UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Monitorización de redes sociales para la detección de usuarios con ideación suicida

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Navarro Arenas, Miguel Angel

Tutor/a: Molina Marco, Antonio

Cotutor/a: Hurtado Oliver, Lluis Felip

CURSO ACADÉMICO: 2022/2023

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN (CASTELLANO)

La depresión y el suicidio son dos temas que están a la orden del día y que preocupan tanto a expertos como a la población en general. El término suicidio viene de las expresiones latinas “sui” y “occidere” que juntas significan: “matarse a sí mismo”. Ambos términos son muy importantes para la salud mental y padecerlos puede condicionar la vida de los individuos. La depresión es una enfermedad mental que puede afectar a cualquier persona, independientemente de su edad, género o estatus social. Si no se trata adecuadamente, la depresión puede llevar a pensamientos suicidas y, en casos extremos, al suicidio mismo. Es importante buscar ayuda si alguien cercano está experimentando síntomas de depresión o pensamientos suicidas. Nuestra idea en este trabajo es ayudar a acercar las inteligencia artificial y, en particular, el procesamiento de lenguaje natural a expertos, para facilitarles las tareas de detección temprana que tan importantes son para evitar las consecuencias de estas enfermedades. Como opción inicial proponemos una interfaz gráfica sencilla en la cual podemos hacer un muestreo sobre los 10 últimos tweets de usuarios para deducir si es posible que muestren indicios de cometer suicidio en un futuro próximo. Pero con el cambio en las políticas de twitter en los últimos meses, no se nos ha hecho posible trabajar con su API ya que los planes nuevos no ofrecen de manera gratuita el acceso a tweets y algunos tokens que tenían los profesores han caducado. Por tanto, proponemos una alternativa como versión 1.0: una interfaz en la que manualmente se introduzca un texto (que pueden ser los tweets) y mostrar por pantalla los resultados. También, como versión 1.1, trabajaremos sobre la API de Reddit para hacer algo similar pero en esta red social, la cual sabemos que si tiene una API gratuita y funcional. Para la parte de procesamiento de lenguaje natural, entrenamos varios modelos de inteligencia artificial para obtener los resultados, que se etiquetan como “suicide” y “non-suicide”. A partir de estos datos obtenidos, será el experto el que haga un estudio de la situación del individuo. Estos datos los tomará como punto de partida para aplicar posibles medidas de detección temprana y apoyo emocional y psicológico al individuo en cuestión. También hemos hecho un estudio de inteligencias artificiales que cumplen una función similar a la nuestra pero que están destinadas al lenguaje inglés.

RESUMEN (INGLÉS)

Depression and suicide are two issues that are the order of the day and that concern both experts and the general population. The term suicide comes from the Latin expressions “sui” and “occidere” which together mean: “to kill oneself”. Both terms are very important for mental health and suffering from them can condition the lives of individuals. Depression is a mental illness that can affect anyone, regardless of age, gender, or social status. If not treated properly, depression can lead to suicidal thoughts and, in extreme cases, suicide itself. It is important to seek help if someone close to you is experiencing symptoms of depression or suicidal thoughts. Our idea in this work is to help bring artificial intelligence and, in particular, natural language processing closer to experts, to facilitate early detection tasks that are so important to avoid the consequences of these diseases. As an initial option, we propose a simple graphical interface in which we can sample the last 10 user tweets to deduce if it is possible that they show signs of committing suicide in the near future. But with the change in twitter's policies in recent months, we have not been able to work with their API as the new plans do not offer free access to tweets and some tokens that teachers had have expired. Therefore, we propose an alternative such as version 1.0: an interface in which text is manually entered (which can be tweets) and the results are displayed on the screen. Also, as version 1.1, we will work on the Reddit API to do something similar but on this social network, which we know does have a free and functional API. For the natural language processing part, we train various AI models to get the results, which are labeled as “suicide” and “non-suicide”. From these data obtained, it will be the expert who makes a study of the situation of the individual. These data will be taken as a starting point to apply possible early detection measures and emotional and psychological support to the individual in question. We have also done a study of artificial intelligences that fulfill a function similar to ours but that are intended for the English language.

RESUMEN (VALENCIANO)

La depressió i el suïcidi són dos temes que estan a l'ordre del dia i que preocupen tant experts com a la població en general. El terme suïcidi ve de les expressions llatines "*sui" i "*occidere" que juntes signifiquen: "matar-se a si mateix". Tots dos termes són molt importants per a la salut mental i patir-los pot condicionar la vida dels individus. La depressió és una malaltia mental que pot afectar qualsevol persona, independentment de la seua edat, gènere o estatus social. Si no es tracta adequadament, la depressió pot portar a pensaments suïcides i, en casos extrems, al suïcidi mateix. És important buscar ajuda si algú pròxim està experimentant símptomes de depressió o pensaments suïcides. La nostra idea en aquest treball és ajudar a acostar les intel·ligència artificial i, en particular, el processament de llenguatge natural a experts, per a facilitar-los les tasques de detecció precoç que tan importants són per a evitar les conseqüències d'aquestes malalties. Com a opció inicial proposem una interfície gràfica senzilla en la qual podem fer un mostreig sobre els 10 últims tuits d'usuaris per a deduir si és possible que mostren indicis de cometre suïcidi en un futur pròxim. Però amb el canvi en les polítiques de *twitter en els últims mesos, no se'ns ha fet possible treballar amb el seu *API ja que els plans nous no ofereixen de manera gratuïta l'accés a tuits i alguns *tokens que tenien els professors han caducat. Per tant, proposem una alternativa com a versió 1.0: una interfície en la qual manualment s'introduïsca un text (que poden ser els tuits) i mostrar per pantalla els resultats. També, com a versió 1.1, treballarem sobre la *API de *Reddit per a fer una cosa similar però en aquesta xarxa social, la qual sabem que si té una *API gratuïta i funcional. Per a la part de processament de llenguatge natural, entrenem diversos models d'intel·ligència artificial per a obtindre els resultats, que s'etiqueten com "suïcide" i "senar-suïcide". A partir d'aquestes dades obtingudes, serà l'expert el que faça un estudi de la situació de l'individu. Aquestes dades els prendrà com a punt de partida per a aplicar possibles mesures de detecció precoç i suport emocional i psicològic a l'individu en qüestió. També hem fet un estudi d'intel·ligències artificials que compleixen una funció similar a la nostra però que estan destinades al llenguatge anglés

PALABRAS CLAVE

Depresión, suicidio, salud mental, enfermedad mental, inteligencia artificial, máquinas de vectores soporte, redes neuronales, BERT, Python, Flask, TensorFlow.

DEDICATORIA

A mi madre, por ser la persona encargada de mi educación siempre y es la que ha hecho que esté aquí hoy en día y me ha enseñado la importancia de aprender, del ser constante y perfeccionista. A Nuria, por ser la mujer de mi vida y apoyarme y quererme en estos últimos años incondicionalmente y por darme el apoyo para construir unos planes a futuro que hacen que me despierte cada día, dando lo mejor de mí para que sean posibles. Por ser la que me ha aguantado estos meses de duro trabajo e incertidumbre (y los que quedan). A mi yaya Pili, por haberme criado cuando los demás se estaban esforzando en sacar la familia adelante y enseñarme la importancia de la familia y el cuidado de estos. A mi padre, por enseñarme la importancia del esfuerzo día a día, del trabajo duro, pero a la vez de disfrutar la vida día a día. A mis tíos y tías, por estar siempre para echar una mano cuando fuera y por estar siempre preocupados de mí. A mis primos, por ser un referente para ellos y que mis conocimientos y ayuda les sirvan para sacar la mejor versión de ellos mismos y luchen por el futuro que quieren. A mis compañeros de piso y carrera, por haber vivido esta experiencia juntos durante estos años, que me han dado a los mejores hermanos postizos.

1.- INTRODUCCIÓN

La depresión y el suicidio son dos enfermedades o problemas de salud mental que afectan a un gran número de personas. A nivel mundial estos problemas son particularmente preocupantes. Se cree que esto se debe en parte a la falta de conexión social en la sociedad moderna, así como a la presión constante a la que se ven sometidos los individuos para tener éxito y ser productivos. Por ejemplo, la OMS estimó en 2019 que el 15% de los adultos en edad de trabajar padecían de algún trastorno mental y que, a lo largo del año se producen unas perdidas económicas de en torno al billón de dólares a causa de la depresión y la ansiedad [1].

1.1.- Contexto y justificación

Entre algunas de las causas que son consideradas como factores de riesgo para el intento de suicidio podemos encontrar el desempleo, la inseguridad laboral y financiera y la pérdida del puesto de trabajo [1]. La depresión y el suicidio son enfermedades complejas que no tienen una sola causa, pero es importante que se tomen medidas para prevenirlas y tratarlas adecuadamente. La OMS también publicó en 2019 en "Suicide worldwide in 2019" unas cifras que son apabullantes: 1 de cada 100 muertes a nivel mundial se produce por suicidio. 703.00 personas cometen suicidio anualmente. Esto nos indica que mueren más personas por suicidio que por VIH [2]. O también podemos saber que cada 40 segundos una persona se suicida o que 10,5 de cada 100.000 habitantes acaban suicidándose [3].

Gráfico de suicidio discriminando por ingresos y edades.

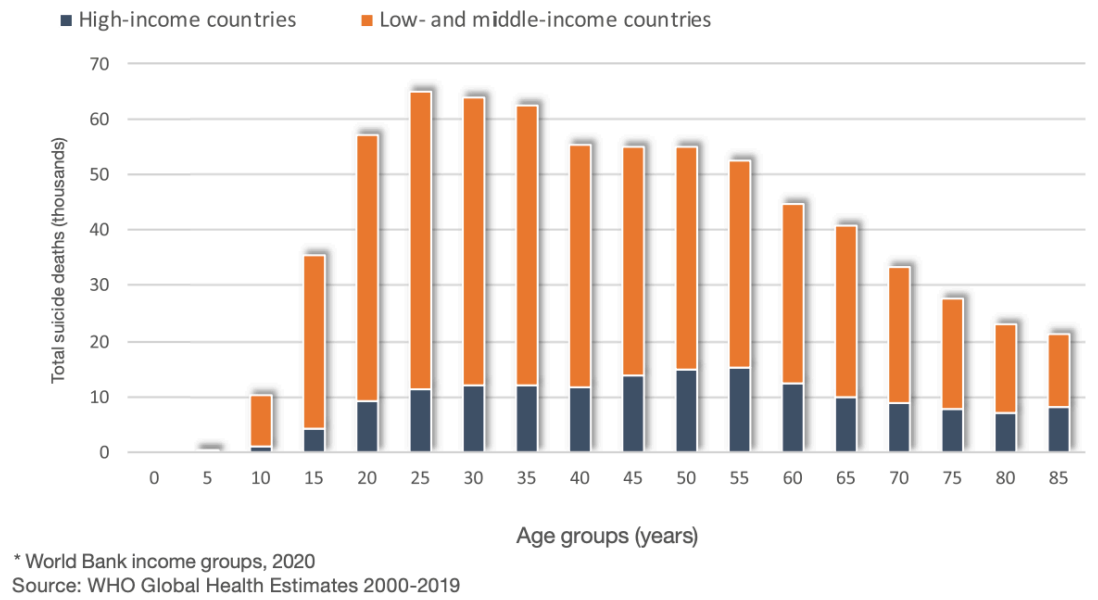

Figura 1: gráfico de suicidio discriminado por ingresos y edades

Gráfico de https://www.who.int/publications/i/item/9789240026643

Como podemos observar, estos problemas son algo que no afecta de la misma medida a ciudadanos que residen en países con una cantidad de ingresos menores. Se ha identificado que, en general, los países con una tasa menor de ingresos tienen mayores tasas de suicidio en comparación con los países más ricos. Esto se debe a varios factores, como la falta de acceso a servicios de salud mental, la falta de oportunidades económicas y laborales, la falta de conexión social, etc. En general, las personas que viven en países más pobres pueden enfrentar una mayor cantidad de estrés y dificultades, lo que puede aumentar el riesgo de desarrollar problemas de salud mental y, en última instancia, llevar al suicidio. A pesar de esta gráfica sabemos que la tasa de suicidios más alta (24,6 por cada 100.000 habitantes) se produjo en un país con una renta y unos ingresos más altos, Corea del Sur, datos del año 2019 [4].

Gráfico sobre el suicidio en personas entre 15 y 29 años

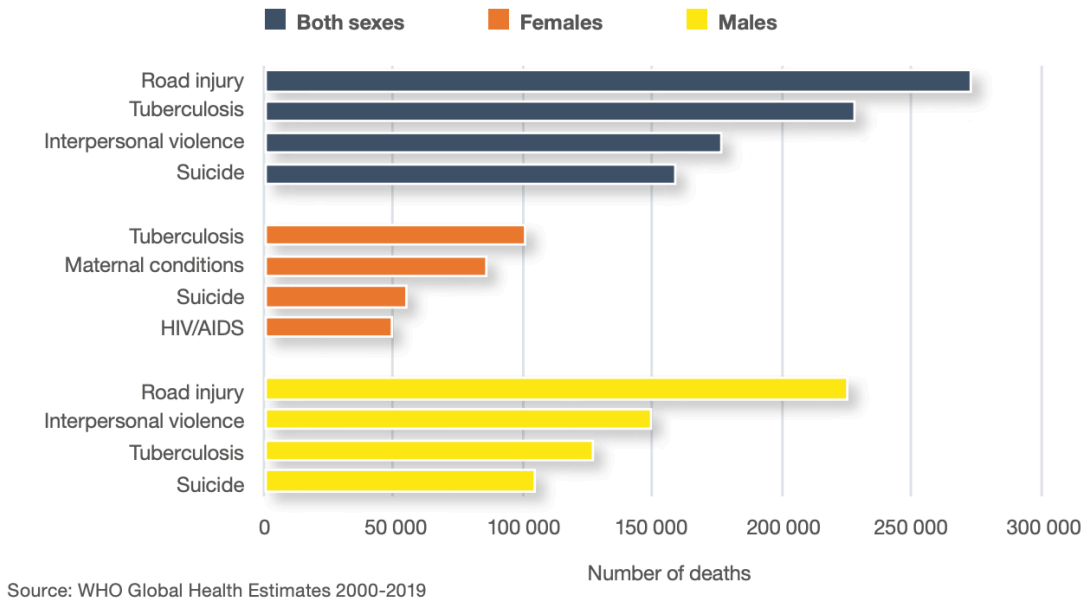

Figura 2: gráfico sobre el suicidio en personas entre 15 y 29 años.

Gráfico de https://www.who.int/publications/i/item/9789240026643

A través de las gráficas anteriores deducimos que uno de los rangos de edades donde se cometen más suicidios es entre los 15 y los 29 años. El suicidio ocupa la tercera causa de muerte entre las mujeres en este rango de edades y el cuarto entre los hombres. También vemos que se produce un mayor número de muertes por suicidio en hombres que en mujeres.

Debemos saber la importancia que tiene la prevención para evitar el suicidio. Se estima que por cada suicidio se producen entre unos 10 y 20 intentos de suicidio y es por ello que la prevención y el evitar la reincidencia es un factor clave para reducir la tasa de suicidio. Según Hawton K y Van Heeringen K el suicidio es una causa multifactorial que nunca es consecuencia de una sola causa o estresante [5], pero los principales y que más

veces se repiten son padecer un trastorno mental, los intentos previos de suicidio, eventos estresantes y las enfermedades crónicas o discapacitantes [6].

Por tanto, la prevención es muy importante para salvar vidas. Una intervención a tiempo hace que brindemos apoyo emocional y conexión social al individuo, además de promover la conciencia sobre la salud mental. Con la prevención, de manera directa, ofrecemos ayuda y acercamos los servicios de salud mental a las personas que más lo necesitan. Si podemos proporcionar esperanza, recuperación y bienestar a una persona en crisis, estamos evitando el suicidio y, por tanto, salvando vidas.

1.2.- Motivación

Una vez que hemos conocido todos los datos que hemos recopilado en torno al suicidio y lo importante que es la prevención para evitar que suceda, nuestra motivación es acercar las herramientas técnicas que tenemos en nuestra mano hoy en día a profesionales del mundo de la psicología, para orientarles a detectar y prevenir el suicidio en un entorno mucho más innovador, prometedor y fiable. Como informáticos, disponemos de la capacidad de analizar grandes volúmenes de datos. A través de ellos somos capaces de con tecnologías como las inteligencias artificiales de capturar patrones en la conducta del ser humano a la hora de expresarse de manera escrita en redes sociales, además de sus interacciones con otros usuarios, y detectar entre ellos aquellos que podemos considerar como susceptibles de conductas suicidas. La motivación personal también supone un gran papel, ya que la creación y estudio de una inteligencia artificial (o en nuestro caso varias) implica un desafío técnico de gran envergadura. Partimos desde el estudio, recopilación y procesamiento de datos, hasta el entrenamiento mismo, junto con la interacción con el usuario. Esto supone un reto personal enorme e involucra un espectro de conocimientos y habilidades muy amplios que atañen a muchas de las ramas principales que componen el grado de ingeniería informática. También, es un gran punto de motivación el conocimiento de que este tipo de herramientas pueden ayudar al bienestar de las personas a través de soluciones innovadoras, con un gran impacto a la sociedad y cumpliendo retos técnicos y científicos los cuales sean capaces de generar resultados significativos para el avance de la sociedad.

1.3.- Objetivos del trabajo.

1.3.1.- Objetivo principal

Nuestra idea principal es acercar la inteligencia artificial a los expertos que tratan la depresión y el suicidio. Los psicólogos, psiquiatras, terapeutas, trabajadores sociales clínicos o enfermeros especializados en la salud mental deberían ser capaces de utilizar herramientas tecnológicas avanzadas que les permitan facilitarles su tarea y acercarles cada vez más a un mundo que tiende a la digitalización y a un ser humano conectado a través de redes sociales. Por tanto, proponemos un sistema que utilice medios en los que el ser humano se sienta más libre de abordar su sufrimiento emocional, como pueden ser las redes sociales, y en nuestro caso en particular, Twitter y Reddit.

En el proyecto se expone cómo se ha creado una inteligencia artificial desde cero, partiendo desde la búsqueda de datasets que contengan la suficiente información y que estén correctamente etiquetadas, pasando por la limpieza de los datos, la traducción de

estos y la ubicación de todos ellos en un dataset resultante sobre el que vamos a trabajar. Nuestro objetivo es crear una herramienta exclusiva para el castellano, ya que tenemos varias ya desarrolladas en el inglés, pero ninguna que sea puramente en español. Posteriormente, estudiaremos varias aproximaciones de aprendizaje automático (máquinas de vectores soporte y redes neuronales) para ver cuál de ellos se ajusta de mejor manera a nuestro modelo. Y, por último, crearemos una interfaz para poner más a mano estas herramientas a los profesionales de la salud mental y hacerles más fácil el manejo de nuevas tecnologías. La interfaz consta de una aplicación web que corre sobre Python y Flask, la cual conecta con la API de Twitter para poder mostrar de vuelta los tweets. Se ha hecho un esfuerzo para utilizar tecnologías de vanguardia y optimizar lo máximo posible tanto los resultados como la interacción con el usuario, adaptando todo el proceso para la creación de una solución óptima y sencilla.

1.3.2.- Objetivos secundarios

Hay que tener en cuenta que las personas que usen la aplicación no son personas expertas en el campo de la informática, por tanto, debemos adaptar el uso de la aplicación para ellos y que sea una herramienta que les apoye en su trabajo de prevención. La idea es mostrarle varios resultados con los cuales ellos serán capaces de inducir una respuesta y actuar en consecuencia.

También es muy importante el estudio de ambos en la sociedad actual. Debemos observar cómo los casos de suicidio han ido aumentando paulatinamente a lo largo de los años en España, los primeros registros que se tienen sobre el suicidio vienen del año 1960, donde se cometieron un total de 1670, hasta el año 2021, donde se han efectuado un total de 4003 suicidios. También, es muy importante saber discernir entre las mejores tecnologías que debemos usar. Para ello, partimos de varios modelos de aprendizaje automático, como las máquinas de vectores soporte o las redes neuronales, entre los cuales debemos saber elegir cuál de ellos se adapta mejor a nuestro modelo, teniendo en cuenta las pruebas que vamos a realizar sobre cada uno, partiendo del dataset que hemos obtenido.

Por último, este proyecto tiene como objetivo mostrar los conocimientos que como alumno he obtenido durante toda la carrera, especializándolos en la rama de computación. Para ello he querido basar mi trabajo final de grado en un proyecto que suponga un reto a nivel personal y en el que pueda demostrar mis conocimientos adquiridos en la rama de computación. A parte de todo lo que atañe a la rama de computación, también he decidido que era necesario añadir parte de otros conocimientos que he adquirido en la carrera, como son el desarrollo de frontend y de tecnologías de aplicaciones web. Para ello, me he apoyado en mi experiencia profesional en la cual he aprendido Flask para desarrollo de aplicaciones web. Creo que al ser un framework que utiliza Python engloba perfectamente todo el desarrollo que es necesario para este trabajo, ya que hace que el eje central del mismo sea Python y que sea el lenguaje íntegro para todo el proyecto, desde principio a fin. Para mi como informático, este proyecto supone el culmen de lo que en nuestro gremio es llamado "**the curse of programming**" que sería algo así como nuestra maldición, algo que hemos firmado sin quererlo al desarrollarnos como programadores y que es algo con lo que tenemos que vivir día a día. Esto en breves palabras es que debemos estar siempre al día, actualizados y teniendo en cuenta las tendencias en las que se mueve nuestro mundo. Desde mi punto de vista es algo sumamente importante y que hace que nuestra profesión sea algo tan difícil pero a la vez intrigante y desafiante. A la vez que crecemos como informáticos y programadores, crecemos como personas y estamos cada

vez más a la orden del día, siendo la vanguardia de nuestro sector y de muchos otros que necesitan de nosotros.

1.3.3.- Identificación de resultados esperados

Nuestra misión es obtener una aplicación web que sea capaz de mostrar por pantalla unos resultados que hayan sido obtenidos con el entrenamiento de una inteligencia artificial. Para ello, esperamos obtener buenos resultados en varios aspectos:

- La aplicación web: esperamos una aplicación que sea capaz de integrar las diferentes inteligencias artificiales previamente entrenadas. Debemos ser capaces de recopilar y analizar los datos relevantes de unos determinados usuarios en Twitter y Reddit. Para ello, aplicamos interfaces claras y concisas para los usuarios expertos. La interfaz debe ser accesible para diversos usuarios especializados, independientemente de sus niveles de habilidad y conocimientos tecnológicos.
- Eficiencia en la detección de textos suicidas y posible notificación de riesgo y ayuda: debemos esperar una alta precisión de nuestros modelos para que la detección sea lo más correcta posible. Nuestros datasets sobre las cuales partimos y los resultados obtenidos deben ser fiables. Debemos de ser capaces de alertar a los usuarios sobre los riesgos detectados en base a nuestros resultados, dando pie a posibles medidas de prevención sobre el suicidio si los datos son muy desfavorables.

Estos resultados constituyen los principales logros que queremos alcanzar y nuestros resultados queremos que signifiquen un avance en la detección temprana y la intervención en situaciones de riesgo, así como la promoción de la salud mental en entornos digitales y redes sociales.

1.4.- Metodología

En este caso se ha seguido una metodología sencilla a la hora de la realización del trabajo. Esta consistía en varias reuniones semanales a través de Microsoft Teams en las que se iba hablando del proceso de la semana anterior y de los siguientes procesos a realizar, algo muy similar a una metodología SCRUM con cliente, en la que los tutores iban dando las pautas a seguir y se debatían cuáles eran los siguientes hitos que se debían cumplir.

Como hemos dicho anteriormente, se ha usado Google colab de manera conjunta para ver el desarrollo en cuanto al código. También se ha usado un repositorio en GitHub para ir albergando algunas partes del código que no se realizaban en colab, así como la parte de la interfaz que se ejecuta a nivel local. El repositorio a parte ha servido como copia backup de todo el proceso, cuando se avanzaba en algún cuaderno de colab se descargaba en formato ".ipynb" y se guardaba en el repositorio para tener los códigos en varias fuentes en caso de que alguna se pierda o no esté accesible.

Dividiremos el desarrollo del trabajo en tres fases: la primera será lo equivalente a un sistema ETL(extracción, transformación y carga de los datos), la segunda fase de

entrenamiento de los distintos mosdelos y tercero la parte de muestra de datos y creación de la aplicación.

- **Primera parte:** durante esta parte hemos realizado una búsqueda de datasets adecuados para nuestro propósito. Hemos hecho hincapié en su contenido y procedencia. Partimos de un documento titulado "Suicidal Ideation Detection: A Review of *Machine Learning* Methods and Applications". Al final del archivo se adjuntan algunos datasets, con sus autores, la cantidad de datos que contienen y de qué red social proviene su información. También hemos hecho énfasis en la manera en la que estas hayan sido etiquetadas, es decir, que lo hayan hecho correctamente con la ayuda de profesionales o con herramientas que hayan demostrado altas capacidades de etiquetado y pocos errores. Posteriormente, a la hora de traducir, optamos por la opción de traducir usando inteligencia artificial preentrenada. Por último, se unieron los datasets traducidas en una base final y se seleccionó la información más relevante para entrenar el modelo. Todo se realizó en Google Colab.
- **Segunda parte:** consistente en todo lo que el entrenamiento atañe. Hemos decidido probar con varios modelos de inteligencia artificial: máquinas de vectores soporte ó SVM, redes neuronales y redes neuronales + BERT. Estos son los modelos que consideramos que eran los que mejor resultados iban a arrojar. Para la parte del entrenamiento hemos usado también google colab y tardis (esta última es una máquina del por el grupo de investigación ELiRF [29] de la UPV con una gran capacidad de cómputo, destacando por tener dos tarjetas gráficas NVIDIA GeForce RTX 3090) para aquellas que ocupan más capacidad de memoria y cómputo como es la ejecución con BERT. También en esta parte entrenaremos modelo de SVM con el corpus por completo en inglés y probaremos a traducir el texto de entrada del español al inglés, para ver si los resultados obtenidos son similares.
- **Tercera parte:** por último, hemos creado la interfaz que muestre los datos obtenidos. Decidimos usar Flask para ello con la finalidad de que todo el proyecto usara Python como motor, dandole uniformidad y menos complejidad para que pueda seguir creciendo en un futuro. La interfaz es sencilla y se ha utilizado HTML y CSS para realizarla, con la ayuda de Bootstrap para darle un mejor aspecto. La conexión con Twitter se ha realizado con la librería tweepy.

1.5.- Estructura

Tenemos que diferenciar dos partes principales en la estructuración del trabajo: por un lado la parte de desarrollo y por otro parte la memoria. La primera parte es donde desarrollaremos todo el código y todos los experimentos que decidimos lanzar. En la segunda, plasmamos todos los resultados que hemos obtenido y las conclusiones a las que hemos llegado, que es la parte de desarrollo de la memoria.

Vamos a dividir la estructura de la memoria en las siguientes partes:

- Estado del arte: donde observamos las herramientas similares a la nuestra que se encuentran ya desarrolladas así como las opciones que hemos tomado y el porqué se han descartado en los procesos anteriormente nombrados.

- Contexto tecnológico: capítulo donde mostramos cuál es el panorama actual de la inteligencia artificial, cuáles son las herramientas más punteras y las decisiones que nos han llevado a usar las tecnologías de nuestro proyecto.
- Análisis del problema: abordamos el problema y entendemos el desglose de las partes que hemos mencionado anteriormente. Aplicamos las técnicas de análisis de proyectos estudiadas en la carrera para solucionar el proyecto.
- Solución propuesta: aquí mostramos nuestras soluciones obtenidas en las fases del desarrollo. Desde el planteamiento inicial de las mismas hasta los resultados obtenidos.
- Pruebas: a través de los resultados obtenidos realizamos pruebas para validar que todo esté correcto en nuestras resoluciones.
- Conclusiones: punto final del proyecto donde también mostraremos el futuro del proyecto y hacia donde queda encaminado.

2.- Estado del arte.

En este apartado mostraremos un análisis del estado actual de aplicaciones y tecnologías relacionadas con la detección del contenido suicida tanto en inglés como en español, así como las soluciones existentes que hay en el mercado que cumplen una misión igual o similar a la nuestra. Analizaremos los enfoques de cada una de ellas, así como sus características, funcionalidades y limitaciones. Con este análisis obtenemos las posibles debilidades y fortalezas de nuestro proyecto, así como las de los competidores. Esto lo haremos con la finalidad de obtener una propuesta de valor y enfocar nuestro proyecto.

El análisis de sentimientos es una herramienta muy extendida dentro del mundo del *machine learning* y que sirve como punto de partida del análisis del estado del arte. Hay muchas herramientas de este tipo, pero las principales están basadas en BERT, que es un encoder para transformers creado por Google [28] que permite un análisis del lenguaje muy completo. La funcionalidades de estos análisis no son solo con temas relacionados con la salud, principalmente son usados en comercio para analizar las tendencias de los clientes y lo satisfechos que están con un determinado servicio o producto.

Dentro de la detección de la ideación suicida en redes sociales, se han establecido varias competiciones a nivel internacional y varios foros que se dedican a objetivos similares, como por ejemplo eRisk, que varios años atrás estudió la prevención de la depresión que sabemos que tan relacionada está con el suicidio [11].

Pero sin duda uno de los proyectos más importantes dentro de la detección de ideación suicida fue publicado en el año 2021: "Community-level Research on Suicidality Prediction in a Secure Environment: Overview of the CLPsych 2021 Shared Task" [30]**.** En ella se explora la predicción del riesgo de suicidio utilizando datos confidenciales de Twitter dentro de un enclave de datos seguro. Alcanzó puntuaciones muy altas con muy alta precisión, pero sin olvidad que se adhirió a estrictos estándares de seguridad y privacidad. Los científicos obtuvieron grandes conocimientos sobre la salud mental en redes sociales, sentando las bases para otros estudios del procesamiento del lenguaje natural junto con la salud mental y las tecnologías más modernas de nuestros tiempos, como las redes sociales.

También es muy importante en este apartado tener en cuenta los datasets que tenemos disponibles. Muchas de ellas las veremos posteriormente en el apartado 5.2, las cuales traen su información sobre todo de Twitter y Reddit. Lo malo de estas es que muchas de ellas no son de dominio público por la protección de datos, por lo que se nos ha hecho muy difícil el acceso a las mismas. También hay algunos datasets, relacionadas con nuestra temática, disponibles en plataformas como Kaggle [12] o HuggingFace [13].

2.1.- Aplicaciones existentes relacionadas

Hay varias aplicaciones existentes que tienen una misión similar a la nuestra, las cuales son interesantes analizar ya que podemos aprender de ellas para mejorar la nuestra. Las principales son STOP (Suicide prevenTion in Social Platforms) [14], Crisis TextLine [15], The Trevor Project [16] y Samaritans Radar [17].

STOP (Suicide prevenTion in Social Platforms)

Es un proyecto de investigación que estudia los problemas mentales en las redes sociales usando inteligencia artificial. Busca patrones de riesgo relacionados con el suicidio, la depresión o los trastornos alimenticios. Proporciona alertas y recursos de ayuda a los usuarios identificados como en situación de riesgo. Está desarrollado en colaboración con la universidad Pampeu Fabra de Barcelona. Está dirigida por Ana Freire, doctora en ingeniería informática que ejerce en la citada universidad.

Crisis TextLine

Proviene de una organización mundial sin ánimo de lucro. Su aplicación consiste en un chatbot a través de mensajes de texto que usa inteligencia artificial para detectar patrones de comportamientos suicida, crisis y riesgo de suicidio en las conversaciones. Está disponible 24 horas al día los 7 días de la semana. Presume de tener el dataset sobre salud mental más grande en el mundo. Permite una intervención temprana y proporciona apoyo a personas en crisis.

The Trevor Project

Es una asociacion sin animo de lucro estadounidense que ofrece servicios de mensajeria, chatbot y llamadas. esta asociacion se encarga de apoyo y prevención a personas jóvenes de la comunidad LGBTQ+. Tiene una gran cantidad de consejeros entrenados y también tiene servicio 24/7. Usa una inteligencia artificial que les permite analizar en tiempo real las conversaciones y detectar mensajes con patrones suicidas.

Samaritans Radar

Es un plug-in para Twitter que usa el procesamiento de lenguaje natural para detectar los posts de las personas. Fue cerrado por completo en el año 2015, después de que se suspendiera en 2014. Consistía en analizar lo que publicaba una persona que era identificada por algún otro usuario o que por sí mismo se daba de alta en la plataforma. Si detectaba un caso positivo, enviaba un email para alertar al usuario por email, dándole las pautas para proteger a esa persona o a si mismo.

Todas estas herramientas son muy similares entre ellas y a la nuestra, quizá la más parecida sea STOP, pero todas ellas están entrenadas y pensadas para usarse con el lenguaje inglés. Nuestra idea es proporcionar una herramienta que sea íntegra del español, desde el entrenamiento hasta su propio uso.

2.2.- Tecnologías de inteligencia artificial

La inteligencia artificial empieza a surgir entre el mundo de la computación en torno a la década de los años 40. Pero sus inicios no tuvieron mucha repercusión, ya que era algo demasiado teórico y poco práctico para la capacidad de cómputo del momento. No fue hasta los años 50 cuando Alan Turing se cuestionó si un computador era capaz de tener propiedades humanas y pensar por sí mismos, fue cuando sentó las bases del “Test de

Turing” [26] Las bases del test son que una persona mantiene una conversación a la vez con una computadora y otra persona, sin saber quien es quien y la misión del primero es saber si la máquina es capaz de responder a las mismas preguntas de un humano. El término de inteligencia artificial se acuñó en una conferencia (Dartmouth)del año 1956 llevada a cabo por los considerados padres de la inteligencia artificial: John McCarty, Marvin Misky, y Claude Shannon [27]. No fue hasta los años 90 que no se hicieron avances significativos en este campo, con la creación de los agentes inteligentes, que son entidades capaces de observar y entender su entorno y ejecutar acciones en base a esos conocimientos que ha adquirido. Las máquinas de vectores soporte fueron creadas en estos años. Son capaces de separar datos en categorías usando un hiperplano que se ha creado a través de un espacio de características. Para muchos, el 1997 fue en año de inflexión de los sistemas inteligentes, cuando por primera vez en la historia, la máquina de IBM, Deep Blue, fue capaz de ganar en una partida de ajedrez al campeón del mundo en ese momento, Gari Kasparov.

Las redes neuronales artificiales son estructuras basadas en el funcionamiento del cerebro humano que han cobrado gran importancia desde la década del 2010. A través del deep learning y su uso conjunto con las redes neuronales multicapa, se han conseguido grandes resultados en el reconocimiento de imágenes y el procesamiento del lenguaje natural.

En el año 2011, otro supercomputador de IBM, llamado Watson, fue capaz de ganar el programa de televisión estadounidense Jeopardy¡. En este año también Apple implementó por primera vez en su modelo de teléfono móvil, iPhone 4s a Siri. No fue hasta el año 2014 que una máquina fue capaz de superar el Test de Turing. Este fue considerado el mayor avance en la inteligencia artificial en muchos años. En el año 2016 el algoritmo de inteligencia artificial, Alpha Go fue capaz de ganar al campeón mundial del popular juego Go. En el año 2017 el software Librantus también fue capaz de vencer a cuatro de los mejores jugadores de póker del momento. El 2018 fue el año de la inteligencia artificial en los automóviles, sentando el precedente Tesla de la conducción autónoma. El año 2020, durante la pandemia, los algoritmos de inteligencia artificial cobraron gran importancia para la detección de focos de contagio, pacientes cero, principales causas, ayuda a médicos y enfermeros, etc.

El año 2015 fue el de la creación de la empresa OpenAI con la finalidad de crear herramientas de IA de alta calidad y librea acceso para la sociedad. Fue en el año 2022 en el que crearon el mayor de los chatbots de inteligencia artificial, ChatGPT, basado en la tecnología GPT-3.5. Para muchos es el próximo gran avance en el mundo moderno, junto con otros grandes hitos como la creación de internet.

3.- Contexto tecnológico

En este apartado trataremos qué tecnologías son las más adecuadas para cada uno de las fases, etapas o ta de nuestro proyecto. Decidiremos cuál se ajusta más a nuestras necesidades de entre las más punteras en su ámbito.

3.1.- Lenguajes de programación para el tratamiento de datos

Los principales lenguajes de programación para el tratamiento de datos para inteligencia artificial son: Python, SQL, R y Scala. Veremos cuáles son las fortalezas y debilidades de cada uno de ellos y veremos cuál es nuestra elección.

Python

Figura 3: logo python

Es un lenguaje de programación dinámico, es decir, cuyas variables pueden tomar diversos valores de diferente tipo. Es también un lenguaje interpretado, es decir, utiliza un intérprete para ejecutar el programa, en lugar de un compilador o ensamblador. Los intérpretes realizan una traducción instrucción a instrucción, a medida que sea necesaria. Es una ventaja muy importante a tener en cuenta ya que un mismo archivo puede ser ejecutado en diversos sistemas. Por otro lado, por lo general, suelen ser más lentos que los programas compilados ya que va traduciendo a medida que se va ejecutando, pero ofrece mayor flexibilidad en la programación y la depuración. Es muy importante que Python es un lenguaje que destaca por la legibilidad de su código, lo cual lo hace un gran candidato para trabajos como el nuestro, en el cual el futuro del programa es muy importante y que cualquiera que lo desee pueda entender el funcionamiento del código sin invertir mucho tiempo en ello.

Fue creado a finales de los años 80por Guido van Rossum. El punto de partida para su creación fue el lenguaje ABC (lenguaje de programación imperativo de propósito general y alto nivel). Python también destaca por ser un lenguaje multiparadigma, es decir, puede ser indistintamente orientado a objeto, imperativo y funcional. Con lo que permite que el usuario no tenga que ajustarse a uno solo y puedan trabajar sobre casi cualquier proyecto con un mismo lenguaje de programación. Otra característica muy importante es la resolución dinámica de nombres. A fecha de hoy, según varios portales online (kodigo.org, keepcoding.io, yeeply.com, etc), Python es el lenguaje de programación más usado y demandado en la actualidad. Por tanto, la comunidad de usuarios online es enorme y la resolución de casi cualquier duda, problema o error se puede encontrar en portales como stackoverflow.

En resumen, las principales ventajas de python son:

- Facilidad de uso: sintaxis clara y legible, enfoque en la legibilidad y simplicidad.
- Gran cantidad de bibliotecas y frameworks: numpy, scikit-learn, tensorflow, pytorch, pandas… Son algunos ejemplos de librerías muy usadas en python y muy importantes para el desarrollo de inteligencias artificiales y tratamiento de datos.
- Comunidad y soporte muy amplios: al ser el lenguaje más utilizado en la actualidad, hay una gran cantidad de usuarios que contribuyen a la comunidad publicando bibliotecas, tutoriales y recursos.
- Versatilidad: al ser un lenguaje de propósito general y ser multiparadigma, puede ser usado en multiples proyectos como aplicaciones web, tratamiento de datos, automatización de tareas…
- Escalabilidad: es un lenguaje que se usa desde sencillos scripts hasta en grandes servidores que están constantemente trabajando y necesitan una gran robustez. Esto nos demuestra que podemos partir de pequeños conjuntos de datos a conjuntos muy grandes y con una gran complejidad.
- Integración con otras tecnologías

SQL

Figura 4: logo SQL

SQL son las siglas en inglés de Structured Query Language diseñado para administrar y recuperar información de bases de datos relacionales. Se basa en los principios operacionales de creación, modificación y consulta. Es un estándar reconocido internacionalmente para interactuar con sistemas de gestión. Está basado en el álgebra y el cálculo relacional para definir, manipular y controlar los datos. Las bases de datos están configuradas con formas de tablas en las que las unidades de información se guardan en cada una de las celdas y se localizan gracias a la fila y la columna en la que se encuentran. El concepto de relación también es importante, ya que para dar forma a la información y asemejarse al mundo real, es necesario establecer relaciones entre tablas. Estas relaciones pueden ser de cuatro maneras: de uno a uno, de uno a muchos, de muchos a uno y de muchos a muchos. Gracias a esta relación también se establecen los términos de claves primarias y claves foráneas. Esto se utiliza para saber qué elemento de la tabla (en este caso las columnas que son las que guardan la información característica) hacemos referencia cuando hacemos una consulta. La clave primaria es una característica que nos permite diferenciar ese elemento del resto. Un ejemplo serían un número de id, un DNI o en caso de ser una clave primaria compuesta, la unión entre un nombre completo, un correo electrónico y un nombre de usuario.

SQL es el lenguaje más extendido para el manejo de bases de datos. Llega a tener tanta importancia que algunos lenguajes de programación lo integran dentro de su código e incluso se pueden hacer consultas directamente con código de SQL. Tiene una amplísima comunidad de personas en activo que lo usan a diario. Casi cualquier duda que pueda aparecer la puedes buscar en portales y estará resuelta por algún usuario. Casi cualquier programador debería conocer los básicos de SQL para conocer cómo suelen ser las

consultas a base de datos, ya que sienta un precedente para cualquier otro sistema de manejo o lenguaje de programación que se dedique a las bases de datos.

R

Figura 5: logo R

Es un entorno de programación y a su vez un lenguaje de programación. Tiene un enfoque hacia el análisis estadístico o la estadística en general (entendiendo estadística cómo una disciplina de las matemáticas encargada de el estudio de la variabilidad, organización y análisis de los datos). También es usado para la visualización de los datos y el desarrollo de aplicaciones dedicadas a la ciencia de datos. Es una reinterpretación del lenguaje S. Destaca su uso en *machine learning*, bioinformática, data mining, marketing o en la economía y las finanzas. Su colección de paquetes es muy extensa y permiten al usuario automatizar tareas como limpieza de datos, análisis, estudio estadístico, aprendizaje automático, etc. Hereda de S el hecho de ser un lenguaje orientado a objetos. Es ampliamente utilizado para el cálculo, llegando a características similares como las de sus competidores en este campo: Matlab y Octave.

En cuanto al lenguaje, es interpretado y de alto nivel, permitiendo hacer operaciones vectoriales y matriciales de manera rápida. Tiene una interfaz gráfica llamada RStudio (es su Integrated Development Environment o IDE), que permite escribir código a sus usuarios. Al ser de código abierto, se fomenta la colaboración y el intercambio de información entre usuarios. Como última mención, tiene una sintaxis flexible y expresiva que hace sencillo su uso y lo convierte en una herramienta muy completa, versátil y popular en el campo de la ciencia de datos.

Scala

Figura 6: logo Scala

Scala es un lenguaje de programación multiparadigma, pero se destaca por ser orientado a objetos, aunque también tiene características de lenguajes funcionales. Usa la "Java Virtual Machine" (JVM) para ejecutarse, por tanto puede usar casi todas las funciones de Java, incluso parte de su código. Su finalidad principal a la hora de su creación fue hacer más eficiente a los desarrolladores la escritura del código y su mantenimiento. Tiene cuatro pilares fundamentales: orientación a objetos, lenguaje funcional, tipificado estático y extensibilidad. Dentro de OO destaca el uso de clases y traits y que cada valor es un objeto. En el lenguaje funcional destaca la técnica del pattern matching, funciones de orden superior, anónimas, anidadas y la currificación. En el tipificado estático destaca el uso de tipos expresivos para que tenga sentido el uso de los tipos y evitemos los errores propios de la no-tipificación. En cuanto a la extensibilidad, todo bebe de Java y de todas sus bibliotecas, además de que gracias a ejecutarse en la

JVM puede acoplarse a aplicaciones existentes en Java sin alterar su funcionamiento original y sin suponer quebraderos de cabeza para los desarrolladores. Ofrece concurrencia y programación reactiva con librerías que permiten la ejecución paralela. Ha ganado en los últimos años una gran popularidad y es usada por empresas punteras en la industria, como Twitter o Airbnb para el tratamiento de datos masivos (”Big Data”). Es muy popular por su uso conjunto con Spark. Spark es un framework de procesamiento distribuido para Big Data que permite la concurrencia, procesamiento paralelo y operaciones en memoria en clústers de computadores.

A pesar de su popularidad, es un lenguaje bastante complicado de aprender, con una curva de aprendizaje muy amplia y que requiere de mucho tiempo para asentar los conocimientos.

Todos estos lenguajes de programación son dignos candidatos a ser el lenguaje que usaremos en nuestro proyecto. Por sencillez, comodidad y su amplio sistema y comunidad, elegimos python para realizar el desarrollo de la aplicación. Como más adelante veremos, usaremos Python como el hilo conductor de todo el proyecto Scala se ha descartado por su gran complejidad, SQL no cumple con todas las funciones que son necesarias para el adecuado desarrollo de la aplicación y R es un lenguaje menos conocido y con una comunidad menor, lo cual puede suponer un problema de cara a la resolución de algunos problemas durante el proceso de creación del trabajo.

Python es el perfecto candidato para nuestra aplicación porque lo podremos usar en las tres fases de desarrollo que hemos marcado anteriormente. También que puede beber de algunas características esenciales de otros lenguajes, como puede ser Spark con el uso de la librería pyspark para en un futuro adaptar la fase de extracción, transformación y carga de datos con Big Data para crear una base de datos más robusta. También permite iteración con otros sistemas de gestión de bases de datos, como SQLAlchemy. Además, la mayoría del desarrollo de inteligencia artificial se hace en Python. Por tanto, nuestro perfecto candidato es Python, para este y el resto de apartados. Ahora debemos cerciorarnos de cuáles son las mejores tecnologías de python para nuestro caso.

3.2.- Librerías para el tratamiento de datos de Python

NumPy

Figura 7: logo NumPy

Su principal utilidad es la creación de grandes vectores y matrices multidimensionales con los cuales podremos hacer cálculos mucho más rápidos. Su funcionalidad principal es el cálculo numérico, proporcionando estructuras de datos mucho más eficientes que las normales, además de funciones matemáticas también optimizadas. Su precursor fue Numeric, creado por Jim Hugunin. Numpy fue creado en el año 2005 por Travis Oliphant. Algunas otras opciones que implementa Numpy son: indexación y selección de datos (operaciones vectorizadas, máscaras booleanas, slicing…), broadcasting (operaciones entre arrays de formas y tamaños diferentes entre ellos de manera automática y eficiente)

e integración con otras librerías (como son las tres siguientes librerías, permitiéndonos análisis de datos más avanzados aprovechando lo mejor de cada una de ellas).

Pandas

Figura 8: logo pandas

Librería especializada en el tratamiento, análisis y manipulación de datos. Su nombre viene, de manera combinada, de "Panel Data" y "Python Data Analysis". Su estructura principal es el DataFrame, que son flexibles y eficientes. Ofrece la posibilidad de manejar tablas y series temporales. Tiene una funcionalidad similar a Excel pero para Python. Algunas de sus características principales, además del tratamiento y manipulación de datos, son: manejo de datos restantes (en python NaN, lo que es crucial para el análisis de datos y más adelante nosotros haremos uso de esta funcionalidad), integración de datos externos (gracias a la posibilidad de leer una amplia variedad de fuentes de datos, como pueden ser archivos CSV, Excel, bases de datos en SQL, archivos JSON…), análisis estadístico (promedios, correlaciones, desviaciones…) o visualización de datos (implementando Matplotlib, por ejemplo, para hacer gráficas atractivas sobre los datos que hemos usado).

Matplotlib

Figura 9: logo matplotlib

Esta librería tiene como función principal la visualización de datos en Python. Permite crear gráficos de una altísima calidad de manera mucho más sencilla. Tiene una gran personalización y es muy usado como complemento de otras librerías para mostrar los resultados. Sabemos que una buena representación de los datos es muy importante a la hora de realizar cualquier estudio sobre datos. Algunas de sus características principales son: variedad de gráficos (de línea, dispersión, barras, histogramas, caja y bigotes…), personalización y control (podemos decidir los estilos de línea, el color, etiquetas de los ejes, leyendas, títulos…), compatibilidad (sobre todo con numpy y pandas), soporte en múltiples interfaces (independientemente de la interfaz, podemos usarlo en distintos entornos) y capacidad de exportación (una vez tenemos nuestros gráficos, los podemos exportar en formatos como PNG, JPEG, PDF…). Casi que toda la parte de visualización de datos de nuestro proyecto la realizaremos con matplotlib.

SciPy

Figura 10: logo sciPy

Su funcionalidad principal son los cálculos matemáticos y científicos, usando sus funciones para analizar datos. Sus características principales son las siguientes: módulos

específicos (como scipy.stats para estadística), funciones numéricas y matemáticas (cálculo de funciones, álgebra lineal, números aleatorios, transformadas de Fourier…), integración (por ejemplo con numpy para aprovechar sus estructuras de datos), algoritmos de optimización (con la finalidad de ajustar modelos, encontrar soluciones óptimas, optimizar funciones…) y herramientas del procesamiento de señales (para el procesamiento de imágenes, audio, señales biomédicas…).

En definitiva, todas estas librerías nos serán muy útiles en nuestra tarea. Tanto en la parte de tratamiento de datos como en la parte de entrenamiento las usaremos todas ellas. Como bien hemos visto por separado son herramientas muy buenas, pero el uso conjunto de las mismas exprime al máximo su potencial y es lo que nosotros usaremos en el proyecto.

3.3.- Lenguajes de programación para el desarrollo de inteligencia artificial

Dentro de los lenguajes de programación, los que más son usados para el procesamiento de inteligencias artificiales son: Python, R, Scala, Java, C++ y Julia. Tanto Ptyhon como R y Scala ya han sido desarrollados anteriormente. Procederemos a mencionar las principales características de Java, C++ y Julia.

Java

Figura 11: logo Java

Java puede ser sin duda el lenguaje de programación por excelencia a lo largo de la historia de la programación (aunque a día de hoy Python le ha comido parte de su terreno). Fue comercializado por primera vez en 1995 por Sun Mycrosystems y desarrollado por James Gosling. Es un lenguaje orientado a objetos muy usado para el desarrollo de software empresarial, aplicaciones móviles, sistemas embebidos y, en menor medida, inteligencia artificial. Destacan sus librerías Deeplearning4j y Weka que permiten el desarrollo de aprendizaje automático en Java. Aunque la comunidad de Java es inmensa, estas dos librerías son menos utilizadas y minoritarias entre la amplia gama de librerías que Java ofrece. Java destaca por tener una gran portabilidad, una amplia biblioteca estándar (Java Development Kit o JDK) y una gran escalabilidad y rendimiento gracias a la capacidad de manejar grandes volúmenes de datos de manera eficiente. Más que un lenguaje de facto para el desarrollo de inteligencia artificial es muy usado en todos los procesos que atañen a este campo, como el procesamiento de datos.

C++

Figura 12: logo C++

Fue creado en el año 1979 por Bjarne Stroustrup. Nació para ser una extensión del lenguaje C, añadiendo la orientación a objetos a este. Posteriormente, se añadieron utilidades de lenguaje de programación estructurada. Es un lenguaje de propósito general que tiene un altísimo rendimiento y gran variedad de aplicaciones. Una de sus características principales es su compatibilidad con otros lenguajes, como puede ser con C de manera implícita o con extensiones con lenguajes como Python. Su principal uso dentro de la inteligencia artificial es la creación de bibliotecas y frameworks para el aprendizaje automático o el procesamiento de imágenes. Algunas ejemplos de bibliotecas que fueron creadas con C++ son Caffe o Tensorflow.

Julia

Figura 13: logo julia

Es un lenguaje de programación nacido hace poco tiempo, en comparación con sus competidores, en 2012. Se destaca por ser un lenguaje homoicónico (según Wikipedia[8] un lenguaje homoicónico es aquel en el que la representación primaria del lenguaje es a su vez una estructura de datos de un tipo primitivo del lenguaje), multiplataforma y multiparadigma. Es de código abierto y de alto nivel, con el objetivo de usarse en aplicaciones científicas y técnicas. Está enfocado en el rendimiento, usando un compilador en tiempo de ejecución capaz de igualar las velocidades de lenguajes de bajo nivel como C o Fortran. Tiene un tipado dinámico y estático a la vez, dando flexibilidad a la hora de escribir su código. Es posible hacer ejecuciones paralelas y computación distribuida con Julia. Permite el uso de librerías de Python, C y R, adaptándose fácilmente a proyectos existentes. Sus librerías y paquetes abarcan el álgebra lineal, estadística, aprendizaje automático, procesamiento de datos, etc. Todas estas características lo hacen un gran candidato.

Como bien hemos dicho anteriormente, Python es el elegido para realizar el desarrollo de la inteligencia artificial. A pesar de ello, Julia podría haber sido la elección más adecuada, pero el desconocimiento por mi parte del lenguaje junto con ser un lenguaje relativamente nuevo y con un gran camino aún por delante hace que la capacidad de resolución de problemas sea menor ya que la cantidad de usuarios activos del mismo es menor que el de Python.

3.4.- Librerías y frameworks de desarrollo de inteligencia artificial

En esta sección veremos que librerías o frameworks son los que principalmente se usan en el desarrollo de la inteligencia artificial. Para ello indagaremos varias opciones y veremos cuáles han sido las que hemos utilizado. Entre las opciones veremos: Tensorflow, Keras, Pytorch, Scikit-Learn o sklearn, Theano y Caffe.

Mucha de la información que vamos a comparar se encuentra alojada en esta página web, la cual hace una comparación entre los diversos softwares para el desarrollo de deep learning.

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Tensorflow

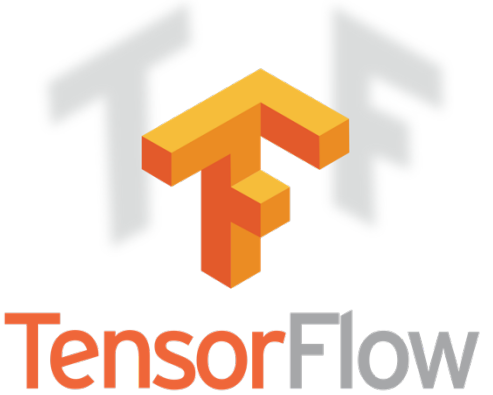

Figura 14: logo TensorFlow

Es una librería de código abierto cuyo objetivo principal es el aprendizaje automático. Fue desarrollado por Google a partir de 1 año 2015. Su predecesor fue una librería de código cerrado llamada DistBelief que a partir del año 2011 comenzó a desarrollarse y que tenía la misma funcionalidad que tiene la actual Tensorflow. Está basado en redes neuronales y permite la creación y entrenamiento de las mismas. Es uno de los frameworks más utilizados a día de hoy, lo que permite que haya una buena comunidad con mucha ayuda online por parte del gran número de usuarios que la usan. Esto permite que haya numerosos recursos disponibles y modelos pre-entrenados, tutoriales y documentación. Tiene un gran número de herramientas y recursos para hacer al usuario más sencillo su uso. Una de sus principales características es que permite una arquitectura flexible y escalable, para poder desarrollar y desplegar los modelos que se crean en diversas plataformas y dispositivos.

Una de sus principales características son las unidades de procesamiento del tensor o TPU. Fueron lanzadas en 2016 por Google y son unos aceleradores de IA diseñados para proporcionar un alto throughput (la tasa de transferencia efectiva o throughput es el volumen de información que fluye en un sistema) que permiten correr modelos de una manera mucho más efectiva. Con esto conseguimos un orden de magnitud mayor que los sistemas tradicionales. Esta característica, junto con el uso de Google colab acelera mucho el desarrollo y ejecución de las redes neuronales en nuestros modelos a crear.

Como Tensorflow tiene una alta compatibilidad y una integración muy sólida con otros frameworks y herramientas, es muy común que se use junto con otras herramientas de aprendizaje automático como puede ser Keras.

En resumen, es un framework muy poderoso y flexible que tiene varias características que la hacen una muy buena candidata para el desarrollo de redes neuronales, sobre todo con la implementación con Keras.

Keras

Figura 15: logo Keras

Es una librería de alto nivel que está escrita sobre Python y se ejecuta sobre Tensorflow, Microsoft Cognitive Toolkit o Theano. Es de código abierto y permite que se experimente con redes de aprendizaje profundo en un menor tiempo siendo amigable para el usuario. Su principal autor es François Chollet. Fue desarrollada por Facebook AI Research y se destaca por su enfoque flexible y dinámico. En el año 2017 recibió soporte por parte del equipo de Tensorflow de Google en su biblioteca de core. Según su autor se trata más de una interfaz que de un framework el cual es simple y con unas abstracciones más intuitivas y de alto nivel. También es ampliamente utilizado en el desarrollo de modelos de aprendizaje profundo y tiene una comunidad que lo enriquece. Tiene unos bloques de construcción predefinidos que están enfocados en la rápida construcción de modelos, como pueden ser capas, funciones de activación, algoritmos de optimización, métricas de evaluación, optimizadores matemáticos, etc. Esto hace que el desarrollo sea más rápido y sencillo. Tiene el código fuente alojado en Github e incluso incluye un canal de Slack para el soporte. Es importante que también ofrece soporte para redes neuronales convolucionales y recurrentes, lo cual es muy interesante de cara a futuro en el proyecto.

Pytorch

Figura 16: logo PyTorch

Es un framework desarrollado también por Facebook AI Research con un enfoque muy dinámico y flexible. Permite construir y modificar grafos computacionales de manera flexible durante el tiempo de ejecución, dando flexibilidad a los desarrolladores para experimentar sobre sus modelos. Tiene una sintaxis intuitiva y un enfoque Pythonic, facilitando la construcción y entrenamiento de los modelos. Tiene un módulo de alto nivel llamado "torch.nm" que facilita la construcción de redes neuronales usando capas y funciones de activación predefinidas. Una de sus fortalezas es la capacidad de aprovechar y exprimir al máximo las GPUs, mediante CUDA (computación paralela de Nvidia), permitiendo que los

cálculos en GPU sean mucho más eficientes. Tiene también una gran comunidad activa y muchos desarrolladores que crean sus propias bibliotecas, herramientas y modelos pre-entrenados que son muy útiles para los usuarios.

Scikit-learn o sklearn

Figura 17: log ScikitLearn

Se trata de una biblioteca en Python que ofrece una amplia gama de herramientas y algoritmos para aprendizaje automático. Es sin duda una de las que tiene un uso más sencillo y tiene un enfoque orientado a objetos, con una interfaz coherente y bien documentada. Se integra con otras bibliotecas de Python como son Pandas y Numpy, para ser mucho más rápido en la manipulación y preprocesamiento de datos. Es muy usado para algoritmos de aprendizaje automático supervisado, como pueden ser Máquinas de Vectores Soporte (SVM), regresión o árboles de decisión. También se suele usar con algoritmos no supervisados como análisis de componentes principales (PCA) o K-vecinos. Son también muy importantes sus herramientas de selección de características, evaluación de rendimiento del modelo, validación cruzada, etc. Tiene una gran cantidad de tutoriales y ejercicios y muchos usuarios activos para dar soporte e incluso contribuir con su propio código.

Theano

Figura 18: logo theano

Es una librería de python principalmente especializada en los cálculos de matrices. Permite definir, optimizar y evaluar matrices gracias a estar escrito sobre NumPy. Por tanto, tiene una interfaz similar a este. Permite el uso de aceleradores GPU dando un rendimiento mucho más alto que con las CPU. También es segura cuando hacemos cálculos con números demasiado grandes sobre operaciones que podrían causar problemas de ejecución o bugs cuando ejecutamos nuestro código.

Es un proyecto de código abierto propiedad de Montreal Institute for Learning Algorithms de la Universidad de Montreal. Su primer lanzamiento fue hace 16 años, en 2007. Su nombre proviene de una filósofa pitagórica del siglo sexto antes de cristo, llamada Theano de Crotone.

Caffe

Figura 19: logo Caffe

Es un framework diseñado con la velocidad y la modularidad como sus principales bazas. Es código abierto y escrito sobre C++ con una interfaz para Python. Entre sus características cabe destacar el soporte a varios modelos de clasificación de imágenes y segmentación de las mismas. Tiene una arquitectura sencilla y expresiva que permite cambiar entre ejecución en GPU y CPU de manera sencilla. Puede procesar grandes números de imágenes con una capacidad más reducida que sus competidores. Hay una gran comunidad detrás de su soporte y mantenimiento, además de numerosos desarrolladores que crean nuevos proyectos y los dan a la comunidad para que los puedan usar. Sobre todo es usado para aplicaciones de visión artificial, procesamiento del habla y multimedia. Nació en el año 2017 por lo que es una tecnología moderna.

Después de tener una visión del panorama actual de librerías y frameworks para el desarrollo de inteligencia artificial, sabemos que para nuestro caso debemos usar más de una tecnología. En nuestro proyecto hemos decidido hacer dos inteligencias artificiales distintas, una basada en redes neuronales y otra en máquinas de vectores soporte. Por tanto la mejor opción para la parte de redes neuronales estaría entre Tensorflow y Pytorch. Por tener una complejidad algo menor, una comunidad mayor y la capacidad de desplegarse en varias plataformas distintas y usar unidades de procesamiento de tensor (TPU) se ha optado por la opción de TensorFlow en lugar de Pytorch. En el caso del desarrollo de SVM la mejor opción es sklearn por su sencillez, herramientas específicas, optimización, etc. Las opciones de Theano y Caffe han sido descartadas desde un primer momento por ser herramientas menos usadas y con mucha menos comunidad e historia, pero con ambas se podría llevar a cabo el desarrollo del trabajo sin ningún problema.

3.5.- Frameworks de desarrollo web

Los principales frameworks de desarrollo web son: Django, Laravel, Angular, React, Vue.js y Flask. Veremos cuál de ellos es el que mejor se adapta a nuestro proyecto y nos ofrece una solución sencilla pero eficiente.

Django

Figura 20: logo Django

Framework de alto nivel y código abierto que está escrito sobre Python. Es muy importante que utiliza el patrón modelo-vista-controlador. Su meta es dar solución a la creación de aplicaciones y páginas web con un nivel de complejidad alto. Sus componentes destacan por el re-uso, conectividad y extensibilidad. Su sistema es muy completo, siendo muy horizontal en todas las tecnologías necesarias para una aplicación web: administración de bases de datos, enrutamiento, autenticación, formularios, etc. Se enfoca principalmente en la eficiencia y la reutilización del código.

Laravel (Symfony)

Figura 21: logo Laravel

Es un framework de código abierto centrado en una sintaxis elegante y expresiva basado en PHP. Su finalidad principal es la creación de servicios y aplicaciones web con un código sencillo pero sin dejar de lado una gran cantidad de utilidades. El desarrollo de Laravel está basado en el desarrollo de sus dependencias, la principal de ellas el Symfony. También sigue el patrón modelo-vista-controlador y promueve las buenas prácticas de desarrollo.

Symfony por su parte es una dependencia de laravel y a su vez un framework. Es muy importante la separación que hace entre la lógica de negocio, la lógica de servidor y la presentación de la web. Destaca su automatización de tareas básicas.

Angular

En este caso es un framework basado en JavaScript o TypeScript. Creado y mantenido por Google y que se crea con la principal característica de que se utiliza para crear aplicaciones web de una sola página (SPA). Usa la capacidad de modelo-vista-controlador. Utiliza un Document Object Model (mas conocido por sus siglas DOM) real, es decir, el estado de la aplicación es dependiente del navegador en el que se visualiza la aplicación. DOM es la estructura de un documento HTML, cuando anidamos las etiquetas de un archivo HTML se forma una estructura interna implícita de árbol llamado árbol de DOM. Se basa en la programación declarativa para generar interfaces de usuario y enlazar los componentes de software.

Figura 22: logo Angular

React

También es un framework basado en JavaScript o TypeScript. Nace en el año 2010 de la mano de Facebook. También está basado en SPA, que son webs que integran dentro de una misma página todo su contenido para mejorar la velocidad y que el usuario tenga una experiencia más fluida. De manera dinámica los objetos y componentes de la aplicación web van apareciendo y desapareciendo en función de la necesidad de los mismos y la interacción con el usuario. Es muy importante la aparición de los elementos JSX, con los que se combina la lógica de programación de JavaScript con los elementos del diseño web. En este caso, se usa un DOM virtual, es decir, eliminamos la necesidad del navegador de reprocesar el contenido para que se apliquen cambios, permitiendo actualizaciones más eficientes y evitando el re-renderizado innecesario. Cuenta con una gran comunidad de usuarios activos y tiene a sus espaldas al gigante tecnológico Facebook.

Figura 23: logo React

Vue.js

Otro framework de JavaScript de código abierto que se basa en la creación de páginas web de una sola página e interfaces de usuario. También trabaja sobre un DOM virtual, al igual que React. Se diferencia de este último en cuanto al soporte que tiene. React es desarrollado y

Figura 24: logo Vue.JS

mantenido por Facebook mientras que Vue.js se mantiene por sus usuarios, es decir, no tiene una empresa que lo respalde. Es un framework del tipo "open-source". Tiene una característica distintiva y es que tiene un enfoque gradual lo que significa que se puede adoptar de forma incremental en proyectos existentes. Su sintaxis es declarativa y tiene unas directivas muy claras para manipular el DOM y enlazar los datos entre las diferentes capas.

Flask

Figura 25: logo Flask

Es un framework basado en Python cuyas características principales son su enfoque en el desarrollo de webs ligeras, sencillas de programar (con el menor número de líneas de código posibles) y flexibles. Por su sencillez, podría ser considerado como un "microframework" de Python. Otro punto muy importante de su uso es la legibilidad del código. Ofrece características como el enrutamiento de URL, soporte para plantillas Jinja2, conexión con bases de datos, formularios, etc. La aplicación web se ejecuta en el servidor, de manera que desde el browser se hace una petición al servidor y este es capaz de proporcionar la información necesaria gracias a Flask.

Nuestra función es hacer una herramienta que sea sencilla pero a la vez altamente escalable, ya que estamos haciendo una versión muy prematura de cara al usuario. También, hemos decidido buscar de alguna manera la uniformidad a lo largo de todo el trabajo, por lo que hemos decidido que Python sea el motor que mueva todo el desarrollo. Es por ello que se han descartado desde un primer momento todas las opciones basadas en JavaScript o TypeScript, no solo por su lenguaje de programación, sino por su curva de aprendizaje tan compleja. Por tanto, hemos barajado la posibilidad de trabajar con Flask o Django, pero nuestra opción ha sido la primera. Esta decisión ha sido principalmente por la sencillez que flask ofrece y que hace que nos centremos más en el grueso del proyecto: el entrenamiento de la IA.

También cabe destacar que junto con Flask hemos usado Bootstrap para mejorar la interfaz con el usuario. Es un framework para el frontend que tiene un conjunto de herramientas que permiten estilizar la vista de las páginas web. Esto nos permite tener una app más responsive y visualmente atractiva. También tiene predefinidos numerosos componentes como botones, formularios, cuadrículas, carruseles, etc.

4.- Análisis del problema

En esta sección analizaremos el problema desde una descripción del mismo hasta los requisitos del sistema. Nuestra finalidad con este análisis y uno de los objetivos más importantes de un trabajo final de grado y durante toda la carrera consiste en saber desgranar un problema para identificar sus puntos clave y poder abordarlo de la manera más óptima posible, pudiendo alcanzar la mejor solución en el menor tiempo y ocupando la menor cantidad de recursos posible, siempre sin dejar ningún aspecto clave del problema al azar u olvidado. Este análisis es muy importante a nivel organizativo, porque de él dependerán las fases del desarrollo del problema.

4.1.- Descripción del problema a resolver

Nuestro principal objetivo es crear una herramienta para acercar la inteligencia artificial a profesionales del mundo de la salud mental para ayudarles en su tarea. La finalidad es detectar y prevenir el suicidio basándonos en la inteligencia artificial en personas que usan plataformas como Twitter o Reddit. Este problema proviene del gran aumento en el número de suicidios cometidos en los últimos años y de la problemática que esto provoca en las etapas tempranas, como pueden ser la depresión, dando lugar a uno de los mayores problemas de la sociedad moderna, que afecta a muchos de los ámbitos de la misma.

Nuestro desafío principal se centra en la detección temprana y en el análisis del proceso del usuario hasta llegar a ese punto. Para ello hacemos uso del procesamiento del lenguaje natural (PLN) para determinar el nivel de riesgo suicida. Con el PLN buscamos unos patrones en el lenguaje o palabras clave que están relacionadas con el suicidio o con la depresión. Por tanto, con esta evaluación del usuario, se puede intervenir de manera temprana y ofrecer apoyo a personas en situación de riesgo.

4.2.- Análisis de requisitos

Según el estándar IEEE 1990 [18], el análisis de requisitos es una parte de un documento en la que se especifican los requisitos inherentes al diseño o comportamiento de un componente de un sistema informático. Es muy común el análisis de requisitos en la ingeniería del software y por lo general se suele dividir en dos: requisitos funcionales y requisitos no funcionales. Estos requisitos se extraen, normalmente, de las necesidades del cliente. En nuestro caso no tenemos un cliente pero la descripción de nuestro problema que hemos ido desarrollando en pasos anteriores, nos puede servir para extraer estas características. Nuestra misión es, por tanto, entregar un producto o aplicación que cumpla con lo requerido por el cliente o, en nuestro caso, por el problema a resolver.

Requisitos funcionales de la aplicación

Número de requisito	RF-A-01
Nombre	Aplicación rápida y funcional
Descripción	La aplicación no debe colgarse o estar demasiado tiempo pensando. Debe cumplir con la muestra de datos precisos y correctos para ser funcional.
Requisito asociado	—

Tabla 1: Requisito funcional app 1

Número de requisito	RF-A-02
Nombre	Extracción de datos de las APIs
Descripción	Obtención de los mensajes y publicaciones de los usuarios en las distintas redes.
Requisito asociado	—

Tabla 2: Requisito funcional app 2

Número de requisito	RF-A-03
Nombre	Detección de riesgo
Descripción	Mediante los modelos, la aplicación debe de ser capaz de procesar los mensajes y publicaciones de los usuarios y obtener unos resultados fiables.
Requisito asociado	--

Tabla 3: Requisito funcional app 3

Número de requisito	RF-A-04
Nombre	Muestra de resultados / Resultado de la predicción
Descripción	Una vez tengamos los datos se deben mostrar al usuario de una manera entendible y sencilla.
Requisito asociado	RF-A-04 // RF-A-02

Tabla 4: Requisito funcional app 4

Requisitos no funcionales de la aplicación

Número de requisito	RNF-A-01
Nombre	Seguridad
Descripción	Los datos de las personas deben ser privados y confidenciales. Los datos deben ser usados de manera ética y legal.
Requisito asociado	—

Tabla 5: Requisito no funcional app 1

Número de requisito	RNF-A-02
Nombre	Escalabilidad
Descripción	La app debería ser capaz de manejar tanto grandes volúmenes de datos como grandes volúmenes de usuarios, sin poner en compromiso el rendimiento de la misma.
Requisito asociado	RF-A-01
Nota	No se ha probado aún este requisito

Tabla 6: Requisito no funcional app 2

Número de requisito	RNF-A-03
Nombre	Usabilidad
Descripción	La app debe ser fácil de usar y accesible para cualquier profesional que la utilice.
Requisito asociado	RF-A-02

Tabla 7: Requisito no funcional app 3

Número de requisito	RNF-A-04
Nombre	Interfaz intuitiva
Descripción	Diseño de una interfaz amigable para el usuario debido a que será un usuario que no tiene porqué tener amplios conocimientos de informática.
Requisito asociado	—

Tabla 8: Requisito no funcional app 4

4.3.- Análisis del marco ético-legal

Hay varios aspectos a tener en cuenta que deben ser importantes para nosotros desde el punto de vista ético y legal, sobretodo los relacionados con los datos de usuarios y la privacidad de los mismos. Analizaremos este marco desde varios puntos de vista:

- La aplicación debe de cumplir con las regulaciones del Reglamento General de Protección de Datos [19] (RGPD) de la unión europea. Esta entidad nos dicta que los datos deben ser privados, confidenciales y seguros. Debemos proteger la información de los usuarios a toda costa. También es muy importante el derecho al olvido y el usuario puede requerirlo en cualquier momento.
- La recolección y uso de los datos debe estar dentro del marco ético, respetando los derechos del usuario. Debemos poder garantizar que el origen de nuestros datos cumpla con estos derechos o que la recopilación de los mismos se haga de manera anónima y consentida. También el propósito es muy importante, ya que los datos deben ser usados con la única intención de la prevención del suicidio. Por último debemos evitar discriminar o sesgar por sexo, edad, raza, orientación sexual, etc. También llegará un momento en el que los usuarios deberán dar su consentimiento explícito a la monitorización (en versiones futuras).
- Debemos tener siempre los datos mínimos y necesarios de cada uno de los usuarios de donde provenga nuestra información, así como de aquellos que vayan a usar la aplicación (en el caso de que, por ejemplo, en versiones futuras se requiera de autenticación de los usuarios).

También son muy importantes dos herramientas a nivel legal español que son importantes a la hora de la creación de páginas web, como son la Ley Orgánica de Protección de Datos de Carácter Personal 15/1999 [31] (LOPD) y la Ley de Servicios de la Sociedad de la Informática y del Comercio Electrónico 34/2002 [32] (LSSI), esta última en caso de que se lance la aplicación al internet.

4.4.- Plan de trabajo

En esta sección abordaremos la división por partes que hemos hecho del trabajo. Para ello, hemos obtenido las siguientes partes (las cuales abordaremos en el siguiente punto para darles solución a las mismas):

- Estudio del problema: como punto de partida hemos estudiado cómo podemos abordar nuestro problema, para dar con un plan de actuación.
- Búsqueda de datasets: debemos encontrar datasets que ya estén etiquetadas por profesionales o que hayan sido ampliamente usadas y verificadas por entidades del mundo de la salud, con la finalidad de cerciorarnos que los datos que obtenemos sean correctos, para que los resultados que obtengamos puedan ser comparados con el estado del arte.
- Traducción de los datos: en esta sección haremos que todos los datos que obtenemos estén en español.
- Creación de un único dataset: aunaremos todos los distintos datasets en una misma con la finalidad de tener unos datos consistentes.
- Entrenamiento de SVM.
- Entrenamiento de redes neuronales.

- Entrenamiento de redes neuronales + BERT.
- Posibilidad de entrenamiento de una red neuronal con BERTO (BERT optimizado para el español).
- Creación de la app: junto con la conexión con twitter o reddit, la obtención de datos y el muestreo de nuestros resultados con respecto a esos datos.
- Documentación del proceso y memoria: en esta sección realizaremos esta memoria y procederemos a documentar, comentar y limpiar todo el código, empaquetándolo para futuras revisiones y posibles futuros trabajos.

A continuación, realizaremos un diagrama de Gantt para mostrar de manera más grafica la distribución en el tiempo del proyecto:

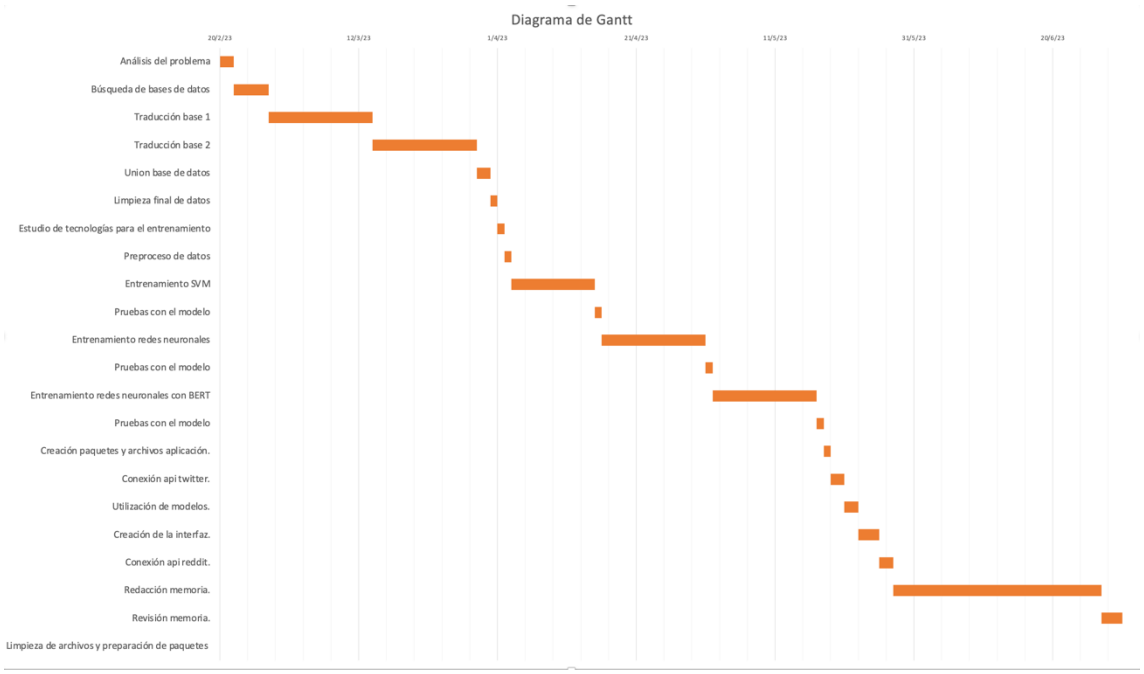

Figura 26: diagrama de Gantt

Este diagrama divide el proceso por etapas básicas, las cuales son las siguientes (corresponden al índice izquierdo del diagrama, por orden): análisis del problema, búsqueda de datasets, traducción dataset 1, traducción dataset 2, unión datasets, limpieza datasets, estudio tecnologías para el entrenamiento, preprocesado de datos, entrenamiento SVM, pruebas SVM, entrenamiento redes neuronales, pruebas redes neuronales, entrenamiento redes neuronales con BERT, pruebas redes con BERT, creación archivos y paquetes aplicación, conexiones api Twitter, utilización de modelos, creación de la interfaz, conexión api reddit, redacción memoria, revisión memoria y limpieza de archivos y preparación de paquetes.

5.- Solución propuesta

En este apartado veremos cómo hemos diseñado la solución del proyecto final de grado desde cero. Veremos, por tanto, de manera íntegra cómo ha sido el proceso de creación de nuestra app y el entrenamiento de los distintos sistemas de inteligencia artificial, así como el proceso de traducción del dataset. Durante el capítulo 3 se ha discutido cuáles son las tecnologías que vamos a usar y porqué se ha decidido optar por ellas, ahora veremos cómo las vamos a utilizar.

5.1.- Arquitectura de la solución (herramientas utilizadas)

En este apartado veremos, de manera general, cómo hemos estructurado nuestra solución y cómo han interactuado las partes entre sí.

La aplicación funciona de manera que vamos a observar en la siguiente figura:

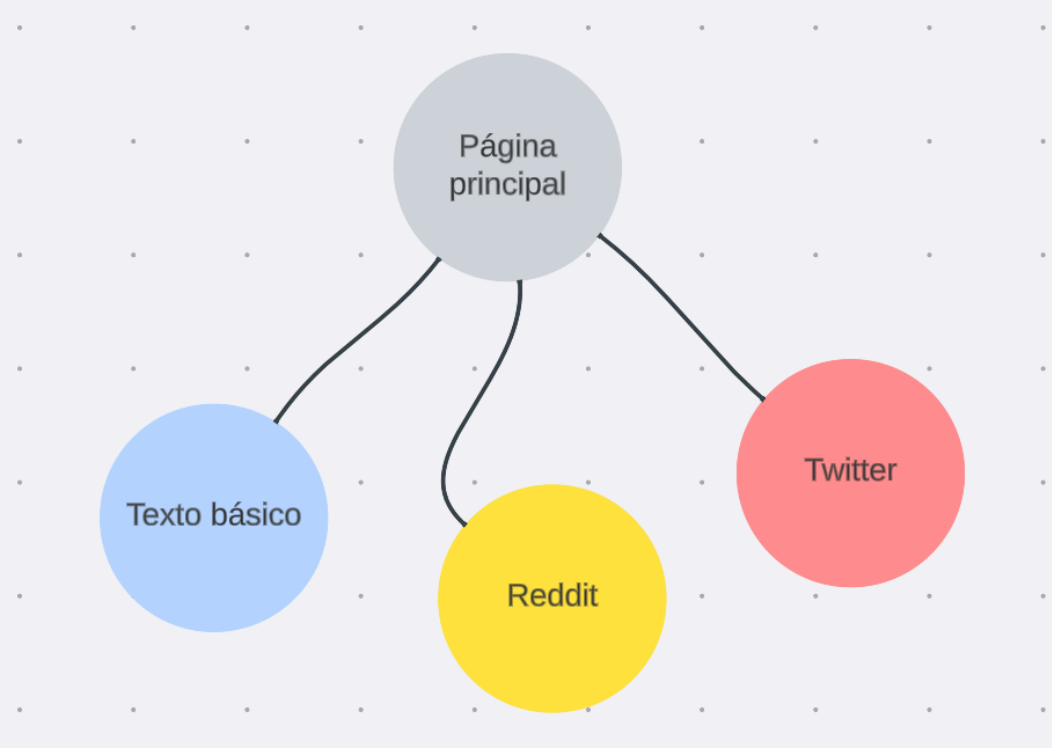

Figura 27: gráfico estructural endpoints aplicación

Como podemos observar en él, la aplicación reside en una app de flask, que se pone en funcionamiento a través del terminal. Posteriormente, una vez está funcionando, el usuario (profesional de la salud) parte de nuestra página principal, la cual tiene tres botones, sobre los cuales elige si usará el texto plano, la conexión con Reddit o la conexión con Twitter (aún sin funcionar).

En cuanto a las tecnologías utilizadas, resumiremos cuáles hemos usado para cada una de las partes (sabiendo que para todas ellas hemos usado Python como lenguaje principal):

- Procesamiento y traducción de datos, diseño del dataset: hemos usado un traductor llamado EasyNMT [25] que está basado en Opus-MT, Helsinki-NLP, mBART50_m2m y M2M_100 (estos dos últimos de Facebook Research). Con el modelo hemos obtenido unos tiempos de traducción similares que con la API de Google Translate, pero siendo EasyNMT gratuito e ilimitado. Para leer nuestro dataset de origen hemos usado pandas y también la hemos usado para hacer partes más pequeñas del dataset para ir traduciendo poco a poco (hemos hecho "chunks" y hemos ido traduciendo cada uno de ellos, lo veremos más adelante). También hemos usado la librería csv de python para leer archivos en formato csv.
- Entrenamiento: usamos pandas, numpy, nltk y sklearn para la parte de las máquinas de vectores soporte. Para la parte de redes neuronales, a parte de las que hemos dicho anteriormente, también usamos la librería de tensorflow y de la librería de transformers BertTokenizer y TFBertForSequenceClassification.
- Creación de la app: hemos usado flask, HTML, CSS y bootstrap para la parte del frontend. También hemos usado tensorflow para leer los modelos, tweepy para conectar con la API de Twitter y pickle también para leer modelos.

5.2.- Procesamiento de datos

Nuestra primera tarea fue la búsqueda de datasets que pudieran sernos útiles para el proyecto. El problema es que los datos que queremos tratar son datos personales que no son tan fácilmente accesibles debido a las restricciones de las redes sociales para compartir datos, información o publicaciones de usuarios a terceros. Por tanto, nos hemos basado en algunas tecnologías similares que usan también datasets que nos pueden resultar útiles, como las que hemos mostrado en el punto 2.1. También, partimos del artículo "**Suicidal Ideation Detection: A Review of *Machine Learning* Methods and Applications" (21).** En este, se muestra una tabla con varios datasets, sus autores, de donde provienen y, en caso de ser de acceso público, dónde se pueden obtener.

Type	Publication	Source	Instances	Public Access
Text	Shing et al. [13]	Reddit	866/11,129	https://tinyurl.com/umd-suicidality
Text	Aladağ et al. [100]	Reddit	508,398	Request to the authors
Text	Coppersmith et al. [33]	Twitter	> 3,200	N.A
Text	Coppersmith et al. [104]	Twitter	1,711	https://tinyurl.com/clpsych-2015
Text	Vioulès et al. [3]	Twitter	5,446	N.A
Text	Milne et al. [101]	ReachOut	65,756	N.A
EHR	Bhat and Goldman-Mellor [79]	Hospital	522,056	N.A
EHR	Tran et al. [69]	Barwon Health	7,746	N.A
EHR	Haerian et al. [102]	CDW & WebCIS	280	N.A
Text	Pestian et al. [80]	Notes	1,319	2011 i2b2 NLP challenge
Text	Gaur et al. [58]	Reddit	500/2,181	Request to the authors
Text	eRisk 2018 [103]	Social networks	892	https://tec.citius.usc.es/ir/code/eRisk.html
Text	RSDD [95]	Reddit	9,000	http://ir.cs.georgetown.edu/resources/rsdd.html

Figura 28: listado datasets (21)

A pesar de que aquí podemos observar muchos datasets, el acceso a ellos no ha sido nada sencillo. Hay algunas de ellas, como "eRisk 2018" [11] que hemos tenido que realizar un gran proceso para poder usarlas, e incluso algunas están aún en trámites para que en el futuro se pueda mejorar el entrenamiento gracias al uso de estas para el entrenamiento. De todas ellas, sólo tenemos acceso a una de ellas "Aladag et al. Source: Reddit. Instances: 508.298" [23], la cual tiene pocos datos etiquetados, pero se ha

procedido a hacer la traducción completa al español de la misma para que se pueda etiquetar también en un futuro y poder usarla.

También obtuvimos un dataset muy bien estructurado y sencillo que obtenía su información de Reddit, de hilo "Suicide Watch" [24], proveniente de Kaggle [8]. En ella, simplemente tenemos dos columnas: el texto y la etiqueta. Es un dataset muy sencillo pero que tiene un gran número de entradas.

Por tanto, una vez que tenemos acceso al menos a estos dos datasets y viendo que nos está costando mucho esfuerzo acceder a otras ya mostradas anteriormente, procedemos a comenzar con la etapa de traducción.

5.3.- Traducción de datos

En la fase de traducción, como tenemos varias dataset que usaremos como fuentes, hemos decidido estructurar el código de manera que la traducción se pueda hacer de la manera más general posible, reutilizando el código al máximo independientemente de la estructura que siga el origen en cuanto a columnas. Hemos hecho el código para tomar como referencia archivos en formato csv, pero se podría utilizar cualquier otro.

Por tanto, partimos de datasets con muchas filas y mucha cantidad de información. El proceso de traducción va a ser costoso, sobre todo en tiempo, por tanto hemos realizado un proceso para asegurar el trabajo echo. Los datasets hemos ido dividiendo en archivos más pequeños, normalmente entre unos 1000 y 2000, los cuales los hemos llamado chunks.

```
df = pd.read_csv('Suicide_Detection.csv')
n = 1000 # number of smaller dataframes
size = len(df) // n  # size of each smaller dataframe
dfs = [df.iloc[i:i+size] for i in range(0, len(df), size)]
for i, small_df in enumerate(dfs):
    small_df.to_csv(f'/content/chunked/smaller_csv_{i}.csv',
index=False)
```

Hemos realizado la traducción de cada uno de ellos de manera individual, para ello hemos primero eliminado los saltos de línea posibles que hubiera en el texto, así como los enlaces que pudiera haber como comentarios, ya que hemos visto que pueden afectar al correcto funcionamiento del traductor.

```
for i, row in enumerate(rows):
      if i != 0:
        for j, cell in enumerate(row):
          if j == 1:
            if cell.strip() and cell!="/n" and "http://" not in cell and
"https://" not in cell:
              translation=model.translate(str(cell),  source_lang="en",
target_lang='es')
              rows[i][j] = translation
```

Guardamos en memoria el archivo una vez se había traducido como "translated_{number}.csv" (por ejemplo el archivo número 1 se llamaba "smaller_csv_1.csv" y el traducido de este archivo "translated_1.csv"). Esto nos permitía asegurarnos del trabajo que íbamos haciendo, ya que la traducción de uno solo de estos datasets podía llegar a tardar unos dos o tres días completos de cómputo sin parar, y podría haber fallos de conexión que hicieran que perdiésemos el avance realizado. Cabe destacar que el proceso de traducción lo hemos realizado en Google colab, ya que es mucho más rápida la traducción de los servidores de google que en local.

```
with open(f"/content/translated/translated_{a}.csv", "w", newline="")
as csvfile:
      csvwriter = csv.writer(csvfile)
      csvwriter.writerows(rows)
      print(f"Traducido archivo n°{a}")
```

Todo este código viene englobado en un bucle for con rango desde 1 hasta el número de chunks que hemos creado previamente. Como google colab puede dar errores de desconexión si estamos mucho tiempo inactivos en una sesión, hemos creado un bloque de código que guarda los trozos que ya llevamos traducidos en nuestra máquina local, para no perder tampoco información:

```
!zip -r /content/file.zip /content/translated
from google.colab import files
files.download("/content/file.zip")
```

5.4.- Diseño del dataset

Primero, debemos realizar la unión de los pequeños trozos de datasets que hemos realizado, para ello usamos pandas. Tenemos todos guardados en una carpeta llamada translated_second. Guardamos el path a esa carpeta en una nueva variable y accedemos a cada uno de los archivos que se encuentra en ella. Posteriormente, creamos un dataframe en el que iremos guardando toda la información que hay en cada chunk. Por último, creamos el archivo "translated_second_ES.csv" a partir del dataframe, una vez ya está lleno. El código es el siguiente:

```
import pandas as pd
import os

folder_path                                                          =
'/Users/miguelangelnavarroarenas/Documents/GitHub/TFG/translated_secon
d'
final_name = 'translated_second_ES.csv'

files    =    [file    for    file    in    os.listdir(folder_path)    if
file.endswith('.csv')]

all_data = pd.DataFrame()

for file in files:
    df = pd.read_csv(os.path.join(folder_path, file))
    all_data = all_data.append(df, ignore_index=True)

all_data.to_csv(final_name, index=False)
```

Como bien hemos dicho anteriormente, tenemos la necesidad de reunir los datos provenientes de distintos datasets. La estructura de cada una de ellas varía y tienen columnas diferentes entre cada una de ellas. Es por tanto que tenemos que establecer una estructura en la cuál guardemos la información necesaria y más importante de cada entrada, pero eliminando del origen la información redundante o innecesaria. Para ello, hemos creado una estructura como la siguiente para el dataset resultante

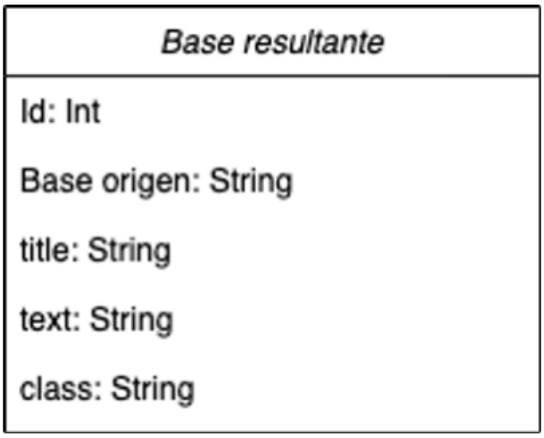

Figura 29: estructura base resultante

Podemos observar la estructura de los datasets traducidos y cómo son diferentes entre ellos:

	name	subreddit	title	selftext	annotation	binary_annotation	confidence
0	tr_1h1j9z	Anxiety	phagophobia-Inability to take pills	no estoy seguro de como manejar esto.Lo he te…	suicide	0.0	3.0
1	t3_4phweu	Anxiety	Does anyone else notice their anxiety gets wor…	Como dice el titulo, he notado que mi ansiedad…	suicide	0.0	3.0
2	t3_29dnph	Anxiety	Not sure what´s been going on, but it´s but he…	Desde hace 3 años, pasé por esta fase obsesiva…	suicide	0.0	3.0
3	t3_v925z	Anxiety	[Help]How can you be a Great SO for someone w…	Recientemente me he vuelto cercana a una chica…	suicide	0.0	3.0
4	t3_2suefa	Anxiety	Thinking of going on medication because my anx…	Me he encontrado muy influeciado por mi ansie…	suicide	0.0	3.0

Tabla 9: Ejemplo dataset Suicide Aladag

- Name: es la codificación del nombre del usuario.
- Subreddit: es el subforo de reddit.
- Title: es el titulo del hilo de reddit.
- Selftext: es el contenido del texto que ha creado el usuario.
- Annotation: es la etiqueta.
- Binary_annotation: es la etiqueta en binario.
- Confidence: es el grado de confianza.

	Unnamed:0	text	Class
0	255141	Ex Wife Threatening SuicideRecently I left my wife for good because she has cheated on me twice and …	Suicide
1	255142	Am I weird I don't get affected by compliments if it's coming from someone I know irl but I feel rea…	Non-suicide
2	255143	Finally 2020 is almost over... So I can never hear "2020 has been a bad year" ever again. I swear to…	Suicide
3	255144	i need help just help me im crying so hard	Suicide
4	255145	I'm so lostHello, my name is Adam (16) and I've been struggling for years and I'm afraid. Through th…	Suicide

Tabla 10: Ejemplo dataset Kaggle

- Unnamed: es el ID.
- Text: es el texto del propio usuario.
- Class: es la etiqueta.

También, hemos observado que las etiquetas de algunos datasets son diferentes a las necesarias. Por ejemplo, en el dataset "SuicideAladag" las etiquetas para el suicidio son muy amplias, como puede ser: "DEPRESSED", "SUICIDE-NOTE", "ATTEMPED", "NON-SUICIDE", etc. Por tanto, necesitamos que todas las etiquetas resultantes sean iguales, ya que sino el clasificador no funcionará correctamente. Como esta es una versión aún inicial del proyecto, vamos a hacer un clasificador lineal con dos clases: suicide y non-suicide (para adaptarnos al dataset que proveniente de Kaggle y así simplificar todo). Por tanto, hemos traducido todas aquellas etiquetas de este primer datset no fueran "NON-SUICIDE" a "suicide" y aquellas que son "NON-SUICIDE" las hemos traducido a "non-suicide". Así, hemos conseguido traducir el dataset y estandarizar las etiquetas del mismo. Esto último, lo hemos realizado con el siguiente código:

```
df = pd.read_csv('translated_annotated_ES.csv')
df['annotation'] = df['annotation'].apply(lambda x: 'non-suicide' if x
== 'NON-SUICIDAL' else 'suicide')
df.to_csv('annotated_ES_finale.csv', index=False)
```

A continuación, tenemos que aunar los datos de la primera y de la segunda en la estructura que hemos creado inicialmente (en futuras versiones, puede variar, ya que puede ser que sea necesario añadir información proveniente de datasets nuevos). Para realizar la unión, hemos usado el siguiente fragmento de código:

```
# CARGAMOS LOS DATASETS ORIGINALES
df1 = pd.read_csv('translated_second_ES.csv')
df2 = pd.read_csv('annotated_ES_finale.csv')

# SELECCIONAMOS SOLO LAS NECESARIAS DE CADA UNA DE ELLAS
df1 = df1[['text', 'class']]
df2 = df2[['title', 'selftext', 'annotation']]

# RENOMBRAMOS PARA DAR CONSISTENCIA
df2 = df2.rename(columns={'selftext': 'text', 'annotation': 'class'})

# CREAMOS LA COLUMNA BASE ORIGEN PARA CADA UNA DE LOS DATASETS
df1['Base origen'] = 'Base de datos 1'
df2['Base origen'] = 'Base de datos 2'
```

```
# UNIMOS SIN TENER EN CUENTA LOS POSIBLES DUPLICADOS
result = pd.concat([df1, df2]).drop_duplicates()
result['title'] = result['title'].fillna('')

# CREAMOS LA COLUMNA TITLE, AJUSTAMOS AL DATASET FINAL Y CREAMOS EL
ARCHIVO
result = result[['Base origen', 'title', 'text', 'class']]
result.to_csv('base_resultante.csv', index=False)
```

Por último, podemos observar la estructura final del dataset que queda como sigue:

	Base origen	title	text	class
0	Base de datos 1	NaN	No puedo elegir cómo Incluso ahora mis pobres ...	suicide
1	Base de datos 1	NaN	Hay un agujero en mis pantalones y no estoy us...	non-suicide
2	Base de datos 1	NaN	Necesito ayuda, soy un chico de 19 años de eda...	suicide
3	Base de datos 1	NaN	Exhausto de tratar de mejorarHe estado intenta...	suicide
4	Base de datos 1	NaN	He terminado de esperar en la fecha Sufro de d...	suicide

Figura 30: resultado dataset resultante traducido

La primera de las columnas indica el dataset de origen de donde viene cada dato. En caso de aparecer base de datos 1 proviene del dataset de Kaggle, en caso de base de datos 2 se trata de SuicideAladag. La intención de guardar el dataset de dónde proviene cada entrada es que podamos hacer backtracking de los usuarios, para saber en qué red social escriben, cuando escribieron o en qué hilo escribieron. Esto lo hacemos con la intención de que para futuras versiones podamos no sólo evaluar a un usuario por un único tweet o publicación, sino que podamos hacer un histórico de lo que publica y cómo evoluciona a lo largo del tiempo.

Ahora vamos a verificar cómo están de balanceados los datos, para así obtener los mejores resultados posibles, por tanto, vamos a contabilizar cuántos datos tenemos con la etiqueta suicide y cuántos tenemos con la etiqueta non-suicide. Obtenemos los siguientes datos:

```
Valores únicos de la columna "class": ['suicide' 'non-suicide']
Conteo de valores en la columna "class":
suicide        116037
non-suicide    116037
Name: class, dtype: int64
```

Figura 31: datos del dataset traducido

Una vez ya tenemos preparado el dataset, procedemos al entrenamiento.

5.5- Entrenamiento

Este proceso lo dividiremos en tres tecnologías diferentes: máquinas de vectores soporte, redes neuronales y, por último, redes neuronales con BERT. Para realizar el entrenamiento y ajustarnos a nuestra idea inicial de prever al usuario los resultados de varios modelos de aprendizaje automático, hemos decidido entrenar 3 modelos de clasificación: una máquina de vectores soporte (SVM), una red neuronal multicapa y un modelo basado en una arquitectura atencional, en este caso, un modelo con BERT. Veremos cómo hemos actuado en cada una de ellas y cuales han sido los resultados obtenidos y si han sido o no satisfactorios. También, por último, nos resultó interesante saber qué resultados obtenemos si entrenamos el mismo modelo pero en inglés, y posteriormente, sólo traducimos la entrada de texto y eso es lo que le pasamos al modelo, para comparar con los resultados obtenidos en nuestro trabajo.

Para cada uno de los entrenamientos hemos usado el mismo dataset, el cuál hemos estandarizado en los pasos anteriores. Para ser lo más estándar posible, antes de cualquier ejecución, hemos dividido el dataset en dos partes: una de entrenamiento y una de validación. Los 3 modelos los entrenaremos con la primera de estas partes y los validaremos con la segunda. Para asegurarnos que se mantiene el equilibrio de etiquetas también en las partes creadas, hemos usado un fragmento de código en python para hacer dicha división. Dicho fragmento divide las entradas con etiquetas ‘suicide’ y ‘non-suicide’. Posteriormente, de cada una de las partes hacemos un train_test_split, en el que subdividimos cada uno de estos grupos en dos, quedándonos cuatro resultantes: train_suicide, val_suicide, train_non_suicide y val_non_suicide. Los conjuntos de validación suponen un 15% del total (esto lo especificamos dentro de los parámetros de entrada de la función train_test_split). Por último, unimos los conjuntos que son de entrenamiento por un lado y los que son de validación por otro y guardamos cada uno de estos resultados en un archivo csv de salida.

Para comprobar que esta división ha sido correctamente hecha, hemos creado otro fragmento de código en python el cual ve que las etiquetas estén correctamente divididas y equilibradas. Para ello, hacemos un conteo de las mismas y mostramos los resultados por pantalla, obteniendo lo siguiente:

Conjunto de entrenamiento	Nº de muestras
Suicide	99077
Non-suicide	98786
Conjunto de validación	Nº de muestras
Suicide	17485
Non-suicide	17433

Tabla 11: Número de muestras dataset entrenamiento y validación

Como podemos observar, los conjuntos tienen casi que el mismo número en ambas divisiones. También podemos observar que los datos del conjunto de validación suponen un 15% del total aproximadamente.

Máquinas de vectores soporte

Como nuestras etiquetas sólo tienen dos categorías, las máquinas de vectores soporte son unas perfectas candidatas a darnos un muy buen rendimiento en nuestra tarea de clasificación. Como bien hemos mencionado anteriormente, para hacerlas funcionar haremos uso de las siguientes tecnologías: pandas, numpy, matplot, sklearn y nltk.

Primero, debemos tener en cuenta que hay varios parámetros que debemos ajustar a la hora de optimizar el uso de las SVM. La primera de ellas es el kernel a utilizar. A grandes rasgos, el kernel es la forma de transformar los datos para "añadir" más dimensiones, si es necesario , y diferenciar los espacios creados para discriminar entre ellos. De manera más técnica, un kernel es una función con dos entradas la cual devuelve un valor. Mide la similitud entre dos vectores del espacio de características. Debido a cómo se estructuran los datos, podemos tener kernels de distintas formas, algunos de ellos incluso pueden llegar a añadir varias dimensiones al plano para poder ajustarse de la mejor manera a los datos (Por ejemplo, podemos añadir varias dimensiones extra para aquellos datos que no podamos clasificar con una línea pero si los podríamos clasificar con un plano, es el caso de, por ejemplo, cuando no tenemos una frontera plana y tenemos otras formas de frontera, como un círculo, el cual puede transformarse en el corte de un plano en 2D a un espacio en 3D. Lo vemos mejor en la siguiente imagen).

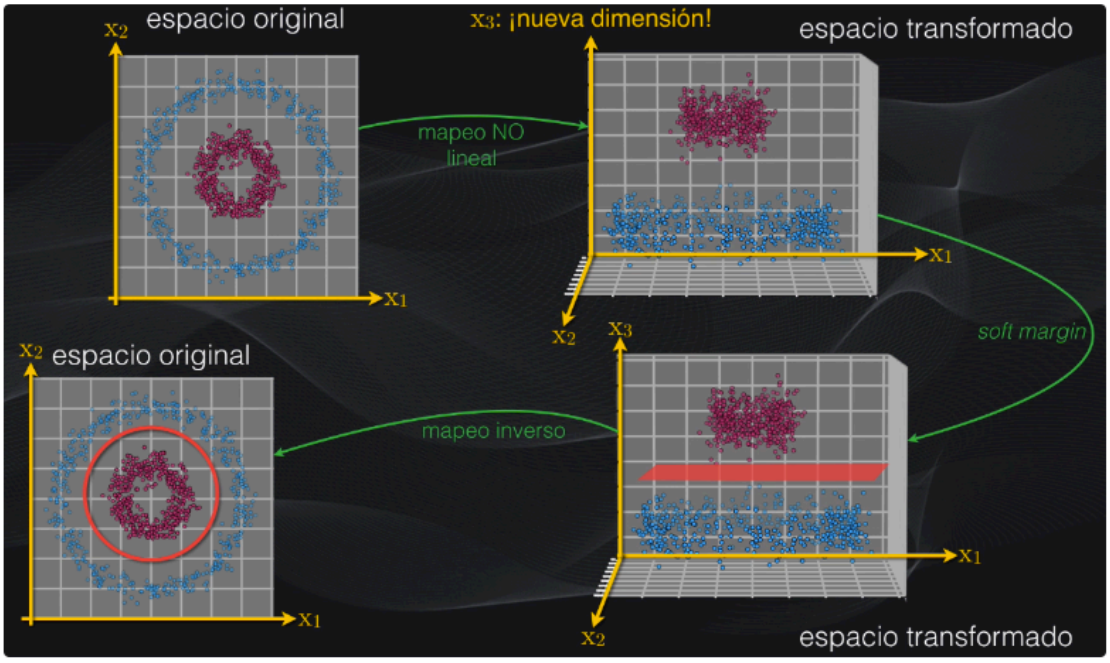

Figura 32: explicación dimensiones máquinas de vectores soporte

Enlace de origen de la imagen: https://www.codificandobits.com/blog/maquinas-de-soporte-vectorial/

A parte del kernel, las máquinas de vectores soporte tienen otros parámetros importantes, como “C”, que es el parámetro de regularización o el grado (degree) de la función polinomial que vayamos a utilizar (aunque para la sigmoide no sea necesario este parámetro, ya que sería lo mismo hacer una sigmoide con degree=2 y degree=3, pero para automatizar el código esto es necesario y así nos es más sencillo ejecutar todo el código de pruebas de una). Por tanto, en la librería de sklearn, tenemos una herramienta capaz de ayudarnos a elegir el mejor estimador. Pero, hacer este trabajo con todo nuestro dataset ocuparía demasiado tiempo, por tanto, hemos cogido sólo el 10% de ella para hacer las pruebas, manteniendo el balanceo de 50% para las dos clases. Usando GridSearchCV de esta librería, hemos estimado como posibles kernels: lineal, poly y sigmoid, como posibles valores de C: 0.1, 1 y 10 y como grado: 2 y 3. Posteriormente, hemos dividido nuestra pequeña base de prueba en conjunto de test, entrenamiento y validación mediante la herramienta train_test_split también de sklearn. Una vez tenemos estos, hacemos un fit del modelo creado a partir de GridSearchCV y los parámetros establecidos, y obtenemos el siguiente código y los siguientes resultados:

```
parameters = {'kernel':['linear','poly','sigmoid'], 'C':[1e-1,1,10],
'degree': [2,3]}
svc = svm.SVC()
model = GridSearchCV(estimator=svc, param_grid=parameters, n_jobs=-1,
cv=10,verbose=2)
model.fit(Train_X_Tfidf, Train_Y)
```

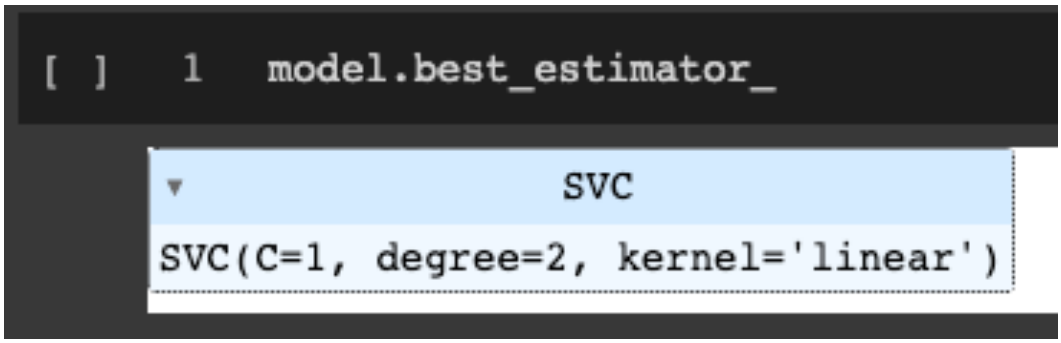

Figura 33: mejores parámetros SVM

Una vez ya tenemos el estimador que va a obtener mejores resultados, debemos preprocesar los datos de nuestro dataset para evitar que algunos símbolos que son muy comunes en internet (como @, #, %) den alguna clase de error, por tanto, usamos expresiones regulares para eliminarlos y que nos quedemos simplemente con el texto.

```
for entry in Corpus["text"]:
  text2 = re.sub('\\W+', ' ', str(entry))
  tokens2 = text2.split()
  Corpus["text"][i] = tokens2
  i+=1
```

Posteriormente, guardamos en una variable los textos para el entrenamiento y en otra las etiquetas. Ahora, usamos encoders para las etiquetas, que cambian de las clases “suicide” y “non-suicide” a 0 y 1 para facilitar el procesamiento de los datos. Normalmente, las

etiquetas con texto con más sencillas de entender para los humanos, pero para las máquinas es mucho más sencillo trabajar con datos numéricos que con cadenas de texto. Por tanto, la columna class pasa a tener sólo los valores 0 y 1.

Antes del entrenamiento, necesitamos vectorizar los textos de entrada. Para ello los convertiremos los textos a vectores TF-IDF (Term Frecuency.-Inverse Document Frecuency). Esto lo hacemos porque los textos son datos de alta dimensionalidad, es decir, cada palabra se podría considerar como una dimensión en un espacio vectorial. Además esto nos va a ayudar a resaltar aquellas palabras que tengan más importancia, ya que no sólo tenemos en cuenta la palabra en sí, sino también en el conjunto completo de los datos. Además, se ha comprobado que estos vectores mejoran el desempeño en algunas tareas, incluida la clasificación de texto.

Al usar el encoder y el paso a vectores TF-IDF, estamos preprocesando nuestros datos y adaptándolos para que sean más apropiados para el entrenamiento.

Por último, pasamos a entrenar el modelo con los parámetros óptimos: C=1.0, kernel=linear, degree=2 y gamma=auto.

Red neuronal

En este caso hemos utilizado tensorflow [9] para nuestra red neuronal. Lo hemos acompañado de las librerías pandas, numpy y sklearn. También hemos usado keras en la parte de tensorflow. Primero, tras cargar las librerías, hemos cargado el dataset con con pandas. Una vez hecho esto, dividimos el conjunto de pruebas, test y validación en con "train_test_split" de sklearn. Posteriormente, procedemos a usar LabelEncoder para codificar las etiquetas de nuestro modelo y tokenizamos los textos de nuestro dataset, para que tengan un formato adecuado para el entrenamiento (deben ser secuencias de números). El código de estos tres últimos pasos queda como a continuación:

```
X_train, X_test, y_train, y_test = train_test_split(train_text,
train_labels, test_size=0.2, random_state=42)

# Codificar las etiquetas utilizando LabelEncoder
label_encoder = LabelEncoder()
train_labels = label_encoder.fit_transform(train_labels)

# Tokenizar los textos
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
X_train_sequences = tokenizer.texts_to_sequences(X_train)
X_test_sequences = tokenizer.texts_to_sequences(X_test)
```

Ahora nos toca saber cuál será la mejor forma de nuestra red neuronal. Para ello nos hemos basado en la página de tensorflow (9) y en cómo estructura la red neuronal para el tratamiento de lenguaje natural. Hemos establecido una capa de Embedding inicial, con

una dimensión de entrada el tamaño de nuestro texto, una dimensión de salida de 100 y un tamaño máximo de entrada de secuencia de 1000. Después de ella, hemos usado una capa GlobalAveragePooling1D, aquí hemos variado un poco con respecto a lo que nos indica Tensorflow. Esta decisión ha sido tomada investigando en algunos foros y por la experiencia propia. Posteriormente, hemos añadido una capa densa con activación relu y tamaño 64 y, por último, otra capa densa de activación softmax y tamaño del número de etiquetas distinto que tenemos, que en nuestro caso es dos. Por último, hemos compilado el modelo con optimizador adam, metrica accuracy y hemos ajustado el parámetro de pérdida a sparse_categorical_crossentropy. El código resultante y su ejecución han resultado como a continuación:

```
    tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index) + 1,
output_dim=100, input_length=max_sequence_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(len(label_encoder.classes_),
activation='softmax')
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 1000, 100)         18326300

 global_average_pooling1d_4   (None, 100)              0
 (GlobalAveragePooling1D)

 dense_6 (Dense)             (None, 64)                6464

 dense_7 (Dense)             (None, 2)                 130

=================================================================
Total params: 18,332,894
Trainable params: 18,332,894
Non-trainable params: 0
_________________________________________________________________
```

Figura 34: resumen parámetros red neuronal

Una vez tenemos todo listo, procedemos a entrenar el modelo con:

```
model.fit(X_train_padded, y_train, validation_data = (X_test_padded,
y_test), epochs=10, batch_size=16)
```

Red neuronal con BERT

En este caso hemos usado las mismas tecnologías que en el anterior, pero le hemos añadido el tokenizador de Bert. Para ello, una vez hemos dividido los conjuntos de entrenamiento y test, pasamos a tokenizar las etiquetas con el LabelEncoder como hemos hecho anteriormente, pero ahora también tokenizamos los textos de entrada de la siguiente manera:

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
X_train_sequences = tokenizer.batch_encode_plus(
    X_train,
    add_special_tokens=True,
    return_attention_mask=True,
    padding='longest',
    truncation=True,
    max_length=100,
    return_tensors='tf'
)

X_test_sequences = tokenizer.batch_encode_plus(
    X_test,
    add_special_tokens=True,
    return_attention_mask=True,
    padding='longest',
    truncation=True,
    max_length=100,
    return_tensors='tf'
)
```

Con este tokenizador conseguimos traducir los textos a secuencias de números. Posteriormente, necesitamos cargar el modelo preentrenado de Bert. Según la documentación, con los modelos tokenizados necesitamos también extraer sus características para que a la hora del entrenamiento del modelo podamos suplir unos parámetros que son necesarios: los inputs y las máscaras de atención. Para extraerlos usamos el siguiente código:

```
X_train_input_ids = X_train_sequences['input_ids']
X_train_attention_mask = X_train_sequences['attention_mask']
X_test_input_ids = X_test_sequences['input_ids']
X_test_attention_mask = X_test_sequences['attention_mask']
```

Una vez tenemos estos parámetros, podemos usar model.compile para ajustar los parámetros del modelo, los cuales quedan como a continuación:

```
Model: "tf_bert_for_sequence_classification_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

 dropout_75 (Dropout)        multiple                  0

 classifier (Dense)          multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
```

Figura 35: resumen parámetros red neuronal con BERT

Una vez compilado el modelo, pasamos a entrenarlo con model.fit y los parámetros que hemos obtenido anteriormente.

6.- Pruebas de los modelos

En nuestro caso enfocamos este apartado a la obtención de resultados en cada una de las fases de la inteligencia artificial, ya sea en SVM, redes neuronales o redes neuronales con BERT. Las pruebas que hemos realizado en cada uno de ellos son: accuracy, matrices de confusión, recall y f1. En la parte de la app no se ha realizado nada de test, pero si se han realizado pruebas de comportamiento en la app y todo ha funcionado correctamente. A continuación, veremos los resultados obtenidos en cada apartado.

Máquinas de vectores soporte

En este caso hemos obtenido los siguientes valores de accuracy, recall y f1-score:

Accuracy score	93.35008699014111
Recall	0.9335268048975127
F1 score	0.9335004207221851

Tabla 12: accuracy, recall, f1 MVS

La matriz de confusión obtenida es la siguiente:

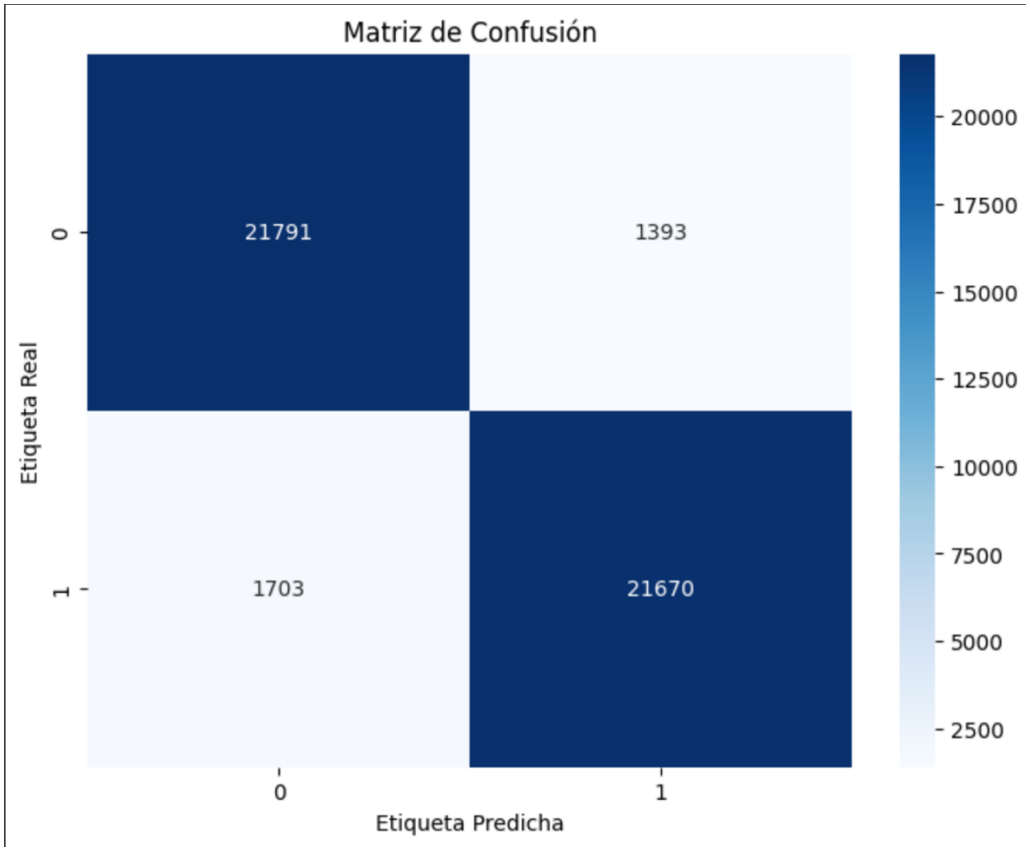

Figura 36: matriz de confusión SVM

En resumen, los resultados obtenidos son los siguientes (redondeados a dos decimales):

	Precisión	Recall	F1-score
0 (suicide)	0.93	0.94	0.93
1 (non-suicide)	0.94	0.93	0.93
Accuracy			0.93
Macro avg	0.93	0.93	0.93
Weighted avg	0.93	0.93	0.93

Tabla 13: resumen resultados SVM

Redes neuronales

Los resultados obtenidos son los siguientes:

Accuracy	93.63
Loss	0.18551714718341827
F1 score	0.9363285227040997
Recall	0.9363361041304208

Tabla 14: accuracy, recall, f1 redes neuronales

La tabla de las métricas de entrenamiento y validación queda como sigue:

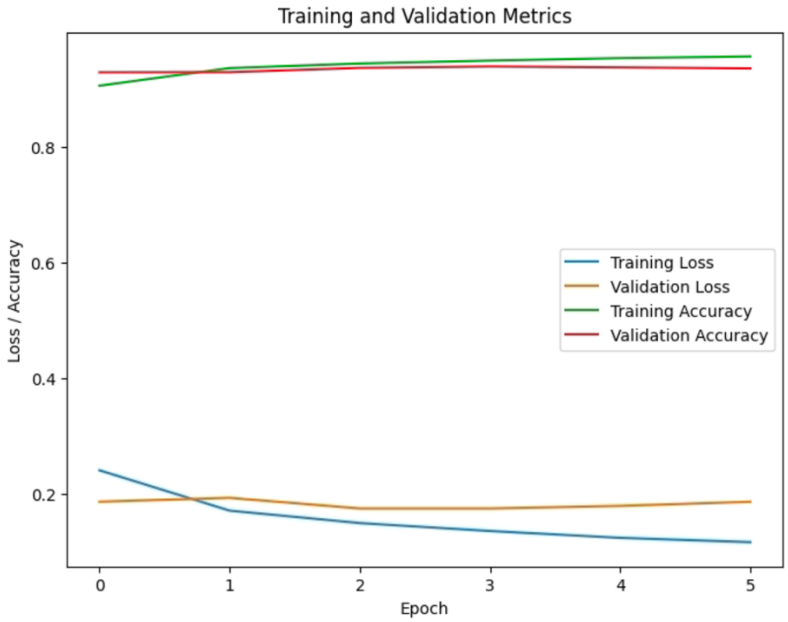

Figura 37: gráfico loss y accuracy red neuronal

La matriz de confusión queda de la siguiente manera:

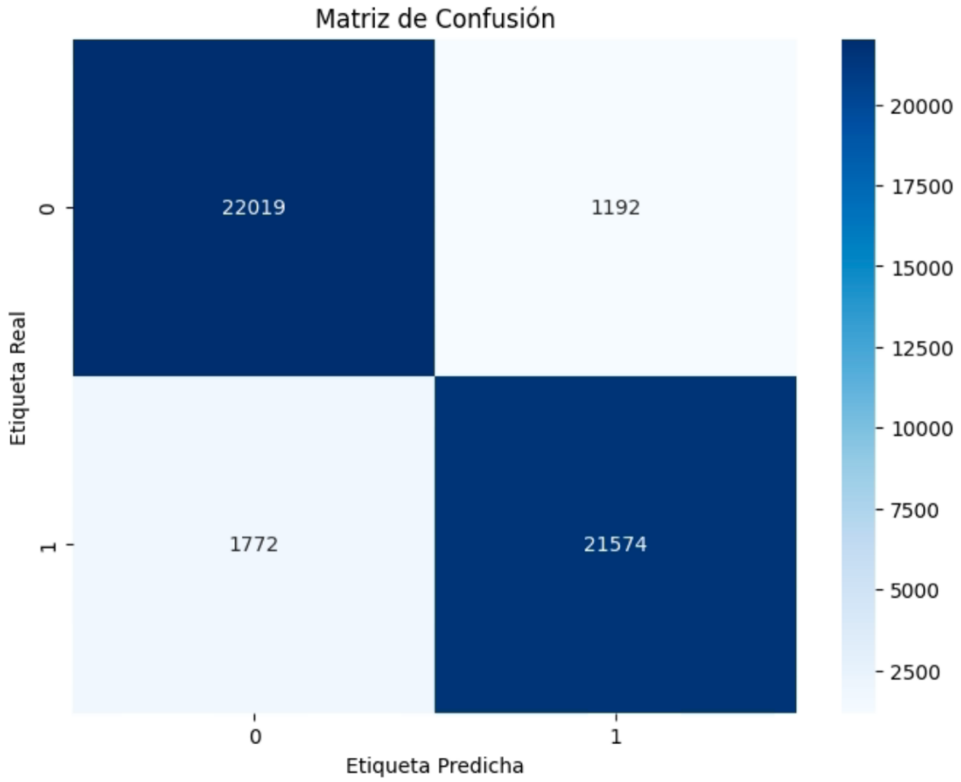

Figura 38: matriz de confusión red neuronal

Redes neuronales con BERT

Los resultados obtenidos son los siguientes:

Accuracy	0.9367374992840369
F1 score	0.9369379656855747
Precision score	0.9353625170998632
Recall score	0.9385187303402917

Tabla 15: accuracy, f1, precisión recall redes neuronales + BERT

Los datos en los conjuntos de entrenamiento y validación han arrojado los siguientes resultados:

Entrenamiento		
	Accuracy	0.9715785780964783
	Loss	0.11038154528162424
Validación		
	Accuracy	0.9367374992840369
	Loss	0.27511359543132796

Tabla 16: resultados entrenamiento-validación redes neuronales + BERT

Matriz de confusión:

Etiqueta real			
0 (suicide)	16299	1134	
1 (non-suicide)	1075	16410	
	0 (suicide)	1 (non-suicide)	Etiqueta predicha

Tabla 17: matriz confusión redes neuronales + BERT

Resumen de los resultados redondeando a dos decimales:

	Precisión	Recall	F1 score
0 (suicide)	0.94	0.93	0.94
1 (non-suicide)	0.94	0.94	0.94
Accuracy			0.94
Macro avg	0.94	0.94	0.94
Weighted avg	0.93	0.94	0.94

Tabla 18: resumen resultados redes neuronales + BERT

7.- Creación de la aplicación

7.1.- Creación del framework (paquetes y archivos básicos)

Para esta app hemos usado Flask. La estructura de carpetas de este framework es muy sencilla y será como sigue:

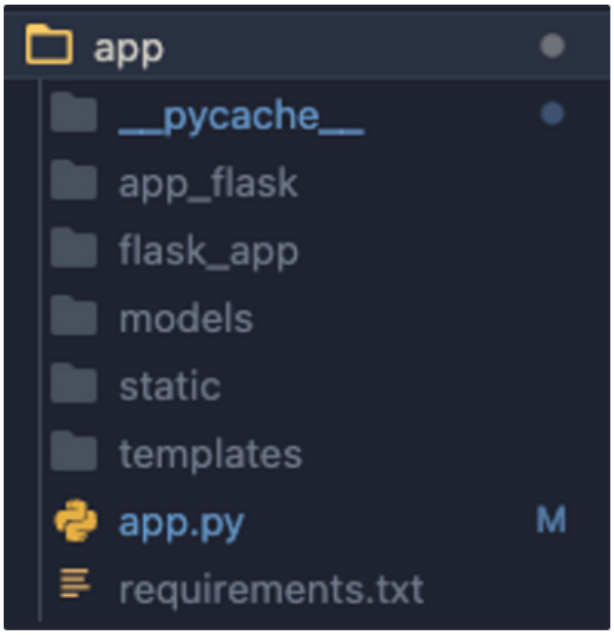

Figura 39: resumen paquetes Flask

Tanto la carpeta "app_flask" como la carpeta "flask_app" se han creado automáticamente a la hora de ejecutar el programa, al igual que la de "**pycache**". Dentro de la carpeta models se encuentran alojados nuestros modelos de SVM, redes neuronales y redes neuronales con BERT. Dentro de la carpeta static encontramos otra carpeta llamada css y dentro de esta el archivo main.css que tiene el siguiente código:

```
/* Estilos generales */
body {
    font-family: Arial, sans-serif;
}

.container {
    margin-top: 40px;
}

h1 {
    margin-bottom: 20px;
}

/* Estilos para el formulario */
.form-group {
    margin-bottom: 20px;
}
```

```
/* Estilos para la tabla de resultados */
.table {
    margin-top: 20px;
}

/* Estilos responsivos */
@media (max-width: 576px) {
    .container {
        margin-top: 20px;
    }
}
```

Dentro de la carpeta templates tenemos seis archivos: index.html, reddit.html, texto.html, resultado_reddit.html, resultado_texto.html y twitter.html. El primero es la página principal y el segundo sirve para visualizar los datos. El código de estos archivos lo veremos en la sección "5.6.3.- Creación de la interfaz.".

Por último, el archivo app.py constituye el corazón de nuestra aplicación, desde donde se ejecutará todo el código que iremos desgranando en las siguientes secciones.

También tenemos el archivo requirements.txt, el cual almacena todas las dependencias necesarias para ejecutar nuestra aplicación en cualquier otro computador.

7.2.- Conexión API Twitter

Para realizar la conexión con la API de Twitter y poder volcar los últimos tweets del usuario, hemos usado la librería tweetpy (previamente la debemos tener instalada). Para hacer uso de la API de twitter (al menos en las versiones anteriores) es necesario tener cuatro claves: la clave del consumidor, el secreto del consumidor, el token de acceso y el secreto del token de acceso. Por tanto, hemos creado cuatro variables generales en las cuales se introducen estos datos.

Para acceder a los tweets, hemos creado la función obtener_tweets, con los parámetros usuario y cantidad (definida por defecto a 10). Esta función hace la conexión con la API como lo indica tweetpy y devuelve los tweets de dicho usuario. El código de esta función es el siguiente:

```
def obtener_tweets(usuario, cantidad=10):
    # Autenticación en la API de Twitter
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)

    # Obtiene los últimos tweets del usuario
    tweets = []
    try:
        for tweet in tweepy.Cursor(api.user_timeline,
screen_name=usuario, tweet_mode="extended").items(cantidad):
            tweets.append(tweet.full_text)
    except tweepy.TweepError:
        return None

    return tweets
```

7.3.- Conexión API reddit

En este caso, es muy similar a la conexión a Twitter, pero usaremos la librería de Python llamada praw. El código necesario para obtener los posts es el siguiente:

```
def obtener_posts_reddit(usuario, cantidad=5):
    try:
        posts = []
        for submission in
reddit.redditor(usuario).submissions.new(limit=cantidad):
            posts.append(submission.title)
        return posts
    except Exception as e:
        print(e)
        return None
```

Cabe mencionar que previamente, es necesario ajustar los parámetros necesarios para la conexión: el clien_id, el client_secret y el user_agent (se deben guardar como variables de entorno). Por último, también es necesario crear un método para realizar la conexión con reddit, muy similar al siguiente:

```
@app.route('/reddit', methods=['GET', 'POST'])
def reddit():
    if request.method == 'POST':
        usuario = request.form['usuario']
        posts = obtener_posts_reddit(usuario)
        if posts is None:
            return "Error: No se pudo obtener las publicaciones de
Reddit del usuario."

        # Aquí se obtienen los resultados de los modelos
        # y se pasan al template resultado_reddit.html
        # resultados = [obtener_resultados_texto(post) for post in
posts]

        return render_template('resultado_reddit.html', posts=posts)

    return render_template('reddit.html')
```

7.4.- Utilización de los modelos

En nuestra app, una vez hemos obtenido los tweets, debemos procesarlos con nuestros modelos. Por tanto, creamos una función que tome como punto de partida estos tweets. Para procesarlos, debemos usar dos tecnologías distintas, una para leer el modelo de SVM y otro para leer los modelos de redes neuronales. Por cada tweet debemos obtener tres resultados, los cuales añadiremos a una tupla formada por el tweet y los resultados. Devolveremos esa tupla en la función que hemos creado para posteriormente visualizarla por pantalla. El código de nuestra función es el siguiente:

```
def obtener_resultados(tweets):
    resultados = []
    for tweet in tweets:
        # Realiza la clasificación con los modelos de inteligencia
artificial
        resultado_1 = modelo_h5.predict([tweet])[0]
        resultado_2 = modelo1_pk1.predict([tweet])[0]
                            resultado_3                             =
modelo2_pk1.predict([tweet])[0]
        resultados.append((tweet, resultado_1, resultado_2))

    return resultados
```

7.5.- Creación de la interfaz

La interfaz se ha creado usando HTML, CSS y bootstrap. Anteriormente hemos visto cuál era la forma de nuestro archivo CSS para añadir algunos estilos a nuestros archivos HTML. Posteriormente, debemos tener en cuenta que tenemos dos páginas que visitar: la principal y la que muestra los resultados. Nuestro archivo index.html sigue el siguiente código:

```
<!DOCTYPE html>
<html>

<head>
    <title>App de Twitter</title>
    <link                                                rel="stylesheet"
href="<https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstra
p.min.css>">
</head>

<body>
    <div class="container">
        <h1>App de Twitter</h1>
        <form method="POST">
            <div class="form-group">
                <label    for="usuario">Nombre    de    usuario    de
Twitter:</label>
                <input  type="text"  class="form-control"  id="usuario"
name="usuario" required>
            </div>
            <button   type="submit"   class="btn   btn-primary">Obtener
Tweets</button>
        </form>
    </div>
</body>

</html>
```

Como podemos observar, tiene simplemente un header con el título “App de twitter”, un formulario con la etiqueta “Nombre de usuario de Twitter:” y un botón para hacer submit.

Por otro lado, el reddit.html es algo distinto:

```
<!DOCTYPE html>
<html>

<head>
    <title>Reddit</title>
</head>

<body>
    <h1>Ingresa un usuario de Reddit</h1>
    <form method="post">
        <input type="text" name="texto"><br>
        <input type="submit" value="Enviar">
    </form>
</body>

</html>
```

Y el texto.html tiene la siguiente forma:

```
!DOCTYPE html>
<html>

<head>
    <title>Texto Plano</title>
</head>

<body>
    <h1>Ingresa un texto</h1>
    <form method="post">
        <textarea name="texto"></textarea><br>
        <input type="submit" value="Enviar">
    </form>
</body>

</html>
```

Y por otro lado, usaremos el archivo de resultado_reddit.html como ejemplo de estructura para los archivos de resultados:

```
<!DOCTYPE html>
<html>

<head>
    <title>Resultados del Texto</title>
</head>

<body>
    <h1>Resultados</h1>
    <table border="1">
        <tr>
            <th>Texto Ingresado</th>
            <th>Resultado Modelo 1 (SVM)</th>
            <th>Resultado Modelo 2 (Redes neuronales)</th>
            <th>Resultado Modelo 3 (Redes con BERT)</th>
```

```
        </tr>
        <tr>
            <td>{{ texto }}</td>
            <td>{{ resultados[0] }}</td>
            <td>{{ resultados[1] }}</td>
            <td>{{ resultados[2] }}</td>
        </tr>
    </table>
    <a href="/"><button>Menú Principal</button></a>
</body>

</html>
```

Por último, veremos algunas capturas de cómo queda nuestra aplicación de manera básica, aún le faltaría mejorar el aspecto con algo más de Bootstrap y CSS:

Figura 40: interfaz index.html

Figura 41: interfaz texto.html

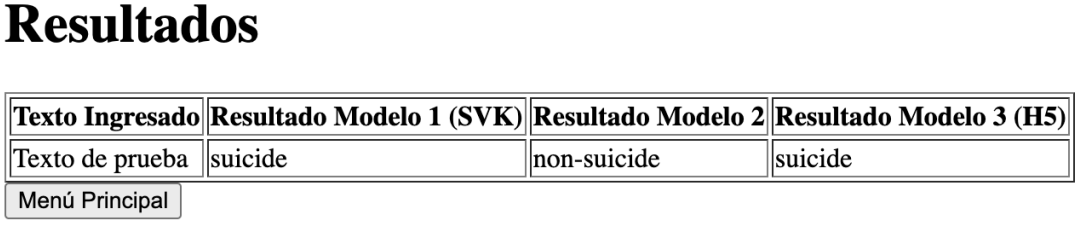

Figura 42: interfaz resultado_texto.html

Figura 43: interfaz reddit.html

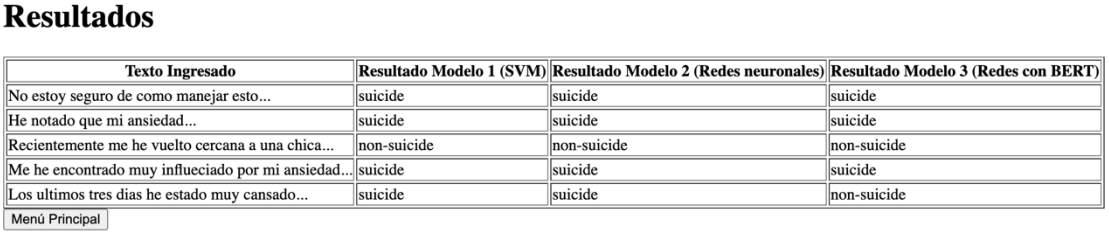

Figura 44: interfaz resultado_reddit.html

Por último, debemos mencionar que la aplicación no se ha podido terminar en su totalidad. Está en proceso de mejora debido a los cambios recientes en la API de Twitter (ahora llamado X). Especificaremos qué requisitos se han cumplido: RF-A-01, RF-A-02, RF-A-03, RF-A-04, RNF-A-01, RNF-A-02, RNF-A-03, RNF-A-04. La implementación de nuevas funcionalidades a la aplicación puede suponer la inclusión de nuevos requisitos funcionales, no funcionales o ambos, por tanto, para futuras versiones se necesitará añadirlos.

8.- Conclusiones

Tras el desarrollo de nuestro de nuestra solución, vamos a comentar las conclusiones que hemos conseguido alcanzar en la realización del mismo.

8.1.- Logro de los objetivos planteados

En cuanto a los objetivos planteados se han cumplido casi que todos, aunque hay algunos que se deben pulir debido los cambios recientes que ha habido en Twitter. Por otro lado, hay algunos objetivos que incluso no se han mencionado y se han dejado marcado para nuevas versiones de la aplicación. Las inteligencias artificiales han resultado todas satisfactorias, desarrollando todas resultados que esperábamos y muy buenos. También hemos conseguido implementarlo en nuestra aplicación, la cual cumple con los requisitos siendo intuitiva, rápida y ofreciendo una conexión y una interfaz que se adecúan con nuestros objetivos marcados.

La funcionalidad de la aplicación la hacen muy óptima para poder ser usada en el ámbito profesional ya que apoyamos a uno de los grandes problemas que tiene la sociedad moderna. Además, en la era de las nuevas tecnologías y la inteligencia artificial, acercamos estas herramientas a profesionales que por la naturaleza de su trabajo no tienen tanto acceso a nuevas herramientas.

8.2.- Evaluación de los resultados

Los resultados son muy buenos, aunque en algunos aspectos puede ser mejorables. Estamos en unos porcentajes de precisión muy altos y con los cuales se puede llegar a una versión inicial muy completa. Cualquier proceso de inteligencia artificial que supere la barrera del 75% de precisión puede considerarse como una buena inteligencia artificial (20). Aunque nosotros nos encontremos en unos valores por encima del 90%, nuestro objetivo es llegar a valores en torno al 95%, lo cual sería perfecto, ya que es muy complicado mejorar estos números. También podemos ver que nuestros números de falsos positivos y falsos negativos en las matrices de confusión son bastante pequeños, lo cual significa que en muy pocos casos se darán resultados erróneos. Además que también, a posteriori, intervendrá en la decisión final la que tome el experto por lo tanto se disminuirá al cero el error provocado.

8.3.- Propuestas de mejora y futuros trabajos

La idea a futuro se puede dividir desde dos puntos de vista: el del entrenamiento de los modelos y el de mejora de la aplicación.

Desde el punto de vista de la mejora de la inteligencia artificial podemos optar por aumentar nuestros datasets para los entrenamientos. También podemos mejorar algunos de los parámetros de los modelos que ya tenemos entrenados para que arrojen mejores resultados. Por último, podemos usar nuevos modelos de entrenamiento y cruzar los datos con los que ya tenemos, para mejorar aún más la experiencia. También una vez tengamos varios modelos, podemos usar los datos obtenidos en cada uno de ellos y usarlos de

manera conjunta, no individual, para obtener un resultado más robusto y así evitar los falsos positivos o los falsos negativos usando varios modelos para ello.

Desde el punto de vista de la aplicación podemos empezar por mejorar la interfaz de la misma, ya que actualmente está con una estructura muy sencilla. Podemos aplicar mejoras responsive para adaptarla a nuevos tamaños de pantalla. En cuanto a la parte de backend de la aplicación, podemos conectar dentro de una misma aplicación con todas las redes sociales y que sea el usuario el que decida en qué red quiere que se busque al usuario que ha introducido. También podemos ofrecerle la posibilidad de que haya un cuadro de texto y de manera manual vaya introduciendo las publicaciones del usuario y se le vayan mostrando los resultados por pantalla.

8.4.- Conclusión

La app y los modelos de inteligencia artificial han sido resueltos con unos resultados, aunque mejorables, muy buenos. Hemos sabido abordar el problema que se nos planteaba al principio del año de una manera eficiente y obteniendo muy buenos resultados en nuestro trabajo. Para mi, como estudiante de la rama de computación supone un broche final muy bonito a la carrera, usando mis conocimientos para apoyar buenas causas como lo es la prevención del suicidio. He usado mis conocimientos y background empresarial para demostrar que soy capaz de plasmarlos en código y papel.

9.- Bibliografía

[1] Organización Mundial de la Salud. (28 de septiembre de 2022). Salud mental en el trabajo. Recuperado de https://www.who.int/es/news-room/fact-sheets/detail/mental-health-at-work

[2] Organización Mundial de la Salud. (2019). Suicide Worldwide 2019. Recuperado de https://www.who.int/publications/i/item/9789240026643

[3] Organización Mundial de la Salud. (9 de septiembre de 2019). Un suicidio cada 40 segundos. Recuperado de https://www.who.int/es/news/item/09-09-2019-suicide-one-person-dies-every-40-seconds

[4] Statista. (8 de junio de 2023). Tasa de suicidios por cada 100,000 habitantes por países. Recuperado de https://es.statista.com/estadisticas/634746/tasa-de-suicidios-en-determinados-paises-por-genero/

[5] Prof Keith Hawton, Prof Kees van Heeringen. (24 de abril de 2009). Suicide. ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S014067360960372X

[6] A. Goñi Sarriés, M. Zandio Zorrilla. (2017). El suicidio, un reto para la prevención. Scielo. https://scielo.isciii.es/scielo.php?pid=S1137-66272017000300335&script=sci_arttext&tlng=en#B3

[7] Datos Macro. (2021). Número de suicidios en España. Recuperado de https://datosmacro.expansion.com/demografia/mortalidad/causas-muerte/suicidio/espana

[8] Komati, N. (2021). Suicide detection database. Kaggle. Recuperado de https://www.kaggle.com/datasets/nikhileswarkomati/suicide-watch

[9] TensorFlow. (n.d.). Clasificación de textos con RNN. Recuperado de https://www.tensorflow.org/text/tutorials/text_classification_rnn?hl=es-419

[10] Centros para el Control y la Prevención de Enfermedades. (n.d.). Does depression increase the risk of suicide? Recuperado de https://www.cdc.gov/suicide/factors/index.html

[11] eRisk. (n.d.). Página principal. Recuperado de https://erisk.irlab.org/

[12] Kaggle. (n.d.). Página principal. Recuperado de https://www.kaggle.com/

[13] HuggingFace. (n.d.). Datasets de suicidio. Recuperado de https://huggingface.co/datasets?sort=downloads&search=suicide

[14] STOP Project. (n.d.). Suicide prevenTion in sOcial Platforms. Recuperado de https://stop-project.github.io/

[15] Crisis TextLine. (n.d.). Página principal. Recuperado de https://www.crisistextline.org/

[16] The Trevor Project. (n.d.). Página principal. Recuperado de https://www.thetrevorproject.org/

[17] Samaritans. (n.d.). Samaritans Radar. Recuperado de https://www.samaritans.org/about-samaritans/research-policy/internet-suicide/samaritans-radar/

[18] IEEE. (n.d.). Estándar IEEE. Recuperado de https://www.ctr.unican.es/asignaturas/is1/ieee830_esp.pdf

[19] Boletín Oficial del Estado. (2016). Reglamento General de Protección de Datos. Recuperado de https://www.boe.es/doue/2016/119/L00001-00088.pdf

[20] Bizagi. (n.d.). Creación de modelos y experimentos, para el porcentaje de precisión de una buena IA. Recuperado de https://help.bizagi.com/automation-service/es/index.html?cloud_ai_create.htm

[21] Shaoxiong Ji, Shirui Pan, Xue Li, Erik Cambria, Guodong Long, Zi Huang. (6 de septiembre de 2022). Suicidal Ideation Detection: A Review of Machine Learning Methods and Applications. Recuperado de https://arxiv.org/abs/1910.12611

[22] eRisk. (2018). eRisk 2018. Recuperado de https://early.irlab.org/2018/index.html

[23] Ahmet Emre Aladağ , Serra Muderrisoglu , Naz Berfu Akbas , Oguzhan Zahmacioglu , Haluk O Bingol. (21 de junio de 2018). SuicideAladag. Pubmed. https://pubmed.ncbi.nlm.nih.gov/29929945/

[24] Reddit. (n.d.). Suicide Watch. Recuperado de https://www.reddit.com/r/SuicideWatch/

[25] UKPLab. (15 agosto 2022). Easy NMT. GitHub. Recuperado de https://github.com/UKPLab/EasyNMT

[26] Wikipedia ("SeroBot"). (n.d.). Test de Turing. Recuperado de https://es.m.wikipedia.org/wiki/Prueba_de_Turing

[27] "cchristofher". (15 abril 2014). Conferencia de Dartmouth. Recuperado de https://darthmouthconference.wordpress.com/

[28] TensorFlow. (n.d.). BERT. Recuperado de https://www.tensorflow.org/text/tutorials/classify_text_with_bert?hl=es-419

[29] ELiRF. (n.d.). Página principal. Recuperado de https://vrain.upv.es/elirf/

[30] Sean MacAvaney, Anjali Mittu, Glen Coppersmith, Jeff Leintz, Philip Resnik. (2021). Community-level Research on Suicidality Prediction in a Secure Environment: Overview of the CLPsych 2021 Shared Task. Recuperado de https://aclanthology.org/2021.clpsych-1.7/

[31] [Autor no concreto]. (Marzo de 2018). Ley Orgánica de Protección de Datos de Carácter Personal 15/1999 (LOPD). Ley Orgánica 3/2018. Recuperado de: https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf

[32] [Autor no concreto] (12 julio 2002). Ley de Servicios de la Sociedad de la Informática y del Comercio Electrónico 34/2002 (LSSI). Boletín oficial del estado. Recuperado de: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**		X		
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**		X		
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.		X		
ODS 11. **Ciudades y comunidades sostenibles**.			X	
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**			X	
ODS 17. **Alianzas para lograr objetivos.**				X

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n. 46022. València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es

Euro-Inf Bachelor awarded by EQANIE

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

La OMS en el año 2015 junto con más de 150 jefes de estado firmaron la llamada agenda 2030. En ella se explicaban los objetivos de desarrollo sostenible (ODS). Se realizó con la finalidad de enfrentar los desafíos sociales, medioambientales, económicos y de salud de nuestro tiempo, con la finalidad de crear un mundo más justo y sostenible para nosotros y las futuras generaciones. En nuestro caso sólo cumplimos con un objetivo de manera directa, el ODS3: salud y bienestar. Nuestra idea es acercar las nuevas tecnologías a los profesionales de la salud para que sean mucho mejores y más eficientes ejerciendo su trabajo, sobre todo en el sector de la salud mental. Nuestra idea intenta apaciguar el gran impacto que provocan los suicidios y la depresión en nuestra sociedad desde el punto de vista humano, social y económico. Con nuestro proyecto podemos abaratar los costes de los servicios de salud, haciendo más accesible y universal la atención psicológica.

Por otro lado, de manera moderada nos relacionamos con: ODS1, ODS4, ODS8 y ODS10. Como bien hemos mencionado durante el proyecto, la depresión y el suicidio puede acarrear un gran peso económico para la sociedad, que acabamos pagando entre todos, permitiéndonos acabar con las diferencias económicas entre individuos al no tener que correr con esos gastos (ODS1). También afectamos de manera indirecta a la educación de las personas, su implementación podría incluir programas educativos sobre la importancia de la atención médica preventiva y cómo acceder a servicios de calidad (ODS4). En cuanto al empleo, estamos ayudando a las personas que trabajan en la psicología, haciéndoles más fácil su trabajo y, por tanto, generando empleo de calidad, lo cual también repercutirá de manera positiva en la salud mental de los trabajadores de este sector (ODS8). Por último, estamos reduciendo las desigualdades haciendo que la salud mental sea más accesible para todos (ODS10).

Por último, afectamos de manera leve a: ODS11 y ODS16. En el primero de los casos, al estar ayudando al sistema de salud, hacemos que las ciudades sean más sanas, provocando un desarrollo social y demográfico mayor, ya que las personas prefieren ciudades o países en los que el acceso a la salud sea gratuito, pero de calidad. En el segundo caso, al reducir la brecha entre las personas con más dinero (que tienen acceso a mejor calidad de salud) y las personas con menos recursos, estamos contribuyendo a la creación de una sociedad más justa.

Los demás ODS no tienen una relación directa con nuestro proyecto y, por lo tanto, no son aplicables en este contexto.

En resumen, nuestro proyecto tiene la capacidad de impactar múltiples ODS a diferentes niveles. Aunque el foco principal es el ODS 3, las implicancias secundarias en otros objetivos demuestran la interconexión inherente entre todos estos objetivos globales y subrayan la importancia de adoptar un enfoque multidimensional en el diseño e implementación de cualquier proyecto sostenible.

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n. 46022. València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es

Euro-Inf Bachelor awarded by EQANIE

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n. 46022. València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es