



Cofinanciado por  
la Unión Europea



Gobierno de España  
MINISTERIO DE INDUSTRIA  
y TURISMO



EOI Escuela de  
organización  
Industrial



Fondos Europeos

SAMSUNG

AI Course

# Capstone Project

## Final Report

For students (instructor review required)

©2023 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of this document.

This document is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this document other than the curriculum of Samsung Innovation Campus, you must receive written consent from copyright holder.



Cofinanciado por  
la Unión Europea



Fondos Europeos

SAMSUNG

# Análisis de emociones y determinación de depresión en textos

02/07/2025

Grupo 3

Ignacio Muñoz Ramírez  
Javier Santos Sintes  
Sandra Martínez Sanabria



Cofinanciado por  
la Unión Europea



Fondos Europeos

SAMSUNG

# Content

## 1. Introduction

- 1.1. Background Information
- 1.2. Motivation and Objective
- 1.3. Members and Role Assignments
- 1.4. Schedule and Milestones

## 2. Project Execution

- 2.1. Data Acquisition
- 2.2. Training Methodology
- 2.3. Workflow
- 2.4. System Diagram

## 3. Results

- 3.1. Exploratory Data Analysis (EDA)
- 3.2. Data Preprocessing
- 3.3. Modeling
- 3.4. User Interface
- 3.5. Testing and Improvements

## 4. Projected Impact

- 4.1. Accomplishments and Benefits
- 4.2. Future Improvements

## 5. Team Member Review and Comment

## 6. Instructor Review and Comment



## 1. Introduction

### 1.1. Background Information

El campo del análisis de texto y, más específicamente, la detección de estados emocionales y de salud mental a través del lenguaje, no es un área nueva de investigación, sino que se asienta sobre una sólida base de estudios en Procesamiento del Lenguaje Natural (PLN) y aprendizaje automático.

Diversas investigaciones han explorado el análisis de sentimientos el análisis de sentimientos en texto. Por ejemplo, estudios como el de Young et al. (2018) proporcionan una revisión exhaustiva de las técnicas de Machine Learning y Deep Learning aplicadas al análisis de sentimientos, demostrando madurez en el campo y variedad de técnicas utilizadas.

**Table 5**

Classification results of the machine learning and deep learning models on the test-set of the employed dataset.

Model	Accuracy (%)	F1 (%)	Precision (%)	Recall (%)	AUROC
KNN	89.1	88.5	<b>97.4</b>	81.1	0.89
SVM	87.0	87.4	87.9	86.8	0.87
AdaBoost	87.9	88.4	87.8	89.0	0.88
XGBoost	88.2	88.6	87.9	89.4	0.88
Decision Tree	85.0	85.4	85.8	85.0	0.85
Random Forest	<b>91.1</b>	<b>91.6</b>	89.5	<b>93.8</b>	<b>0.90</b>
Logistic Regression	83.4	83.7	84.7	82.8	0.83
MLP	81.8	87.8	87.1	88.6	0.75
CNN	83.1	88.8	87.3	90.4	0.76

Cuando hablamos de detección de depresión a través de texto, existe una corriente creciente en la literatura. Estudios como el de Hsu et al.(2023) sugieren trabajar de forma híbrida con Machine Learning, Deep Learning, Modelos de Lenguaje e IA Explicable. Maji y Kumar (2022) combinan Redes Neuronales híbridas, SBERT y CNNs para detección de depresión en plataformas como Reddit.

La base de estos avances radica en el trabajo de Devlin et al. (2019), que revolucionaron el NLP al proporcionar representaciones contextuales ricas del lenguaje, mejorando el rendimiento de tareas de clasificación de texto y análisis de sentimiento.

En este sector, los métodos de IA más comunes son:

- Redes Neuronales Convolucionales (CNNs)
- Redes Neuronales Recurrentes (RNNs) y sus variantes (LSTM, GRU)
- Modelos de Lenguaje Pre-entrenados (PLMs) / Transformers (ej. BERT, SBERT)
- Redes Neuronales Híbridas (ej., SBERT-CNN)
- Algoritmos de Machine Learning Tradicionales (KNN, SVM, Random Forest,...)
- Técnicas de IA Explicable (XAI)

### 1.2. Motivation and Objective

La depresión es un problema creciente que afecta en gran medida a nuestra sociedad. En España, la depresión es un problema de salud mental significativo, con alrededor de 3



Cofinanciado por  
la Unión Europea



Fondos Europeos



millones de personas afectadas, lo que representa aproximadamente el 5-10% de la población. Esta cifra se basa en estimaciones y encuestas de salud, y algunos estudios sugieren que podría haber aún más personas con depresión no diagnosticada.

**La depresión es la segunda causa de baja laboral en España** y tiene un impacto económico considerable, con un **gasto estimado de 23.000 millones de euros anuales**. De la misma manera, la preocupación por la salud mental también está creciendo y las redes sociales se han convertido en un espacio donde la gente expresa sus sentimientos y estados de ánimo, a veces indicadores de depresión. El análisis de datos de redes sociales se puede convertir en una herramienta útil y complementaria para identificar patrones de depresión, facilitando así su apoyo temprano.

El objetivo de este proyecto fue crear dos sistemas basados en Deep Learning para el análisis de texto y así poder facilitar la detección temprana de depresión en los trabajadores de una empresa. Al identificar proactivamente a los trabajadores en riesgo, no solo mejoramos su salud y bienestar emocional, sino que también se traduce en una reducción sustancial de los costes asociados a bajas laborales y un aumento de la productividad y rentabilidad de las empresas.

**Es muy importante recordar que un modelo de IA para detectar depresión es una herramienta de apoyo y nunca deberá reemplazar el diagnóstico de un profesional de la salud mental.**

### 1.3. Members and Role Assignments

Aunque en este proyecto todos los miembros hemos colaborado en cada fase, cada uno se especializó en un área concreta para ser más eficientes. Este es el resumen de esta distribución del proyecto en aspectos generales:

- Idealización (Sandra, Javier, Ignacio)
- Búsqueda datasets (Sandra)
- Tratamiento dataset depresión (Sandra)
- Entrenamiento modelo depresión (Javier, Ignacio)
- Tratamiento datasets emociones (Sandra)
- Entrenamiento modelo emociones (Javier, Ignacio)
- Fine tuning (Javier, Ignacio)
- Memoria proyecto (Sandra, Javier, Ignacio)
- Preparación exposición (Sandra, Javier, Ignacio)

Se ha utilizado Trello para la asignación de roles y el seguimiento del trabajo.



Cofinanciado por  
la Unión Europea



Fondos Europeos

SAMSUNG

The screenshot shows a Trello board with the following structure:

- Tareas pendientes del modelo de sentimientos (Future):**
  - Crear Frontend de la aplicación
  - + Añade una tarjeta
- Memoria:**
  - Introducción: members and role assignments
  - Project execution: system diagram
  - Results: User interface
  - Project impact: accomplishments and benefits
  - Project impact: future improvements
  - Team member review and comment
  - Instructor review and comment
  - + Añade una tarjeta
- En progreso (In progress):**
  - Introducción: Background information
  - Introducción: Motivation and objective
  - Introduction: schedule and milestone
  - Project execution: training methodology
  - Project execution: workflow
  - Results: EDA
  - Results: Modeling
  - Results: testing and improvements
  - Fine-tuning del modelo
  - + Añade una tarjeta
- Hecho (Done):**
  - Elección del modelo
  - Arquitectura del modelo
  - Entrenamiento del modelo
  - Ánalisis del score
  - Elección del modelo
  - Arquitectura del modelo
  - Results: data preprocessing
  - Project execution: Data acquisition
  - Entrenamiento del modelo
  - Ánalisis del score
  - Reevaluación del análisis del score
  - + Añade una tarjeta

Ilustración 1: Roles asignados mediante Trello

## 1.4. Schedule and Milestones

Para la gestión y seguimiento del proyecto, hemos elaborado un diagrama de Gantt. Este detalla las distintas fases, desde la idealización hasta la exposición final, junto con las fechas de inicio y fin previstas para cada tarea.

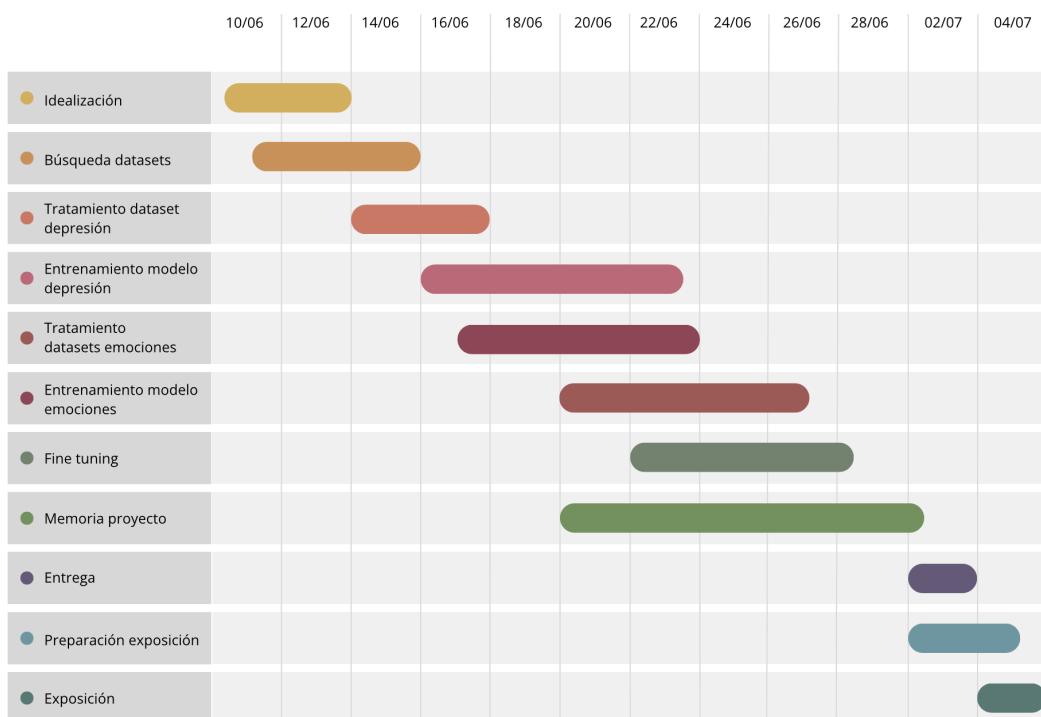


Ilustración 2: Diagrama de Gantt



## 2. Project Execution

### 2.1. Data Acquisition

Para el desarrollo de este proyecto, se utilizaron dos conjuntos de datos diferentes, uno para el modelo inicial que detecta depresión en comentarios de Reddit y otro para el modelo que analiza los sentimientos encontrados en un texto.

#### 2.1.1 Dataset depresión

El dataset relacionado con la depresión ha sido utilizado para crear un modelo que determine si un texto contiene indicios de depresión. Fue obtenido de Kaggle, y consiste en 2470778 publicaciones en Reddit en distintos foros. Este dataset se obtuvo en formato csv e incluía las siguientes variables:

- Subreddit: comunidad donde se publicó el post
- Title: título del post
- Body: contenido principal del post
- Upvotes: número de votos positivos recibidos
- Created UTC: fecha y hora de publicación (formato epoch)
- Number of comments: número de respuestas generadas
- Label: variable objetivo, con valores 0 (no depresivo) o 1 (depresivo)

En este dataset las variables más importantes son Title, Body y Label. Title y Body se han unificado para obtener un texto más completo. La variable Label, según se especifica en la información del dataset, se obtiene desde la variable Subreddit. Los comentarios en las comunidades 'SuicideWatch' y 'Depression' se ha determinado como 1 (depresivo) y los comentarios en las comunidades 'happy', 'DeepThoughts' y 'teenagers' como 0 (no depresivo).

#### 2.1.2 Dataset emociones

El dataset utilizado para el análisis de sentimientos viene constituido por 4 datasets distintos, tres de ellos se obtuvieron en github pero provienen de google research. Se tratan de 211225 comentarios en distintos foros de Reddit. El dataset se encontraba dividido en tres por lo que lo primero fue unificar los tres para obtener un dataset más completo. Las variables de este dataset son las siguientes:

- Text: comentario del usuario
- id: identificador único del comentario
- Author: nombre de usuario
- Subreddit: comunidad/foro donde se publicó el comentario
- Link\_id: identificador del post original
- Parent\_id: identificador del comentario
- Created\_utc: fecha y hora de publicación (formato epoch)
- Rater\_id: identificador del anotador que asignó la emoción al texto
- Example\_very\_unclear: indicador booleano que señala si el comentario era difícil de clasificar emocionalmente



Cofinanciado por  
la Unión Europea



Fondos Europeos

SAMSUNG

- 28 emociones: admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realization, relief, remorse, sadness, surprise, neutral.

En cuanto al cuarto dataset, se ha obtenido de Hugging Face donde especifican que se trata de un dataset en inglés de mensajes en Twitter con 6 emociones: anger, fear, joy, love, sadness y surprise. La página desde donde se obtuvo daba la oportunidad de descargarlo ya dividido en train/validation/test pero obtuvimos el dataset sin dividir para realizar todo el procesamiento del mismo. Por lo tanto contenía 416809 filas sin nulos.

En la página de Hugging Face de este dataset se aporta la siguiente información para la citación de este dataset:

#### Autor del Dataset

```
@inproceedings{saravia-etal-2018-carer,  
title = "{CARER}: Contextualized Affect Representations for Emotion Recognition",  
author = "Saravia, Elvis and  
Liu, Hsien-Chi Toby and  
Huang, Yen-Hao and  
Wu, Junlin and  
Chen, Yi-Shin",  
booktitle = "Proceedings of the 2018 Conference on Empirical Methods in Natural  
Language Processing",  
month = oct # "-" # nov,  
year = "2018",  
address = "Brussels, Belgium",  
publisher = "Association for Computational Linguistics",  
url = "<https://www.aclweb.org/anthology/D18-1404>",  
doi = "10.18653/v1/D18-1404",  
pages = "3687--3697",  
abstract = "Emotions are expressed in nuanced ways, which varies by collective or  
individual experiences, knowledge, and beliefs. Therefore, to understand emotion, as  
conveyed through text, a robust mechanism capable of capturing and modeling different
```



linguistic nuances and phenomena is needed. We propose a semi-supervised, graph-based algorithm to produce rich structural descriptors which serve as the building blocks for constructing contextualized affect representations from text. The pattern-based representations are further enriched with word embeddings and evaluated through several emotion recognition tasks. Our experimental results demonstrate that the proposed method outperforms state-of-the-art techniques on emotion recognition tasks.",

{}

La información se encontraba en formato diccionario, por lo que fue necesaria la transformación a DataFrame pandas. Además, la columna label definía las emociones con números: [0: sadness, 1: joy, 2: love, 3: anger, 4: fear, 5: surprise]. Se realizó un one-hot encoding mediante get\_dummies de pandas, proporcionando los nombres de las columnas posteriormente y modificando las columnas booleanas por columnas binarias.

## 2.2. Training Methodology

Este proyecto se basa en dos sistemas: uno para clasificar sentimientos (multiclasificación) que utiliza un conjunto de datasets, y otro para detectar depresión (binario) a través de un dataset con mensajes de Reddit. Para ambos proyectos, en rasgos generales, cargamos, limpiamos y preprocesamos los datos, posteriormente creamos las redes neuronales y las entrenamos con los datos vectorizados, y finalmente validamos y visualizamos los resultados.

### 2.2.1 Preprocesamiento de textos

Tras la limpieza del dataset (gestión de nulos y duplicados), nos quedamos con el texto en crudo y debemos limpiar y estandarizar los datos lo que mejora su calidad y permite ser interpretado por los algoritmos de aprendizaje automático. Estos pasos son:

- **Normalización del texto** → conversión a minúsculas
- **Eliminación de elementos irrelevantes** → URLs, @, #, caracteres especiales, puntuaciones excesivas,...
- **Tokenización** → división del texto en unidades pequeñas
- **Eliminación de Stop Words** → supresión de palabras muy comunes y que no aportan valor semántico
- **Lematización o Stemming** → reducción de las palabras a un base/raíz
- **Manejo de OOV** → palabras que aparecen en el conjunto de prueba pero fuera del vocabulario de entrenamiento

### 2.2.2 Vectorización de Textos

Una vez preprocesados los textos, deben ser transformados a representaciones numéricas (vectores) para que los modelos de Deep Learning los puedan procesar

- **Vectorización Externa para el modelo de detección de depresión.** Los textos se vectorizaron previamente a la división del dataset y a su entrada a la MLP mediante el vectorizador “TfidfVectorizer”.
- Vectorización Interna para el modelo de análisis de sentimiento. En este modelo se aplicó una CNN con una Capa de Embedding como primera capa para vectorizar cada texto

### 2.2.3 Creación y Configuración de las Redes Neuronales

- Red Neuronal Multilayer Perceptron (MLP)  
Compuesta por capas densas conectadas y regularización Dropout, diseñada para clasificación binaria. Una MLP es una arquitectura sencilla y eficiente para tareas de clasificación cuando los datos de entrada están bien representados numéricamente (gracias a TF-IDF).
  - **TfidfVectorizer.** Este vectorizador transforma el texto preprocesado (lematizado y sin stopwords) en un formato numérico, que es una representación numérica y compacta del texto.
  - **Dense (256).** Es la primera capa oculta de la red, que recibe los datos de entrada y donde comienza el proceso de aprendizaje de la red.
  - **Dropout (0.3).** Esta capa se aplica como técnica de regularización para prevenir el overfitting. Esto lo consigue “desactivando” el 30% (0.3) de las neuronas de la capa anterior de manera aleatoria en cada paso del entrenamiento.
  - **Dense (128).** Esta capa recibe la salida de la capa anterior, profundiza en la red y permite aprender representaciones aun más abstractas.
  - **Dropout (0.3).** Continúa el proceso de regularización.
  - **Dense (1).** Esta es la capa final de la red en la que se usa una sola neurona. La función sigmoide comprime la salida en un rango de 0 a 1. Es la elección estándar para problemas de clasificación binaria.

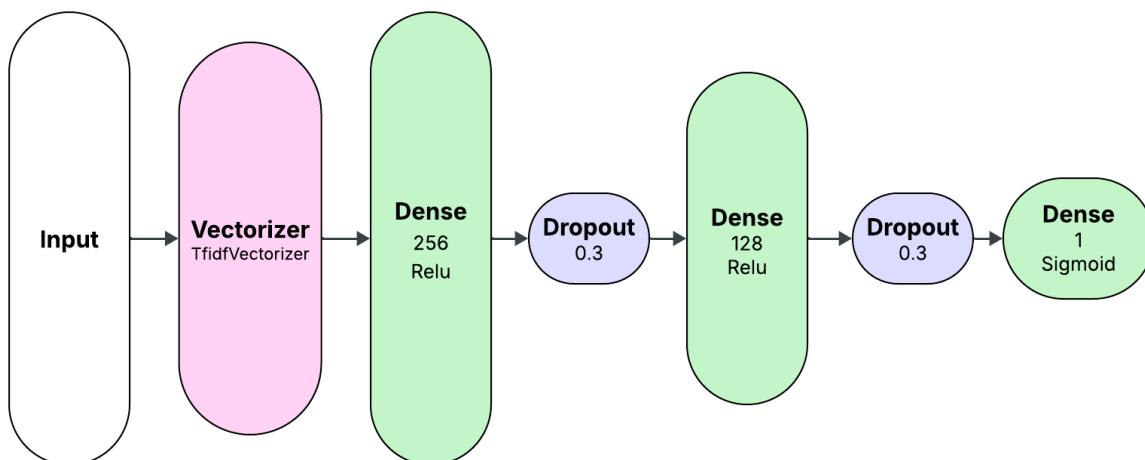


Ilustración 3: Representación modelo MLP

- Red Neuronal Convolucional (CNN)

Compuesta por una capa de embedding, capas convolucionales, una capa de pooling, capas de dropout y capas densas. Las CNN se utilizan comúnmente para tareas de análisis de sentimiento multiclas, además, su diseño permite capturar patrones y características locales en las secuencias de palabras.

- **Embedding.** Es la primera capa de la red. Toma los índices numéricos de cada palabra y los convierte en vectores densos de una dimensión específica.
- **Conv1D (128, 5).** Esta capa convolucional aplica 128 filtros con un kernel size de 5 a las secuencias. Cada filtro está diseñado para detectar un patrón específico como un n-grama o una combinación de palabras en cualquier parte del texto. Estos n-gramas son importantes para determinar el sentimiento, sin importar su posición exacta en la frase.
- **Dropout (0.3).** Primer paso para la regularización.
- **Conv1D (128, 3).** Esta capa convolucional funciona de la misma manera que la anterior pero disminuimos el kernel size a 3, para detectar n-gramas más pequeños.
- **Dropout (0.3).** Continúa el proceso de regularización.
- **GlobalMaxPooling1D.** Las capas de pooling reducen drásticamente la dimensionalidad de los datos, haciendo que las capas posteriores sean más eficientes. Su principal beneficio es que captura la presencia de patrones, sin importar en qué parte se encuentra del texto.
- **Dense (64).** Esta es una capa fully connected que recibe la salida completa de GlobalMaxPooling1D, procesan las características y preparan la clasificación final. Permite al modelo aprender combinaciones complejas de las características extraídas por las capas convolucionales, creando representaciones aún más refinadas para la tarea de clasificación.
- **Dropout (0.4).** Continúa el proceso de regularización y lo refuerza.
- **Dense (6).** Esta es la capa final con 6 neuronas, una para cada sentimiento. La función softmax, la elección estándar para la clasificación multiclas, convierte la salida de estas neuronas en una distribución donde la suma de todas las probabilidades de 1. La clase con probabilidad más alta se considera la predicción del modelo.

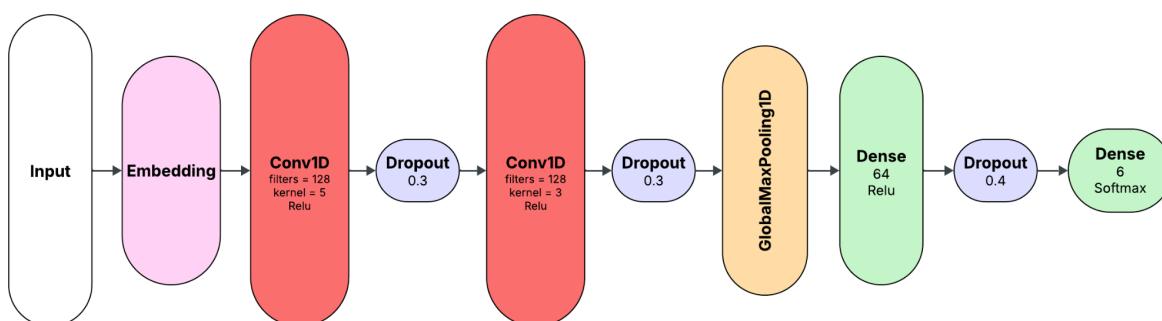


Ilustración 4: Representación modelo CNN

## 2.2.4 Entrenamiento

El entrenamiento de los modelos se realiza utilizando los conjuntos de datos preprocesados y vectorizados (entrenamiento y validación).

Se usaron callbacks para controlar el proceso de entrenamiento (EarlyStopping mediante “val\_accuracy” y ReduceLROnPlateau mediante “val\_loss”)

## 2.2.5 Validación del Modelo y Visualización

Una vez finalizado el entrenamiento, se validaron rigurosamente.

**Evaluación en el Conjunto de Prueba:** El rendimiento final del modelo se mide en el conjunto de prueba, un subconjunto de datos no visto durante el entrenamiento o la validación

**Métricas de Evaluación.** Se calculan métricas clave como:

- **Precisión (Accuracy):** Proporción de predicciones correctas.
- **Pérdida (Loss):** Valor de la función de pérdida en el conjunto de prueba.
- **Precisión (Precision), Recall y F1-Score:** Métricas por clase y promediadas (macro, micro, weighted) que proporcionan una visión más detallada del rendimiento, especialmente útil en conjuntos de datos desequilibrados.
- **Matriz de Confusión:** Visualización que muestra el número de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos para cada clase, revelando dónde el modelo comete errores.

**Gráficas de la evolución diferentes métricas durante el proceso de entrenamiento.**

## 2.3. Workflow

En este apartado se describe detalladamente el flujo de trabajo que se ha seguido desde el momento de idealización hasta la presentación del proyecto final. En la siguiente ilustración se puede observar cómo se encadenan los diferentes procesos, incluyendo la búsqueda de datasets, el tratamiento de dichos datasets y la creación de los modelos.

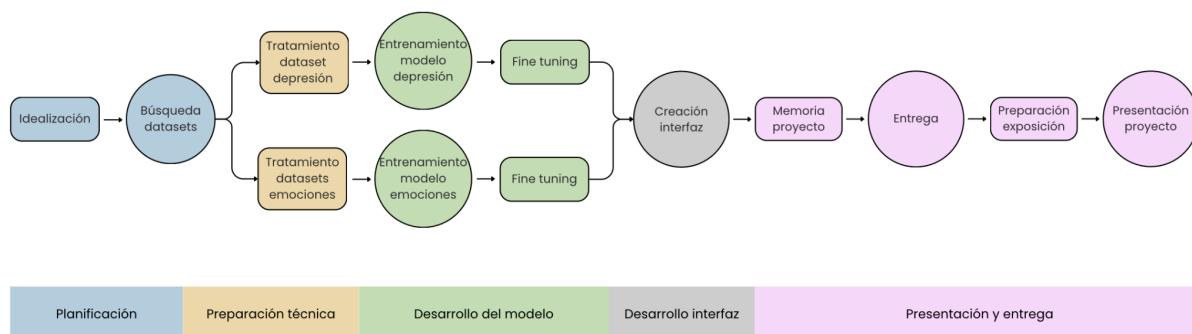


Ilustración 5: Workflow

## 2.4. System Diagram

En la siguiente ilustración se va a proporcionar una representación visual del sistema de inteligencia artificial desarrollado, mostrando de manera esquemática cómo fluye el proceso de trabajo.

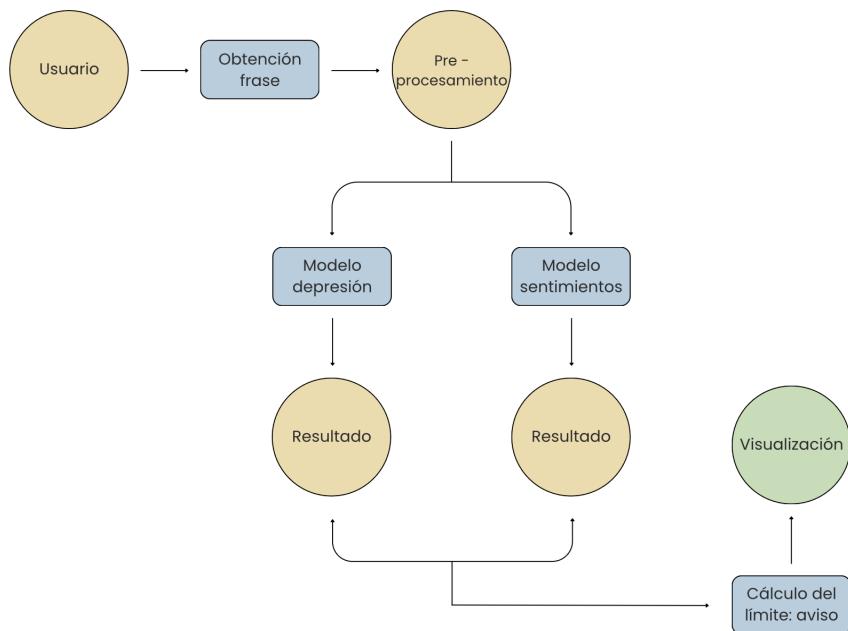


Ilustración 6: System Diagram

## 3. Results

### 3.1. Exploratory Data Analysis (EDA)

El análisis exploratorio de datos fue una etapa clave para comprender la estructura y las características de los datasets obtenidos antes de realizar el entrenamiento de los modelos. Se realizaron análisis por separado de los dos datasets utilizados en el proyecto: el de detección de depresión y el de análisis de sentimientos.

#### 3.1.1 Dataset depresión

Empezando por el dataset de depresión, de las ocho columnas que tenía inicialmente el dataset nos centramos únicamente en tres de ellas, 'title', 'body' y 'label'. Pero previamente se realizó un estudio para conocer cómo estaba estructurado el dataset, valores nulos o cualquier error que hubiera.

De las 2470778 filas, 461051 tenían nulos en la columna 'body', por lo que procedimos a su eliminación dado que el dataset era lo suficientemente extenso como para entrenar al modelo sin esas filas.

# Eliminar filas donde la columna 'body' es nula



```
df_cleaned = df.dropna(subset=['body']).copy()
```

La columna 'label', una de las más importantes, seguía conteniendo 84 nulos pero en este caso como conocíamos la clasificación realizada en dicha columna se procedió a llenar nulos. En este caso, para las filas en las que en 'subreddit' tuvieran un resultado de: 'teenagers', 'happy' o 'DeepThoughts' se asignó un 0 a la columna label. Mientras que las filas en las que en 'subreddit' tuvieran un resultado de 'depression' o 'SuicideWatch' se asignó un 1 a la columna label.

```
# Identificar las filas donde 'label' es nulo y 'subreddit' es 'depression' o 'SuicideWatch'  
condition = df_cleaned['label'].isnull() & df_cleaned['subreddit'].isin(['depression',  
'SuicideWatch'])
```

```
# Rellenar los valores nulos de 'label' con 1 para esas filas  
df_cleaned.loc[condition, 'label'] = 1
```

```
# Identificar las filas donde 'label' es nulo y 'subreddit' está en la lista especificada  
condition_zero = df_cleaned['label'].isnull() & df_cleaned['subreddit'].isin(['teenagers',  
'happy', 'DeepThoughts'])
```

```
# Rellenar los valores nulos de 'label' con 0 para esas filas  
df_cleaned.loc[condition_zero, 'label'] = 0
```

Dado que seguían habiendo 42 nulos en la variable 'label' se revisó la columna 'subreddit' ya que tenía valores numéricos no siendo estos posibles en dicha columna. Dicho estudio resultó en un error en la introducción de los datos, ya que los valores de las columnas se encontraban mezclados. Para evitar problemas con esos datos se procedió a eliminar dichas filas.

```
# Eliminar las filas donde la columna 'subreddit' tenga los valores 4, 5, 6, 7, 8  
df_cleaned = df_cleaned[~df_cleaned['subreddit'].isin(['4', '5', '6', '7', '8'])].copy()
```

```
# Eliminar las filas donde la columna 'label' es nula  
df_cleaned.dropna(subset=['label'], inplace=True)
```

Otra operación realizada fue la de unificar las variables 'title' y 'body' ya que algunos valores de body eran nulos mientras que en title sí contenían información. Por lo que al unificar las columnas obtuvimos una variable 'text' que contenía toda la información de texto del dataset.

```
# Concatenar 'title' y 'body' en una nueva columna 'text'  
# Usamosfillna("") para asegurar que los valores nulos no causen errores y añadimos un  
espacio entre título y cuerpo  
df_cleaned['text'] = df_cleaned['title'].fillna("") + ' ' + df_cleaned['body'].fillna("")
```

Seguidamente se realizó un estudio de la distribución de la columna label, dando como resultado un desbalance de 77% de los datos asignados como no depresión y un 23% de los datos asignados como depresión.

```
df_cleaned['label'].value_counts(normalize=True)
```

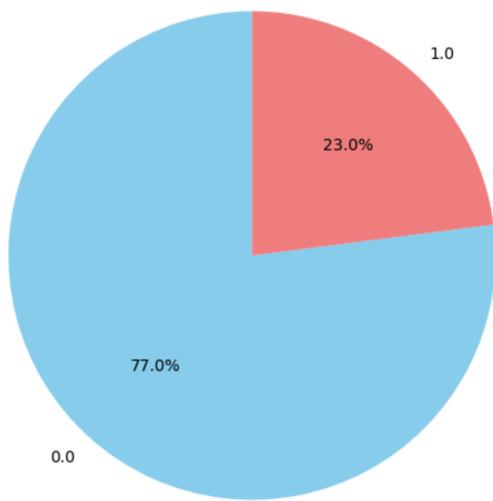


Ilustración 7: Distribución de la columna Label

Esta distribución es debida al desequilibrio de clases en la variable 'subreddit', donde 'teenagers' es de donde más información se ha obtenido, dando como resultado más información para datos no depresivos.

```
df_cleaned['subreddit'].value_counts(normalize=True)
```

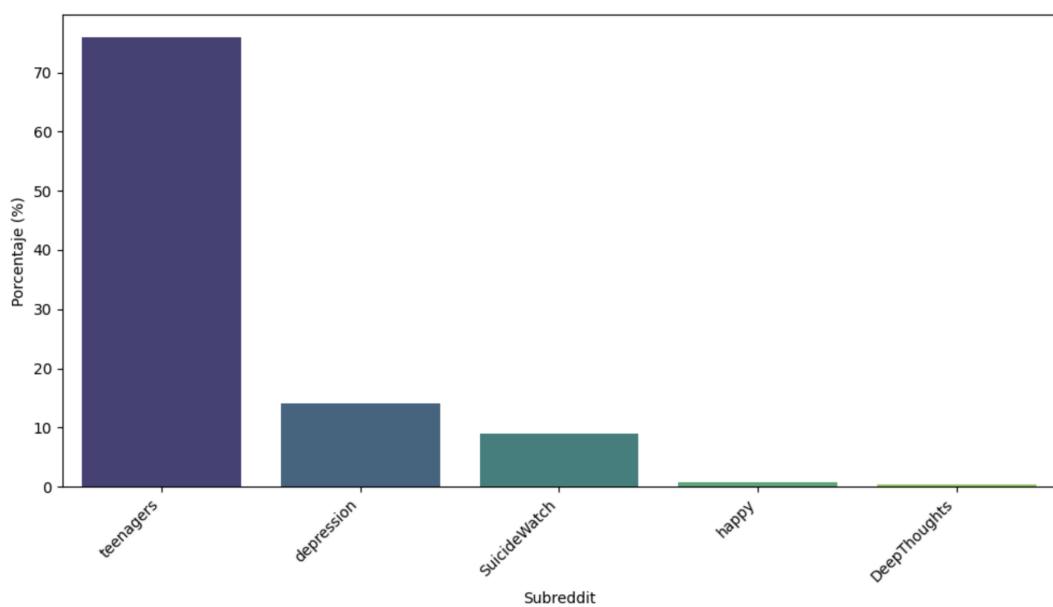


Ilustración 8: Distribución porcentual de la columna Subreddit



Además se generó una ilustración para mostrar las diez palabras más representativas de los textos definidos como depresivos.

```
depressive_texts_df = df[df['label'] == 1]

all_depressive_words = ''.join(depressive_texts_df['processed_text'].tolist())

all_depressive_words = re.sub(r'\s+', ' ', all_depressive_words).strip()

words = all_depressive_words.split()

word_counts = Counter(words)

n_common_words = 20
most_common_words =
word_counts.most_common(n_common_words)

words_list = [word for word, count in most_common_words]

counts_list = [count for word, count in most_common_words]
```

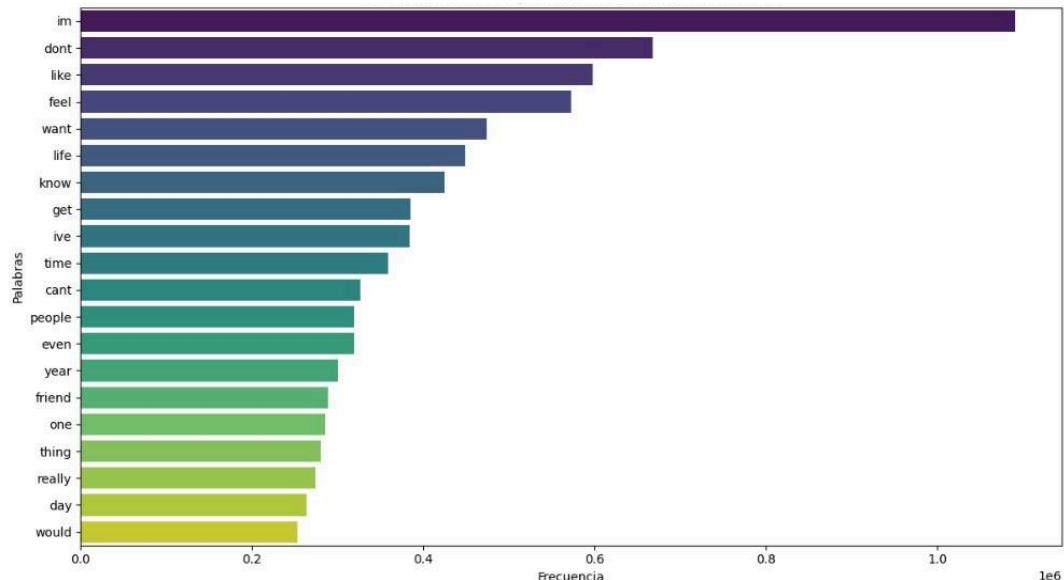


Ilustración 9: Frecuencia de las 20 palabras más comunes de indicios de depresión

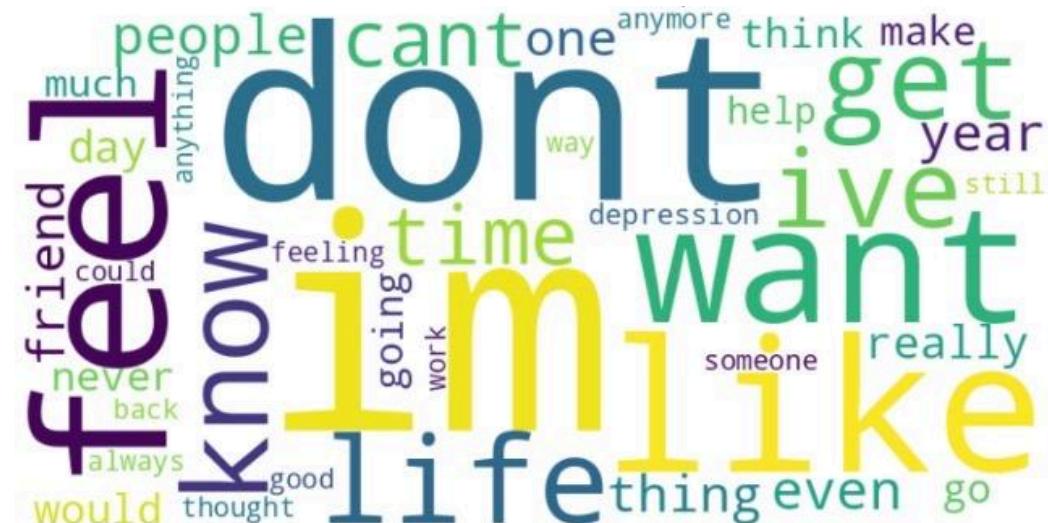


Ilustración 10: Nube de palabras de textos de depresión

### **3.1.2 Datasets emociones**

En cuanto al dataset del análisis de sentimientos, lo primero fue unificar los tres datasets de Github en los que, tras eliminar la columna rater\_id, se encontraron 56737 filas duplicadas que fueron eliminadas dejando 154488 filas en el dataset.

```
# Concatenar los datasets de goemotions
goemotions_df = pd.concat([goemotions_1_df, goemotions_2_df, goemotions_3_df],
                           ignore_index=True)

# Eliminar la columna 'rater_id'
goemotions_df = goemotions_df.drop('rater_id', axis=1)

# Comprobar filas duplicadas
duplicates = goemotions_df.duplicated().sum()

# Comprobación de que los dos dataframes tienen las mismas columnas
goemotions_cols = goemotions_df.columns.tolist()
emotions2_cols = emotions2_df.columns.tolist()

common_cols = list(set(goemotions_cols) & set(emotions2_cols))

goemotions_cols = goemotions_df.columns.tolist()
emotions2_cols = emotions2_df.columns.tolist()

missing_cols_in_emotions2 = list(set(goemotions_cols) - set(emotions2_cols))

for col in missing_cols_in_emotions2:
    emotions2_df[col] = 0
```



## # Concatenación de los datasets

```
emotions2_df = emotions2_df[goemotions_df.columns]

unified_df = pd.concat([goemotions_df, emotions2_df], ignore_index=True)
```

Tras esta comprobación se unificaron el dataset de Github y el dataset de Hugging Face, rellenando con 0 las columnas que no contenía el dataset de Hugging Face. Posteriormente se redujo el número de variables para obtener únicamente los valores de 'text' y los 28 sentimientos. De esta manera la estructura del dataframe quedó con 548310 filas y 29 columnas.

## # Rellenar las columnas sin información con 0

```
unified_df = unified_df.fillna(0)
```

## # Crear un nuevo dataset solo con el texto y las emociones

```
text_col = unified_df['text']
emotion_cols_df = unified_df.loc[:, 'admiration':]
df = pd.concat([text_col, emotion_cols_df], axis=1)
```

Además se realizó un estudio de la cantidad de emociones por fila. En un principio trabajamos con más de una emoción por fila, pero se decidió trabajar solo con una emoción por fila. Por lo que en el estudio de cuantas emociones había por fila se encontraron 2733 filas sin ninguna emoción y, a pesar de que la gran mayoría tenían una sola emoción, se hallaron filas con hasta 12 emociones.

## # Identificar las columnas de emociones (sin texto)

```
emotion_cols = df.columns.drop('text').tolist()
```

## # Calcular la suma de las emociones por fila

```
df['emotion_count_per_row'] = df[emotion_cols].sum(axis=1)
```

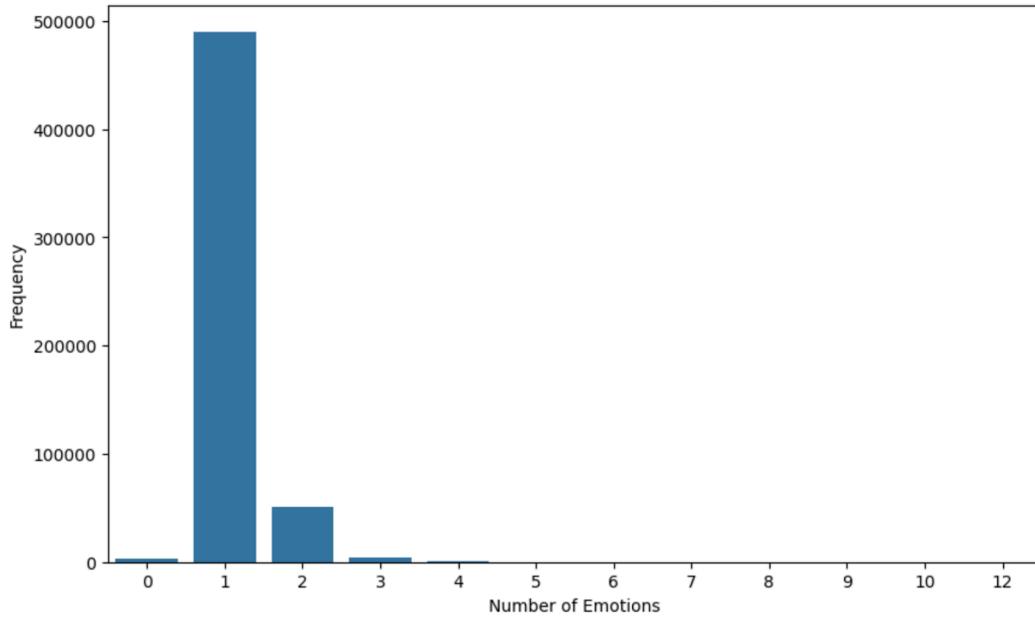


Ilustración 11: Distribución del número de emociones por fila

El primer paso para dejar una emoción por fila fue eliminar las filas sin emociones, ya que no nos aportan valor. El segundo paso fue expandir las filas con más de una emoción de tal manera que tengan una emoción por fila (duplicando las filas en la variable ‘text’ para obtener cada fila con una emoción). De modo que el dataframe pasó a tener 606555 filas.

```
# Mantener filas donde la suma de las columnas de emoción sea mayor que 0
df_filtered = df[df['emotion_count_per_row'] > 0].copy()

# Elimina la columna de recuento de emociones temporales
df_filtered = df_filtered.drop('emotion_count_per_row', axis=1)

# Verifique que ninguna fila tenga un recuento de emociones de 0 en el marco de datos filtrado
emotion_cols_filtered = df_filtered.columns.drop('text').tolist()
df_filtered['emotion_count_per_row_filtered'] =
df_filtered[emotion_cols_filtered].sum(axis=1)

# Elimina la fila temporal
df_filtered = df_filtered.drop('emotion_count_per_row_filtered', axis=1)

# Duplicadas las filas con más de una emoción
import pandas as pd
# Identifica las filas con más de una emoción
rows_with_multiple_emotions = df[(df.drop('text', axis=1).sum(axis=1)) > 1]

# Crea una lista para almacenar las filas duplicadas
duplicated_rows_list = []
```



```
# Itera de las filas identificadas
for index, row in rows_with_multiple_emotions.iterrows():
    text = row['text']
    # Obtén las columnas donde la emoción es valor 1
    active_emotions = row[emotion_cols][row[emotion_cols] == 1].index.tolist()

    # Para cada emoción activa, crea una nueva fila
    for emotion in active_emotions:
        new_row = {'text': text}
        # Inicia todas las columnas con emociones a 0
        for emo_col in emotion_cols:
            new_row[emo_col] = 0
        # Establece la emoción activa actual en 1
        new_row[emotion] = 1
        duplicated_rows_list.append(new_row)

# Crea un nuevo DataFrame a partir de la lista de filas duplicadas
duplicated_df = pd.DataFrame(duplicated_rows_list)

# Separa filas con exactamente una emoción del df original
rows_with_single_emotion = df[(df.drop('text', axis=1).sum(axis=1)) == 1].copy()

# Concatena las filas de una sola emoción y las filas duplicadas recién creadas
df_expanded = pd.concat([rows_with_single_emotion, duplicated_df], ignore_index=True)

# Asegúrese de que las columnas estén en el mismo orden que el df original (excluyendo
# la columna de suma temporal si aún existe)
original_cols_order = [col for col in df.columns if col != 'emotion_count_per_row']
df_expanded = df_expanded[original_cols_order]

# Verifica la suma de emociones por fila en el nuevo marco de datos
emotion_cols_expanded = df_expanded.columns.drop('text').tolist()
df_expanded['emotion_count_per_row_expanded'] =
df_expanded[emotion_cols_expanded].sum(axis=1)

# Elimina la columna temporal
df_expanded = df_expanded.drop('emotion_count_per_row_expanded', axis=1)
```

1	490166
2	50759
3	4062
4	400
5	106
6	53
7	20
8	6
9	3
10	1
12	1

Ilustración 12: Distribución de emociones por fila

Finalmente, se realizó un estudio de la distribución de las emociones en el dataframe completo. En este caso se realizó en percentil, obteniendo como resultado que 7 emociones eran las más relevantes.

```
emotion_counts = df.loc[:, 'admiration':].sum()  
  
total_emotion_sum = emotion_counts.sum()  
emotion_percentages = (emotion_counts / total_emotion_sum) * 100
```

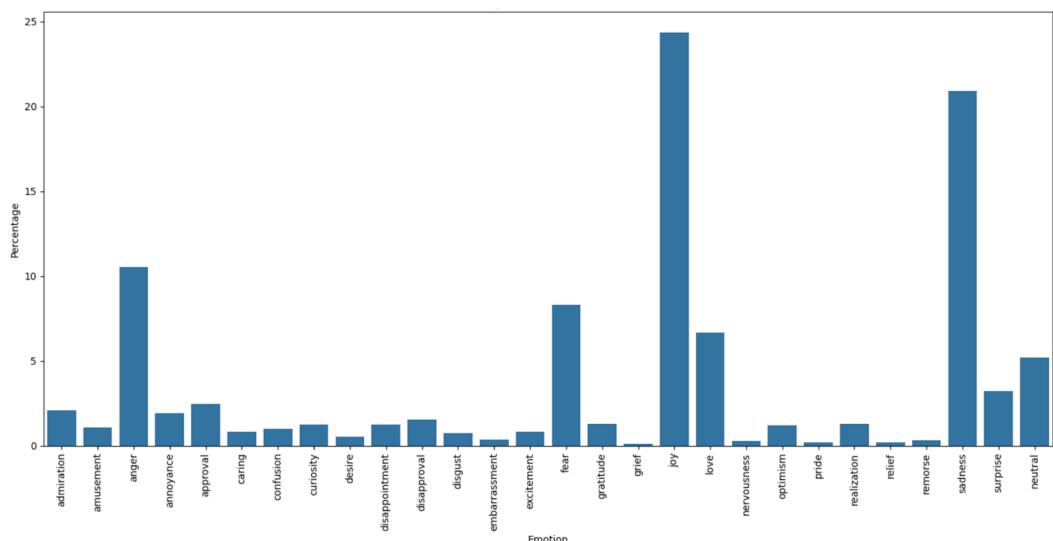


Ilustración 13: Distribución de las 28 emociones

Por esta razón, se procedió a realizar dos reducciones en la dimensión del dataset. Por una parte se redujo a 10 emociones y por otra parte a 6 emociones. Estos tres dataset se

utilizaron para entrenar los modelos y saber cuál era el que mejor resultado obtenía. El de 28 emociones, debido a su desbalance de clases fue descartado desde un principio. Y la decisión final fue la de utilizar el dataset con 6 emociones porque era el que mejores resultados obtenía con el modelo.

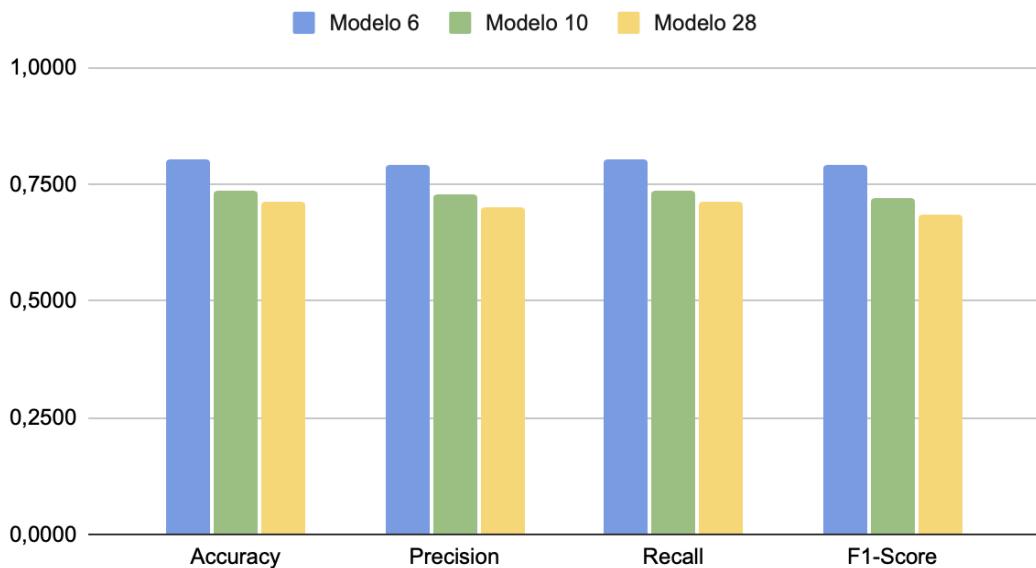


Ilustración 14: Métricas obtenidas al entrenar el modelo con los datasets de 6, 10 y 28 sentimientos

En el caso del dataset con 10 emociones, se llevó a cabo un agrupamiento de emociones de manera que se obtuvieron las siguientes nuevas emociones:

- Alegría: joy, amusement, excitement, optimism
- Aprecio: love, caring, admiration
- Tristeza: sadness, grief, remorse
- Ira: anger, annoyance, disapproval
- Ansiedad: fear, nervousness, relief
- Sorpresa: surprise, realization, confusion
- Reconocimiento: approval, gratitude, pride
- Malestar: embarrassment, disgust, disappointment
- Interés: desire, curiosity
- Neutro: neutral

```
# Define el mapeo de emociones originales a nuevas categorías
emotion_mapping = {
    'alegria': ['joy', 'amusement', 'excitement', 'optimism'],
    'aprecio': ['love', 'caring', 'admiration'],
    'tristeza': ['sadness', 'grief', 'remorse'],
    'ira': ['anger', 'annoyance', 'disapproval'],
    'ansiedad': ['fear', 'nervousness', 'relief'],
    'sorpresa': ['surprise', 'realization', 'confusion'],
```

```
'reconocimiento': ['approval', 'gratitude', 'pride'],
'malestar': ['embarrassment', 'disgust', 'disappointment'],
'interes': ['desire', 'curiosity'],
'neutro': ['neutral']
}
# Crea nuevas columnas para las emociones unificadas
for new_emotion, old_emotions in emotion_mapping.items():
    df[new_emotion] = df[old_emotions].sum(axis=1)
```

De esta manera el nuevo dataframe tenía 11 columnas y la distribución de emociones (en porcentaje) fue la siguiente:

```
# Crear una lista de los nuevos nombres de columnas de emociones
new_emotion_columns = list(emotion_mapping.keys())

# Cree el nuevo DataFrame con 'preprocessed_text' y las nuevas columnas de emoción
df_diez = df[['preprocessed_text']] + new_emotion_columns

# Calcular la suma de cada columna de emoción en df_diez
emotion_counts = df_diez[new_emotion_columns].sum().sort_values(ascending=False)

# Calcular el número total de instancias (filas)
total_instances = len(df_diez)

# Calcula el porcentaje de cada emoción
emotion_percentages = (emotion_counts / total_instances) * 100
```

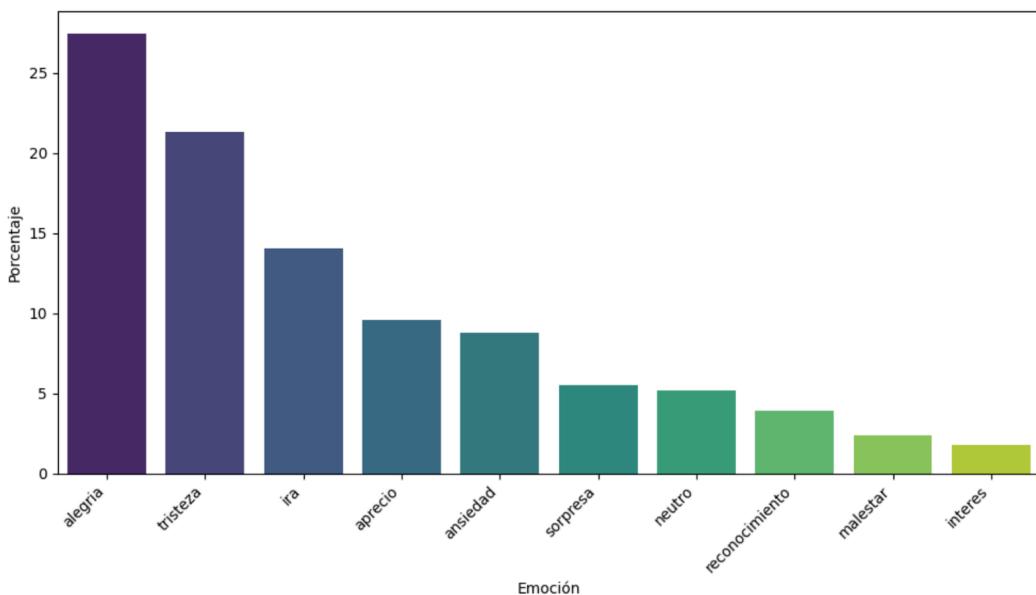


Ilustración 15: Distribución de las 10 emociones



Por otro lado, el dataset con 6 emociones se unificó de la siguiente manera:

- Alegría: joy, amusement, gratitude, love, excitement, pride, optimism, admiration, caring, approval
- Tristeza: sadness, grief, disappointment, remorse, embarrassment, realization
- Ansiedad: fear, nervousness, confusion
- Ira: anger, annoyance, disgust, disapproval
- Sorpresa: surprise, curiosity, desire
- Neutro: neutral, relief

```
# Definir el mapeo de las emociones originales a las nuevas 6 categorías
emotion_mapping_6 = {
    'alegria': ['joy', 'amusement', 'gratitude', 'love', 'excitement', 'pride', 'optimism',
    'admiration', 'caring', 'approval'],
    'tristeza': ['sadness', 'grief', 'disappointment', 'remorse', 'embarrassment', 'realization'],
    'ansiedad': ['fear', 'nervousness', 'confusion'],
    'ira': ['anger', 'annoyance', 'disgust', 'disapproval'],
    'sorpresa': ['surprise', 'curiosity', 'desire'],
    'neutro': ['neutral', 'relief']
}

# Crear nuevas columnas para las emociones unificadas en el df original
for new_emotion, old_emotions in emotion_mapping_6.items():
    # Comprueba si todas las old_emotions existen en las columnas de DataFrame
    existing_columns = [col for col in old_emotions if col in df.columns]
    if existing_columns:
        df[new_emotion] = df[existing_columns].sum(axis=1)
    else:
        df[new_emotion] = 0

# Crea una lista con los 6 nuevos nombres de columnas de emociones
new_emotion_columns_6 = list(emotion_mapping_6.keys())

# Cree el nuevo DataFrame con 'preprocessed_text' y las nuevas 6 columnas de
# emociones
df_seis = df[['preprocessed_text']] + new_emotion_columns_6
```

Tras la unificación en 6 emociones, el nuevo dataframe contenía 7 columnas.

En cuanto a la distribución de las nuevas emociones (en porcentaje), se obtuvo el siguiente gráfico:

```
# Calcular la suma de cada columna de emoción en df_seis
emotion_counts_seis =
df_seis[new_emotion_columns_6].sum().sort_values(ascending=False)
```

# Calcular el número total de instancias (filas)

```
total_instances_seis = len(df_seis)
```

# Calcula el porcentaje de cada emoción para las 6 categorías

```
emotion_percentages_seis = (emotion_counts_seis / total_instances_seis) * 100
```

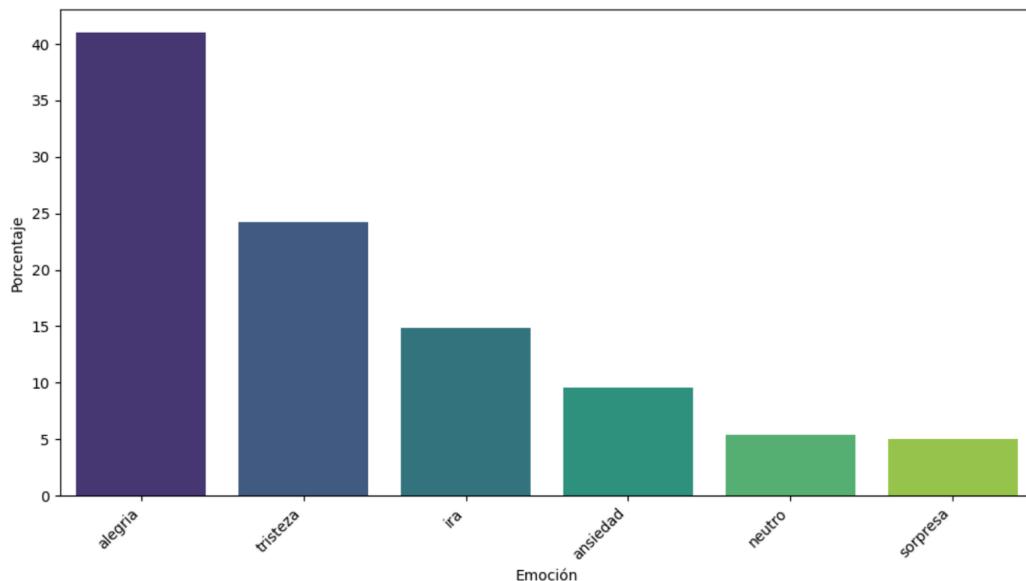


Ilustración 16: Distribución de las 6 emociones

Una vez finalizado el análisis exploratorio y la limpieza del dataset, se procedió al preprocesamiento de los textos, explicado en el siguiente apartado.

### 3.2. Data Preprocessing

Antes de entrenar los modelos, era necesario aplicar un preprocesamiento a los textos con el fin de asegurar la calidad y coherencia de los datos. Dado que se trabajó con lenguaje natural (texto libre), los pasos de limpieza fueron esenciales en ambos datasets. Además, se siguieron pasos similares en ambos conjuntos de datos para que a la hora de unificar los modelos no hubiera problema.

```
import re

def clean_text_regex(text):
    if isinstance(text, str):
        # Quitar los urls
        text = re.sub(r'http\S+|https\S+|www\S+', "", text, flags=re.MULTILINE)
        # Quitar las menciones
        text = re.sub(r'@\w+', "", text)
        # Quitar los hashtags
```



```
text = re.sub(r'#\w+', " ", text)
# Quitar signos especiales y puntuación
text = re.sub(r'^\w\s', " ", text)
return text
else:
    return ""

df['cleaned_text'] = df['text'].apply(clean_text_regex)
```

En la parte de detección de depresión el primer paso fue realizar una limpieza del texto con regex, eliminando URLs, menciones, hashtags, puntuación, etc.

```
import nltk
nltk.download('punkt', quiet=True)

from nltk.tokenize import word_tokenize

def tokenize_text(text):
    if isinstance(text, str):
        return word_tokenize(text)
    else:
        return []

# Aplica la tokenización a la columna 'cleaned_text'
df['tokens'] = df['cleaned_text'].apply(tokenize_text)
```

El segundo paso fue la tokenización en la que se dividió el texto limpio en unidades más pequeñas.

```
import nltk
nltk.download('stopwords', quiet=True) # Descarga la lista de stopwords

from nltk.corpus import stopwords

# Obtén la lista de stopwords en inglés
stop_words = set(stopwords.words('english'))

def remove_stopwords(tokens):
    return [word for word in tokens if word.lower() not in stop_words]

# Aplica la función remove_stopwords a la columna 'tokens'
df['tokens_no_stopwords'] = df['tokens'].apply(remove_stopwords)
```

El tercer paso fue la eliminación de stopwords, palabras que no aportan mucho significado. En este caso fueron palabras en inglés ya que los textos estaban escritos en inglés.



```
import nltk
nltk.download('wordnet', quiet=True) # Descarga WordNet lexicon
nltk.download('omw-1.4', quiet=True) # Descarga Open Multilingual WordNet

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def lemmatize_tokens(tokens):
    return [lemmatizer.lemmatize(word) for word in tokens]

# Aplica la función de lematización a la columna 'tokens_no_stopwords'
df['lemmatized_text'] = df['tokens_no_stopwords'].apply(lemmatize_tokens)

def join_tokens(tokens):
    return " ".join(tokens)

# Aplica la función join_tokens a la columna 'lemmatized_text'
df['processed_text'] = df['lemmatized_text'].apply(join_tokens)
```

Por último, se realizó una lematización, es decir una reducción de las palabras a su forma base para agrupar términos relacionados.

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('punkt_tab')

import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Obtén las stop words en inglés
stop_words = set(stopwords.words('english'))

# Inicializa el WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

# Función para simplificar el part-of-speech tag para lematización
def get_wordnet_pos(word):
    tag = nltk.pos_tag([word])[0][1].upper()
```



```
tag_dict = {"J": wordnet.ADJ,
            "N": wordnet.NOUN,
            "V": wordnet.VERB,
            "R": wordnet.ADV}
return tag_dict.get(tag, wordnet.NOUN)

def preprocess_text(text):
    # Tokenización
    tokens = word_tokenize(text)
    # Pasar a minúsculas
    tokens = [word.lower() for word in tokens]
    # Elimina los signos de puntuación y los caracteres especiales
    tokens = [re.sub(r'[^a-zA-Z0-9]', " ", word) for word in tokens]
    # Elimina los strings vacíos
    tokens = [word for word in tokens if word]
    # Elimina stop words
    tokens = [word for word in tokens if word not in stop_words]
    # Lematización
    tokens = [lemmatizer.lemmatize(word, get_wordnet_pos(word)) for word in tokens]
return " ".join(tokens)

# Aplica el procesado a la columna 'text'
df_expanded['preprocessed_text'] = df_expanded['text'].apply(preprocess_text)
```

Por otra parte, en el análisis de sentimientos, se realizó todo en conjunto para ahorrar tiempo. Se crearon dos funciones, una general que procesaba el texto: tokenización, minúsculas, eliminación de puntuación o caracteres especiales, eliminación de stopwords y lematización; y otra función que se incluía en la anterior ayudando en la lematización. Esta última función definía si las palabras encontradas se trataban de adjetivos, sustantivos, verbos o adverbios para que fuera más fácil encontrar la raíz de las palabras.

### 3.3. Modeling

#### 3.3.1 Modelo Depresión

Una vez limpiado el dataset pasamos a realizar el modelo del dataset de depresión el cual indica si un usuario está depresivo a través de la columna label donde 0 es no depresión y 1 es depresión, además tenemos la columna processed\_Text que es donde almacenamos el texto finalmente el dataset se nos queda en 2 columnas para entrenar el modelo.

```
df.info()
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   label        106165 non-null  float64 
 1   processed_text 106165 non-null  object 
```



```
dtypes: float64(1), object(1)
memory usage: 1.6+ MB
```

```
#Cambiamos el type de label de float a int
df['label'] = df['label'].astype(np.int64)
df.info()
# Column      Dtype
---  -----
0   label      int64
1   processed_text  object
dtypes: int64(1), object(1)
memory usage: 28.7+ MB
```

Ahora procedemos a vectorizar los textos a través del TfidfVectorizer aquí tenemos los pasos:

```
# 1. Inicializa el vectorizador
tfidf = TfidfVectorizer(max_features=500)

# 2. Limpia los datos de texto
# Rellena los valores null con una cadena vacía
df['processed_text'] = df['processed_text'].fillna("")

# 3. Aplica el vectorizador y crea la matriz numérica
X = tfidf.fit_transform(df['processed_text']).toarray()

# 4. Finalmente asignamos a la y el label
y = df['label']
```

Dividimos los datos en entrenamiento y test:

```
# Datos entrenamiento y test
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.15, stratify=y, random_state=42)

# Datos Validación
X_train_final, X_val, y_train_final, y_val = train_test_split(X_train, y_train, stratify=y_train,
test_size=0.15, random_state=42)
```

Ahora desarrollamos el modelo:

```
model = Sequential([
    Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
```



```
Dense(128, activation='relu'),  
Dropout(0.3),  
Dense(1, activation='sigmoid')  
])  
early_stop = EarlyStopping(  
    monitor='val_loss',  
    patience=5,  
    restore_best_weights=True  
)  
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  
  
history = model.fit(X_train_final, y_train_final, epochs=10, batch_size=32,  
validation_data=(X_val, y_val), callbacks = [early_stop])
```

El cual nos arroja el siguiente resultado:

```
Epoch 1/10  
45359/45359 ━━━━━━━━━━━━━━━━ 44s 968us/step - accuracy:  
0.9189 - loss: 0.2089 - val_accuracy: 0.9288 - val_loss: 0.1845  
Epoch 2/10  
45359/45359 ━━━━━━━━━━━━━━━━ 43s 946us/step - accuracy:  
0.9300 - loss: 0.1836 - val_accuracy: 0.9300 - val_loss: 0.1826  
Epoch 3/10  
45359/45359 ━━━━━━━━━━━━━━━━ 45s 1ms/step - accuracy:  
0.9323 - loss: 0.1788 - val_accuracy: 0.9306 - val_loss: 0.1803  
Epoch 4/10  
45359/45359 ━━━━━━━━━━━━━━━━ 46s 1ms/step - accuracy:  
0.9344 - loss: 0.1737 - val_accuracy: 0.9304 - val_loss: 0.1836  
Epoch 5/10  
45359/45359 ━━━━━━━━━━━━━━━━ 45s 1ms/step - accuracy:  
0.9360 - loss: 0.1705 - val_accuracy: 0.9305 - val_loss: 0.1825  
Epoch 6/10  
45359/45359 ━━━━━━━━━━━━━━━━ 45s 994us/step - accuracy:  
0.9372 - loss: 0.1677 - val_accuracy: 0.9305 - val_loss: 0.1825  
Epoch 7/10  
45359/45359 ━━━━━━━━━━━━━━━━ 46s 1ms/step - accuracy:  
0.9383 - loss: 0.1660 - val_accuracy: 0.9300 - val_loss: 0.1842  
Epoch 8/10  
45359/45359 ━━━━━━━━━━━━━━━━ 45s 1ms/step - accuracy:  
0.9390 - loss: 0.1639 - val_accuracy: 0.9304 - val_loss: 0.1819
```



Cofinanciado por  
la Unión Europea



Fondos Europeos



Lo evaluamos:

#### # Evaluación del modelo

```
loss, accuracy = model.evaluate(X_test, y_test)
9418/9418 ----- 3s 357us/step - accuracy:
0.9299 - loss: 0.1811
```

Realizamos la predicción:

```
y_pred_probs = model.predict(X_test) # Probabilidades
y_pred = (y_pred_probs > 0.5).astype(int) # Convertir a clases (0 o 1)
```

Sacamos los resultados de precisión, recall y F1-score, utilizamos binary por que es una clasificación binaria:

```
precision = precision_score(y_test, y_pred, average='binary')
recall = recall_score(y_test, y_pred, average='binary')
f1 = f1_score(y_test, y_pred, average='binary')

print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-score: {f1:.2f}")
```

Los resultados son:

Precision: 0.87

Recall: 0.82

F1-score: 0.84

Y finalmente obtenemos la matriz de confusión :

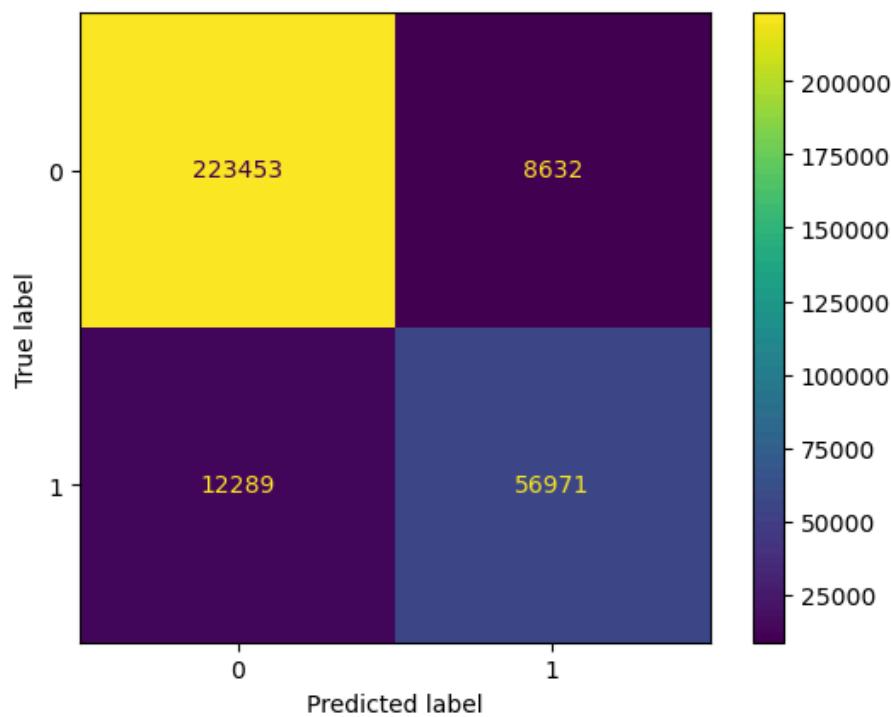


Ilustración 17: Matriz de confusión modelo depresión

### 3.3.2 Modelo Emociones

El modelo de emociones contamos con 28 emociones las tenemos en one hot encoding:



Data columns (total 29 columns):				
#	Column	Non-Null Count	Dtype	
0	admiration	606555	non-null	int64
1	amusement	606555	non-null	int64
2	anger	606555	non-null	int64
3	annoyance	606555	non-null	int64
4	approval	606555	non-null	int64
5	caring	606555	non-null	int64
6	confusion	606555	non-null	int64
7	curiosity	606555	non-null	int64
8	desire	606555	non-null	int64
9	disappointment	606555	non-null	int64
10	disapproval	606555	non-null	int64
11	disgust	606555	non-null	int64
12	embarrassment	606555	non-null	int64
13	excitement	606555	non-null	int64
14	fear	606555	non-null	int64
15	gratitude	606555	non-null	int64
16	grief	606555	non-null	int64
17	joy	606555	non-null	int64
18	love	606555	non-null	int64
19	nervousness	606555	non-null	int64
20	optimism	606555	non-null	int64
21	pride	606555	non-null	int64
22	realization	606555	non-null	int64
23	relief	606555	non-null	int64
24	remorse	606555	non-null	int64
25	sadness	606555	non-null	int64
26	surprise	606555	non-null	int64
27	neutral	606555	non-null	int64
28	preprocessed_text	606254	non-null	object

Ilustración 18: Columnas del dataset de 28 emociones

Medimos la longitud máxima que puede tener un texto dentro del dataset y también obtenemos la media.

```
text_lengths = df['preprocessed_text'].str.len()
max_length = text_lengths.max()
max_length
685.0 #máxima longitud del texto

text_lengths = df['preprocessed_text'].str.len()
median_length = text_lengths.mean().round(2)
median_length
np.float64(53.04) #media longitud del texto
```



Preparamos los datos:

```
# 1. Identificar columnas de texto y etiquetas
text_column = df.columns[28]
sentiment_columns = df.columns[:28]
# Convertir a lista de strings para el Tokenizer
texts = df[text_column].astype(str).tolist()
# Obtener los valores de las etiquetas como un array NumPy
labels = df[sentiment_columns].values
```

Configuramos el tokenizer:

```
# 2. Configuración del Tokenizer
vocab_size = 10000 #tamaño máximo del vocabulario
oov_tok = "<unk>"

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(texts)

# 3. Convertimos textos a secuencias de números
sequences = tokenizer.texts_to_sequences(texts)
```

Ahora vamos con el padding:

```
# 4. Padding
# Necesitamos que todas las secuencias tengan la misma longitud para alimentar la red
# neuronal.
# maxlen: la longitud máxima de las secuencias.
# Es común elegir la longitud del 90-95% de las secuencias o la longitud media +
# 2*desviación estándar.
sequence_lengths = [len(s) for s in sequences]
max_sequence_length = int(np.percentile(sequence_lengths, 95)) # Usamos el percentil
# 95

print(f"\nLongitud máxima de secuencia (percentil 95): {max_sequence_length}")

padded_sequences =
pad_sequences(sequences, maxlen=max_sequence_length, padding='post',
truncating='post')
```

Dividimos el dataset en train/test/validation:

```
test_size = 0.15
random_state = 42
X_train, X_test, y_train, y_test =
```



Cofinanciado por  
la Unión Europea



Ministerio de Industria,  
Energía y Turismo



Escuela de  
organización  
industrial



Fondos Europeos

SAMSUNG

```
train_test_split(padded_sequences, labels, test_size=test_size,  
random_state=random_state)
```

Empezamos con la estructura del modelo hemos escogido una CNN

```
embedding_dim = 100 # Dimensión de los vectores de palabra  
model = Sequential()  
model.add(Embedding(input_dim=vocab_size + 1,  
                     output_dim=embedding_dim,  
                     input_length=max_sequence_length))  
#Capas Convolucionales 1D  
model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))  
model.add(Dropout(0.3))  
  
model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))  
model.add(Dropout(0.3))  
  
model.add(GlobalMaxPooling1D())  
  
model.add(Dense(units=64, activation='relu'))  
model.add(Dropout(0.4))  
  
model.add(Dense(units=28, activation='softmax'))  
  
model.compile(optimizer=Adam(learning_rate=0.001),  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])
```



Cofinanciado por  
la Unión Europea



Fondos Europeos

SAMSUNG

Arquitectura del modelo emotions:

Model: "sequential"		
Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
conv1d (Conv1D)	?	0 (unbuilt)
dropout (Dropout)	?	0
conv1d_1 (Conv1D)	?	0 (unbuilt)
dropout_1 (Dropout)	?	0
global_max_pooling1d (GlobalMaxPooling1D)	?	0
dense (Dense)	?	0 (unbuilt)
dropout_2 (Dropout)	?	0
dense_1 (Dense)	?	0 (unbuilt)

Ilustración 19: Arquitectura del modelo de emociones

Entrenamiento del modelo:

```
epochs = 20
batch_size = 64

early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=5,
                               min_delta=0.0001,
                               mode='max',
                               restore_best_weights=True)

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                             factor=0.2,
                             patience=3,
                             min_lr=0.00001)
history = model.fit(X_train, y_train,
                     epochs=epochs,
                     batch_size=batch_size,
                     validation_data=(X_test, y_test),
                     callbacks=[early_stopping, reduce_lr])
```

Hemos obtenido:

val\_accuracy: 0.7100

val\_loss: 0.9446

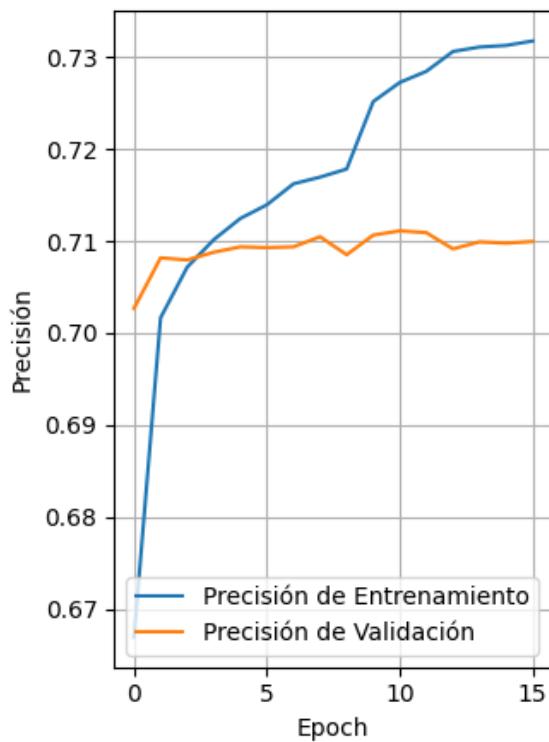


Ilustración 20: Precisión durante el entrenamiento de la CNN

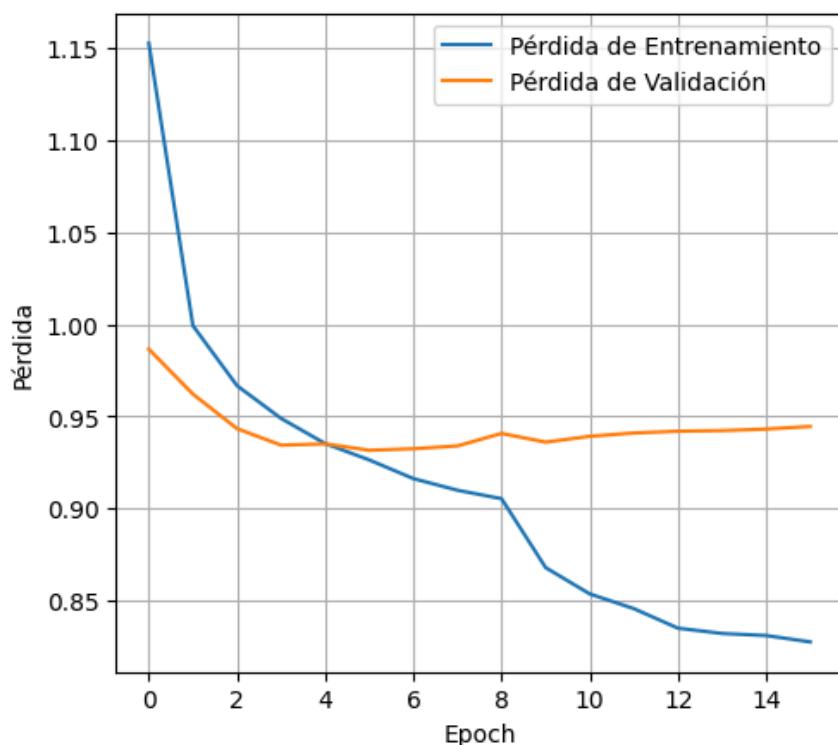


Ilustración 21: Pérdida durante el entrenamiento de la CNN



Cofinanciado por  
la Unión Europea



EOI Escuela de  
organización  
industrial



Fondos Europeos

SAMSUNG

Evaluación del modelo:

```
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
```

Pérdida en el conjunto de prueba: 0.9393

Precisión en el conjunto de prueba: 0.7111

```
# Convertimos los arrays en un vector de posición entre 0 - 27
y_test_classes = np.argmax(y_test, axis=1)
```

Calculamos precision, recall, f1-score:

```
precision, recall, f1_score, support = precision_recall_fscore_support(
    y_test_classes,
    y_pred_classes,
    average='weighted',
    zero_division=0
)
```

Precisión: 0.7015

Recall: 0.7111

F1-Score: 0.6861

Reporte de clasificación por sentimiento, aquí vemos que muchos sentimientos tienen muy poco datos para poder tener unos valores considerables en las métricas de precisión, recall y F1.



	precision	recall	f1-score	support
admiration	0.26	0.43	0.32	2514
amusement	0.33	0.60	0.42	1320
anger	0.84	0.87	0.86	12949
annoyance	0.16	0.10	0.12	2302
approval	0.18	0.00	0.00	2958
caring	0.22	0.00	0.00	1054
confusion	0.33	0.00	0.01	1189
curiosity	0.38	0.01	0.03	1533
desire	0.22	0.02	0.04	649
disappointment	0.06	0.00	0.00	1547
disapproval	0.16	0.00	0.00	1923
disgust	0.19	0.06	0.09	912
embarrassment	0.22	0.00	0.01	422
excitement	0.52	0.01	0.02	1025
fear	0.90	0.77	0.83	10019
gratitude	0.42	0.71	0.53	1483
grief	0.00	0.00	0.00	122
joy	0.89	0.93	0.91	29286
love	0.75	0.66	0.70	8086
nervousness	0.00	0.00	0.00	338
optimism	0.27	0.13	0.17	1483
pride	0.00	0.00	0.00	258
realization	0.00	0.00	0.00	1639
relief	0.00	0.00	0.00	247
remorse	0.24	0.30	0.27	386
sadness	0.92	0.93	0.92	25499
surprise	0.64	0.79	0.71	3863
neutral	0.23	0.67	0.34	6305
accuracy			0.71	121311
macro avg	0.33	0.29	0.26	121311
weighted avg	0.70	0.71	0.69	121311

Ilustración 22: Reporte de clasificación por emoción (28 sentimientos)



Cofinanciado por  
la Unión Europea



EOI Escuela de  
organización  
industrial



Fondos Europeos

SAMSUNG

### 3.4. User Interface

Hemos desarrollado un pequeño programa con django y haciendo una interfaz básica la cual le pasas una frase y este con el modelo que hemos entrenado anteriormente tanto de emociones como de depresión te da una, en esta pequeño demo solo se puede utilizar el idioma el inglés como input:



Ilustración 23: Interface modelo depresión



Cofinanciado por  
la Unión Europea



Fondos Europeos

SAMSUNG

## ❤️ Análisis de Emociones

### 🏆 Emoción dominante: Tristeza (68,98999786376953%)

Distribución de emociones:



Ilustración 24: Interfaz modelo emociones

Hemos desarrollado unas reglas para concatenar los dos modelos para dar una mejor versión del pronóstico sobre si una persona tiene riesgo de depresión las cuales son:

#### Alerta Moderada:

- **Condición:** Tristeza + Ira + Ansiedad > 50%
- **Acción:** Recomendación preventiva

#### Alerta Máxima:

- **Condición:** Emociones negativas > 50% Y Score depresión > 0.3
- **Acción:** Recomendación fuerte de consulta profesional



**Sistema de Detección Combinado**

**¿Cómo funciona nuestro sistema de alerta?**

**💡 Alerta Moderada**

- **Condición:** Tristeza + Ira + Ansiedad > 50%
- **Acción:** Recomendación preventiva

**⚠ Alerta Máxima**

- **Condición:** Emociones negativas > 50% Y Score depresión > 0.3
- **Acción:** Recomendación fuerte de consulta profesional

**💡 Metodología:** Combinamos análisis de emociones específicas con un modelo entrenado para detectar patrones de lenguaje asociados con depresión, proporcionando una evaluación más completa.

Ilustración 25: Funcionamiento del sistema de alerta

### 3.5. Testing and Improvements

#### 3.5.1 Mejora modelo depresión

Conseguimos mejorar el modelo cambiando los parámetros de la vectorización los cuales fueron los siguientes:

```
tfidf = TfidfVectorizer(  
    max_features=1000, # Aumentado de 500 a 1000 características  
    ngram_range=(1, 3), # Incluir unigramas, bigramas y trigramas  
    max_df=0.90, # Ignorar palabras que aparecen en más del 90% de los textos  
    min_df=5, # Ignorar palabras que aparecen en menos de 5 textos  
    binary=False # False = usar frecuencias, True = solo presencia/ausencia  
)  
  
df['processed_text'] = df['processed_text'].fillna("")  
X = tfidf.fit_transform(df['processed_text']).toarray()
```

Nos dimos cuenta una cosa muy importante (la cual afectaba más al modelo) es que no estábamos usando el binay=False el cual cuenta las palabras repetidas (frecuencia), además de añadir bigramas trigramas, poner un max df y un min df los cuales ignoran tanto palabras que se repiten mucho como que aparecen poco.

Una vez vectorizado de nuevo procedimos a probar el modelo y este mejoró dándonos los siguientes valores en cuanto a scores:

Precision: 0.88

Recall: 0.84

F1-score: 0.86

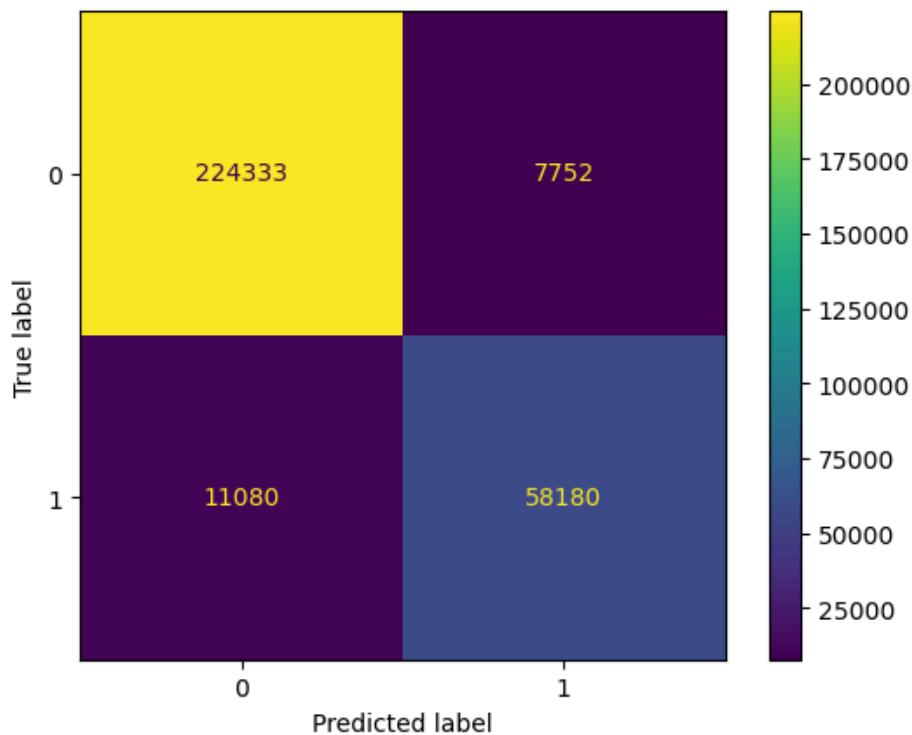


Ilustración 26: Matriz de confusión modelo depresión mejorado

### 3.5.2 Mejora modelo emociones

En el modelo de emociones tenemos un total de 28 emociones las cuales no todas tienen los suficientes datos para poder tener unos buenos scores (precisión, f1, recall), entonces nos decantamos por crear dos modelos reduciendo las emociones primero probamos con uno de 10 emociones y finalmente con uno de 6, he aquí los resultados obtenidos y como lo llevamos a cabo:

Modelo de 10 emociones:

En el modelo de 10 emociones nos quedamos con el top 10 con más datos los cuales eran los siguientes:



Data columns (total 11 columns):			
#	Column	Non-Null Count	Dtype
0	preprocessed_text	604574 non-null	object
1	alegria	604574 non-null	int64
2	aprecio	604574 non-null	int64
3	tristeza	604574 non-null	int64
4	ira	604574 non-null	int64
5	ansiedad	604574 non-null	int64
6	sorpresa	604574 non-null	int64
7	reconocimiento	604574 non-null	int64
8	malestar	604574 non-null	int64
9	interes	604574 non-null	int64
10	neutro	604574 non-null	int64

Ilustración 27: Columnas del dataset de 10 emociones

```
X = df['preprocessed_text']
y = df.iloc[:, 1:]

tfidf = TfidfVectorizer(max_features=750)
X_tfidf = tfidf.fit_transform(X).toarray()
```

```
model = Sequential([
    Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(6, activation='sigmoid')
])
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy', 'AUC']
)
```

```
loss, accuracy, auc = model.evaluate(X_test, y_test)
2834/2834 ----- 8s 3ms/step - AUC: 0.9450 -
accuracy: 0.6827 - loss: 0.1534
```

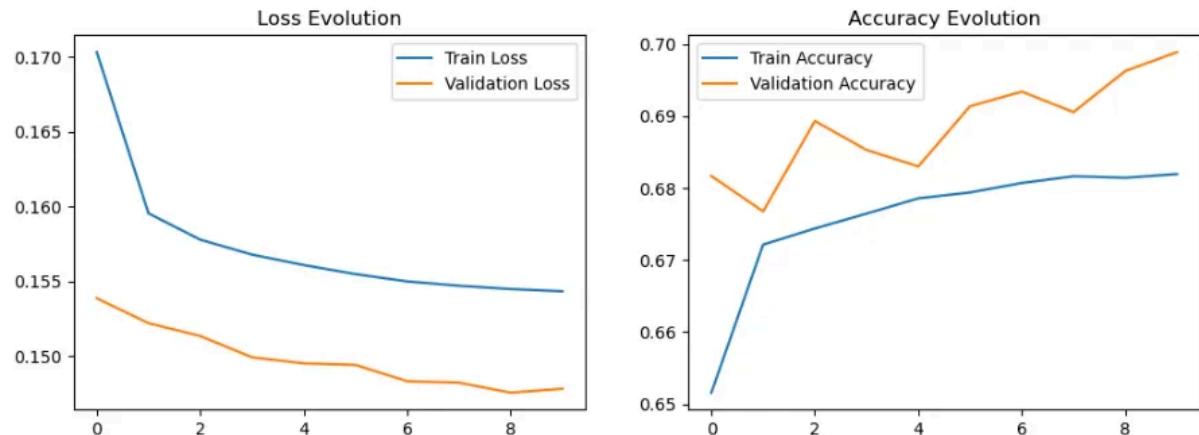


Ilustración 28: Pérdida y accuracy modelo emociones mejorado

En este modelo de 10 emociones los resultados son los siguientes:

Precisión: 0.7268

Recall: 0.7384

F1-Score: 0.7225

El reporte de clasificación esta ocasión es mejor aunque seguimos viendo algunos sentimientos los cuales tienen pocos datos y por lo tanto malos resultados.

	precision	recall	f1-score	support
alegria	<b>0.85</b>	<b>0.84</b>	<b>0.84</b>	24737
aprecio	<b>0.59</b>	<b>0.77</b>	<b>0.67</b>	8791
tristeza	<b>0.91</b>	<b>0.92</b>	<b>0.92</b>	19513
ira	<b>0.65</b>	<b>0.82</b>	<b>0.72</b>	12736
ansiedad	<b>0.84</b>	<b>0.83</b>	<b>0.84</b>	8089
sorpresa	<b>0.65</b>	<b>0.37</b>	<b>0.47</b>	5037
reconocimiento	<b>0.52</b>	<b>0.25</b>	<b>0.34</b>	3554
malestar	<b>0.25</b>	<b>0.00</b>	<b>0.00</b>	2230
interes	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	1534
neutro	<b>0.27</b>	<b>0.40</b>	<b>0.32</b>	4718
accuracy			<b>0.74</b>	90939
macro avg	<b>0.55</b>	<b>0.52</b>	<b>0.51</b>	90939
weighted avg	<b>0.73</b>	<b>0.74</b>	<b>0.72</b>	90939

Ilustración 29: Reporte de clasificación por emoción (10 sentimientos)



En el modelo de 6 emociones unificamos a 6 emociones los datos obtenidos:

Data columns (total 7 columns):				
#	Column	Non-Null Count	Dtype	
0	preprocessed_text	604574	non-null	object
1	alegria	604574	non-null	int64
2	tristeza	604574	non-null	int64
3	ansiedad	604574	non-null	int64
4	ira	604574	non-null	int64
5	sorpresa	604574	non-null	int64
6	neutro	604574	non-null	int64

Ilustración 30: Columnas del dataset de 6 emociones

Hemos llevado el mismo proceso que el modelo de 10 emociones y los resultados son bastante mejores que los anteriores modelos ,por lo tanto nos hemos decantado por este modelo como modelo final:

Precisión: 0.7994

Recall: 0.8024

F1-Score: 0.7944

El reporte de clasificación obtenido fue este el cual todos los sentimientos dan unos resultados buenos y no hay ningún sentimiento que tenga una puntuación mala , salvo neutro el cual tenemos en cuenta pero como creemos que no influye demasiado en la performance del modelo.

	precision	recall	f1-score	support
alegria	0.83	0.93	0.87	37082
tristeza	0.90	0.84	0.87	22185
ansiedad	0.93	0.68	0.78	8909
ira	0.67	0.80	0.73	13425
sorpresa	0.73	0.46	0.56	4450
neutro	0.32	0.20	0.25	4888

Ilustración 31: Reporte de clasificación por emoción



## 4. Projected Impact

### 4.1. Accomplishments and Benefits

El desarrollo de este proyecto ha culminado con la creación de dos sistemas de análisis de texto basado en técnicas avanzadas de Deep Learning, con beneficios tangibles en el campo de la salud mental. Con el objetivo de mejorar la eficiencia laboral de empresas, además conseguimos mejorar la calidad de vida de las personas y su bienestar.

Los principales **Logros** del proyecto son:

- **Detección temprana y precisa de la Depresión.** El modelo MLP de clasificación binaria es capaz de identificar indicadores de depresión en mensajes de RRSS.
- **Análisis de Sentimiento para una mayor comprensión.** El modelo de CNN para clasificación multiclase nos ofrece una visión más completa del estado emocional subyacente en los mensajes.
- **Potencial.** Este proyecto demuestra la capacidad de aplicar un monitoreo no invasivo y a gran escala a través de la IA, para determinar la salud mental a través de mensajes, implementando intervenciones tempranas y personalizadas. Todo ello derivando en una reducción de la prevalencia de bajas laborales.

#### Beneficios para las Grandes Empresas:

Este modelo ofrece a la sección Recursos Humanos (RRHH) una gran herramienta. Al integrar este sistema discretamente en plataformas como Microsoft Teams, los emails y otras plataformas de comunicación de sus trabajadores (siempre bajo un estricto protocolo de privacidad y consentimiento), podría detectar patrones lingüísticos indicativos de que puede desarrollar una depresión.

El enfoque de la propuesta se basa en trabajar de manera preventiva. En lugar de intervenir cuando el empleado ya está de baja, el sistema de RRHH permite identificar señales tempranas, facilitando así una aproximación empática por parte de la empresa e incluso la opción de apoyo psicológico o asesoramiento.

En España el coste medio anual por baja por depresión es de unos 23.000 millones de euros. Una implementación exitosa del proyecto podría aspirar a reducciones de entre 10% y 30%. Si se pudiese aplicar a todas las empresas de España, se traduciría en un potencial ahorro de entre 2.300 millones y 6.900 millones anuales. Además, estudios en salud laboral demuestran que las intervenciones tempranas en salud mental son significativamente más efectivas y menos costosas que las intervenciones tardías.

#### KPI's:

- Reducción del porcentaje de bajas relacionadas con la salud mental en un 10-30%.
- Disminución del coste económico anual por depresión en un 20%.



- Reducción de la duración promedio de las bajas por salud mental de 88,24 días a al menos 70 días.
- Incremento en los índices de productividad del equipo en un 10%.
- Aumento de las tasas de retención de empleados en un 10%.

### Protección de Datos:

Para que la implementación de este modelo se haga con la máxima rigurosidad en cuanto a la protección de datos y la privacidad, sería imprescindibles:

- Consentimiento Explícito
- Anonimización de los datos antes de ser procesados
- Seguimientos a través de canales confidenciales y convencionales
- Transparencia
- Seguridad de Datos

### Beneficios para plataformas de Redes Sociales:

Las plataformas de redes sociales tienen una responsabilidad creciente en la salud mental de sus usuarios. Este modelo ofrece una oportunidad para diferenciarse y construir una comunidad más segura y solidaria. Al integrar el sistema en el análisis de publicaciones o mensajes (siempre con el consentimiento del usuario y bajo políticas de privacidad claras), la plataforma podría detectar proactivamente signos de depresión o angustia en el contenido generado por los usuarios.

El objetivo no sería reemplazar la interacción humana o el diagnóstico profesional, sino actuar como un primer intervintente. Cuando se detecten patrones lingüísticos asociados a la depresión y un sentimiento predominantemente negativo o de desesperanza, el sistema podría generar una alerta que, de forma privada y discreta, sugiera al usuario recursos de apoyo en salud mental, líneas de ayuda o contenido de bienestar, directamente dentro de la aplicación.

Las plataformas que implementen este sistema lo podrían usar para atraer nuevas fuentes de financiación, sobre todo si es una red social a gran escala como Instagram, X, Facebook,...

Estos ingresos podrían ser directos, a través de patrocinios o alianzas estratégicas. Empresas farmacéuticas, grandes aseguradoras de salud o fundaciones de salud mental están dispuestas a intervenir en programas de impacto y visibilidad. Una empresa con millones de usuarios activos podría asegurar acuerdos con 2-3 grandes socios podrían generar entre 10 y 50 millones de euros anuales por esta vía.

Además, si una plataforma logra un incremento perceptible en su reputación y valor de marca, podría experimentar un incremento de ingresos publicitarios de otro tipo de anunciantes en un 0.5% al 2% anual.



### KPI's:

- Número de intervenciones proactivas (ej., mensajes de sugerencia de recursos) realizadas en un 25%.
- Reducción en el volumen de informes de contenido relacionado con la salud mental/autolesiones en un 12%.
- Incremento en el uso de recursos de salud mental ofrecidos por la plataforma en un 10-15%
- Mejora en las métricas de retención de usuarios a largo plazo.
- Mejora en las encuestas de satisfacción del usuario relacionadas con la seguridad y el soporte en al menos 1 punto sobre 10.

### Protección de Datos:

Para que la implementación de este modelo se haga con la máxima rigurosidad en cuanto a la protección de datos y la privacidad, sería imprescindibles:

- Anonimización de los datos antes de ser procesados
- Privacidad del Contenido
- Transparencia y Consentimiento
- Seguridad de Datos
- Minimización de datos
- Identificador de Riesgo, no de Diagnóstico

## 4.2. Future Improvements

En cuanto a mejoras, queremos optimizar el programa desarrollado para que sea capaz de evaluar múltiples frases por usuario de manera simultánea, ya que con una sola frase no es posible determinar de forma precisa el estado emocional o el posible nivel de depresión. Para ello, estamos barajando distintas opciones, como analizar un conjunto de 10 frases por usuario y considerar solo las que reflejan sentimientos negativos, de forma que si el número de frases negativas supera un umbral (por ejemplo, 4 de 10), o si el porcentaje de frases negativas es mayor a un cierto valor (por ejemplo, un 40%), el sistema pueda indicar un posible estado de riesgo. Los cuales iremos testando en el futuro analizando casos reales de personas con depresión para ver qué es lo mas óptimo.

Por otro lado, para mejorar el rendimiento del modelo de detección de depresión. Dado que los datos que detectan la depresión provienen de Reddit y definen que un usuario tiene depresión por haber escrito en un foro con nombre 'Depression' o 'SuicideWatch', es necesario encontrar unos datos avalados por psicólogos que confirmen que esos textos tienen rasgos de depresión. Así el entrenamiento del modelo de depresión se ajustaría más a la realidad y nos permitiría mejorar la precisión y la robustez del modelo. Además, sería recomendable aumentar la cantidad de información con la que se entrena el modelo de emociones para mejorar los resultados obtenidos.

De cara a un futuro en el que este proyecto fuera llevado a empresas o bien en redes sociales, se podría realizar un ajuste para detectar usuarios que realicen comentarios o envíen mensajes que puedan provocar depresión a otros usuarios o incluso el suicidio de



los mismos. De esta manera, ya no solo informar de casos de depresión y enviar la alerta pertinente, si no de detectar personas que puedan estar haciendo daño a los demás y, en caso de las empresas tomar medidas o en el caso de las redes sociales bloquear sus perfiles.

## 5. Team Member Review and Comment

<ATTACH A TEAM PICTURE HERE>

NAME	REVIEW and COMMENT
Ignacio	<p>Este proyecto ha sido un gran reto para nosotros, no solo en lo técnico y lo logístico, sino también como equipo, sobre todo por nuestra falta de experiencia. Pero lo bueno es que, a nivel humano, nos entendimos genial desde el principio. Todos hemos aportado y nos hemos ayudado en todas las situaciones cuando alguien se atascaba con algo, y eso fue clave. Una de las dificultades más grandes fue no tener ordenadores lo bastante potentes para correr los modelos cuando se hacían grandes, o para manejar el dataset de dos millones de datos. Ahí nos tocó tirar de ingenio y buscarle la vuelta para solucionar esos problemas. Además, cada vez que probábamos los modelos, íbamos encontrando algún pequeño fallo o algo que se podía mejorar, gracias al testing lo veíamos rápido y lo arreglábamos sin mucha complicación.</p> <p>En general, fue un proyecto muy interesante. En mi caso, me ha servido para entender cómo se trabaja en un entorno más profesional: lo importante que es llevar un buen schedule y saber repartir las tareas. Y aunque hubo complicaciones, al final conseguimos sacar el trabajo adelante, que es lo que cuenta.</p>
Sandra	Durante este proyecto tuve la oportunidad de profundizar en el procesamiento de lenguaje natural, un área en la que tenía cierta dificultad. Me encargué principalmente de la limpieza de los datos, el EDA y el preprocesamiento de los textos, lo que me ayudó a mejorar mis habilidades en estas áreas. En cuanto a la realización de los modelos estuve siempre pendiente de lo que realizaban los compañeros con la finalidad de aprender y aportar cada uno su punto de vista para mejorar los modelos.



	<p>Uno de los mayores desafíos fue limpiar y transformar los datos de manera eficiente, especialmente porque venían de fuentes distintas y no siempre estaban bien etiquetados.</p> <p>Llevar a cabo un proyecto de esta magnitud en tan poco tiempo ha sido difícil porque también hay que saber parar el proyecto y que no suponga una alta carga de trabajo. Podría haberse mejorado mucho con más tiempo y más recursos, pero el proyecto ha seguido un buen ritmo para llegar a los plazos deseados. Ha sido una grata experiencia trabajar en equipo para realizar un proyecto sobre inteligencia artificial y es, para mí, un punto de inflexión en mi carrera, ayudándome a desarrollar mis capacidades en el análisis de datos.</p>
Javier	<p>El desarrollo de un proyecto de esta envergadura ha sido tanto desafiante como entretenido. La colaboración con compañeros y el continuo feedback y aprendizaje en este campo, me ha servido enormemente para enriquecer y afianzar mis conocimientos, sobre todo en el mundo del Deep Learning. Sin duda, tengo que destacar la suerte que he tenido con el grupo que me ha tocado, mis compañeros son muy trabajadores, las dinámica de trabajo ha sido agradable y la colaboración y el trabajo en equipo fue excelente.</p> <p>En relación al propio proyecto y su desarrollo, es cierto que la búsqueda y obtención de datos fue complicada ya que al ser un proyecto basado en el análisis de texto es muy importante trabajar con datasets de gran tamaño y calidad. Aun así, los datasets escogidos han sido correctos para el entrenamiento del modelo pero es algo que se podría mejorar. Los modelos han dado buenos resultados, aunque el que se basa en la clasificaciones de emociones tiene rango de mejora, principalmente por el volumen del dataset escogido.</p> <p>Aunque el resultado y la precisión es mejorable, hemos obtenido buenos resultados. Alguno de los factores que nos impidió crear sistemas de más calidad fue la capacidad computacional de nuestros ordenadores y no poder dedicarle mucho más tiempo al entrenamiento, ya que no podíamos arriesgarnos a crear modelos que requiriese mucho tiempo de entrenamiento y diesen al final malos resultados.</p> <p>En general, el proyecto lo hemos desarrollado sin mucha dificultad pero con gran trabajo, siento que los tres hemos aprendido mucho y que hemos hecho un trabajo contundente. Personalmente, creo que se deberían seguir desarrollando proyectos en este campo ya que, independientemente del impacto económico de la empresa, lo importante son las personas. En las últimas décadas el aumento de enfermedades ligadas a la salud mental y los suicidios han aumentado de manera notable. Tengo la suerte de que ningún familiar, amigo o conocido lo ha sufrido, pero conozco varias personas licenciadas en psicología que sin duda me han hecho darme</p>



Cofinanciado por  
la Unión Europea



Ministerio de Industria  
y Comercio

EOI

Escuela de  
organización  
Industrial



Fondos Europeos

SAMSUNG

	cuenta del problema y de que la sociedad debe hacer algo por solucionarlo o mitigarlo.
--	--



Cofinanciado por  
la Unión Europea



Ministerio de Industria  
y Turismo

EOI Escuela de  
organización  
Industrial



Fondos Europeos

SAMSUNG

## 6. Instructor Review and Comment

CATEGORY	SCORE	REVIEW and COMMENT
IDEA	__/10	
APPLICATION	__/30	
RESULT	__/30	
PROJECT MANAGEMENT	__/10	
PRESENTATION & REPORT	__/20	
TOTAL	__/100	