-zA-Z]*	Negativa
Descontento	(descont)[a-zA-Z]*	Negativa
Morir	(morir)[a-zA-Z]*	Negativa
Suicidarse	(suicid)[a-zA-Z]*	Negativa
Horrorizar	(horrori)[a-zA-Z]*	Negativa
Atemorizado	(depri|deprim|depresi)[a-zA-Z]*	Negativo

Amenazas	(amenaz)[a-zA-Z]*	Negativo
Aburrir	(aburr)[a-zA-Z]*	Negativo
Susto	[a*](sust)[ad|ar|e]*[a-zA-Z]*	Negativo
Enfado	(enfad|enoj(a|o))[a-zA-Z]*	Negativo
Alterar	(alter)[a-zA-Z]*	Negativo
Nervios	(enerv|nervios)[a-zA-Z]*	Negativo
Enfurecerse/Enfurruñarse	(enfurec|enfurr)[a-zA-Z]*	Negativo

Tabla 1 – Expresiones regulares relacionadas con determinados términos de sentimientos

Tras estas comprobaciones, con recuento, tratando de modificar solo la intensidad en vez de invertir la polaridad (de positivo a negativo y viceversa), establecemos dos condiciones de modificación del *dataset*:

- Si hay más términos positivos relevantes que negativos o, en su defecto, hay más términos positivos que negativos, cambiamos de "positivo" a "muy positivo".
- Si hay más términos negativos relevantes que positivos o, en su defecto, hay más términos negativos que positivos, cambiamos de "negativo" a "muy negativo".

Tanto al cargar los registros como aplicar la ampliación de clases categóricas posibles, se ha tratado de evitar un desequilibrio de datos basado en un exceso de muestras de un tipo frente a un porcentaje muy bajo de algunas de las restantes (*overfitting* o *undersampling*).

De acuerdo con la distribución normal de Gauss, los valores más extremos tienden a no situarse en el área central y mayoritaria. Esto se da en nuestra clasificación ya que, de manera muy aproximada, habría dos tercios de valores positivos y negativos y otro tercio de "extremos".

Para entender mejor esta explicación, se puede observar el siguiente diagrama de barras, en el que podremos corroborar que incluso la clase "negativo" tiene más valores que "positivo", pero que esta última ve que "muy positivo" sí supera a "muy negativo":

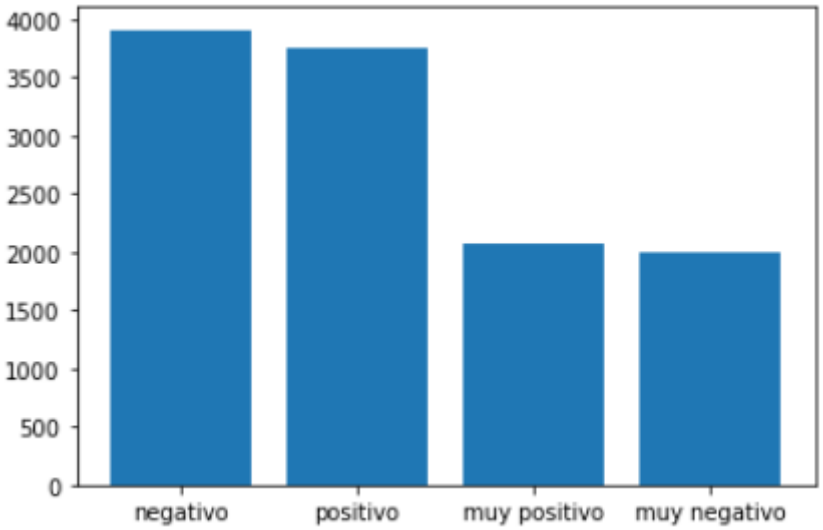


Figura 2 – Diagrama de barras con la cantidad de registros en función del valor de la variable objetivo, relacionada con el tipo de sentimiento

3.3. Almacenamiento de los datos

Antes de poner en marcha la aplicación, incorporamos todos los registros cuyo procesamiento deseamos en esta solución a un archivo de formato abierto separado por comas (el formato de tipo CSV).

Mientras, tanto los términos de parada de los que hemos hablado antes (*stopwords*) como relaciones de sentimientos positivos y negativos se almacenaron en una base de datos desplegada en un clúster de *MongoDB* (bases de datos documentales de tipo NoSQL).

Ahora bien, los datos de entrada, por parte del usuario, para su análisis, se cargarán de un archivo CSV. Los registros son composiciones de combinaciones de una fecha y de un cuerpo del mensaje.

3.4. Modelado de la estrategia predictiva

Para llevar a cabo la clasificación predictiva, hemos establecido un *pipeline* o tubería de datos, basado en una secuencia de fases de subprocesamiento de los datos que finalizan con la aplicación de un algoritmo de aprendizaje predictivo.

En nuestro caso concreto, la secuenciación obedecerá a este orden:

I. Vectorización con inversión de frecuencia (TF-IDF): aplicación del modelo de espacio vectorial (transformación de toda expresión de lenguaje natural a vectores de pesos de términos, mediante técnicas algebraicas) basada en el cálculo del producto de la frecuencia de términos en un documento y la llamada frecuencia inversa documental, que calcula el logaritmo de base 2 del cociente entre el total de documentos y el número total de veces que en ellos aparece un término.
II. Procesamiento del lexicón: Se aplican operaciones de transformación sobre un vector normalizado que resulta del número de términos positivos y negativos que se encuentran, teniendo en cuenta palabras las palabras de reversión (justamente las mismas que no se tuvieron en cuenta para su eliminación al tratar de limpiar el *dataset* de términos de parada u *stopwords*). Se tuvo en cuenta, en otras palabras, la llamada polarización terminológica. Asimismo, igual que la primera fase, será implícita a una estrategia de concatenación de resultados de lo que son múltiples (dos) transformaciones. Eso sí, de cara a la normalización, daremos previamente el doble de importancia a los términos extremos ("muy positivos" o "muy negativos").
III. Selección de términos más característicos: Tomando como base aplicaciones de prueba y contraste de la técnica del chi-cuadrado entre cada término y clase no negativa, tratamos de seleccionar un tanto por ciento de los términos con mayores puntuaciones. La función basada en la aplicación del chi-cuadrado o chi-2 tomaría dos arreglos y devolvería una unidad o un par de estos, con puntuaciones. En nuestro caso consideramos que cuanto mayor sea la proporción de términos con buenas puntuaciones a evaluar, mejor será la precisión.
IV. Clasificación mediante máquinas de soporte vectorial (SMV): Se trta de un método de predicción y regresión que separa linealmente (en verdad, no es el único kernel, pero sí el que utilizaremos nosotros), al menos, dos regiones, siendo todas ellas grupos en las que la mayoría de valores pertenezcan a una clase concreta. Utilizaremos un coeficiente de regularización con valor del 90% dado que queremos reducir al máximo posible el margen de clasificación, lo cual dará lugar a una mayor exactitud en las predicciones,

aunque dejando un margen de error prudencial. Asimismo, se harán contrastes en base al criterio *one-vs-one* dado que estaremos trabajando con varias clases (cuatro, una por cada tipo de sentimiento: positivo, negativo, muy negativo o negativo) y queremos obtener los detalles de cada una de estas.

Eso sí, cabe no olvidar explicar de qué manera se han seleccionado, previamente, los datos a entrenar y evaluar. Para ello, del mismo modo que en las fases de ETL se quiso evitar un desbalanceo muy acentuado de ajustes, aquí se estará tratando de evitar un sobreajuste.

3.5. Concreciones de modelado respecto a SMV

El propósito y fin de los algoritmos SVM es buscar un hiperplano con suficiente margen para separar grupos de datos clasificados, de modo que no se mezclen. El espacio puede ser n-dimensional, por cuanto y en tanto que existe un número *n* de características.

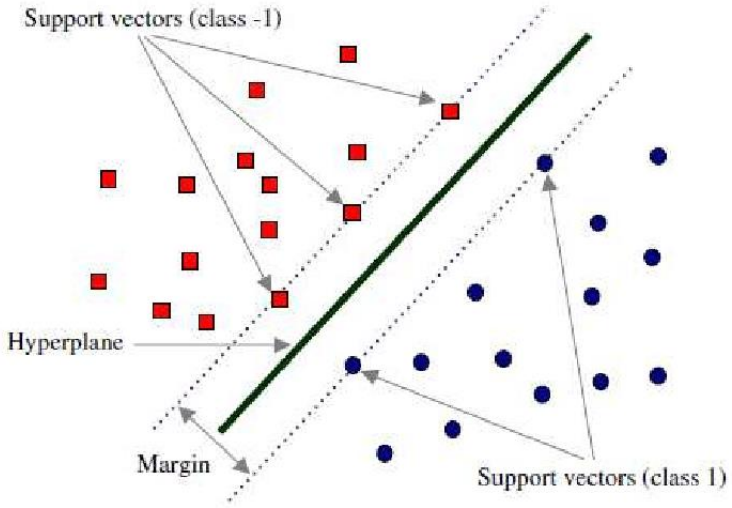
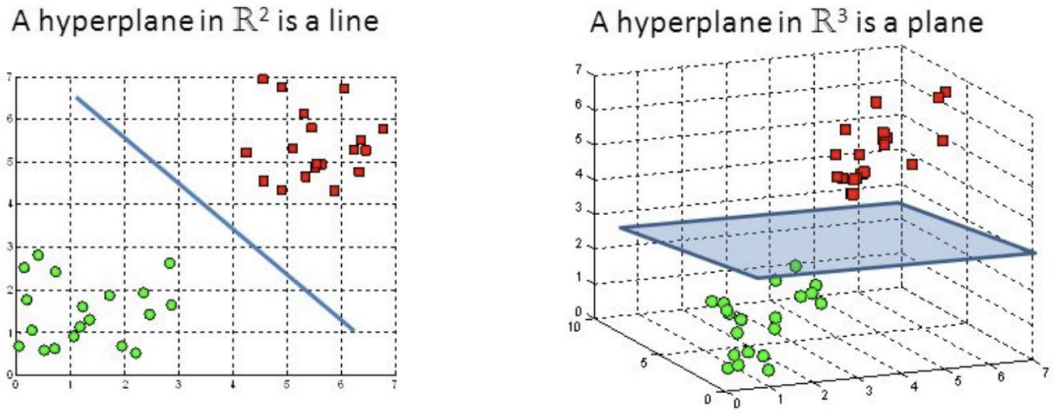


Figura 3 – Ejemplo gráfico de separación clasificadora en SVM

En el ejemplo de la imagen observamos una separación, con fines de clasificación, entre dos clases, denotadas, en cierto modo, con valores -1 y 1, respectivamente (intuición del margen mayor). Vemos también cómo existe un hiperplano de separación bajo margen.

Pero en base a lo que se ha indicado, el hiperplano tiene unas dimensiones que dependen del número de características. Así pues, podemos considerarlo como una línea si hay 2 dimensiones, o como un plano bidimensional si estas fueran cúbicas.

Eso sí, cabe indicar que nosotros hemos establecido un modelo lineal multiclase, aunque igualmente, el ejemplo anterior puede servir para obtener una comprensión genérica e inicial sobre estas concreciones de matematización.



3.5.1. Formulación matemática

La función objetivo de SVM en modalidad lineal (la cual utilizaremos para resolver estos problemas), conocida como fórmula primal, es esta:

$$J(\beta) = {}^{1}/_{2}\,\beta'\beta + C\sum_{n=1}^{N}(\xi_n + \xi^*{}_n)$$

La misma expresión está sujeta a las siguientes restricciones:

$$\forall n:\ y_n - \left(x'{}_n\beta + b\right) \leq\ \varepsilon + \xi_n$$

$$\forall n: \left(x'{}_n\beta + b\right) - y_n \leq\ \varepsilon + \xi^*{}_n$$

$$\forall n:\ \xi^*{}_n\ \geq 0$$

$$\forall n:\ \xi_n\ \geq 0$$

La constante C, un parámetro de regularización, es una restricción de caja basada en un valor positivo que controla la penalización que se impone en las observaciones subyacentes en el margen épsilon e impide el sobremuestreo.

Asimismo, la función de pérdida insensitiva ε hace caso omiso a los errores que se comprenden en la distancia ε demarcada para el valor observado, tratándolos como valores equivalentes a cero. Para concretar, cabe indicar que se mide de la siguiente forma:

$$L_\varepsilon = \begin{cases} 0\ si\ |y\ -\ f(x)|\ \leq\ \varepsilon \\ |y\ -\ f(x)|\ -\ \varepsilon\ si\ |y\ -\ f(x)|\ \geq\ \varepsilon \end{cases}$$

Hay que saber, además, que existe la llamada función de pérdida de bisagra, la cual ayuda a maximizar el margen que está comprendido entre los puntos de los datos y lo que denominamos como hiperplano. Esta cuenta con el parámetro de regulación (C) y se expresa así:

$$\min_{w} C\ ||w||^2 + \sum_{i\,=\,1}^{n}(1\ -\ y_i\langle x_i, w\rangle)_+$$

Una vez que tenemos constancia de esta función de pédida, para actualizar los pesos en nuestro modelo, utilizaremos unos gradientes, cuyo resultado de búsqueda resultará de derivadas parciales, a exponerse a continuación:

$$\frac{\partial}{\partial w_k} C\ ||w||^2 \ = \ 2Cw_k$$

$$\frac{\partial}{\partial w_k}(1 \ - \ y_i\langle x_i, w\rangle)_+ \ = \begin{cases} 0\ si\ y_i\langle x_i, w\rangle \ \geq \ 1 \\ -y_i x_{ik}\ si\ y_i\langle x_i, w\rangle \ < \ 1 \end{cases}$$

3.5.2. *Kernels* no lineales

Hay problemas que pueden resolverse igualmente con SVM, pero no admiten un modelo lineal. En este, mapearíamos los datos en torno a un espacio de características de una dimensión que probablemente sea infinita. La función del espacio de entrada no será líneal (solo la de decisión).

Los tipos de *kernels* (recordemos que SVM aplica transformaciones gracias al llamado "truco del kernel", que nos ayuda a lidiar, muy eficientemente, con datos de altas dimensiones) más habituales son los siguientes:

Tipo	Definición	Expresión
Polinómico	Se considera como una representación más genérica, pero menos eficiente y eficaz (precisión de resultados) del modelo lineal.	$k(x_i, x_j) = (x_i{}^T x_j \ + \ C)^p$
Gaussiano	Se emplea cuando no hay conocimientos previos de un conjunto de datos que se nos da.	$k(x_i, x_j) = e^{-\frac{\|x_j - x_i\|^2}{2\sigma^2}}$
Sigmoide	Es habitual en redes neuronales, por sus semejanzas con el perceptrón bicapa de las redes neuronales.	$k(x_i, x_j) = \tanh(\alpha x_i y \ + \ C)$
Función de Bessel (J)	Principalmente se emplea para eliminar los términos cruzados de las funciones matemáticas (variables no cuadradas).	$k(x_i, x_j) = \frac{J_{v+i}(\sigma\|x_i - x_j\|)}{\|\|x_i - x_j\|\|^{-n(v+1)}}$

3.6. Complejidad algorítmica

En este apartado, que en cierto modo supone un adelanto de detalles sobre la implementación de la solución, conviene comenzar acerca de los algoritmos más importantes que se están utilizando para resolver nuestro problema.

La manera más clarificadora puede ser la exposición de la información en una tabla como la que se expone a continuación. Eso sí, no todos han sido desarrollados por nosotros (en estos casos, los habríamos instanciado).

Nombre funcional	Descripción	Complejidad
extender_clasificacion()	En función de la cantidad de términos negativos y positivos, ya sean considerados como "relevantes" o no, se decide si incrementar la intensidad del polo negativo o positivo.	$O(n^2)$ – Orden cuadrático
ProcesadorLexicon.medir_puntuacion_mensaje()	Se tokeniza el mensaje en bloques de 3 unidades (ngramas), tratando, por separado, los términos previos (dos primeros) y los actuales (el último). Se considera la polaridad sentimental y se tiene en cuenta la inversión promovida por las cláusulas de negación	O(n) – Orden lineal
TfidfVectorizer()	Se hace una vectorización que tenga en cuenta la frecuencia de términos en documentos así como la llamada inversión de la frecuencia en documentos (el factor IDF).	O(nL log nL) – Orden logarítmico
SVC()	Implementación de la librería *libsvm* en la que el tiempo de ajuste escala, al menos, de manera cuadrática, con el número de muestras.	$O(n^3)$ – Orden cúbico

Tabla 2 – Algoritmos más importantes que se utilizan en la solución desarrollada

Con lo cual, teniendo en cuenta el peor de los casos, podemos decir que, en general, la complejidad conjunta del problema, partiendo de sus algoritmos más importantes, sería de tipo $O(n^3)$.

3.7. Estrategias de visualización

Consideramos que para una mejor comprensión de los resultados de nuestro análisis predictivo, no basta con exponer conjuntos de variables y mediciones, sino tratar de recurrir a la vía gráfico-ilustrativa.

Por ello, en base a los resultados del análisis, se ha considerado utilizar dos tipos de gráficos, tal y como se indica a continuación:

I. Diagrama circular multinivel: El primer nivel (el más externo) representa la cantidad de sentimientos negativos y de sentimientos positivos. Mientras, el segundo y último nivel, más interno, hace un desglose de esa clasificación inicial, en función de la intensidad de esos sentimientos (nivel normal y nivel extremo). Para los indicadores de negatividad se utilizan tonos rojizos de distinta intensidad mientras que, del mismo modo, para indicar positividad, empleamos tonos verdosos.

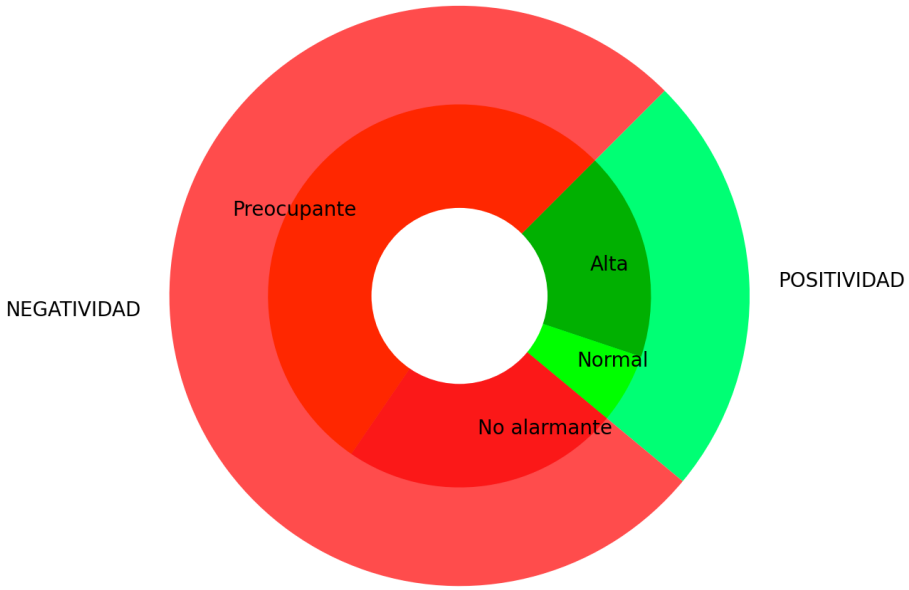


Figura 4 – Ejemplo de diagrama circular multinivel empleado dentro del marco de estrategias de visualización

II. Diagrama de barras: Se quiere mostrar una serie cuantitativa de sentimientos a lo largo de distintas fechas, ordenadas cronológicamente. Cada barra representa fragmentos cuyo tamaño varía en función del nivel de sentimientos positivos y negativos (sin considerar las clasificaciones subcategóricas). Al mismo tiempo, superponemos un gráfico de líneas que hace referencia a la cantidad de sentimientos muy negativos a lo largo de esa misma secuencia temporal de fechas.

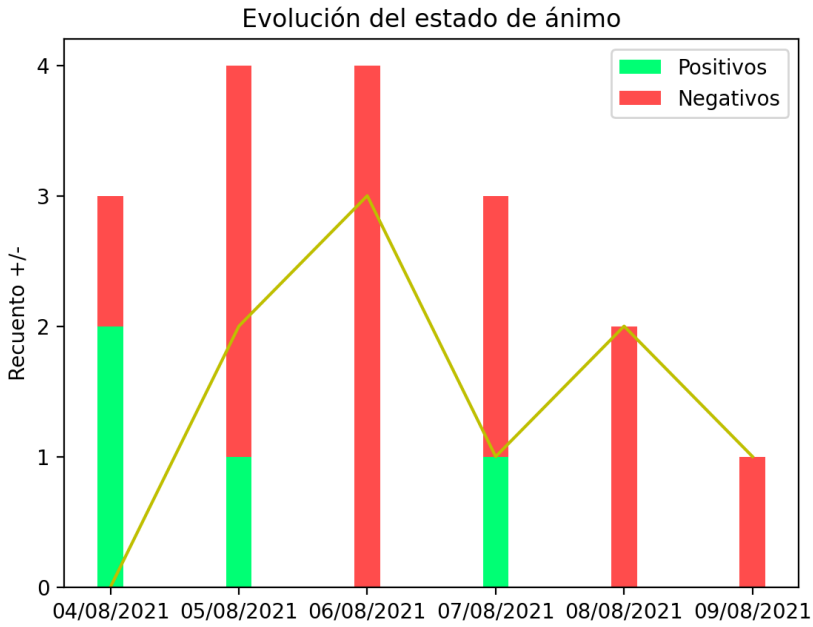


Figura 5 – Ejemplo de combinación entre un diagrama de barras multiclasificación y un diagrama de líneas, a fin de relacionar la evolución cronológica del estado de ánimo con un recuento bipolar de estados.

3.8. Esquema de construcción de la solución

En base a lo explicado en los puntos anteriores, se ha implementado una aplicación web, recurriendo a la librería *Streamlit* (esto nos ha permitido diseñar una sencilla y elegante solución de *front-end* por medio de unas pocas de funciones invocadas en una clase de *Python*).

La aplicación tiene la siguiente secuencia de funcionamiento:

I. Carga del *dataset* de entrenamiento (también obtenemos el lexicón y los términos de parada almacenados en nuestra base de datos).
II. Procesamiento del *dataset* de entrenamiento, tras una previa distribución entre la partición de prueba y la de evaluación, por medio de un *pipeline* que considera también unos procedimientos previos considerando los lexicones, la vectorización mediante técnica IDF y la selección de un percentil de resultados elevados, a lo cual sigue la aplicación de la técnica de clasificación SVC.
III. Muestra de un campo de entrada para cargar el *dataset* (conjunto de mensajes con una fecha asignada) que queremos evaluar. Será un archivo de texto separado por comas (extensión CSV).
IV. Representación de los diagramas mencionados en el apartado 3.6.

Mientras, para su despliegue, al margen de la solución presupuestada (véase la sección posterior a los anexos), se ha valorado, de manera temporal y preliminar, recurrir al servicio de plataforma como servicio (PaaS por sus siglas en inglés) *Heroku*.

Esta solución nos permitió desplegar fácilmente la aplicación, "bebiendo" del código que se subió al repositorio de *Github*, a través de *Heroku*, sin costes adicionales. La solución está accesible en la URL https://sientmeasistes.herokuapp.com.

4. IMPLEMENTACIÓN

4.1. Introducción

Aunque en el punto 3.8 ya hemos hecho una especie de esquema sobre el desarrollo de la solución, vamos a dedicar un capítulo específico a la implementación del código, que está escrito en el lenguaje *Python*.

A lo largo de los siguientes apartado, explicaremos el propósito y el sentido de los distintos fragmentos que componen el código fuente de nuestra aplicación, repartido en dos librerías (*main*, que es la clase principal, y *procesadorlexicon*).

4.2. Descarga de las bases de datos

La primera función que se ha definido en la clase principal tiene el cometido de descargar los registros de las tablas relacionadas con el lexicón (términos positivos y negativos) así como con los términos de parada (con ciertas excepciones).

Con esta intención se programa una instancia de conexión, vía *MongoClient*, al clúster de *MongoDB* que tenemos solicitado (se trata de una única base de datos, la cual está compuesta de varias tablas). Asimismo, el código utilizado será el siguiente:

```
def descargarDatosDB(lexicon_terminos_negativos,
lexicon_terminos_positivos, stopwords):
    from pymongo import MongoClient
    import urllib
    usuario = urllib.parse.quote("amgc")
    contrasena = urllib.parse.quote("am1996")
    connection = MongoClient("mongodb+srv://" + usuario + ":" +
contrasena + "@cluster0.uxnqk.mongodb.net/tfm-
amgc?retryWrites=true&w=majority")
    db = connection['tfm-amgc']
    for termino_neg in db['lexicon-neg'].find({}, {'_id' : 0}):
        lexicon_terminos_negativos.append(termino_neg['termino'])
    for termino_pos in db['lexicon-pos'].find({}, {'_id' : 0}):
        lexicon_terminos_positivos.append(termino_pos['termino'])
    for sw in db['stopwords'].find({}, {'_id':0}):
        if (sw['termino'] not in ['no', 'nada', 'nunca', 'ni',
'jamas', 'tampoco', 'siquiera']):
            stopwords.append(sw['termino'])

    connection.close()
```

Como se puede observer, nos autenticamos con un usuario y almacenamos los registros descargados en registros locales, tras lo cual procedemos a cerrar la conexión, dado que no volveremos a necesitarla.

4.3. Carga de los datos de entrenamiento

```
@st.cache
def cargarDatasetEntrenamiento(lexicon_terminos_negativos,
lexicon_terminos_positivos, stopwords):
    mensajesDF = pd.read_csv('mensajes.csv', sep=',')

    # Eliminamos las columnas anómalas, sin datos, de la estructura
DataFrame.
    mensajesDF.drop(mensajesDF.columns[[2,3,4]], axis = 1, inplace =
True)

    # Eliminamos registros vacíos, con valores no disponibles
    mensajesDF.dropna(inplace=True)

    # Eliminamos los marcadores de tabulación, retorno y tabulación
de las cadenas
    mensajesDF['mensaje'].replace(to_replace=[r"\\n|\\t|\\r",
"\t|\n|\r"], value="", regex = True, inplace = True)
    # Eliminamos los nicks de usuario de Twitter así como los
hashtags
    mensajesDF['mensaje'].replace(to_replace=[r"(@|#)[^\s]+",
"(@|#)[^\s]+"], value="", regex = True, inplace = True)
    # Eliminamos los enlaces externos (URLs)
    mensajesDF['mensaje'].replace(to_replace=[r"^http[s]?://(?:[a-
zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+",
                                              "http[s]?://(?:[a-zA-Z]|[0-
9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+"], value="",
regex = True, inplace = True)

    # Eliminamos los signos de puntuación
    mensajesDF['mensaje'].replace(to_replace=[r"[...|.|,|;]+"],
value="", regex = True, inplace = True)

    # Eliminamos los indicadores de retuiteo (RT)
    mensajesDF['mensaje'].replace(to_replace=[r"RT"], value="",
regex = True, inplace = True)

    # Eliminamos los EMOJIS
    import re
    regrex_pattern = re.compile(pattern = "["
            u"\U0001F600-\U0001F64F"
            u"\U0001F300-\U0001F5FF"
            u"\U0001F680-\U0001F6FF"
            u"\U0001F1E0-\U0001F1FF"
                               "]+", flags = re.UNICODE)

    mensajesDF['mensaje'] = mensajesDF['mensaje'].apply(lambda x :
regrex_pattern.sub(r'',x))

    # Ponemos todo texto en minúsculas
    mensajesDF['mensaje'] = mensajesDF['mensaje'].str.lower()

    # Eliminamos las llamadas STOPWORDS

    #print(mensajesDF[mensajesDF.sentimiento == "positivo"].count)
    mensajesDF['mensaje'] = mensajesDF['mensaje'].str.split('
').apply(lambda x: ' '.join(k for k in x if k not in stopwords))

    # print(pd.isnull(mensajesDF['mensaje']).describe())
```

```
    # Optimizamos el dataset en cuanto a categorización de
sentimientos
    from nltk.util import ngrams
    i = 0
    for indice, registro in mensajesDF.iterrows():
        registro_aux = re.sub("[^\w\s]", "",
str(registro['mensaje'])).split()
        terminos_positivos = 0
        terminos_negativos = 0
        terminos_en_total = len(registro_aux)

        if terminos_en_total > 0:
            fragmentos_mensaje = list(ngrams(registro_aux, 3,
pad_left=True))
            hay_termino_positivo_relevante = False
            terminos_positivos_relevantes = 0
            hay_termino_negativo_relevante = False
            terminos_negativos_relevantes = 0
            for fragmento in fragmentos_mensaje:
                terminos_previos = fragmento[:2]
                termino_actual = fragmento[2]

                if termino_actual in lexicon_terminos_positivos:
                    if any(termino_ in terminos_previos for termino_
in ['no', 'nada', 'nunca', 'ni', 'jamas', 'tampoco', 'siquiera']):
                        terminos_negativos += 1
                    else:
                        terminos_positivos += 1
                        # Tratamos de buscar un término relevante,
que detemine valores extremos
                        #if (hay_termino_positivo_relevante !=
True):
                        emocion = re.match(r"(emoci|emotiv)[a-zA-
Z]*", termino_actual)
                        alegria = re.match(r"(alegr)[a-zA-Z]*",
termino_actual)
                        felicidad = re.match(r"(felic|feliz)[a-zA-
Z]*", termino_actual)
                        maravilla = re.match(r"(maravill)[a-zA-Z]*",
termino_actual)
                        encanto = re.match(r"(encant)[a-zA-Z]*",
termino_actual)
                        pasion = re.match(r"[a]*(pasion)[a-zA-Z]*",
termino_actual)
                        amor =
re.match(r"(enamor|am(a|e|o)|amab|amar[^g])[a-zA-Z]*",
termino_actual)
                        gustar = re.match(r"(gust)[a-zA-Z]*",
termino_actual)
                        agrado_paladar = re.match(r"(sabr|delici)[a-
zA-Z]*", termino_actual)
                        excitar = re.match(r"(excit)[a-zA-Z]*",
termino_actual)
                        impresionar = re.match(r"(impresi[o-ó]n)[a-
zA-Z]*", termino_actual)
                        agradecido = re.match(r"(agradez|agradec)[a-
zA-Z]*", termino_actual)
                        querer =
re.match(r"(quiero|querem|querre)[a-zA-Z]*", termino_actual)
                        adorar = re.match(r"(adoro|adora|adore)[a-
zA-Z]*", termino_actual)
```

```
                        divertir =
re.match(r"(diverti|diviert|divertid)[a-zA-Z]*", termino_actual)
                        hay_termino_positivo_relevante = divertir or
adorar or querer or impresionar or agradecido or emocion or alegria
or felicidad or maravilla or encanto or pasion or amor or gustar or
agrado_paladar or excitar
                        if (hay_termino_positivo_relevante):
                            terminos_positivos_relevantes+=1
                elif termino_actual in lexicon_terminos_negativos:
                    if any(termino_ in terminos_previos for termino_
in ['no', 'nada', 'nunca', 'ni', 'jamas', 'tampoco', 'siquiera']):
                        terminos_positivos += 1
                    else:
                        terminos_negativos += 1
                        depresion =
re.match(r"(depri|deprim|depresi)[a-zA-Z]*", termino_actual)
                        angustia = re.match(r"(angust)[a-zA-Z]*",
termino_actual)
                        desanimo = re.match(r"(desanim)[a-zA-Z]*",
termino_actual)
                        decepcion = re.match(r"(decepcion)[a-zA-
Z]*", termino_actual)
                        tristeza = re.match(r"(trist)[a-zA-Z]*",
termino_actual)
                        desesperacion = re.match(r"(desespe)[a-zA-
Z]*", termino_actual)
                        descontento = re.match(r"(descont)[a-zA-
Z]*", termino_actual)
                        morir = re.match(r"(morir)[a-zA-Z]*",
termino_actual)
                        suicidarse = re.match(r"(suicid)[a-zA-Z]*",
termino_actual)
                        horrorizar = re.match(r"(horrori)[a-zA-Z]*",
termino_actual)
                        atemorizado = re.match(r"(atemori)[a-zA-
Z]*", termino_actual)
                        amenazas = re.match(r"(amenaz)[a-zA-Z]*",
termino_actual)
                        aburrir = re.match(r"(aburr)[a-zA-Z]*",
termino_actual)
                        susto = re.match(r"[a*](sust)[ad|ar|e]*[a-
zA-Z]*", termino_actual)
                        enfado = re.match(r"(enfad|enoj(a|o))[a-zA-
Z]*", termino_actual)
                        estres = re.match(r"(estres)[a-zA-Z]*",
termino_actual)
                        alterar = re.match(r"(alter)[a-zA-Z]*",
termino_actual)
                        nervios = re.match(r"(enerv|nervios)[a-zA-
Z]*", termino_actual)
                        enfu_erse = re.match(r"(enfurec|enfurr)[a-
zA-Z]*", termino_actual)
                        hay_termino_negativo_relevante = enfu_erse
or alterar or nervios or enfado or decepcion or susto or amenazas or
aburrir or horrorizar or morir or suicidarse or depresion or
angustia or desanimo or tristeza or desesperacion or descontento
                        if (hay_termino_negativo_relevante):
                            terminos_negativos_relevantes+=1


            if (registro['sentimiento'] == 'positivo' and
```

```
                (terminos_negativos_relevantes <
terminos_positivos_relevantes or terminos_positivos >
terminos_negativos)):
                registro['sentimiento'] = 'muy positivo'

            if(registro['sentimiento'] == 'negativo' and
                (terminos_positivos_relevantes <
terminos_negativos_relevantes or terminos_negativos >
terminos_positivos)):
                registro['sentimiento'] = 'muy negativo'

    return mensajesDF
```

Posteriormente, bajo un indicador peticionario de almacenamiento en caché (útil para ahorrar tiempo de procesamiento en caso de recargar la página para, por ejemplo, proceder al tratamiento de los datos de entrada en fichero que remita el usuario), definimos otra función.

Esta función servirá para cargar los datos de entrenamiento, que estarán almacenados en un archivo con miles y miles de registros, cuyo nombre y extensión, serán, ya expuestos de manera combinada, como nombre de archivo, *mensajes.csv*.

Con ello, procedemos a hacer una limpieza del *DataFrame* recién cargado, por medio de las siguientes acciones:

- Eliminación de las columnas que representan anomalías basadas en la carencia de datos.
- Eliminación de registros vacíos, que tienen valores no disponibles.
- Eliminación de los marcadores de retorno, tabulación y saltos de línea.
- Eliminación de los *hashtags* y los *nicks* de los usuarios de Twitter que pudieran estar incorporados.
- Eliminación de los enlaces externos (URLs).
- Eliminación de los signos de puntuación.
- Eliminación de los indicadores de retuiteo de tuits (marcados con los caracteres “RT”).
- Eliminación de emoticonos de tipo *emoji*, rigiéndonos por una serie de códigos en formato UNICODE.
- Puesta de todo el texto en letras minúsculas.
- Eliminación de los términos de parada o *stopwords*.

Después, nos encargaremos de hacer una optimización del conjunto de datos a la hora de categorizar los sentimientos, de una manera determinante en cierto modo para definir las categorías extremas (“muy negativo” o “muy positivo”).

Por cada fila, crearemos listas en las que cada elemento sean las distintas palabras que componen el mensaje encadenado, sin tener en cuenta los espacios en blanco (más bien, estos serán el factor determinante al separar cada palabra).

Lo primero que se hace con la misma es fragmentar, en función de los espacios en blanco, las cadenas de mensajes de cada uno de los registros, para crear una lista de palabras independientes (en cada una de las iteraciones por fila).

Con ello, cogeremos “puñados” de tres palabras, teniendo en cuenta que el significado de una palabra puede verse alterado por una cláusula a interpretar como un adverbio que sea de intensidad, frecuencia o negación, o que haya verbos que hagan de “previo indicador”.

Por cada uno de esos “puñados”, que consideraremos como fragmentos, haremos una separación de arreglos entre el último término y los dos previos, que son los que podrían alterar el concepto estándar.

Tras ello tratamos de comprobar si el término último se encuentra en el lexicón de términos positivos o en el de términos positivos. Al mismo tiempo, dicho sea que establecemos cuatro contadores, en función de la categoría del término.

Si el término está registrado como positivo pero tiene cláusulas de negación, lo contamos como negativo. Si no, como positivo, pero también, si y solo si cumple con un patrón de expresión regular de términos relevantes, como término positivo relevante.

En cambio, si estuviera registrado como negativo, contaríamos el término como positivo si estuviera precedido de cláusulas de negación y como negativo si se da el caso contrario. De darse lo último, comprobaríamos si cumple con expresión de término negativo relevante.

Mejor explicado, dicho sea lo que se ha abordado en los dos párrafos anteriores depende del lexicón en el que esté almacenado el término tercero de cada fragmento (términos positivos o negativos).

Una vez hechos estos recuentos, trataremos de valorar si se dan las condiciones necesarias para que se extreme la categorización de un registro clasificado como “positivo” o como “negativo”. Estas son las siguientes:

- Si la cantidad de términos negativos relevantes fuese inferior a la de positivos negativos o si en su defecto hubiera más términos positivos que negativos, entonces se pasaría de “positivo” a “muy positivo”.
- Si la cantidad de términos negativos relevantes fuese superior a la de positivos negativos o si en su defecto hubiera más términos negativos que positivos, entonces se pasaría de “negativo” a “muy negativo”.

4.3. Preparación del *pipeline* de procesamiento

Tomando como parámetros las particiones de datos (clasificación y contenido) de entrenamiento y de evaluación, definimos una función cuya invocación preparará y pondrá en marcha el *pipeline* que determinará los resultados de procesamiento.

```
@st.cache
def prepararPipeline(data_train, data_test, target_train,
target_test):
    from sklearn.feature_extraction.text import TfidfVectorizer
    # Create feature vectors
    vectorizer = TfidfVectorizer(use_idf = True, lowercase=True)
    from procesadorlexicon import ProcesadorLexicon
    lexicon = ProcesadorLexicon(lexicon_terminos_negativos,
lexicon_terminos_positivos)
    from sklearn.svm import SVC, LinearSVC
    svm = SVC(C = 0.9, kernel='linear', decision_function_shape =
'ovo')
    from sklearn.feature_selection import SelectPercentile, chi2
    selectPercentile = SelectPercentile(chi2, percentile=95)
```

```
    from sklearn.pipeline import Pipeline, FeatureUnion
    pipeline = Pipeline([
        ('feats', FeatureUnion([
            ('vectorizer', vectorizer),
            ('lexicon', lexicon)
        ])),
        ('select', selectPercentile),
        ('classifier', svm)
    ])

    pipeline.fit(data_train, target_train)

    preds = pipeline.predict(data_test)

    from sklearn.metrics import accuracy_score
    print("Exactitud =", accuracy_score(target_test,preds))
    from sklearn.metrics import classification_report
    print("Classification report:")
    clas_report = classification_report(target_test,preds)
    print(clas_report)

    from sklearn.metrics import multilabel_confusion_matrix
    print(multilabel_confusion_matrix(target_test,preds,labels=["muy
negativo", "muy positivo", "negativo", "positivo"]))

    return pipeline
```

Tengamos en cuenta que para procesar los datos aplicaremos una serie de técnicas basadas en las siguientes acciones:

- Vectorización por medio de *TfidfVectorizer*, utilizando la inversión de frecuencia y asegurándonos de que todos los términos a procesar estarán en mayúsculas.
- Obtención de un vector normalizado en función de la cantidad de términos de distintas categorías (en la clase *ProcesadorLexicon*, de la que se hablará más adelante).
- Elección del noventa y cinco por ciento de términos más característicos mediante una instancia a *SelectPercentile* (bajo la técnica del chi-cuadrado).

Con todo ello, tratamos con una instancia de *SVC*, que se base en una clasificación múltiple de la aplicación lineal de las SVM, con un coeficiente de regularización (C) del noventa por ciento y una función de decisión que haga comparaciones de uno en uno.

Tras ello, imprimimos en consola unos resultados que indiquen la exactitud de nuestros cálculos así como una tabla donde, por cada categoría, se indiquen la exactitud, la cobertura, la precisión y el marcador F1, aparte de considerar algunos promedios.

Acto seguido, se imprimen varias matrices de confusión (una por cada categoría), en el siguiente orden: “muy negativo”, “muy positivo”, “negativo” y “positivo”, teniendo en cuenta los datos de evaluación, a comparar con las predicciones.

4.4. Hacer recuento de clasificaciones en el *dataset* del usuario

Una vez que disponemos del *pipeline* y lo hemos puesto en funcionamiento, se da la opción de que, al hacer predicciones sobre unos datos que remitamos, podamos obtener cuántos de ellos obedecen a las distintas clasificaciones.

Para ello, definimos una función cuyo código exponemos a continuación, la cual retornará un arreglo con las cantidades de términos positivos, negativos, muy positivos y muy negativos que pueda haber en el conjunto.

```
def hacerRecuentoDatasetUsuario(mensajesUsuario, pipeline):
    terminos_pos = 0
    terminos_neg = 0
    terminos_muypos = 0
    terminos_muyneg = 0
    for mensaje in mensajesUsuario:
        clasificacion_mensaje = pipeline.predict([mensaje])
        if (clasificacion_mensaje == "positivo"):
            terminos_pos+=1
        elif (clasificacion_mensaje == "muy positivo"):
            terminos_muypos+=1
        elif (clasificacion_mensaje == "negativo"):
            terminos_neg+=1
        elif (clasificacion_mensaje == "muy negativo"):
            terminos_muyneg+=1

    return     [terminos_pos,     terminos_neg,     terminos_muypos,
terminos_muyneg]
```

4.5. Procesamiento del lexicón para optimizar cálculos predictivos

En este apartado hablaremos sobre las funciones principales (omitimos la de inicialización) de la clase *procesadorlexicon.py*, que en cierto modo va a depender también de la clase *sklearn*. El código se expone a continuación.

```
def transform(self, relacion_mensajes, y=None):
    resultado_a_normalizar = []

    for mensaje in relacion_mensajes:
      puntuacion_neg_pos_mensaje = self.medir_puntuacion_mensaje(mensaje)
      resultado_a_normalizar.append(puntuacion_neg_pos_mensaje)

    return preprocessing.normalize(resultado_a_normalizar)

  def medir_puntuacion_mensaje(self, mensaje):
    terminos_positivos = 0
    terminos_negativos = 0
    mensaje_tokenizado = re.sub("[^\w\s]", "", str(mensaje)).split()

    fragmentos_mensaje = list(ngrams(mensaje_tokenizado, 3, pad_left=True))
    terminos_positivos_relevantes = 0
    terminos_negativos_relevantes = 0
```

```
        for fragmento in fragmentos_mensaje:
            terminos_previos = fragmento[:2]
            termino_actual = fragmento[2]

            if termino_actual in self.terminos_positivos:
                if any(termino_ in terminos_previos for termino_ in self.PALABRAS_REVERSION):
                    terminos_negativos += 1
                else:
                    terminos_positivos += 1
                    # Tratamos de buscar un término relevante, que detemine valores extremos
                    emocion = re.match(r"(emoci|emotiv)[a-zA-Z]*", termino_actual)
                    alegria = re.match(r"(alegr)[a-zA-Z]*", termino_actual)
                    felicidad = re.match(r"(felic|feliz)[a-zA-Z]*", termino_actual)
                    maravilla = re.match(r"(maravill)[a-zA-Z]*", termino_actual)
                    encanto = re.match(r"(encant)[a-zA-Z]*", termino_actual)
                    pasion = re.match(r"[a]*(pasion)[a-zA-Z]*", termino_actual)
                    amor = re.match(r"(enamor|am(a|e|o)|amab|amar[^g])[a-zA-Z]*",
termino_actual)
                    gustar = re.match(r"(gust)[a-zA-Z]*", termino_actual)
                    agrado_paladar = re.match(r"(sabr|delici)[a-zA-Z]*", termino_actual)
                    excitar = re.match(r"(excit)[a-zA-Z]*", termino_actual)
                    impresionar = re.match(r"(impresi[o-ó]n)[a-zA-Z]*", termino_actual)
                    agradecido = re.match(r"(agradez|agradec)[a-zA-Z]*", termino_actual)
                    querer = re.match(r"(quiero|querem|querre)[a-zA-Z]*", termino_actual)
                    adorar = re.match(r"(adoro|adora|adore)[a-zA-Z]*", termino_actual)
                    divertir = re.match(r"(diverti|diviert|divertid)[a-zA-Z]*", termino_actual)
                    hay_termino_positivo_relevante = divertir or adorar or querer or impresionar or
agradecido or emocion or alegria or felicidad or maravilla or encanto or pasion or amor or
gustar or agrado_paladar or excitar
                    if (hay_termino_positivo_relevante):
                        terminos_positivos_relevantes+=1
            elif termino_actual in self.terminos_negativos:
                if any(termino_ in terminos_previos for termino_ in self.PALABRAS_REVERSION):
                    terminos_positivos += 1
                else:
                    terminos_negativos += 1
                    depresion = re.match(r"(depri|deprim|depresi)[a-zA-Z]*", termino_actual)
                    angustia = re.match(r"(angust)[a-zA-Z]*", termino_actual)
                    desanimo = re.match(r"(desanim)[a-zA-Z]*", termino_actual)
                    decepcion = re.match(r"(decepcion)[a-zA-Z]*", termino_actual)
                    tristeza = re.match(r"(trist)[a-zA-Z]*", termino_actual)
                    desesperacion = re.match(r"(desespe)[a-zA-Z]*", termino_actual)
                    descontento = re.match(r"(descont)[a-zA-Z]*", termino_actual)
                    morir = re.match(r"(morir)[a-zA-Z]*", termino_actual)
                    suicidarse = re.match(r"(suicid)[a-zA-Z]*", termino_actual)
                    horrorizar = re.match(r"(horrori)[a-zA-Z]*", termino_actual)
                    atemorizado = re.match(r"(atemori)[a-zA-Z]*", termino_actual)
                    amenazas = re.match(r"(amenaz)[a-zA-Z]*", termino_actual)
                    aburrir = re.match(r"(aburr)[a-zA-Z]*", termino_actual)
                    susto = re.match(r"[a*](sust)[ad|ar|e]*[a-zA-Z]*", termino_actual)
                    enfado = re.match(r"(enfad|enoj(a|o))[a-zA-Z]*", termino_actual)
                    estres = re.match(r"(estres)[a-zA-Z]*", termino_actual)
```

```
            alterar = re.match(r"(alter)[a-zA-Z]*", termino_actual)
            nervios = re.match(r"(enerv|nervios)[a-zA-Z]*", termino_actual)
            enfu_erse = re.match(r"(enfurec|enfurr)[a-zA-Z]*", termino_actual)
            hay_termino_negativo_relevante = enfu_erse or alterar or nervios or enfado or
decepcion or susto or amenazas or aburrir or horrorizar or morir or suicidarse or depresion
or angustia or desanimo or tristeza or desesperacion or descontento
            if (hay_termino_negativo_relevante):
               terminos_negativos_relevantes+=1

    return [terminos_positivos, terminos_negativos, terminos_positivos_relevantes*2,
terminos_negativos_relevantes*2]


  def fit(self, X, y=None):
    return self
```

La función *fit*, que es la última, nos permitirá hacer una estimación de parámetros, tomando como argumentos los datos de entrenamiento. Precisamente calculará la media aritmética y la desviación estándar.

Mientras, en la función *transform*, normalizamos los vectores que resultan de las distintas secuencias de mensajes, de modo que la norma vectorial (la suma de sus valores) tenga como valor el equivalente a una unidad.

En cuanto a *medir_puntuacion_mensaje()*, hay que decir que es una función similar, prácticamente idéntica, a la que hemos empleado para contar los términos de cada categoría en la fase de preprocesamiento.

Pero en este caso, retornamos algo, que viene a ser un vector de cuatro elementos en el que se duplica la puntuación de las cantidades de términos positivos y negativos de carácter relevante, a fin de darles más importancia, por lo que de extremos tienen.

4.6. Preparación de los datos a procesar

Tras haber definido las funciones principales y explicado en qué consiste la librería de procesamiento del lexicón que nos servirá en el *pipeline*, procedemos a cargar los datos de entrenamiento, así como a su tratamiento.

```
data_load_state = st.text('Cargando datos de entrenamiento...')
lexicon_terminos_negativos = []
lexicon_terminos_positivos = []
stopwords = []
descargarDatosDB(lexicon_terminos_negativos,
lexicon_terminos_positivos, stopwords)
mensajesDF = cargarDatasetEntrenamiento(lexicon_terminos_negativos,
lexicon_terminos_positivos, stopwords)
data_train, data_test, target_train, target_test =
train_test_split(mensajesDF['mensaje'],mensajesDF['sentimiento'],tra
in_size=0.75,random_state=0)
data_load_state.text('Comenzando procesamiento...')
pipeline = prepararPipeline(data_train, data_test, target_train,
target_test)
```

Como podemos observar, nos descargamos los registros de la base de datos que tengan que ver con los lexicones de términos positivos y negativos así como con los términos de parada o *stopwords*.

Tras ello, una vez que tengamos cargado nuestro *DataFrame* de entrenamiento (*mensajesDF*), procederemos a hacer un particionamiento del conjunto en el que el setenta y cinco por ciento de los datos tengan un propósito de *training*.

Finalmente, nos encargaremos de invocar a la realización de las funciones propias del *pipeline*, tras lo cual podremos saber, consultando en consola, si consideramos oportunos o no los resultados.

4.7. Remisión de los mensajes de usuario a evaluar

Ahora, se muestra al usuario un elemento de entrada mediante el cual podrá subir un archivo al servidor, si y solo si este es de tipo "CSV". En caso de que la carga sea exitosa, se procede a la predicción, mediante la función *predict*, invocada por *hacerRecuentoDatasetUsuario()*.

En función de los resultados, obtendremos un arreglo con la cantidad de términos que hay con cada categoría. Esto nos servirá para unas representaciones gráficas que ya abordamos en el punto 3.7 de esta memoria. El código se expone en el siguiente bloque.

```
filename = st.file_uploader("Elegir dataset de mensajes: ",
type=['csv'])

if filename is not None:
    data_load_state.text('Procesando dataset a evaluar...')
    mensajes_usuario = pd.read_csv(filename.name, sep=',')
    st.write("MENSAJES DEL USUARIO")
    st.write(mensajes_usuario["mensaje"])
    total_clasificacion =
hacerRecuentoDatasetUsuario(mensajes_usuario["mensaje"], pipeline)
    total_mensajes = len(mensajes_usuario)

    tags_sentimientos = ['POSITIVIDAD', 'NEGATIVIDAD']
    dimensiones = [total_clasificacion[2]+total_clasificacion[0],
total_clasificacion[1]+total_clasificacion[3]]
    tags_subniveles = ['Alta', 'Normal', 'No alarmante',
'Preocupante']
    dimensiones_subniveles = [total_clasificacion[2],
total_clasificacion[0], total_clasificacion[1],
total_clasificacion[3]]
    colores_tags_sentimientos = ['#00FF74', '#FF4C4C']
    colors_tags_subniveles = ['#01B001', '#00FF00', '#FB1818',
'#FF2700']

    bigger = mp.pie(dimensiones, labels=tags_sentimientos,
colors=colores_tags_sentimientos,
                    startangle=45, frame=True, counterclock = False)
    smaller = mp.pie(dimensiones_subniveles, labels=tags_subniveles,
                     colors=colores_tags_subniveles, radius=0.66,
                     startangle=45, labeldistance=0.7, counterclock =
False)
    centre_circle = mp.Circle((0, 0), 0.3, color='white')
```

```
    fig = mp.gcf()
    fig.gca().add_artist(centre_circle)
    mp.axis('equal')
    mp.tight_layout()
    st.write(fig)
    mp.savefig('sentimientos_usuario_XXX.jpg')
    #mp.show()

    etiquetas_fechas = mensajes_usuario["fecha"].unique()
    clasif_mensajes_porfechas = []
    recuento_positivos = []
    recuento_negativos = []
    recuento_muynegativos = []
    for fecha in etiquetas_fechas:
        recuento =
hacerRecuentoDatasetUsuario(mensajes_usuario[mensajes_usuario["fecha
"] == fecha]["mensaje"], pipeline)
        clasif_mensajes_porfechas.append([fecha, recuento])
        recuento_positivos.append(recuento[0] + recuento[2])
        recuento_negativos.append(recuento[1] + recuento[3])
        recuento_muynegativos.append(recuento[3])

    anchor_barras = 0.2

    from matplotlib.ticker import MaxNLocator

    fig1, ax = mp.subplots()

    ax.bar(etiquetas_fechas, recuento_positivos, anchor_barras,
label='Positivos', color=colores_tags_sentimientos[0])
    ax.bar(etiquetas_fechas, recuento_negativos, anchor_barras,
bottom=recuento_positivos, label='Negativos',
color=colores_tags_sentimientos[1])
    mp.plot(recuento_muynegativos, 'y')

    ax.yaxis.set_major_locator(MaxNLocator(integer=True))
    ax.set_ylabel('Recuento +/-')
    ax.set_title('Evolución del estado de ánimo')
    ax.legend()
    st.write(fig1)
    mp.savefig('evolucion_sentimientos_XXX.jpg')
    mp.show()


    data_load_state.text('He aquí los resultados del procesamiento
evaluador de los mensajes:')
```

5. RESULTADOS

5.1. Métricas predictivas

Tenemos más de 21’3 millares de registros de entrenamiento, que están clasificados bien como positivos o como negativos (cabe recordar que en el preprocesamiento, modificaremos la clasificación, de modo que pueda haber sentimientos extremos).

En base a estos, que obedece a una distribución normal gaussiana, y la aplicación de un *pipeline* que combina la aplicación de las SVMs lineales múltiples con técnicas previas de optimización en torno a la vectorización IDF y el estudio del lexicón, obtenemos estos resultados:

```
Exactitud = 0.7866492146596858
Classification report:
              precision    recall  f1-score   support

muy negativo       0.83      0.97      0.89       574
muy positivo       0.77      0.89      0.83       594
    negativo       0.76      0.67      0.71       946
    positivo       0.80      0.73      0.76       942

    accuracy                           0.79      3056
   macro avg       0.79      0.81      0.80      3056
weighted avg       0.79      0.79      0.78      3056

[[[2366  116]
  [  19  555]]

 [[2302  160]
  [  63  531]]

 [[1906  204]
  [ 312  634]]

 [[1942  172]
  [ 258  684]]]
```

Figura 6 – Métricas de evaluación aplicadas sobre el modelo predictivo de clasificación en uso para esta solución

Como se puede observar, la exactitud, que mide la tasa de casos en los que el modelo predictivo ha acertado, es superior al setenta y ocho por ciento (de hecho, podríamos aplicar un redondeo decimal que derivase en el incremento de una unidad).

Mientras, en nuestra tabla de clasificación, observamos que a la hora de clasificar sentimientos extremos, la precisión (contempla los positivos tanto verdaderos como falsos) es menor con respecto a la cobertura (en vez de valorar los falsos positivos, contempla falsos negativos).

No obstante, nos interesa más el hecho de que la puntuación F1 sea, en ambos casos, superior al ochenta por ciento. Apreciamos así un rendimiento combinado bastante satisfactorio dado que la ponderación conjunta es bastante superior a los dos tercios.

Así nos conformamos de que los sentimientos extremos, que son los más destacables, sobre todo los negativos (por si hubiera alguna posibilidad de desembocadura en suicidio o acto lesivo), son controlados con una "precisión general" muy elevada (casi 9/10).
En cambio, en relación a las categorías no extremas, sí que nos interesa que la precisión sea superior a la cobertura, sobre todo, por lo importante que es no generar falsas alarmas o asignaciones incorrectas, derivadas de clasificaciones imprecisas.

No nos interesa poner en riesgo la precisión del diagnóstico, lo cual, por motivos obvios, es bastante más grave cuando pasan desapercibidos los sentimientos negativos, extremos o no (en estos casos sí que puede haber un cuadro patológico a tratar).

A continuación, exponemos una serie de matrices de confusión, de las cuales, hay una por cada posible categoría de clasificación, en el orden que sigue: "muy negativo", "muy positivo", "negativo" y "positivo".

No hay caso alguno de "matriz perfecta", es decir, de resultado algebraico en el que hubiese una diagonal con ceros única y exclusivamente, que se basase en la ausencia de valores erróneamente clasificados.

En todos los casos, apreciamos cómo hay más "positivos verdaderos" que "positivos falsos" así como menos "negativos falsos" que "negativos verdaderos". Eso sí, la cantidad de falsas asignaciones negativas es relativamente ínfima cuando hablamos de categorías extremas.

5.2. Ejemplo de evaluación (caso concreto)

Como se dijo anteriormente, una vez que se hicieron las labores de entrenamiento y evaluación del conjunto de datos o *dataset* previamente asignado, remitimos al sistema un conjunto de mensajes con una fecha asignada.

Los casos de uso son muy heterogéneos pero sirvámonos de esta relación:

Fecha	Mensaje
04/08/2021	El menú del día es delicioso
04/08/2021	El salmón sí que ha estado delicioso hoy también
04/08/2021	Hemos estado viendo una película muy entretenida. Me ha gustado la escena de los dos animales.
05/08/2021	Estoy aburrido porque no tengo planes de fiesta
05/08/2021	Mi mejor amiga me ha invitado a su casa.
05/08/2021	Hemos celebrado una fiesta maravillosa
05/08/2021	Me hizo mucha ilusión recibir una colonia de regalo
06/08/2021	Mañana me voy a la playa con mis grandes amigos
06/08/2021	Estoy emocionada, hacía tiempo que no veraneaba
06/08/2021	El mar transmite calma
06/08/2021	Me duele la cabeza, estoy cansada
07/08/2021	Me da miedo nadar a gran distancia
07/08/2021	Las medusas me ponen muy nerviosa. Me entra ansiedad
07/08/2021	La paella de hoy ha estado deliciosa
08/08/2021	Me gustaría pasear por la orilla
08/08/2021	Pasear no me pareció mala idea
09/08/2021	Estoy muy entretenida. No echo de menos mi casa.

Tabla 3 – Relación de mensajes de usuario (ejemplo) que se remite para su clasificación

En base a la misma, obtenemos los siguientes resultados:

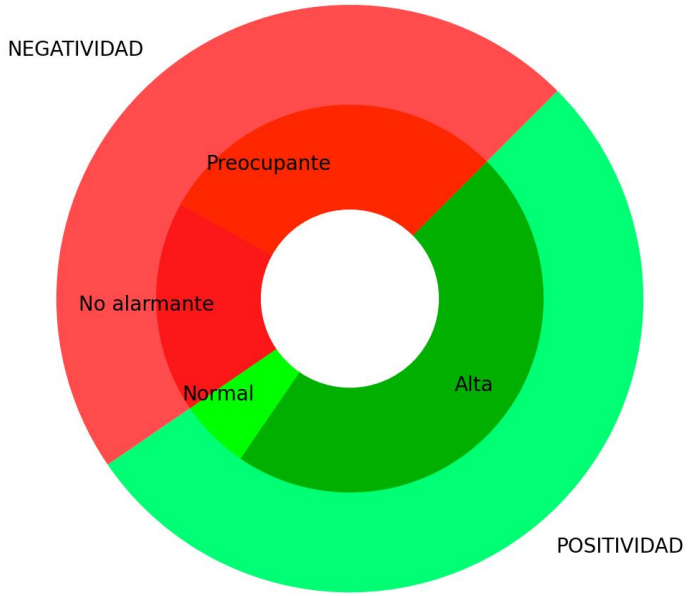


Figura 7 – Ejemplo de diagrama circular multinivel para el caso concreto a estudiar

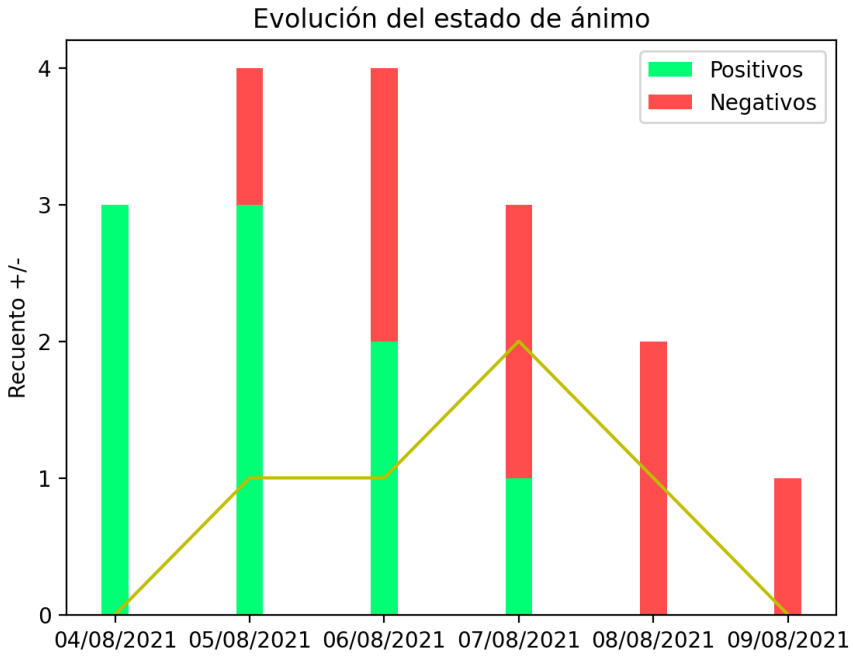


Figura 8 – Ejemplo de diagrama de barras con superposición de línea para el caso concreto a estudiar

Se puede observar que no solo que hay más sentimientos positivos que negativos, sino que la mayoría de sentimientos favorables presentan una positividad bastante elevada, aunque va decayendo a lo largo de la secuencia temporal.

Al mismo tiempo, observamos no solo que los sentimientos positivos empiezan a decae cuando van apareciendo los negativos (a partir del 6 de agosto), sino que en la mayoría de casos hay, al menos, un sentimiento de negatividad extrema (factor alarmante).

6. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

6.1. Expectativas de la solución

Consideramos que como prototipo, la solución sienta unas bases en la provisión de medios informáticos que ayuden al facultativo a detectar posibles cuadros de patologías psiquiátricas así como a prevenir males mayores derivados de estos.

Saber si un usuario tiene, por ejemplo, demasiados sentimientos negativos, y la mayoría de estas transmisiones son de una categoría extrema, puede ayudar bastante a un facultativo a aumentar su seguridad sobre el juicio clínico a dictaminar.

Por cierto, se insiste, en las ejemplificaciones, en el uso de la misma por parte de un facultativo, dado que, por el momento, creemos que ha de servir como una herramienta auxiliar, y no como un sustituto del personal sanitario así como tampoco una mera herramienta de autodiagnóstico.

Dicho esto, es posible que la solución sea más sofisticada en un largo plazo, trascendiendo la mera clasificación analítica de sentimientos dentro de un rango que contiene entre dos y cuatro categorías. Pero sobre esta cuestión profundizaremos en el siguiente apartado.

En cualquier caso, para finalizar, dadas las aspiraciones de prototipado y base, se puede decir que se han cumplido unas expectativas iniciales (no interpretemos esto como una incompletud en el sentido más literal).

6.2. *Deep Learning*, en aras de una solución más sofisticada

Podemos decir que el *Deep Learning* (DL) será uno de los principales responsables del hecho de sofisticación y potenciamiento de la atención médica y psicosocial por parte de lo que conocemos como Inteligencia Artificial (IA).

Uno de sus ámbitos de aportación más relevantes será el diagnóstico por imagen, por medio de las redes neuronales convolucionales , diseñadas para trabajar con datos de imágenes bidimensionales, aunque pueden trabajar con una o con tres dimensiones.

Estas llevan a cabo una operación lineal llamada “convolución”, la cual multiplica con la entrada un conjunto de pesos, bien en un mero arreglo unidimensional o una estructura de dos dimensiones conocida como núcleo o filtro, por medio de la entrada.

El filtro en cuestión puede diseñarse para detectar un tipo específico de característica en la entrada, lo que permite al filtro, por medio de la imagen de entrada entera, descubrir esa característica en cualquier punto de la imagen.

De todos modos, no vamos a dedicar una proporción mayor del apartado a explicar el funcionamiento de estas redes neuronales. Básicamente se ha querido hacer una especie de pincelada para “orientar” en cierta medida al lector.

Más bien, se mencionan, porque podrían ayudar a detectar tanto cuadros externos de autolesión como, especialmente, patologías neurológicas que pudieran desembocar en serias alteraciones de la salud mental del paciente.

Al mismo tiempo, conviene no olvidar la repercusión que puede tener el uso de redes neuronales en el procesamiento de lenguaje natural, en la medida en la que se pueden clasificar elementos no necesariamente gramaticales. Los resultados pueden tener un origen de audio (voz).

Con ello, se plantea un desafío dado que al tratar con información médica, las labores de detección, análisis y clasificación resultan ser mucho más complejas (hay que detectar los elementos clínicos, establecer relaciones con significado y obtener información provechosa).

Cabe indicar, a la luz de lo anterior, que una posible mejora del sistema, en sí, implicaría poder dar indicaciones estimadas sobre enfermedades concretas (partiendo de manuales como el DSM), más allá de contrastar la negatividad o positividad de los sentimientos.

6.3. Liberación del código

Uno, personalmente, se reserva el derecho a la objeción de conciencia en lo concerniente a la propiedad intelectual, que es considerada, por parte del mismo, como una especie de proteccionismo intelectual con rango censor y obstaculizador.

Con lo cual, el código de este proyecto será liberado, de modo que se pueda decir que hablamos de un prototipado de código abierto. En nuestro caso, estableceremos las directrices de la Licencia Pública General de GNU.

7. CASO DE NEGOCIO

7.1. Introducción

Aunque se trate este proyecto de una especie de prototipo, no lo desconsideramos para su puesta en producción tras ofrecerlo a alguna entidad, dentro de lo que pudiera ser un proyecto piloto que fomentase los avances en telemedicina.

El propósito concreto de nuestra aplicación es tener un sistema que facilite a los profesionales sanitarios el diagnóstico de enfermedades mentales o, en base a ello, una mera prevención de otros males mayores.

La cuestión es que en ocasiones es mejor monitorizar, bien en directo o en diferido, cómo se expresa el paciente durante un rango de tiempo, porque en función de su situación, utilizará unos términos y expresiones u otras.

En base a ello, definiremos una serie de elementos del caso, sin entrar a detallar los detalles presupuestarios, que entrarán a formar parte del mismo (se insiste en que, para este trabajo, se considerará, más bien, lo relacionado con el proyecto piloto).

7.1. Clientes

Nuestros clientes potenciales serán, al menos, al principio, profesionales sanitarios que bien trabajen de manera autónoma o en equipos integrados en clínicas, universidades, institutos u hospitales.

Para nuestra fase inicial y piloto, llegaremos a un acuerdo con el equipo de psiquiatría de un hospital comarcal del área central de España. Este está compuesto por unos cuatro facultativos, de los cuales, dos trabajan en turnos de urgencias.

7.2. Unidades de producto

Nosotros ofreceríamos al cliente la instalación y despliegue de la solución en una instancia del servicio de *cloud* llamado *Azure*, de Microsoft. El “piloto” solo tendría que pagar una comisión de 120 euros cada mes (algo como el pago por uso, que servirá para compensar otros gastos).

Eso sí, puestos a definir una unidad de producto, podríamos referirnos a la solución en sí (mejor dicho, a la aplicación web, ya desplegada, sin excluir bajo ningún concepto lo relacionado con las bases de datos).

7.3. Personal implicado

Al menos, para este prototipo cuya puesta en producción es una aspiración, se considera que con un desarrollador y un médico-psiquiatra que le ayude a la hora de verificar los conjuntos de datos y los resultados, quizá sea suficiente.

En cuanto al personal sanitario, se preferirá a alguien que tenga entre 10 y 20 años de experiencia en la materia, sin importar sus conocimientos técnicos. Mientras, se espera que el desarrollador tenga conocimientos de Python, bases de datos SQL y NoSQL, y *machine learning*.

7.4. Gastos de marketing

Para el proyecto piloto, no ha sido necesario entrar en gestiones de una mayor trascendencia, puesto que se ha gestionado todo a través del jefe del departamento sanitario en cuestión, al que se le planteó de manera altruista la posibilidad de colaborar.

No obstante, de cara a futuro, no se descartaría pagar anuncios en *Google Ads* así como tampoco pagar algunas cuñas publicitarias en periódicos sanitarios y crear campañas específicas en redes sociales como *Linkedin* y *Twitter*.

ANEXOS

A.I. Licencia GNU GPL v3

Yo siento, tú me asistes (Sientmasistes) – Detector-clasificador de sentimientos que facilite conclusiones médicas posteriores

PLANIFICACIÓN Y PRESUPUESTO

En este apartado, expondremos el valor presupuestario del proyecto en curso, con independencia de las fases en las que se encuentre (desarrollo, producción, mantenimiento, etc.). Consideramos los siguientes conceptos:

- Recursos humanos
- Recursos de desarrollo
- Infraestructura
- Mantenimiento

Cabe recordar que el propósito de esta solución informática es sentar las bases de un prototipo práctico que facilite, tomando las ventajas del *machine learning*, la detección de posibles cuadros patológicos de enfermedades mentales o meros problemas de supervisión psicológica.

Mediante un conjunto de datos de considerable volumen, previamente entrenado y evaluado, podremos remitir mensajes de texto (procedentes de publicaciones en redes sociales, tablones de comentarios o conversaciones) para hacer una valoración predictiva.

En base a los resultados, podremos determinar si lo que se expresa contempla un sentimiento positivo, negativo, muy negativo o negativo. Posteriormente se visualizarán, permitiendo así al interesado hacer valoraciones clínicas sobre el usuario, conforme al caso de negocio.

Una vez dicho esto, se seguirá un orden de planificación para cuya mejor comprensión utilizaremos un cronograma relativamente sencillo de objetivos y fases, el cual expondremos a continuación:

Actividad	**1Q-JUL21**	**2Q-JUL21**	**1Q-AGO21**	**2Q-AGO21**	**1Q-SEP21**
Análisis y concreción de requisitos (15h)	X				
Extracción de datos de fuentes varias (25h)	X	X			
Preparación de la infraestructura de datos (10h)		X			
Diseño de la solución (15h)		X			
Implementación de la solución (45h)		X	X		
Pruebas (20h)			X	X	
Configuración de la infraestructura de despliegue (5h)				X	
Elaboración de la memoria (45h)	X	X	X	X	X

La relación de costes, a modo de propuesta presupuestaria (tomando como referencia un trimestre), se expone aquí (téngase en cuenta que de manera provisional, para el despliegue, hemos recurrido al servicio gratuito de *Heroku*):

Concepto	Unidad	Descripción	Coste unitario	Coste total
Desarrollador	1 (180 horas)	Personal encargado de diseñar, desarrollar e implementar la solución.	35€/hora	6300 €

Médico-psiquiatra	1 (14 horas)	Personal encargado de asesorar al desarrollador al evaluar los resultados y definir ciertos datos preprocesamiento.	12,5€/hora	175 €
Despliegue en infraestructura	1 (3 meses)	Instancia A2 v2 de Microsoft Azure en Red Hat Enterprise, con 4GiB de RAM y 20GiB de almacenamiento temporal.	97’2663€/mes	291,8 €
				6766,8 €

Aparte de ello, se consideraría recargar una comisión de uso que rondase los ciento veinte euros mensuales, siendo el valor más bajo el que abonasen las entidades sin ánimo de lucro (organizaciones sociales no gubernamentales).

Asimismo, se recuerda que el desarrollo de nuestra solución se inspira en una especie de simbiosis entre las metodologías de desarrollo prototipado e incremental (esta solución es ampliable, pero sirve como prototipo de base en fase posterior a la preproducción).

BIBLIOGRAFÍA

En este apartado expondremos la lista de referencias bibliográficas que nos han servido de respaldo y soporte a la hora de desarrollar este trabajo de planteamiento e investigación:

1) https://www.efe.com/efe/espana/sociedad/la-pandemia-de-trastornos-mentales-agudos-se-instala-con-el-fin-covid/10004-4560644
2) S. Liu, "AI market size 2018-2025," *Statista*, 07-Dic-2020. [En línea]. Disponible en: https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/. [Leído: 10-Jul-2021].
3) "Descubre los principales beneficios del Machine Learning," *Iberdrola*. [En línea]. Disponible en: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico. [Leído: 10-Jul-2021].
4) D. Faggella, "Machine Learning for Medical Diagnostics - 4 Current Applications," *Emerj*, 14-Mar-2020. [En línea]. Available: https://emerj.com/ai-sector-overviews/machine-learning-medical-diagnostics-4-current-applications/. [Leído: 10-Jul-2021].
5) "Tendencias de desarrollo en eHealth," *Campus Sanofi*. [En línea]. Disponible en: https://campussanofi.es/smart-hospital/noticias/tendencias-de-desarrollo-en-ehealth/. [Leído: 10-Jul-2021].
6) D. Jegen y C. Byers, "Healthcare is the next wave of data liberation," *TechCrunch*, 29-Abr-2021. [En línea]. Disponible en: https://techcrunch.com/2021/04/29/healthcare-is-the-next-wave-of-data-liberation/. [Leído: 10-Jul-2021].
7) "10 startups de salud que representan el futuro sanitario en España," El blog de Startupxplore, 16-May-2019. [En línea]. Disponible en: https://startupxplore.com/es/blog/10-startups-futuro-sanitario-espana/. [Leído: 10-Jul-2021].
8) J. C. Sobrino Sande, "Análisis de sentimientos en Twitter". Barcelona, España: Universitat Oberta de Catalunya, 2018. [En línea]. Disponible en: https://openaccess.uoc.edu/webapps/o2/bitstream/10609/81435/6/jsobrinosTFM0618memoria.pdf. [Leído: 10-Jul-2021].
9) N. Bianchi, *6 Business Examples of Sentiment Analysis in Action*, 01-Feb-2021. [En línea]. Disponible en: https://www.repustate.com/blog/sentiment-analysis-real-world-examples/. [Leído: 10-Jul-2021].
10) I. Tkachenko, *5 Sentiment Analysis Real-Life Applications & Examples*, 24-Abrr-2019. [En línea]. Disponible en: https://theappsolutions.com/blog/development/sentiment-analysis-for-business/. [Leído: 10-Jul-2021].
11) Gandhi, R., 2018. Support Vector Machine — Introduction to Machine Learning Algorithms. [en línea] Towards Data Science. Disponible en: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Leído: 26-Jul-2021].
12) Reza Baghaee, H., 2020. General classification hyperplane representation of SVM algorithm. imagen Disponible en: https://www.researchgate.net/figure/General-classification-hyperplane-representation-of-SVM-algorithm_fig5_330557084 [Leído: 26-Jul-2021].
13) MathWorks. 2021. Understanding Support Vector Machine Regression- MATLAB & Simulink. [en línea] Disponible en: https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html [Leído: 27-Jul-2021].
14) Van Vaerenbergh, S. and Santamaría, I., 2021. Métodos kernel para clasificación. [en línea] GTAS. Disponible en:

https://gtas.unican.es/files/docencia/APS/apuntes/07_svm_kernel.pdf [Leído: 27-Jul-2021].
15) Awasthi, S., 2021. Seven Most Popular SVM Kernels. [en línea] Dataaspirant. Disponible en: https://dataaspirant.com/svm-kernels/#t-1608054630728 [Leído: 27-Jul-2021].
16) "Sklearn objects: FIT() Vs transform() vs FIT_TRANSFORM() vs predict()," Analytics Vidhya, 28-Abr-2021. [en línea]. Disponible en: https://www.analyticsvidhya.com/blog/2021/04/sklearn-objects-fit-vs-transform-vs-fit_transform-vs-predict-in-scikit-learn/. [Leído: 6-Jul-2021].
17) J. Brownlee, "How do convolutional layers work in deep learning neural networks?," How Do Convolutional Layers Work in Deep Learning Neural Networks?, 16-Abr-2020. [en línea]. Disponible en: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. [Leído: 06-Sep-2021].
18) "¿Qué ES el DSM? ¿Afectará la NUEVA versión DSM-5 AL TDAH?," ¿Qué es el DSM? ¿Afectará la nueva versión DSM-5 al TDAH? [en línea]. Disponible en: https://www.fundacioncadah.org/web/articulo/que-es-el-dsm-iv-tr-afectara-la-nueva-version-dsm-5-al-tdah.html. [Leído: 06-Sep-2021].
19) "Licencias - proyecto GNU - free Software Foundation," GNU. [en línea]. Disponible en: https://www.gnu.org/licenses/licenses.es.html. [Leído: 01-Sep-2021].
20) R. Celis, Heroku: qué es, cómo funciona y para qué sirve, 2017. [en línea]. Disponible en: https://platzi.com/blog/que-es-heroku/. [Leído: 01-Sep-2021].