Universidade Vigo

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria del Trabajo de Fin de Grao que presenta

D. Erik Pereira Fernández

para la obtención del Título de Graduado en Ingeniería Informática

Etiquetado Inteligente: Predicción Automática de Flairs en Reddit

Julio , 2025

Traballo de Fin de Grao Nº: EI 24/25-11

Tutor: Anália García Lourenço
Co-tutor: Guillermo Blanco González
Área de coñecemento: Linguaxes e Sistemas Informáticos
Departamento: Informática

Dedicatoria

A mi familia por su apoyo durante este camino.
A quién ha estado ahí tanto en los momentos más complicados como en los más felices durante los últimos años.

Agradecimientos

Agradezco en primer lugar a mis tutores por su sabiduría e interés en el proyecto. Sin sus conocimientos y guía no habría sido posible la realización del mismo.

También quiero agradecer a mi familia, amigos y a todos los que han hecho de este camino algo más ameno, y que han estado para mí en todo momento, especialmente durante estos últimos años.

Índice

Índice de figuras

Índice de tablas

1. Introducción

Hoy en día la forma en la que las personas expresan sus emociones, comparten su día a día o incluso buscan apoyo social ha cambiado en parte debido a las plataformas digitales y redes sociales [1]. Una de estas plataformas, Reddit [2] destaca como una especie de espacio híbrido entre red social y foro temático, donde millones de usuarios participan de una forma activa en *discusiones* organizadas en una especie de comunidades específicas llamadas *subreddits*. Uno de estos subredits más importantes en el contexto del bienestar emocional y el que más nos atañe a efectos de este TFG es el de `r/mentalhealth`, que viene a ser como un espacio abierto para aquellas personas que quieren hablar de salud mental, tanto para contar sus experiencias personales, hacer consultas, desahogarse o reflexionar.

Las publicaciones dentro de Reddit pueden ser acompañadas de etiquetas conocidas como *flairs*, que permiten clasificar el contenido en función de su naturaleza. Estas etiquetas son opcionales pero cumplen un rol muy importante en la organización del contenido, ya que permiten por ejemplo diferenciar las publicaciones que buscan apoyo moral (*Need Support*) de aquellas que expresan mejoras (*Good News*) o simplemente de opiniones (*Thoughts / Opinions*). Hoy en día este proceso de etiquetado es manual, subjetivo y heterogéneo, lo que genera una alta de publicaciones sin flair o con un flair incorrecto.

Todo el crecimiento de estos espacios digitales permite surgir a nuevas formas de análisis social, psicológico y lingüístico. La gran cantidad de volumen de datos al que se puede acceder va a abrir nuevas forma de investigación en lo relativo a la salud digital. Surge de aquí la pregunta de: ¿Es posible aplicar técnicas de inteligencia artificial para mejorar la organización y comprensión del discurso sobre la salud mental en este tipo de plataformas?

En los últimos años, la inteligencia artificial (IA) y específicamente el aprendizaje automático (*machine learning, ML*), ha demostrado su amplia capacidad para abordar problemas complejos en los que los métodos tradicionales son insufucientes o costosos, o, lo que también es beneficioso, complementando tareas ya existentes, como la previsión del tiempo [3]. Se han usado con éxito sobre todo en visión por ordenador, predicción médica e incluso análisis financiero, lo que ha motivado la aplicación en el campo de la salud mental digital. Existen trabajos en los que se ha demostrado que es posible identificar síntomas de depresión únicamente mediante un anális lingüístico automatizado de publicaciones de Twitter [4], e incluso en otras se ha explorado una detección temprana de las ideas acerca el suicidio en publicaciones [5].

Todas esas propuestas constituyen lo que hoy en día es una nueva categoría de investigación conocida como **psiquiatría computacional**, que se basa en el uso de técicas de NLP (procesamiento del lenguaje natural), modelos estadísticos y teoría clínica para ofrecer nuevas herramientas de monitoreo y evaluación en salud mental [6]. Por supuesto estas herramientas no deben de ser sustitutivo del diagnóstico médico, pero tienen un gran valor como sistemas de apoyo a la decisión o detección temprana.

Reddit ofrece una serie de ventajas: alto volumen de datos, un anonimato relativo, estructura temática y presencia de una serie de comunidades activas en torno a la salud mental. Por otra parte también presenta una serie de desafios: un lenguaje informal en los posts, expresiones coloquiales, errores gramaticales, sarcasmo y desequilibrios entre categorías de datos.

La clasificación automática de flairs puede definirse como una tarea de clasificación compleja donde el objetivo es predecir una categoría discreta a partir del texto de la publicación. Esta tarea implica construir modelos que sean capaces de interpretar el contenido emocional del lenguaje, tarea que puede no resultar trivial en entornos informales como Reddit.

El análisis de publicaciones relacionadas con la salud mental requiere también de una serie de precauciones adicionales. El tratamiento de los datos plantea cuestiones relacionadas con la privacidad, el consentimiento y el uso responsable de información sensible [7]. Cualquier proyecto que trabaje con este tipo de datos debe de seguir los principios de transparencia, anonimato y respeto por los usuarios, incluso si estos datos han sido obtendios de fuentes públicas.

Este trabajo propone una metodología para realizar una clasificación automatizada de flairs en publicaciones de Reddit mediante técnicas de NLP y aprendizaje supervisado. Se busca contribuir a mejorar la accesibilidad y la estructuración de comunidades virtuales centradas en el bienestar psicológico, incluso facilitando el trabajo para los moderadores, investigadores y usuarios.

Este proyecto no solo responde a una problemática técnica, sino también a la necesidad social de comprender mejor los discursos digitales sobre salud mental, y dotar a las plataformas de herramientas que contribuyan a su uso ético, responsable y empático.

2. Objetivos

2.1. Objetivo principal

El objetivo principal de este Trabajo de Fin de Grado (TFG) ha sido desarrollar un sistema de clasificación automática de posts del subreddit `r/mentalhealth` (extensible a otros subreddits) mediante técnicas de procesamiento del lenguaje natural [8] y modelos de aprendizaje automático. Se busca predecir de el *flair* (etiqueta) que corresponde a una publicación, basándose únicamente en su contenido textual.

El sistema busca mejorar la organización del contenido generado por los usuarios en las comunidades digitales. Al automatizar esta clasificación se espera hacer más sencilla la tarea de moderación, aumentar la acceibilidad de contenido relevante y sentar las bases para futuras herramientas en los entornos digitales.

No se ha buscado sustituir en ningún momento el jucio humano ni en la moderación ni en la interpretación del contenido, sino complementarlo aportando un sistema que hace las veces de primer filtro de clasificación o un asistente en tareas de etiquetado masivo. El sistema se ha concebido como una herramienta de investigación y exploración, no como una solución clínica o diagnóstica.

2.2. Objetivos específicos

Para alcanzar el objetivo principal se han planteado los siguientes objetivos específicos, que han guiado el desarrollo metodológico y técnico del trabajo:

- **Objetivo específico 1: Recolección y almacenamiento de datos.** Diseñar un sistema robusto para extraer publicaciones de Reddit a gran escala, filtrando específicamente aquellas correspondientes al subreddit `r/mentalhealth`, procesando los campos que nos interesan y almacenándolas en una base de datos estructurada para su posterior análisis.
- **Objetivo específico 2: Creación del dataset para el entrenamiento.** Seleccionar de todos los posts existentes un subconjunto de datos con el que realizar el posterior entrenamiento.
- **Objetivo específico 3: Preprocesamiento y depuración del lenguaje natural.** Aplicar técnicas de procesamiento del lenguaje natural (tokenización, lematización, eliminación de ruido, vectorización) para transformar el texto *raw* en representaciones aptas para su análisis, garantizando la calidad y coherencia del dataset.
- **Objetivo específico 4: Entrenamiento y evaluación de modelos de clasificación.** Implementar y comparar distintos algoritmos de clasificación supervisada —incluyendo *Gaussian Naive Bayes*, *Support Vector Machines* y *Random Forest*— para predecir automáticamente el flair más adecuado para cada publicación.
- **Objetivo específico 5: Análisis de resultados y validación del sistema.** Evaluar el rendimiento de los modelos entrenados utilizando métricas como precisión, recall y F1-score. Analizar errores comunes y posibles causas de mal rendimiento.

Los objetivos definidos en este apartado han servido como base estructural para el desarrollo de todo el trabajo.

3. Resumen de la solución propuesta

Con el ojetivo de satisfacer los objetivos planteados en el capítulo 2.2, se desarrolló un sistema completo para la clasificación automática de flairs en publicaciones del subreddit *r/mentalhealth*, basado en técnicas de procesamiento del lenguaje natural (NLP) y modelos de aprendizaje automático (ML). Esta solución ha permitido predecir la flair más adecuada para cada post, facilitando la organización y el análisis del contenido.

La Figura 1 muestra un esquema de la arquitectura general de la solución, que integra módulos de recolección de datos, creación de datasets y preprocesamiento textual y entrenamiento y evaluación de modelos.

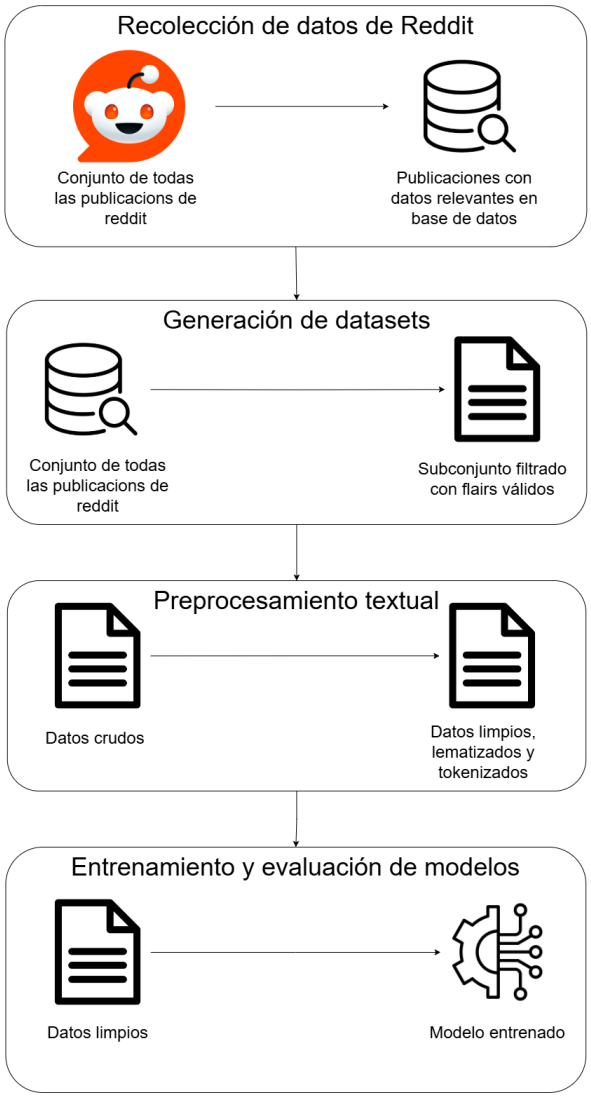

Figura 1: Resumen del flujo general del sistema propuesto para la clasificación automática de flairs a partir de publicaciones textuales.

La implementación del sistema precisó de una serie de etapas:

- **Obtención de datos:** Se extrajo un conjunto enorme de publicaciones de Reddit a partir de un volcado histórico descargado vía torrent. Acto seguido, se filtraron las publicaciones del subreddit `r/mentalhealth` (560.000 en total), y se almacenaron en una base de datos relacional para su análisis posterior.
- **Generación de datasets:** Generación del dataset que fue usado de forma futura para el entrenamiento de los modelos.
- **Preprocesamiento textual:** Al contenido textual de los posts se le aplicó un proceso de limpieza. Esto incluyó tareas como la eliminación de URLs, eliminación de símbolos de puntuacion y caracteres no ASCII, así comoo la tokenización, eliminación de stopwords [9] y lematizacion [10]. Esta fase es esencial para reducir el ruido de los datos y preparalos para su posterior vectorización y entrenamiento de modelos [11].
- **Entrenamiento y evaluación de modelos:** Se entrenaron modelos de clasificación supervisada utilizando las representaciones vectoriales TF-IDF [12] de los datos. Entre los algoritmos que han sido usados están *Gaussian Naive Bayes* [13], *Support Vector Machines (SVM)* [14] y *Random Forest*[15].Se usaron métricas como la precisión, recall y F1-score para ealuar el rendimiento de cada modelo y seleccionar aquel más adecuado.
- **Integración de módulos:** En último lugar, todos los componentes del sistema se integraron en una arquitectura modular, que permite ejecutar todo el flujo desde la consulta a la base de datos hasta la predicción automática del flair para nuevas publicaciones.

Metodología empleada

Durante la etapa de planificación y desarrollo del sistema, se analizaron diversas metodologías de trabajo aplicables a proyectos de minería de datos e inteligencia artificial. Tras dicha revisión, se seleccionó la metodología **CRISP-DM** (Cross-Industry Standard Process for Data Mining), ampliamente reconocida en el ámbito de la analítica de datos y el aprendizaje automático [16].

Esta metodología se compone de seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Para este TFG, únicamente se han implementado las cinco primeras fases, excluyendo la de despliegue, dado que el enfoque del proyecto se centra en realizar un análisis experimental y no en la implementación en producción.

El uso de CRISP-DM resulta muy útil en este tipo de proyectos, ya que permite adoptar un enfoque iterativo de ser necesario, ideal para contextos donde las herramientas y modelos evolucionan rápidamente. De hecho, durante el desarrollo de este trabajo se ha tenido que hacer varias iteraciones en las fases de procesado de datos y modelado, al no resultar satisfactorio el resultado de algunos modelos.

La Figura 2 representa el ciclo de vida seguido en este proyecto, basado en la metodología CRISP-DM. Esta metodología estructurada en seis fases permite abordar proyectos de análisis de datos de forma iterativa. A continuación, se describe cómo se ha aplicado cada una de ellas:

- **Comprensión del negocio:** Se definieron los objetivos del proyecto centrados en la clasificación automática de flairs en Reddit, en particular en el subreddit `r/mentalhealth`.

- **Comprensión de los datos:** Se exploró el volcado de datos de Reddit para identificar las variables relevantes y comprender su estructura.
- **Preparación de los datos:** Se procesaron los textos y se construyó el dataset final utilizado para el modelado.
- **Modelado:** Se entrenaron y compararon diferentes modelos de clasificación supervisada utilizando validación cruzada.
- **Evaluación:** Se analizaron los resultados obtenidos por cada modelo mediante métricas de precisión, recall y F1-score para determinar su rendimiento, así como con gráficos.

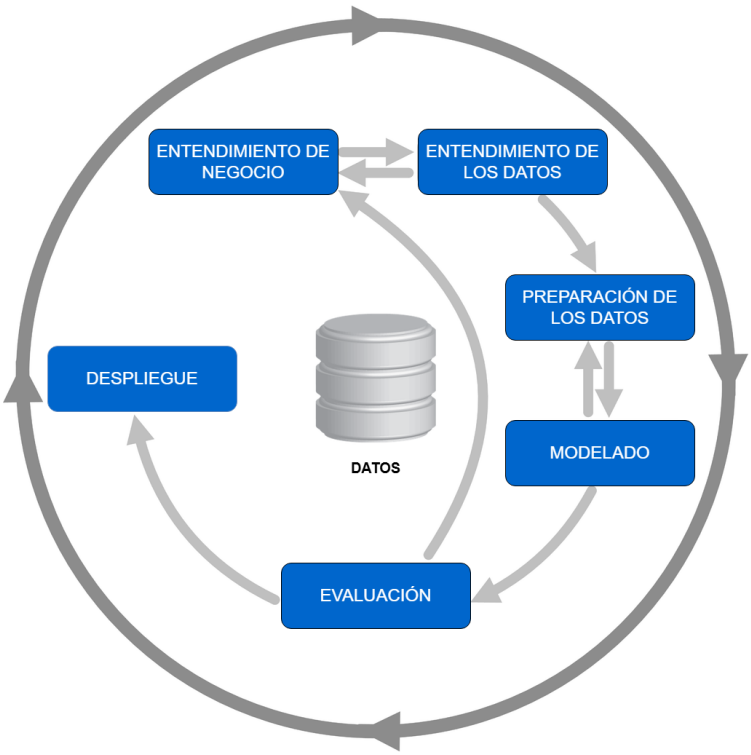

Figura 2: Fases de la metodología CRISP-DM [17].

4. Planificación y seguimiento

4.1. Planificación

Como se mencionó, la metodología elegida para la realización de este Trabajo de Fin de Grado ha sido CRISP-DM, ampliamente adoptada en proyectos de minería de datos y aprendizaje automático por su flexibilidad y enfoque iterativo.

La carga total estimada del proyecto es de **300 horas**, en línea con los 12 créditos ECTS asignados al TFG del Grado en Ingeniería Informática. El desarrollo se planificó entre **diciembre de 2024** y **junio de 2025**, con una dedicación media de aproximadamente 1 hora y media diaria.

Con base en esta estimación, se estableció la siguiente planificación dividida por fases de CRISP-DM:

Tabla 1: Fase 1. Compresnión del negocio

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN
T1	**Definición de los objetivos del proyecto** Se establecieron las metas generales del proyecto, centradas en la clasificación automática de flairs en Reddit.	5
T2	**Traducción a objetivos de ML** A partir de los objetivos generales, se determinaron los retos específicos desde el punto de vista del aprendizaje automático, incluyendo definición de tareas, inputs y outputs esperados.	5
T3	**Planificación de fases** Se definió la estructura del proyecto bajo la metodología CRISP-DM, sin incluir el despliegue.	5
T4	**Comprensión del dominio Reddit y flairs** Se analizó el funcionamiento de Reddit, el subreddit `r/mentalhealth`, y el uso de flairs. Esta fase ha permitido establecer el contexto y alcance del problema.	15
	HORAS TOTALES DE LA FASE	**30**

Tabla 2: Fase 2. Comprensión de los datos

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN
T5	**Recolección y verificación de datos** Se hizo una descarga de los datos de posts de reddit para el subreddit `r/mentalhealth`..	30
	HORAS TOTALES DE LA FASE	**30**

Tabla 3: Fase 3. Preparación de los datos

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN
T6	**Limpieza del texto** Se conviertió el texto a minúscula, se eliminaron URLs, signos de puntuación, caracteres no ASCII, se eliminaron stopwords y se aplicó un proceso de lematización .	15
T7	**Definición y estructuración del dataset** Se construyó el conjunto de datos con columnas bien definidas (texto limpio, flair). Se almacenó en formato CSV (Comma Separated Values) [18] y Pickle [19] listo para entrenamiento.	20
T8	**Verificación y depuración final** Se inspeccionó manual y automáticamente una muestra del dataset para asegurar consistencia en el número de clases, ausencia de valores nulos y distribución equilibrada de flairs.	5
	HORAS TOTALES DE LA FASE	**40**

Tabla 4: Fase 4. Modelado

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN
T9	**Definición de métricas** Se definieron métricas clave para evaluar los modelos: F1-score, precision, recall, accuracy y matriz de confusión.	10
T10	**Selección de algoritmos** Se han comparado modelos de clasificación supervisada como Gaussian Naive Bayes, SVM y Random Forest para determinar los más adecuados al problema.	10
T11	**Generación de características adicionales** Se ha explorado la creación de features adicionales (como longitud del texto, frecuencia de palabras clave, etc.) para mejorar el rendimiento.	20
T12	**Entrenamiento de modelos** Se entrenaron los modelos seleccionados usando los datos preprocesados, aplicando técnicas de validación cruzada y conjuntos de prueba.	40
T13	**Evaluación intermedia** Análisis de los resultados iniciales, detección de errores comunes y evaluación de flairs difíciles de clasificar.	10
T14	**Ajuste de hiperparámetros** Optimización de los parámetros de los modelos entrenados.	20
	HORAS TOTALES DE LA FASE	**120**

Tabla 5: Fase 5. Evaluación

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN
T15	**Validación final del modelo** Se ha evaluado el mejor modelo con un conjunto de prueba independiente, verificando su rendimiento general y comparando métricas.	30
T16	**Documentación de resultados** Se registraron resultados finales, se generaron visualizaciones (matriz de confusión, gráficas comparativas) y se redactó el análisis.	15
	HORAS TOTALES DE LA FASE	**50**

Tabla 6: Fase 6. Documentación y presentación

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN
T17	**Redacción del TFG** Elaboración del documento final. Redacción académica con citas y referencias.	30
T18	**Elaboración de figuras y anexos** Creación de tablas, gráficas, diagramas y anexos para apoyar el contenido del documento.	10
T19	**Corrección y revisión** Revisión ortográfica, sintáctica y comprobación del formato.	5
	HORAS TOTALES DE LA FASE	**50**

La Figura 3 muestra la planificación temporal mediante un diagrama de Gantt, utilizado para visualizar la duración estimada de cada fase.

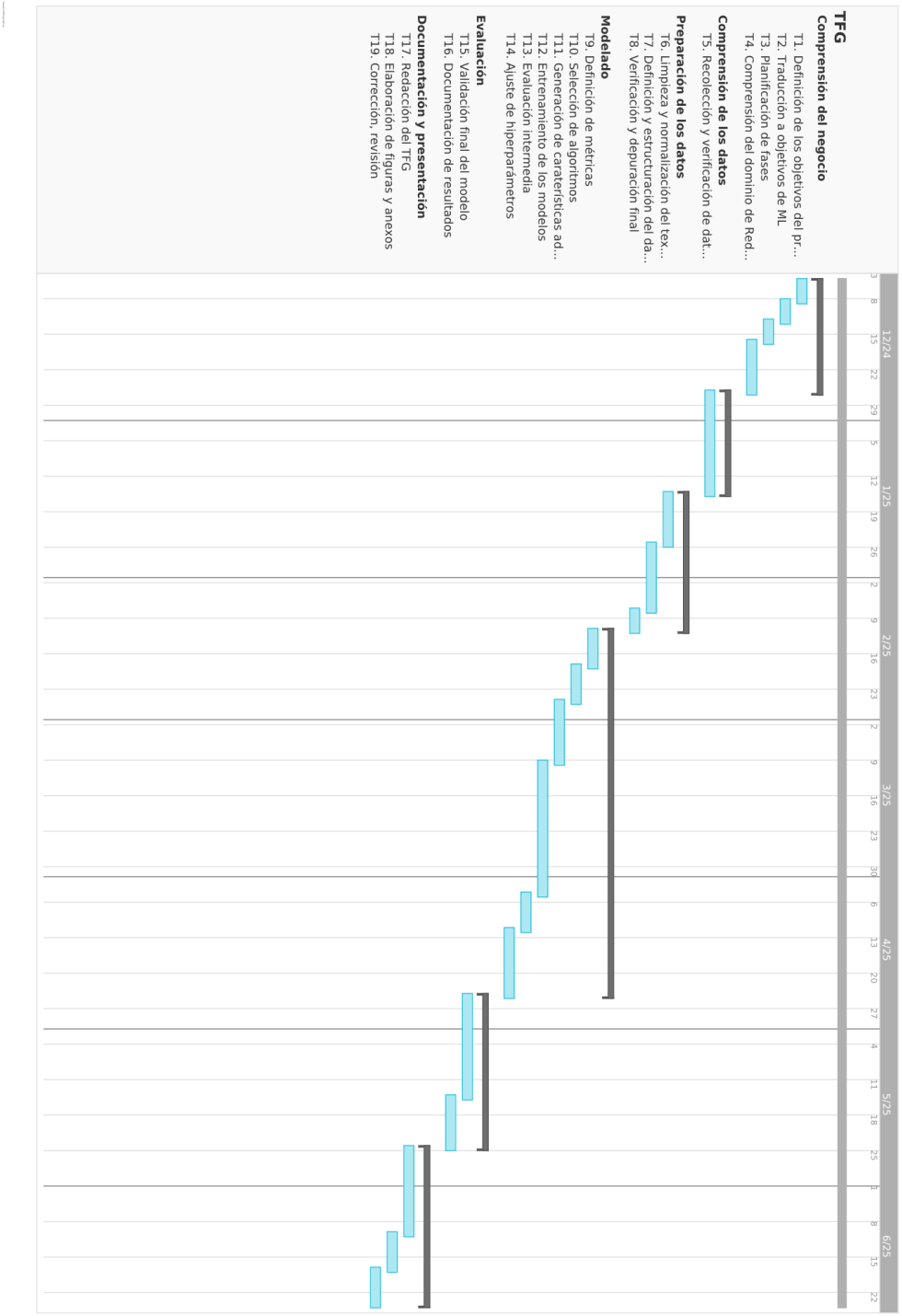

Figura 3: Diagrama de Gantt de la planificación inicial del proyecto.

Esta estructura detallada no solo facilitó una gestión eficiente del tiempo, sino que también

permitió una constante alineación con los objetivos definidos para el proyecto.

4.2. Seguimiento

La duración real del proyecto ascendió a aproximadamente **314 horas**, un poco por encima de lo previsto, debido a varios factores imprevistos que afectaron principalmente a la recolección de datos y al entrenamiento de modelos.

A continuación se muestran las horas reales consumidas en cada fase del proyecto:

Tabla 7: Fase 1. Comprensión del negocio - Seguimiento

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN ESTIMADA	DURACIÓN REAL
T1	Definición de los objetivos del proyecto	5	5
T2	Traducción a objetivos de ML	5	5
T3	Planificación de fases	5	5
T4	Comprensión del dominio Reddit y flairs	15	8
	HORAS TOTALES DE LA FASE		**23**

Tabla 8: Fase 2. Comprensión de los datos - Seguimiento

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN ESTIMADA	DURACIÓN REAL
T5	Recolección y verificación de datos.	30	55
	HORAS TOTALES DE LA FASE		**55**

Tabla 9: Fase 3. Preparación de los datos - Seguimiento

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN ESTIMADA	DURACIÓN REAL
T6	Limpieza y normalización del texto	15	18
T7	Definición y estructuración del dataset	20	20
T8	Verificación y depuración final	5	6
	HORAS TOTALES DE LA FASE		**44**

Tabla 10: Fase 4. Modelado - Seguimiento

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN ESTIMADA	DURACIÓN REAL
T9	Definición de métricas	10	10
T10	Selección de algoritmos	10	10
T11	Generación de características adicionales	20	14
T12	Entrenamiento de modelos.	40	50
T13	Evaluación intermedia	10	10
T14	Ajuste de hiperparámetros	20	20
	HORAS TOTALES DE LA FASE		**114**

Tabla 11: Fase 5. Evaluación - Seguimiento

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN ESTIMADA	DURACIÓN REAL
T15	Validación final del modelo	30	20
T16	Documentación de resultados	15	15
	HORAS TOTALES DE LA FASE		**35**

Tabla 12: Fase 6. Documentación y presentación - Seguimiento

TAREA	NOMBRE Y DESCRIPCIÓN	DURACIÓN ESTIMADA	DURACIÓN REAL
T17	Redacción del TFG	30	28
T18	Elaboración de figuras y anexos	10	10
T19	Corrección y revisión	5	5
	HORAS TOTALES DE LA FASE		**43**

Las principales desviaciones vienen derivadas de:

- **Trabas en la recolección de datos:** En un primer momento se planeó el uso de la API de Reddit mediante PRAW. Sin embargo existía una serie de restricciones tanto en la recuperación por fechas como en el número total de publicaciones que se podían descargar. Todo esto forzó a cambiar a un volcado masivo de datos vía torrent. Este volcado contenía archivos comprimidos (.zst) de un gran tamaño y requirió de un proceso complejo de filtrado, descompresión y parsing en paralelo.
- **Limitaciones computacionales:** Algunos modelos (especialmente Random Forest) exigieron mayores recursos de memoria de los disponibles en el equipo personal. Fue necesario recurrir a ordenadores de la facultad con mayor capacidad para completar el entrenamiento sin errores de segmentación o *out-of-memory*.
- **Falta de experiencia previa:** Al no contar con experiencia directa en minería de datos sobre texto de Reddit, se invirtió un tiempo considerable en el estudio de herramientas específicas como TF-IDF, uso de los modelos, etc.

Estos factores derivaron en desviaciones en los tiempos reales, sobre todo en la parte de descarga de los datos y de entrenamiento de modelos.

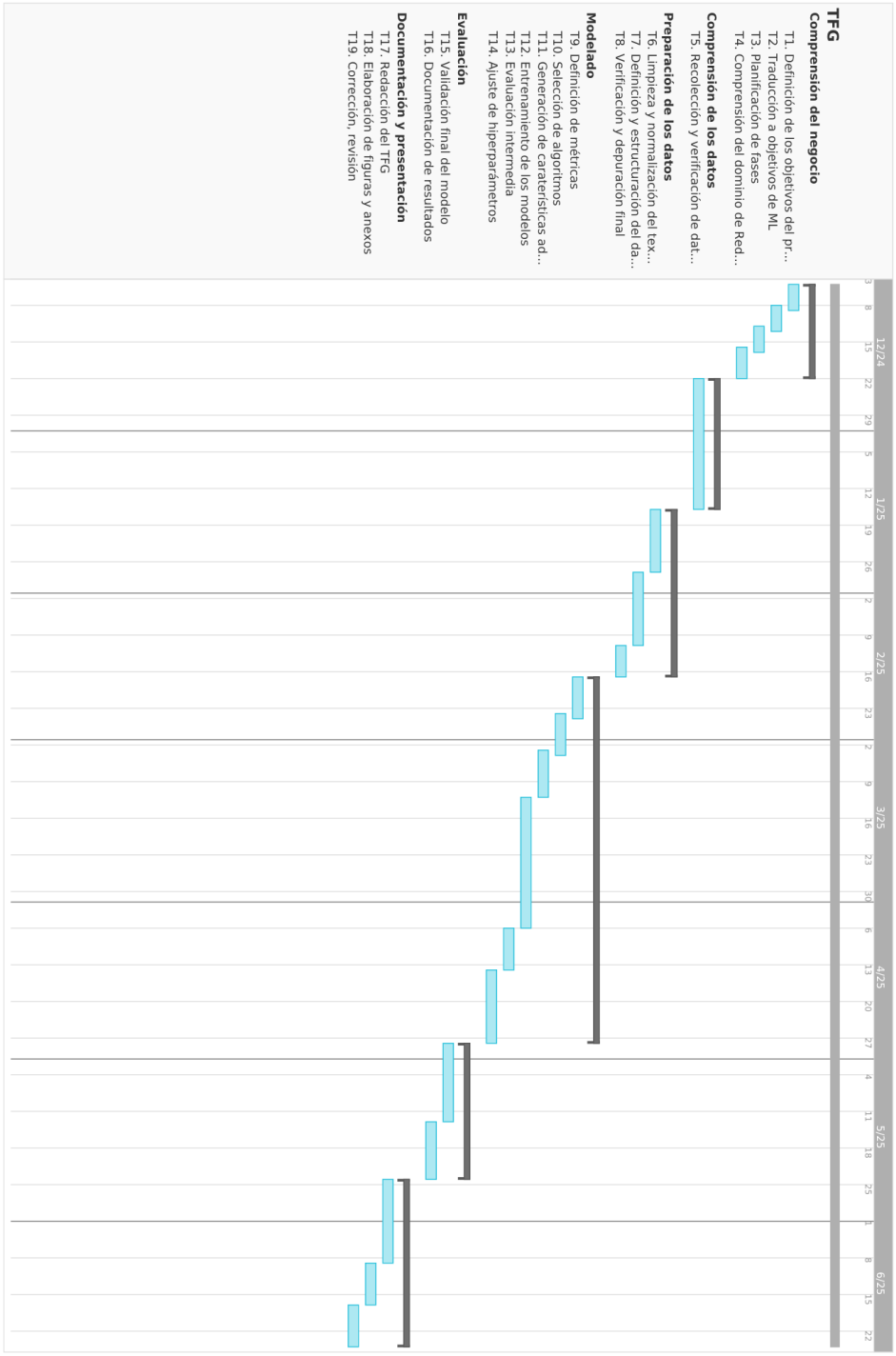

Figura 4: Diagrama de Gantt real del proyecto con desviaciones.

5. Arquitectura

5.1. General

La arquitectura del sistema propuesto se estructuró en cuatro componentes fundamentales que han permitido la automatización del flujo de trabajo completo para la clasificación automática de flairs en publicaciones del subreddit `r/mentalhealth`. Estos bloques abarcan desde la obtención y almacenamiento de datos, hasta el modelado y evaluación de modelos de aprendizaje automático.

En primer lugar, se partió de un volcado masivo de Reddit que se ha procesado mediante scripts paralelizados en Python y se almacenó en una base de datos relacional MySQL. Posteriormente, se realizó una selección y limpieza de los datos relevantes, aplicando técnicas de Procesamiento de Lenguaje Natural (NLP), como normalización, eliminación de ruido textual, y tokenización.

La etapa de modelado se ha apoyado en algoritmos clásicos de clasificación supervisada como `Gaussian Naive Bayes`, `SVM` y `Random Forest`. Estos modelos fueron implementados y ajustados manualmente, utilizando como soporte bibliotecas especializadas de Python, como scikit-learn. La evaluación de los resultados se llevó a cabo mediante métricas como accuracy, precision, recall y F1-score, apoyándose en representaciones visuales como matrices de confusión y reportes de clasificación. . En conjunto, el pipeline completo `-desde la extracción y preprocesamiento del texto, hasta el almacenamiento, entrenamiento, evaluación-` fue desarrollado empleando tecnologías de propósito general como Python, pandas y bibliotecas especializadas.

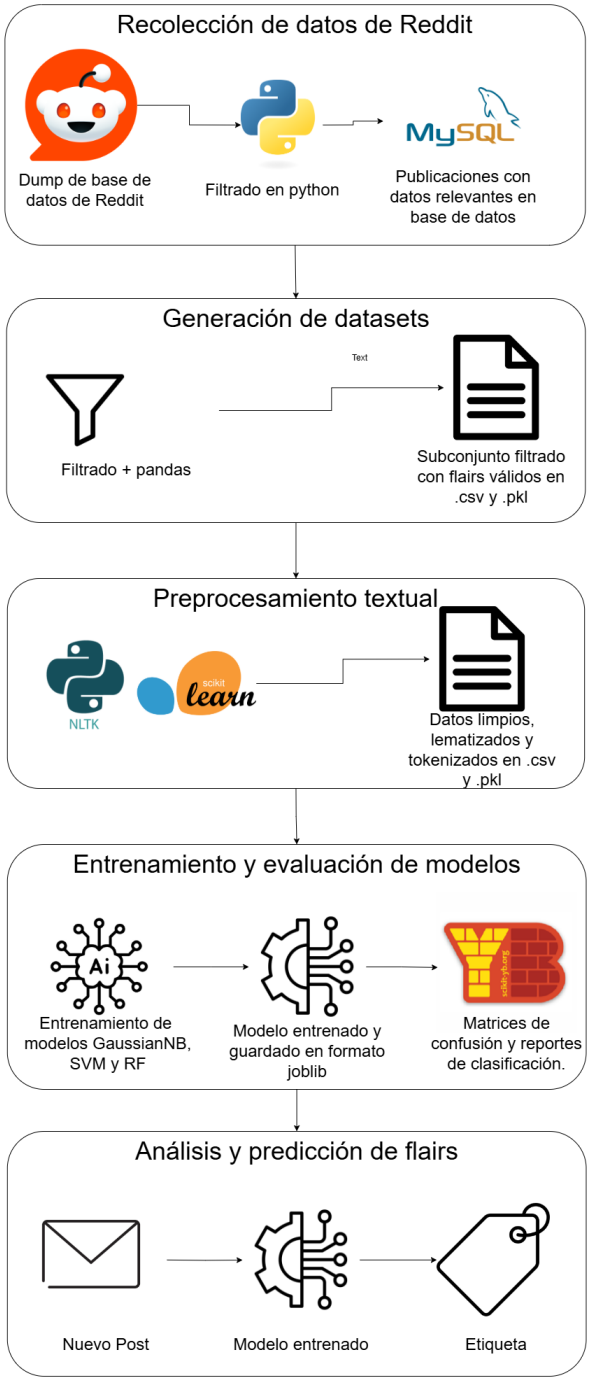

Figura 5: Arquitectura general del proyecto.

5.2. Componentes

1. Extracción y almacenamiento de datos

Este componente se encarga de obtener y organizar los datos necesarios para el análisis.

- **Fuente de datos:** Se usó un volcado masivo de Reddit obtenido a través de un archivo torrent. La API oficial no ofrecía acceso completo al histórico de posts y el número de posts del que se podía hacer la descarga era muy pequeño.
- **Procesamiento inicial:** Los archivos comprimidos (.zst) fueron descomprimidos y filtrados utilizando varios hilos de procesamiento en Python para extraer exclusivamente los posts del subreddit `r/mentalhealth`.
- **Base de datos:** Los posts fueron almacenados en una base de datos MySQL [20], guardando aquellas columnas que se condiraron necesarias como el título, texto, autor, fecha, flair, etc., permitiendo consultas SQL para facilitar el filtrado posterior.

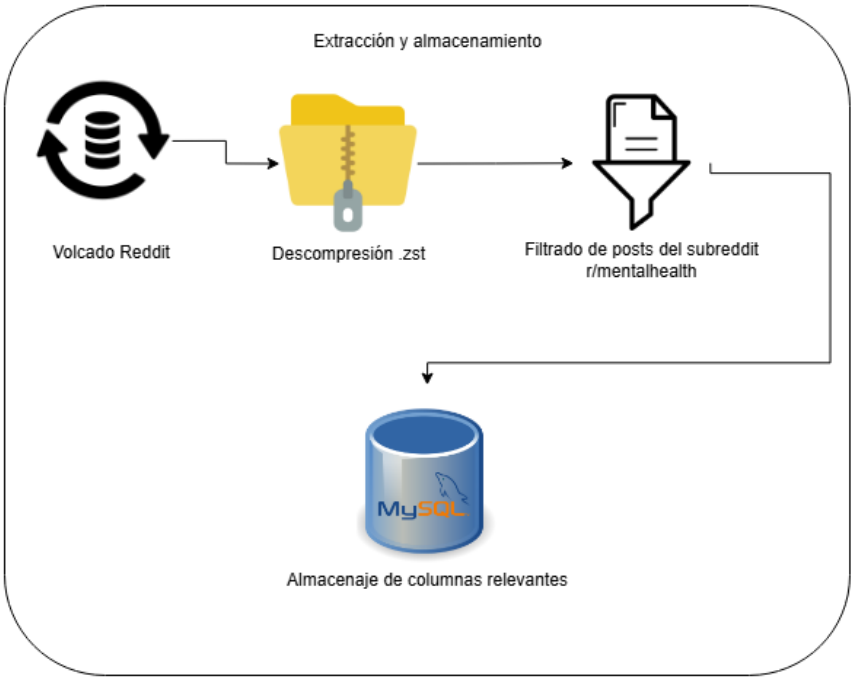

Figura 6: Proceso de extracción y almacenamiento de datos.

2. Preprocesamiento, construcción del dataset y transformación

Esta etapa comprende tanto la selección de los datos relevantes desde la base de datos general como el preprocesamiento textual para su análisis mediante técnicas de PLN.

- **Selección y construcción del dataset:** A partir de la base de datos completa generada desde el volcado de Reddit, se filtraron las publicaciones del subreddit `r/mentalhealth` de las fechas comprendidas entre 01/01/2024 y 30/06/2024, almacenando su contenido en un dataset en CSV y Pickle con los datos en crudo.

- **Limpieza:** Se eliminaron URLs, símbolos de puntuación, caracteres no ascii y stopwords. El texto se normalizó a minúsculas.
- **Lematización:** Se aplicó un proceso de reducción de palabras a su forma base, lo que permitió tratar como equivalentes distintas formas flexionadas de un mismo término (por ejemplo, "corriendo" y "correr").
- **Tokenización:** Se realizó la tokenización del texto por palabras.
- **Verificación:** Se realizó una inspección manual de una muestra del dataset para asegurar que los datos no contienen elementos nulos, tanto en el flair como en el texto preprocesado.
- **Formato final:** El dataset se almacenó en formatos CSV y Pickle para facilitar su posterior carga durante el modelado. Las columnas incluidas para realizar la clasificación fueron la flair y el texto limpio obtenido.

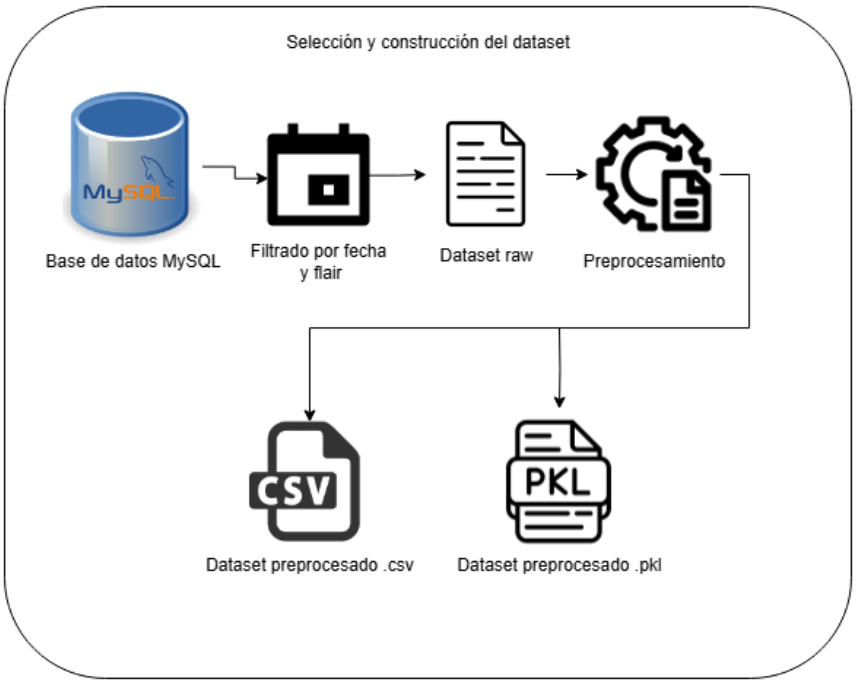

Figura 7: Proceso de selección y construcción del dataset desde la base de datos general.

3. Modelado y clasificación

En este bloque se desarrollan y entrenan los modelos de clasificación automática.

- **Modelos utilizados:** Se evaluaron clasificadores supervisados como `Gaussian Naive Bayes`, `SVM` y `Random Forest`.
- **Entrenamiento:** Se utilizó validación cruzada para mejorar la capacidad de generalización del modelo.

- **Evaluación:** Las métricas utilizadas fueron *accuracy*, *F1-score*, precisión, y matriz de confusión.

La Figura 8 resume el proceso de entrenamiento y validación de modelos.

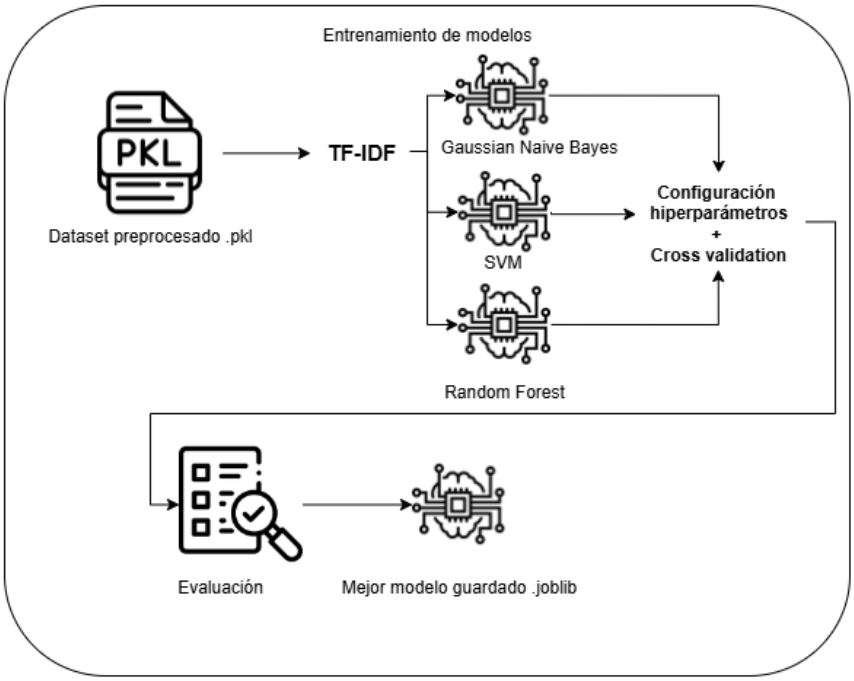

Figura 8: Fase de modelado y entrenamiento de modelos de clasificación.

4. Análisis y predicción de flairs

El objetivo del sistema es predecir de forma automática el flair más probables para una nueva publicación textual.

- **Entrada:** Texto de un post del subreddit `r/mentalhealth`.
- **Salida esperada:** Flair sugerido por el modelo entrenado.

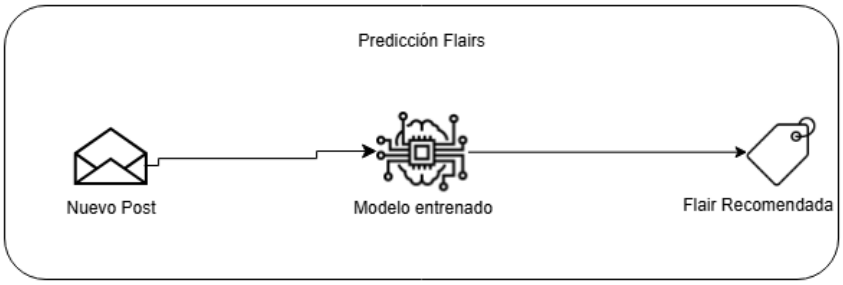

Figura 9: Predicción automática de flairs a partir de texto libre.

6. Tecnologías e integración de productos de terceros

Para este proyecto se han usado varias tecnologías seleccionadas para satisfacer la problemática de la clasificación automática de flairs en los posts del subreddit `r/mentalhealth`.

6.1. Lenguajes de programación

- **Python 3.12 [21]:** Lenguaje principal del proyecto. Ha sido utilizado para tareas de preprocesamiento de texto, gestión de la base de datos, modelado y evaluación de modelos de aprendizaje automático.

6.2. Bibliotecas y marcos de trabajo en Python

A lo largo del desarrollo del proyecto se utilizaron múltiples librerías del ecosistema de Python:

- **pandas 2.2.3 [22]:** Usada para la manipulación y el análisis de datos estructurados, así como para transferir los datos de memoria a archivos CSV y pkl.
  Motivo de su utilización: Usado para la manipulación y el análisis de los datos así como para transferir los datos de memoria a archivos CSV y pkl.

- **joblib 1.5.1 [23]:** Serialización de modelos de aprendizaje automático y ejecución de tareas en paralelo.
  Motivo de su utilización: Se ha utilizado principalmente para el guardado de modelos entrenados.

- **SQLAlchemy 2.0.41 [24]:** ORM (Object Relational Mapper) para la interactuar con la base de datos MySQL, utilizado tanto para escritura como consulta.
  Motivo de su utilización: Permite cómodamente desde Python recuperar e insertar datos dentro de la BD MySQL.

- **scikit-learn 1.6.1 [25]:** Biblioteca principal para el aprendizaje automático. Proporciona herramientas simples pero eficientes.
  Motivo de su utilización: Se ha usado para implementar los modelos de aprendizaje automático *Naive Bayes*, *Random Forest*, y *SVM*, además de herramientas de preprocesamiento, *pipelines* y validación cruzada.

- **nltk 3.9.1 [26]:** Proporciona un conjunto de herramientas para el procesamiento de lenguaje natural.
  Motivo de su utilización: Se ha empleado para realizar tareas como *tokenización*, *lemmatización*, y eliminación de *stopwords*.

- **re (Python 3.12) [27], string (Python 3.12) [28], datetime (Python 3.12) [29] :** Utilizadas para procesamiento de texto (regex, limpieza de puntuación) y gestión temporal.

- **argparse (Python 3.12) [30]:** Permite realizar la gestión de argumentos de línea de comandos para scripts.
  Motivo de su utilización: Se ha utilizado para gestionar los parámetros de entrada de los scripts así como para ofrecer la ayuda para su construcción.

- **pickle (Python 3.12):** Permite la serialización de objetos en disco.
  Motivo de su utilización: Ha sido particularmente útil para almacenar el dataset y resultados intermedios.
- **zstandard 0.23.0 [31]:** Permite de archivos `.zst` desde Python.
  Motivo de su utilización: Ha sido utilizada para la descompresión de los archivos de la base de datos descargados desde el archivo torrent.
- **praw 7.8.1 [32]:** Librería oficial de Reddit para acceder a información mediante su API.
  Motivo de su utilización: Inicialmente fue empleada para extracción de datos pero acabó siendo descartada para usar el dump de la base de datos de Reddit descargado desde el torrent.
- **yellowbrick 1.5 [33]:** Permite la visualización de métricas de modelos de aprendizaje automático.
  Motivo de su utilización: Ha sido utilizado para visualizar las matrices de confusión de los diversos modelos probados.
- **logging (Python 3.12) [34]:** Permite un registro estructurado del comportamiento de los scripts, errores y métricas de ejecución.
  Motivo de su utilización: Se ha utilizado para tener un registro de la descompresión y de los datos por el script ingesta de datos.
- **mysql-connector-python 9.3.0 [35]:** Conector oficial de MySQL para Python que permite la conexión directa a bases de datos sin necesidad de configurar un ORM.
  Motivo de su utilización: Utilizado para realizar pruebas directas de conexión y ejecución de consultas sobre la base de datos MySQL durante la fase de desarrollo inicial y verificación de la estructura de tablas.

6.3. Bases de datos

- **MySQL 8.0.40 :** Sistema gestor de bases de datos utilizado para almacenar los datos filtrados del volcado de Reddit. Se empleó como intermediario para almacenar los datos después del volcado original y antes del preprocesamiento.

6.4. Entornos y herramientas de desarrollo

- **Visual Studio Code 1.99.3 [36]:** Editor de código gratuito, creado por Microsoft que permite la edición y visualización de múltiples lengajes, es ligero y pueden instalarse multitud de extensiones.
  Motivo de su utilización: Se utilizó tanto para el desarrollo del código python como para visualizar datasets y ejecutar los scripts directamente desde su terminal.
- **GitHub [37]:** Plataforma web de desarrollo colaborativo que utiliza Git como sistema de control de versiones.
  Motivo de su utilización: Mayoritariamente para el respaldo del código fuente del proyecto.
- **Google Drive [38]:** Plataforma de almacenamiento en la nube que permite compartir documentos y archivos.

Motivo de su utilización: Se ha usado mayoritariamente para compartir archivos entre distintos dispositivos.

- **Overleaf [39]:** Herramienta colaborativa en línea para redactar documentos en LaTeX.
  Motivo de su utilización: Ha sido utilizada para la redacción de la memoria de este TFG.
- **Draw.io:** Plataforma web para la creación de diagramas.
  Motivo de su utilización: Ha sido utilizado para la creación de los diagramas de arquitectura, flujo de trabajo y esquemas de procesamiento de datos.
- **TeamGantt [40]:** Herramienta para la planificación temporal de proyectos mediante diagramas de Gantt.
  Motivo de su utilización: Se ha utilizado para la realizacion de los diagramas de Gantt de la planificación y el seguimiento.
- **TeamViewer 15.66.5 [41]:** Herramienta de acceso remoto que permite conectarse a otros equipos.
  Motivo de su utilización: Ha sido utilizado para conectarse a equipos de la facultad con mayores capacidades de procesamiento durante el entrenamiento de modelos como Random Forest.

6.5. Productos de terceros

- **Volcado masivo de Reddit (Pushshift [42] Torrent):** Fuente principal de datos, compuesta por archivos comprimidos en formato `.zst` con todos los posts de Reddit. Permitió el acceso a millones de publicaciones de diversas fechas sin las restricciones que tenía la API oficial de Reddit.

7. Especificación y análisis de requisitos

En esta sección se detallan los requisitos necesarios para un correcto desarrollo del proyecto, abarcando aspectos funcionales y no funcionales.

7.1. Requisitos funcionales

1. **Integración del sistema de clasificación con la base de datos Reddit**
   - **Descripción:** El sistema debe de conectarse a una base de datos que contenga un volcado masivo de publicaciones del subreddit `r/mentalhealth`.
   - **Objetivo:** Obtener un subconjunto de datos relevantes para ser clasificados automáticamente según su *flair*.
2. **Preprocesamiento de datos**
   - **Descripción:** El sistema debe realizar automáticamente la limpieza del texto de los posts (eliminación de URLs, símbolos, normalización a minúsculas, etc.).
   - **Objetivo:** Estandarizar los datos para asegurar una entrada limpia y coherente a los modelos de ML.
3. **Clasificación automática mediante algoritmos de ML**
   - **Descripción:** Implementación de modelos supervisados (Naive Bayes, SVM, Random Forest) para predecir el flair más probable de una publicación.
   - **Objetivo:** Automatizar la clasificación de posts del subreddit `r/mentalhealth`.
4. **Evaluación de modelos y visualización de resultados**
   - **Descripción:** Evaluar los modelos entrenados mediante métricas como *precision*, *recall*, *F1-score* y matriz de confusión.
   - **Objetivo:** Determinar la calidad del modelo y generar reportes que permitan su interpretación.

7.2. Requisitos no funcionales

1. **Rendimiento**
   - **Descripción:** El sistema debe optimizar el uso de CPU y RAM durante el entrenamiento y la inferencia.
   - **Objetivo:** Garantizar un tiempo de ejecución aceptable incluso con modelos y datasets moderadamente grandes (Difícil en modelos complejos como Random Forest).
2. **Escalabilidad**
   - **Descripción:** El sistema debe poder adaptarse al uso de datasets más amplios o nuevos subreddits sin necesidad de reescribir el código.
   - **Objetivo:** Facilitar la extensión del sistema a otros casos similares con mínima intervención técnica.

3. **Mantenibilidad**
    - **Descripción:** El código debe ser claro, modular y comentado.
    - **Objetivo:** Permitir que futuros desarrolladores puedan entender, adaptar o mejorar fácilmente el sistema.
4. **Interoperabilidad**
    - **Descripción:** El sistema debe garantizar la correcta integración entre componentes desarrollados en distintos lenguajes (Python y SQL/MySQL).
    - **Objetivo:** Evitar incompatibilidades entre versiones o entornos que puedan afectar el flujo de trabajo.

8. Diseño del software

El diseño del software se estructuró en torno al procesamiento y análisis de datos textuales extraídos del subreddit `r/mentalhealth`, para así poder clasificar automáticamente las publicaciones según su *flair*. En esta sección se describe el flujo de trabajo que va desde la obtención inicial de los datos hasta la evaluación del rendimiento de los modelos de ML.

8.1. Modelado

8.1.1. Creación del conjunto de datos

El dataset utilizado fue generado a partir de un volcado masivo de la base de datos de Reddit previamente almacenado en una base de datos MySQL. Para el proceso se realizó:

- **Consulta SQL:** Se extrajeron los posts del subreddit `r/mentalhealth` entre enero de 2024 y junio de 2024 (ambos incluidos).
- **Filtrado:** Se eliminaron entradas con flairs poco frecuentes, ya que no aportaban suficiente representación para permitir un entrenamiento fiable de los modelos de clasificación supervisada. Esta decisión se tomó para evitar clases desbalanceadas que pudieran afectar negativamente al rendimiento y la generalización del modelo. Al final nos hemos quedado con las flairs de *'Need Support', 'Venting', 'Opinion / Thoughts', 'Sadness / Grief', 'Good News / Happy', 'Inspiration / Encouragement', 'Resources'*.
- **Exportación:** Los datos se almacenaron tanto en formato `.csv` y `.pkl` para facilitar su posterior tratamiento en Python.

La Figura 10 nos muestra la distribución de los flairs dentro de nuestro dataset. Cabe destacar que el total de publicaciones del dataset es de 31225 y el número de palabras por cada post varía desde las 3 del que menos tiene, a 5800 del que más, teniendo una media de palabras de unas 240 por post.

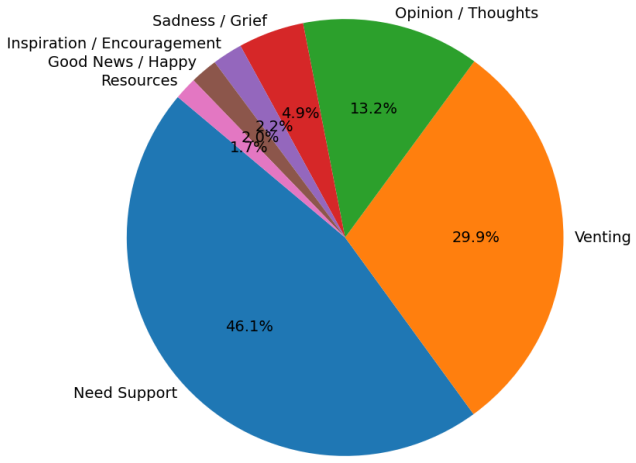

Figura 10: Distribución de flairs en el dataset.

8.1.2. Preprocesamiento del texto

Antes del modelado, se aplicó una limpieza exhaustiva del texto para eliminar elementos que pudieran dificultar el análisis al modelo de Machine learning. Las principales transformaciones fueron:

- Conversión a minúsculas.
- Eliminación de URLs (enlaces web), signos de puntuación, emojis, y caracteres no ascii que no aportaban información semántica relevante para el análisis.
- Tokenización y eliminación de *stopwords*.
- Lemmatización.

Estas tareas se implementaron con las librerías `nltk`, `re`, y `scikit-learn`, encapsuladas en un pipeline de preprocesamiento reutilizable para otros módulos y scripts.

8.1.3. Vectorización del texto mediante TF-IDF

Antes del entrenamiento de los modelos, fue necesario convertir el texto preprocesado en una representación numérica que pudiera ser entendida por los algoritmos de aprendizaje automático. Para llevar esto a cabo, se utilizó la técnica **TF-IDF (Term Frequency-Inverse Document Frequency)**, una forma de vectorización que asigna un peso a cada palabra en función de su frecuencia en un documento y su rareza en el conjunto total de documentos.

El objetivo de TF-IDF es destacar términos relevantes que ayudan a diferenciar entre clases, penalizando palabras muy comunes que aparecen en todos los textos (por este motivo se realiza la eliminación de stopwords) y otorgando más peso a aquellas que son representativas de un documento en particular. Esta técnica es especialmente adecuada para tareas de clasificación de texto.

Además, para mejorar la capacidad del modelo de capturar patrones en el texto, se experimentó con el parámetro `n_gram_range` del vectorizador TF-IDF. Este parámetro permite especificar si se emplean *unigramas* (palabras individuales), *bigramas* (pares consecutivos de palabras), o una combinación de ambos en la representación numérica. La inclusión de *n-gramas* más largos puede ayudar a capturar contexto y relaciones entre palabras que los unigramas solos no representan, lo que en algunos casos mejora la precisión del modelo.

Gracias a esta representación y a la optimización del rango de *n-gramas*, fue posible transformar el texto en una matriz numérica que sirvió como entrada para el entrenamiento de los modelos de clasificación.

8.1.4. Selección de los modelos

Para hacer frente al problema de la clasificación automática de flairs en posts de Reddit se eligieron tres algoritmos de aprendizaje supservisado, que son muy utilizados en tareas de procesamiento de lenguaje natural (NLP): Gaussian Naive Bayes, Máquinas de Vectores de Soporte (SVM) y Random Forest.

- **Gaussian Naive Bayes (GNB):**

  Está basado en el teorema de Bayes, supone independencia condicional entre las palabras de un texto. Aunque esta independecia rara vez se cumple en textos reales ya que las palabras tienden a correlacionarse, GNB suele obtener buenos resultados.

Ventajas:

- Entrenamiento rápido.
- Robusto a conjuntos de datos pequeños o medianos.
- Funciona bien cuando las representaciones numéricas del texto tienen muchos ceros, como con TF-IDF.

Limitaciones:

- Asume independencia entre características, lo que no es frecuente.
- Su rendimiento puede verse afectado en clases con mucha superposición semántica, es decir cuando el significado de sus datos tiende a ser similar.

Fórmula matemática:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

- **Máquinas de Vectores de Soporte (Linear SVM):**

Las SVM buscan la línea o plano que maximice el margen entre clases. En problemas de texto, donde las dimensiones son altas (cada palabra o n-grama genera una dimensión) las SVM lineales tienden a funcionar especialmente bien. En esta implementación, se utilizó el clasificador `LinearSVC` de `scikit-learn`, con una estrategia de clasificación *one-vs-rest* para tareas multiclase, lo que busca es entrenar múltiples clasificadores, donde cada uno trata como positivo a su clase y como negativo a todas las demás, de forma que cuando clasifica mira cuál es el clasificador con el que tiene más confianza para decidir la clase final.

Ventajas:

- Muy eficaz en espacios de muchas dimensiones.
- Tienden a ser tolerantes al sobreajuste (over-fitting [43]) gracias a su parámetro de regularización, que permite aumentar la generalización.
- Rápido tanto en entrenamiento como en predicción para grandes conjuntos de texto.

Limitaciones:

- Sensible al desbalanceo de clases.
- A diferencia de otros modelos como el Naive Bayes, SVM no da una probabilidad de pertenencia a cada clase, sino solo la predicción final.

Fórmula matemática:

$$\begin{aligned} \min_{w,b,\zeta} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to} \quad & y_i (w^T \phi(x_i) + b) \geq 1 - \zeta_i, \\ & \zeta_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

- **Random Forest Classifier (RF):**

  Random Forest es un método basado en árboles de decisión. Entrena múltiples árboles sobre subconjuntos aleatorios del conjunto de datos y selecciona la clase final por una votación mayoritaria. Es un modelo no lineal, lo que le permite capturar interacciones complejas entre las características.

  Ventajas:

  - Alta precisión y rendimiento, especialmente para conjuntos de datos grandes y complejos
  - Muy resistente al over-fitting.
  - Alta versatilidad y aplicabilidad, gracias al soporte tanto para tareas de clasificación como de regresión.

  Limitaciones:

  - Consumo elevado de memoria y tiempo, ya que entrena muchos árboles.

Fórmula matemática:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Cada uno de estos modelos se entrenó sobre una representación vectorizada del texto (mediante TF-IDF), se validó con validación cruzada, y fue evaluado utilizando métricas como F1-score y matriz de confusión.

8.1.5. Entrenamiento y ajuste

Se aplicó un enfoque de entrenamiento y ajuste basado en técnicas estándar en problemas de clasificación de texto.

En primer lugar, el conjunto de datos fue dividido en dos subconjuntos: entrenamiento (80 %) y validación (20 %), lo que se conoce como **holdout** [44], una técnica estándar y sencilla para evaluar el desempeño de los algoritmos de aprendizaje automático. Se ha utilizando muestreo estratificado para preservar la distribución de clases, esto es, que una clase tenga una proporción similar tanto para el conjunto de entrenamiento como para el conjunto de validación. Dado el desbalanceo significativo en la distribución de etiquetas del dataset, con algunas *flairs* representando un alto porcentaje y otras apenas un pequeño porcentaje (por ejemplo, la *flair* 'Need Support' representaba aproximadamente el 46 % de las muestras, mientras que 'Resources' no superaba el 2 % —)se decidió emplear la opción `class_weight="balanced"` en modelos como SVM y Random Forest.

Esta configuración ajusta automáticamente el peso de cada clase durante el entrenamiento, otorgando mayor importancia a las clases minoritarias para compensar el desbalance y evitar que el modelo favorezca predominantemente las clases mayoritarias.

Inicialmente, se entrenaron modelos sin este ajuste, observándose un sesgo hacia las clases más frecuentes y un rendimiento pobre en las minoritarias. Posteriormente, con el parámetro `class_weight="balanced"` se mejoró la capacidad del modelo para reconocer las clases menos representadas.

A continuación, se construyó un pipeline completo compuesto por:

- Un vectorizador `TF-IDF`, que transforma el texto preprocesado en vectores numéricos.

- Un clasificador ajustable, dependiendo del modelo especificado (GaussianNB, LinearSVC o RandomForest).
- Un transformador intermedio (`DenseTransformer`) en el caso de Naive Bayes, ya que este clasificador requiere una entrada densa, no proporcionada por TF-IDF, que devuelve una matriz dispersa.

El entrenamiento se completó utilizando `GridSearchCV` [45], con validación cruzada [46] de 10 folds (`cv=10`), para identificar automáticamente los hiperparámetros más adecuados en cada caso:

- Para **Naive Bayes**, se ajustó el parámetro `var_smoothing`, el cual controla la estabilidad numérica.
- Para **SVM lineal**, se configuraron distintas regularizaciones (`C`), tolerancias de convergencia y número máximo de iteraciones.
- Para **Random Forest**, se emodificaron los valores de profundidad máxima (`max_depth`), número de árboles (`n_estimators`) y criterios de poda de árbol como `min_samples_split` y `min_samples_leaf`.
- Para todos los modelos se utilizó el parámetro de n-gram range del vectorizador, para encontrar si era más óptimo hacer uso de unigramas, bigramas, o ambos.

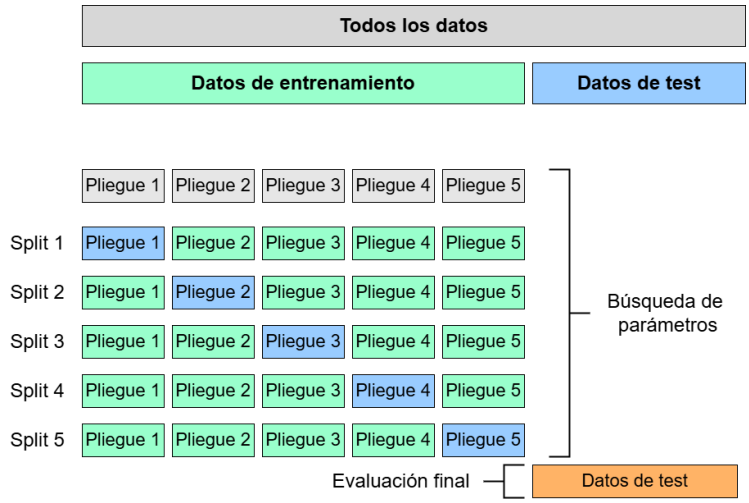

Figura 11: Ejemplo de cross-validation con 5 folds.

La Figura 11 muestra el proceso utilizado para la búsqueda de hiperparámetros óptimos mediante *GridSearchCV*. En primer lugar, el conjunto de datos se divide en un conjunto de entrenamiento y otro de prueba. A continuación, el conjunto de entrenamiento se subdivide en múltiples pliegues (cinco en la imagen, diez para el proyecto) y se aplica validación cruzada: en cada iteración, uno de los pliegues actúa como conjunto de validación mientras que el resto se utiliza para entrenar. Este proceso se repite para cada combinación de parámetros. Finalmente, los mejores hiperparámetros encontrados se validan sobre el conjunto de test reservado, obteniendo así la evaluación final del modelo.

Una vez encontrado el mejor modelo entrenado para cada clasificador, este fue serializado y guardado utilizando `joblib`, junto con la generación automática de dos resultados clave:

1. **Matriz de confusión** sobre el conjunto de validación, utilizando `Yellowbrick`, para representar gráficamente los aciertos y errores por clase.
2. **Informe de clasificación** (`classification_report`), incluyendo métricas como son `precision`, `recall`, `F1-score` y soporte por clase.

Este proceso permitió una buena evaluación y comparación entre los distintos modelos probados, estableciendo así una buena base para la selección final.

8.1.6. Evaluación del rendimiento

Las métricas empleadas para la comparación de modelos incluyeron:

- **Precision, Recall, F1-score y Support** [47]:
  - Precision: Habilidad del clasificador de no etiquetar un dato negativo como positivo. Se denota por la fórmula
    $$\text{Precisión} = \frac{TP}{TP + FP}$$
    donde TP son los verdaderos positivos y FP los falsos positivos.
  - Recall: Habilidad del clasificador de encontrar todas las muestras positivas. Se denota por la fórmula
    $$\text{Recall} = \frac{TP}{TP + FN}$$
    donde TP son los verdaderos positivos y FN los falsos negativos.
  - F1-score: Se puede definir como una media harmónica ponderada de los dos anteriores, siendo su mejor valor 1 y su peor valor 0.
  - Support: Es el número de ocurrencias de cada clase en el conjunto de datos.
- **Matriz de confusión** (visualizada con `yellowbrick`).

8.2. Flujo general del sistema

1. Extracción de datos desde la base de datos.
2. Preprocesamiento del texto.
3. Creación del conjunto final para el entrenamiento en CSV y `pkl`.
4. Entrenamiento de varios modelos de clasificación.
5. Evaluación de los modelos.
6. Documentación y visualización de resultados.

El hecho de que el flujo sea tan modular permite modificar o sustituir componentes del sistema, pudiendo hacerlo muy escalable y mantenible.

9. Gestión de datos e información

En esta sección se va a describir como se ha llevado a cabo toda la gestión de los datos en el proyecto, tanto la obtención como el tratamiento, almacenamiento y uso de la información necesaria para hacer uso de los modelos de aprendizaje automático.

9.1. Obtención de los datos

Como ya se ha mencionado anteriormente los datos se obtuvieron a partir de un volcado masivo de la base de datos de Reddit en un formato comprimido `.zst`, debido a las limitaciones y trabas de la API oficial. Estos comprimidos contenían posts de múltiples subreddits, por lo que se hizo un proceso de filtrado que extraía únicamente las publicaciones pertenecientes al subreddit `r/mentalhealth`, y de todas ellas se quedase sólo con los datos que nos interesaban: título, texto, autor, fecha de publicación, ...

Para ello, se implementó un sistema de descompresión y lectura línea a línea del fichero con `zstandard`, para después realizar un filtrado que validaba tanto la integridad como la relevancia de cada línea. Finalmente, los datos válidos se almacenaban en una base de datos MySQL.

9.2. Creación del dataset

Una vez los datos están dentro de la base de datos se procede a crear el conjunto de datos para realizar el entrenamiento, validación y prueba de los modelos. Para ello se realizó una consulta a la base de datos que extrajo los posts de la primera mitad del año 2024, quedándose con su título y texto (agrupándolo en una única columna), y la `flair` asignado por el usuario o moderador. Se han tenido en cuenta las 7 etiquetas más presentes, al ser el resto muy minoritarias sólo introducirían error en el entrenamiento. Dichas flairs son:

- **Desahogo** *(Venting)*: Publicaciones donde el autor expresa emociones intensas, frustraciones o pensamientos personales sin buscar necesariamente una respuesta o solución por parte de la comunidad.
- **Buenas noticias / Felicidad** *(Good News / Happy)*: Entradas con contenido positivo que comparten logros personales, momentos de alegría o avances significativos en el bienestar del autor.
- **Necesito apoyo** *(Need Support)*: Publicaciones donde el usuario solicita explícitamente ayuda, consejo o acompañamiento emocional ante una situación complicada.
- **Inspiración / Motivación** *(Inspiration / Encouragement)*: Mensajes destinados a animar, inspirar o motivar a otros usuarios, muchas veces con reflexiones positivas, frases alentadoras o experiencias superadas.
- **Recursos** *(Resources)*: Entradas que comparten herramientas útiles, enlaces, documentos, libros, aplicaciones o cualquier otro tipo de recurso práctico relacionado con la salud mental.
- **Tristeza / Pena** *(Sadness / Grief)*: Publicaciones centradas en sentimientos de pérdida, duelo o tristeza profunda, donde el usuario expone su malestar emocional de forma introspectiva.

- **Opinión / Pensamientos** *(Opinion / Thoughts)*: Textos donde los usuarios comparten ideas, reflexiones o perspectivas personales sobre temas relacionados con la salud mental, sin que necesariamente haya una carga emocional directa o petición de ayuda.

El resultado fue almacenado en un fichero `.csv` y `pkl` con dos columnas:

- **text_raw**: Texto en crudo extraído.
- **label_post**: Clase o categoría del flair.

9.3. Preprocesamiento

Al texto en crudo se le aplicó un proceso de limpieza, compuesto por:

- Eliminación de URLs.
- Eliminación de símbolos de puntuación.
- Eliminación de caracteres no ASCII.
- Eliminación de stopwords
- Conversión a minúsculas
- Eliminación de palabras muy cortas
- Lemmatización.

En un principio también se contempló la posibilidad de realizar *stemming* en vez de lematización, pero se descartó por la pérdida de precisión semántica que implica. El *stemming* aplica reglas heurísticas para recortar palabras hasta su raíz, lo cual puede generar formas no reconocibles o ambiguas (por ejemplo, "comput" en lugar de "computer" o "computing"). En cambio, la lematización tiene en cuenta el contexto gramatical y transforma las palabras a su forma canónica o *lema* (por ejemplo, "better" a "good"), lo cual resulta más adecuado para el análisis de texto en dominios sensibles como la salud mental, donde el significado preciso de las palabras es esencial para una correcta interpretación.

9.4. Almacenamiento de modelos

Una vez entrenados los modelos, el mejor modelo para cada clasificador se almacenó utilizando el módulo `joblib` en Python. Esto permite guardar todo el modelo facilitando su reutilización posterior sin necesidad de volver a entrenarlo.

El modelo se guarda con un nombre que identifica tanto al algoritmo utilizado como al conjunto de datos:

`data/trained_model/best_model_{modelo}_{dataset}.joblib`

9.5. Resultados y métricas

Cada ejecución del pipeline de aprendizaje automático genera dos archivos:

- Un archivo `.txt` con el informe de clasificación detallado (`precision`, `recall`, `f1-score`).
- Una matriz de confusión en formato imagen `.png`, generada mediante la librería Yellowbrick.

Estos archivos se almacenaban en el directorio `data/results` y se emplearon para seleccionar el modelo final.

10. Pruebas llevadas a cabo

Para evaluar el rendimiento de los modelos de clasificación automática de flairs en Reddit, se realizó una prueba utilizando el mismo dataset para los tres modelos de clasificacion elegidos: *Naive Bayes* (NB), *Random Forest* (RF) y *Support Vector Machine* (SVM).

10.1. Descripción de la prueba

El conjunto de datos utilizado fue dividido en un 80 % para entrenamiento y un 20 % para validación, aplicando una división estratificada para preservar la proporción original de las distintas clases de flairs. Para cada modelo, se realizó un proceso de ajuste de hiperparámetros mediante validación cruzada utilizando *GridSearchCV* con 10 folds, con el fin de optimizar su desempeño.

Una vez entrenados, los modelos fueron evaluados sobre el conjunto de validación, obteniéndose métricas clave de rendimiento tales como precisión, recall y F1-score, así como las matrices de confusión correspondientes.

10.2. Matrices de confusión

A continuación se presentan las matrices de confusión obtenidas para cada uno de los modelos evaluados:

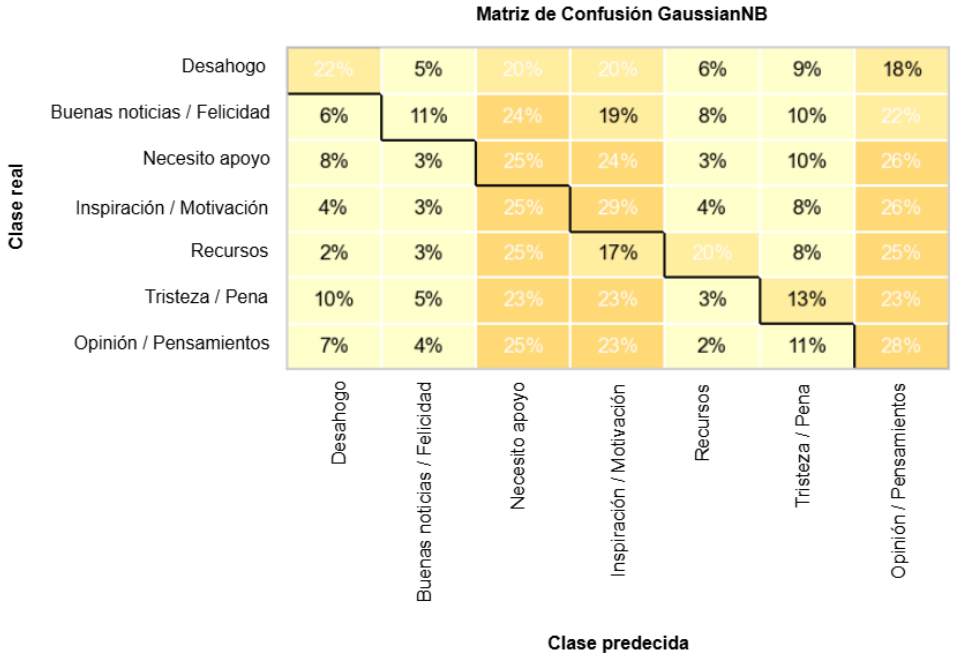

Figura 12: Matriz de confusión del modelo Naive Bayes

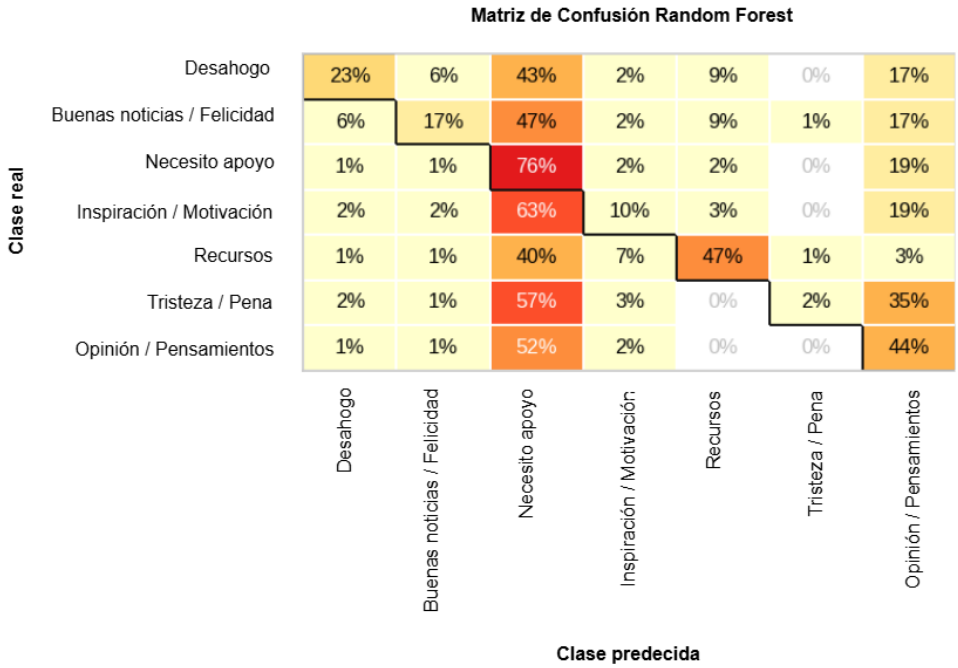

Figura 13: Matriz de confusión del modelo Random Forest

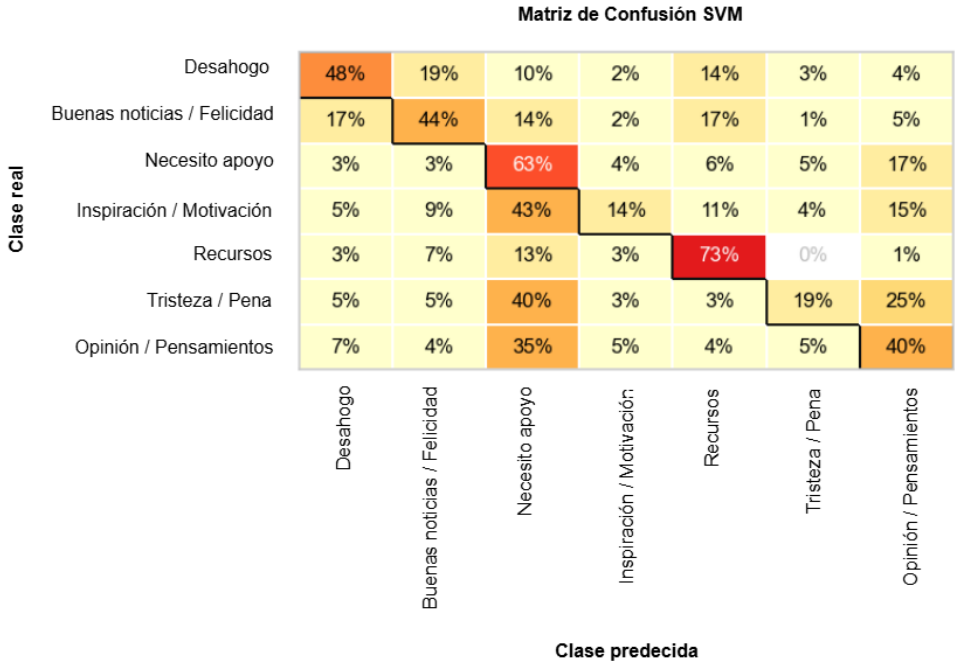

Figura 14: Matriz de confusión del modelo Support Vector Machine

10.3. Informe de clasificación

El informe detallado con las métricas de evaluación para cada modelo se presenta a continuación. Este informe contiene precisión, recall, F1-score y soporte para cada clase, proporcionando un análisis exhaustivo del rendimiento en la clasificación de flairs.

Tabla 13: Informe de clasificación - Gaussian Naive Bayes

Etiqueta	**Precisión**	**Recall**	**F1-score**	**Soporte**
Desahogo	0.06	0.22	0.09	127
Buenas noticias / Felicidad	0.06	0.11	0.08	139
Necesito apoyo	0.46	0.25	0.32	2880
Inspiración / Motivación	0.16	0.29	0.20	822
Recursos	0.10	0.20	0.14	104
Tristeza / Pena	0.06	0.13	0.08	304
Opinión / Pensamientos	0.32	0.28	0.30	1869
Precisión global (accuracy)	0.25 (sobre 6245 muestras)			
Promedio macro	0.18	0.21	0.17	6245
Promedio ponderado	0.34	0.25	0.28	6245

Tabla 14: Informe de clasificación - SVM

Etiqueta	**Precisión**	**Recall**	**F1-score**	**Soporte**
Desahogo	0.18	0.48	0.26	127
Buenas noticias / Felicidad	0.18	0.44	0.26	139
Necesito apoyo	0.61	0.63	0.62	2880
Inspiración / Motivación	0.33	0.14	0.19	822
Recursos	0.17	0.73	0.27	104
Tristeza / Pena	0.18	0.19	0.18	304
Opinión / Pensamientos	0.52	0.40	0.45	1869
Precisión global (accuracy)	0.47 (sobre 6245 muestras)			
Promedio macro	0.31	0.43	0.32	6245
Promedio ponderado	0.50	0.47	0.47	6245

Tabla 15: Informe de clasificación - Random Forest

Etiqueta	Precisión	Recall	F1-score	Soporte
Desahogo	0.28	0.23	0.25	127
Buenas noticias / Felicidad	0.32	0.17	0.22	139
Necesito apoyo	0.55	0.76	0.63	2880
Inspiración / Motivación	0.44	0.10	0.17	822
Recursos	0.31	0.47	0.37	104
Tristeza / Pena	0.26	0.02	0.04	304
Opinión / Pensamientos	0.49	0.44	0.46	1869
Precisión global (accuracy)	0.51 (sobre 6245 muestras)			
Promedio macro	0.38	0.31	0.31	6245
Promedio ponderado	0.49	0.51	0.47	6245

10.4. Análisis de Resultados

10.4.1. Resumen global de rendimiento

Tabla 16: Comparación global entre modelos

Modelo	Accuracy	Macro F1-score	Weighted F1-score
Gaussian Naive Bayes	0.25	0.17	0.28
Support Vector Machine	0.47	0.32	0.47
Random Forest	**0.51**	0.31	**0.47**

`Random Forest` presenta el mejor rendimiento global en términos de *accuracy* (51 %), mientras que `SVM` obtiene resultados muy similares en F1 ponderado. Por el contrario, `Gaussian Naive Bayes` ofrece un rendimiento sustancialmente inferior. Esto nos indica que en este caso los modelos de RF y SVM son capaces de trabajar de una mejor forma con datasets con clases desbalanceadas.

10.4.2. Rendimiento por clase

- **Need Support**: Clase mayoritaria. Todos los modelos consiguen identificarla razonablemente bien, en especial `Random Forest` (F1 = 0.63).
- **Venting** y **Good News / Happy**: Son clases minoritarias con mejor comportamiento en `SVM`, especialmente en términos de recall.
- **Inspiration / Encouragement** y **Sadness / Grief**: Todas las técnicas presentan dificultades para capturar patrones en estas clases.
- **Resources**: `SVM` logra un recall notable (73 %), lo que indica que es eficaz identificando esta clase concreta.
- **Opinion / Thoughts**: Tanto `SVM` como `RF` ofrecen F1 superiores a 0.45, lo que refuerza su utilidad como modelos base.

10.4.3. Observaciones y líneas de mejora

- **Naive Bayes** no se adapta bien al contexto del problema, probablemente por la independencia condicional que asume.
- **SVM** demuestra solidez en clases minoritarias y generalización aceptable.
- **Random Forest** destaca por su capacidad para modelar relaciones no lineales, especialmente en clases frecuentes.

A partir de estos resultados, se plantean posibles mejoras:

- Aplicación de técnicas de rebalanceo como `SMOTE` o `class weighting`.
- Experimentación con nuevos clasificadores como `Logistic Regression`, `XGBoost` o arquitecturas basadas en `Transformers` (*e.g.* Bidirectional Encoder Representations from Transformers *BERT* [48]).
- Análisis de errores detallado clase a clase para afinar el preprocesado y la ingeniería de características.

11. Manual de Usuario

11.1. Introducción

En este manual se ayuda a los usuarios técnicos a utilizar el proyecto basado en la automatización de la clasificación de flairs en publicaciones de reddit, en concreto bajo el subreddit `r/mentalhealth` mediante técnicas de procesamiento del lenguaje natural y aprendizaje automático. Se explicac como instalar el entorno y ejecutar los scripts para el procesamiento del volcado de Reddit, para la generación de los datasets y su procesamiento y para el entrenamiento de los modelos de clasificación.

11.2. Requisitos mínimos

Software:

- Python 3.12 o superior

Hardware:

- GPU con al menos 6 GB de memoria dedicada
- Memoria RAM mínima de 32 GB
- Procesador de 64 bits
- Espacio en disco libre mínimo 20 GB (un mes actual del volcado de Reddit está pesando sobre unos 16 GB).

11.3. Instalación del entorno

A continuación se indican los pasos para instalar y configurar el entorno Python necesario para ejecutar los scripts de procesamiento y ML.

11.3.1. Entorno Python

Preparación del entorno de Python donde se realizarán las tareas de ingesta de datos, preprocesamiento, entrenamiento y clasificación:

```
# Cambiar al directorio raíz del proyecto
cd <ruta_del_proyecto>

# Crear un entorno virtual para aislar las dependencias
python -m venv venv

# Activar el entorno virtual
# Windows:
venv\Scripts\activate

# macOS/Linux:
source venv/bin/activate

# Instalar las dependencias del proyecto
pip install -r requirements.txt
```

11.4. Manual del Desarrollador

Aquí se explican tanto los scripts del proyecto como su ejecución, así como la estructura de carpetas de todo el proyecto.

11.4.1. Descripción de Scripts

- **extract_reddit_zst_to_db.py**: Procesa archivos comprimidos `.zst` de Pushshift y los filtra según diversos criterios como son fecha, subreddit y profundidad de comentarios, quedándose únicamente con las columnas que interesan. Inserta los resultados en una base de datos relacional MySQL.
- **dataset_creator.py**: Extrae datos desde la base de datos y genera datasets etiquetados por flairs. Guarda los datos en formato `.pkl` y `.csv`.
- **text_preprocessor.py**: Preprocesa texto eliminando URLs, puntuación, palabras cortas, stopwords, y lematiza las palabras. Se puede tanto ejecutar como script como importar como módulo, para un procesamiento del dataset en crudo y generación y guardado del dataset `limpio` se ejecuta como script, mientras que para proprocesar una entrada del usuario para realizar una clasificación se importa como módulo.
- **ml_combined.py**: Entrena y evalúa modelos de clasificación (Naive Bayes, Random Forest, SVM) con GridSearchCV. Permite seleccionar el modelo al ejecutar desde CLI.

11.4.2. Ejecución de Scripts

A continuación se muestra la ayuda de los scripts y ejemplos de uso con sus argumentos:

extract_reddit_zst_to_db.py

Uso:

python extract_reddit_zst_to_db.py <input_folder> [opciones]

Opciones:

––output	Carpeta de salida para archivos procesados.
––working	Carpeta de archivos temporales (default: "temp_files").
––field	Campo de filtrado (default: "subreddit").
––value	Valor del campo a comparar (default: "mentalhealth").
––processes	Número de procesos paralelos (default: 10).
––start_date	Fecha inicial (YYYY−MM).
––end_date	Fecha final (YYYY−MM).
––database	Nombre de la base de datos (default: "reddit_bc").
––comment_depth	Profundidad de comentarios (−1 sin comentarios).

Como podemos observar se ha dejado la posibilidad de descargar comentarios hasta diferente nivel de los posts. No se ha podido abordar este tema por dificultades técnicas en la descarga de los mismos al obtener errores 429 al hacer la descarga y ser una descarga extremadamente lenta. Hablamos de que un mes pasaba de tardar unos 8-10 minutos en ejecutarse sin hacer la descarga de los comentarios a ejecutarse en más de 3 horas.

dataset_creator.py

Uso:

python dataset_creator.py --database <nombre_bd> --table <tabla> --start_date <YYYY-MM> --end_date <YYYY-MM> --dataset_name <nombre_dataset>

Parámetros:

--database	Nombre de la base de datos (obligatorio).
--table	Nombre de la tabla con los datos de Reddit (obligatorio).
--start_date	Fecha de inicio del filtrado (formato YYYY-MM).
--end_date	Fecha de fin del filtrado (formato YYYY-MM).
--dataset_name	Nombre del dataset a generar (por defecto: dataset_flairs).

Este script extrae publicaciones de Reddit desde una base de datos relacional, filtradas por flairs y rango de fechas, y genera un dataset etiquetado en formato CSV y PKL.

text_preprocessor.py

Uso:

python text_preprocessor.py --input_pkl <archivo.pkl> [opciones]

Opciones:

--output_dir	Carpeta destino de salida.
--min_word_length	Longitud mínima de palabras (default: 3).
--stopwords_language	Idioma para stopwords (default: "english").
--custom_stopwords	Archivo de stopwords personalizados.

ml_combined.py

Uso:

python ml_combined.py --dataset_path <dataset_clean.csv> --model <nb|rf|svm>

11.4.3. Estructura del Proyecto

La estructura del proyecto es modular por carpetas. Árbol de directorios:

```
|-- DB
|   |-- connect.py
|   |-- __init__.py
|
|-- ingest
|   |-- extract_reddit_zst_to_db.py
|   |-- __init__.py
|
|-- ml
|   |-- dataset_creator.py
|   |-- ml_combined.py
|   |-- predict_text.py
|   |-- text_preprocessor.py
|
|-- models
|   |-- Comment.py
|   |-- Submission.py
|   |-- __init__.py
|
|-- utils
|   |-- file_util.py
|   |-- logger.py
|   |-- process.py
|   |-- progress.py
|   |-- __init__.py
|
|-- data
|   |-- datasets
|   |   |-- csv
|   |   |-- pickle
|   |
|   |-- results
|   |-- trained_model
|
|-- logs
|
|-- temp_files
|-- requirements.txt
```

Descripción de carpetas:

- `DB/`: Contiene las funciones necesarias para estableces y gestionar la conexión con la base de datos MySQL a través de SQLAlchemy.
- `ingest/`: Incluye los scripts del proceso de *ingesta* de datos. Permite descomprimir comprimidos `.zst` del volcado de Reddit, insertando los datos relevantes en la base de datos relacional.

- `ml/`: Agrupa los módulos del pipeline de aprendizaje automático:
  - `dataset_creator.py`: Extrae datos desde la base de datos y los transforma en datasets etiquetados por flair.
  - `text_preprocessor.py`: Aplica una limpieza y transformación de texto (lemmatización, stopwords, etc.).
  - `ml_combined.py`: Entrena y evalúa clasificadores como Naive Bayes, SVM y RF sobre los datos procesados.
  - `predict_text.py`: Permite realizar predicciones de flair sobre nuevos textos con un modelo específico.
- `models/`: Define entidades del dominio como clases Python. Representan objetos como publicaciones y comentarios en Reddit, proporcionando una estructura para manipular los datos en Python durante la *ingesta*.
- `utils/`: Conjunto de funciones auxiliares reutilizables en el proyecto. Incluye herramientas para logging, manejo de archivos, gestión del progreso y procesamiento paralelo.
- `data/`: Almacena las salidas derivadas de la ejecución de los scripts de procesamiento y entrenamiento de modelos. Se subdivide en:
  - `datasets/csv/`: Contiene versiones en formato CSV de los datasets crudos y preprocesados, sirve sobre todo para hacer un análisis visual.
  - `datasets/pickle/`: Incluye los mismos datasets pero serializados en formato `.pkl` para facilitar la carga eficiente desde Python.
  - `results/`: Guarda los reportes del entrenamiento (precisión, recall, F1) y las matrices de confusión de los modelos entrenados.
  - `trained_model/`: Contiene los modelos entrenados serializados en formato `.joblib`, listos para ser usados en predicciones o evaluaciones.
- `logs/`: Carpeta destinada al almacenamiento de logs de ejecución generados durante la ejecución del proceso de ingesta de datos.
- `temp_files/`: Incluye indicadores de progreso e información adicional durante el proceso de ingesta de datos.

12. Principales aportaciones

El análisis de datos a gran escala de plataformas como Reddit supone un desafío computacional y metodológico. El procesamiento eficaz de los datos requiere una arquitectura flexible, modular y bien organizada.

Este proyecto presenta una solución compuesta por **múltiples pipelines especializados**, que nos ha permitido abordar todo el ciclo de vida de un dato, desde la extracción en crudo del dump, hasta su uso en las tareas de predicción.

Las principales aportaciones del proyecto, alineadas con los objetivos propuestos, son las siguientes:

- **Cumplimiento del Objetivo específico 1: Recolección y almacenamiento de datos.** Se diseñó e implementó un sistema robusto para extraer posts de Reddit, concretamente del subreddit `r/mentalhealth` (aunque extensible a otros). El sistema procesa los campos de interés y almacena la información en una base de datos relacional MySQL mediante SQLAlchemy, facilitando su posterior análisis y manipulación.
- **Cumplimiento del Objetivo específico 2: Creación del dataset para el entrenamiento.** Se desarrolló un pipeline dedicado a seleccionar y construir un subconjunto de datos representativo, partiendo de los datos almacenados, asegurando la calidad y relevancia del conjunto para el entrenamiento de modelos de aprendizaje automático.
- **Cumplimiento del Objetivo específico 3: Preprocesamiento y depuración del lenguaje natural.** Se implementaron técnicas de procesamiento de lenguaje natural, para transformar el texto sin procesar en un texto limpio listo para vectorizar.
- **Cumplimiento del Objetivo específico 4: Entrenamiento y evaluación de modelos de clasificación.** Se desarrollaron y compararon diversos algoritmos supervisados, como son Gaussian Naive Bayes, Support Vector Machines y Random Forest, para predecir automáticamente el flair más adecuado para cada publicación. Se evalua el rendimiento de los modelos mediante métricas estándar y se guardan los modelos entrenados para su reutilización.
- **Arquitectura modular y multi-pipeline.** Se estructuró el proyecto en múltiples pipelines especializados, facilitando la mantenibilidad, escalabilidad y reutilización del código.
- **Persistencia y trazabilidad.** Se implementó el almacenamiento sistemático de resultados intermedios y finales en formatos `.csv`, `.pkl` y `.joblib`, facilitando la trazabilidad, reproducibilidad y análisis posteriores.
- **Estructura del proyecto clara y documentada.** La organización del código y documentación facilita tanto la comprensión, como la extensión y el mantenimiento del sistema, separando las responsabilidades en módulos específicos.

De esta manera, el proyecto ha cumplido con su objetivo principal: *facilitar el análisis y clasificación automática de datos de Reddit mediante una arquitectura modular basada en aprendizaje automático.* La solución planteada permite acelerar el proceso de etiquetado de los posts manteniendo iertos niveles de calidad.

13. Conclusiones

Se presentan las conclusiones desde un punto de vista técnico y personal derivadas del desarrollo del proyecto.

13.1. Aspectos técnicos

El proyecto ha alcanzado sus objetivos principales, superando las dificultades que surgieron durante la ejecución del mismo, muchas de ellas motivadas por la falta de conocimiento acerca de las tecnologías.

La incorporación del aprendizaje automático a la clasificación automática de flairs aporta en parte un valor añadido, ya que permite acelerar los procesos manteniendo cierta cantidad de los resultados. Quizá no permita un reemplazo total de los moderadores que puedan chequear los flairs, pero si es una muy buena ayuda inicial.

También el hecho de utilizar CRISP-DM permitió un avance ordenado en la ejecución del proyecto, que se ajustó a las necesidades y retos encontrados.

13.2. Reflexiones personales

A título personal, este proyecto ha supuesto tanto un gran desafío como una oportunidad de crecimiento profesional, ya que me ha permitido tomar contacto con áreas poco exploradas como el el machine learning.

La gestión integral del proyecto han potenciado mi capacidad para abordar problemas técnicos de manera efectiva.

Los retos y problemas encontrados han sido enriquecedores para fortalecer mi perfil profesional, dándome la oportunidad de mejorar en el análisis y resolución de problemas así como la adaptación a nuevas herramientas.

En definitiva, este trabajo ha sido muy positivo para mi persona, aportándome conocimientos y competencias fundamentales para futuros desarrollos y proyectos.

14. Vías de trabajo futuro

Aquí se plantena posibles líneas de desarrollo y ampliaciones que podrían estar basadas en este proyecto:

- **Optimización y diversificación de modelos de aprendizaje automático:**
  Se podrían explorar otras técnicas y algoritmos para realizar la clasificación, como podrían ser redes neuronales o métodos de aprendizaje por refuerzo, que podría permitir mejorar tanto la precisión como la eficiencia. Puede probarse a sustituir el TF-IDF por embeddings basadas en modelos preentrenados (como BERT), y probar con arquitecturas basadas en Deep Learning (redes neuronales tipo LSTM).

- **Optimización y actualización continua del dataset:**
  Mantener y ampliar el conjunto de datos mediante la incorporación de nuevos posts y flairs que aparezcan en el subreddit r/mentalhealth, para garantizar que el modelo se mantenga actualizado frente a cambios en la comunidad.

- **Integración y despliegue definitivo en la plataforma Reddit:**
  La consecución del proyecto sería implementar un sistema de clasificación automática de flairs en tiempo real dentro del entorno de Reddit, ya sea mediante bots o extensiones, mejorando así la experiencia de los usuarios y de los moderadores. Para esto se debería validar y ajustar el modelo en escenarios reales, asegurando que es escalable, eficiente y adaptable a la evolución de los subreddits.

- **Monitoreo y mantenimiento continuo:**
  Deería de establecerse un sistema o forma de monitoreo que permita detectar desviaciones en el rendimiento del modelo después de ser desplegado, permitiendo también automatizar procesos de reentranamiento y actualización de forma que se mantenga su precisión a lo largo del tiempo.

15. Referencias

[1] Ajin S. *The Digital Heartbeat: How Social Media Shapes Emotional Expression and Connection — reflections.live*. https://reflections.live/articles/14591/the-digital-heartbeat-how-social-media-shapes-emotional-expression-and-connection-article-by-ajin-s-20636-m7pzv3e9.html. Accedido el 12-06-2025.

[2] Alexey Medvedev, Renaud Lambiotte y Jean-Charles Delvenne. «The Anatomy of Reddit: An Overview of Academic Research». En: mayo de 2019, págs. 183-204. ISBN: 978-981-13-6240-8. DOI: 10.1007/978-3-030-14683-2_9.

[3] *Machine Learning in Forecasting vs. Traditional Methods — stxnext.com*. https://www.stxnext.com/blog/machine-learning-in-forecasting-vs-traditional-forecasting-methods. Accedido el 12-06-2025.

[4] Munmun De Choudhury et al. «Predicting Depression via Social Media». En: *Proceedings of the International AAAI Conference on Web and Social Media* 7.1 (ago. de 2021), págs. 128-137. DOI: 10.1609/icwsm.v7i1.14432. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14432.

[5] Cecilia Lao, Jo Lane y Hanna Suominen. «Analyzing Suicide Risk From Linguistic Features in Social Media: Evaluation Study». En: *JMIR Formative Research* 6.8 (ago. de 2022). ©Cecilia Lao, Jo Lane, Hanna Suominen. Originally published in JMIR Formative Research, e35563. ISSN: 2561-326X. DOI: 10.2196/35563. URL: https://formative.jmir.org/2022/8/e35563/.

[6] Philip Resnik et al. «Beyond LDA: Exploring Supervised Topic Modeling for Depression-Related Language in Twitter». En: *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*. Denver, Colorado: Association for Computational Linguistics, jun. de 2015, págs. 99-107. DOI: 10.3115/v1/W15-1212. URL: https://aclanthology.org/W15-1212/.

[7] Stevie Chancellor et al. «A Taxonomy of Ethical Tensions in Inferring Mental Health States from Social Media». En: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. FAT* '19. Atlanta, GA, USA: Association for Computing Machinery, 2019, págs. 79-88. ISBN: 9781450361255. DOI: 10.1145/3287560.3287587. URL: https://doi.org/10.1145/3287560.3287587.

[8] Badry Mustofa y Wawan Saptomo. «Use of Natural Language Processing in Social Media Text Analysis». En: *Journal of Artificial Intelligence and Engineering Applications (JAIEA)* 4 (feb. de 2025), págs. 1235-1238. DOI: 10.59934/jaiea.v4i2.875.

[9] Serhad Sarica y Jianxi Luo. «Stopwords in Technical Language Processing». En: *CoRR* abs/2006.02633 (2020). arXiv: 2006.02633. URL: https://arxiv.org/abs/2006.02633.

[10] Divya Khyani et al. «An Interpretation of Lemmatization and Stemming in Natural Language Processing». En: *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology* 22 (ene. de 2021), págs. 350-357.

[11] Jose Camacho-Collados y Mohammad Taher Pilehvar. «On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis». En: *arXiv preprint arXiv:1707.01780* (2017).

[12] «TF–IDF». En: *Encyclopedia of Machine Learning*. Ed. por Claude Sammut y Geoffrey I. Webb. Boston, MA: Springer US, 2010, págs. 986-987. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_832. URL: https://doi.org/10.1007/978-0-387-30164-8_832.

[13] Vikramkumar, Vijaykumar B y Trilochan. *Bayes and Naive Bayes Classifier*. 2014. arXiv: 1404.0933 [cs.LG]. URL: https://arxiv.org/abs/1404.0933.

[14] Thorsten Joachims. «Text Categorization with Support Vector Machines». En: *Proc. European Conf. Machine Learning (ECML'98)* (ene. de 1998). DOI: 10.17877/DE290R-5097.

[15] Leo Breiman. «Random Forests». En: *Machine Learning* 45.1 (1 de oct. de 2001), págs. 5-32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[16] Peter Chapman. «CRISP-DM 1.0: Step-by-step data mining guide». En: 2000. URL: https://api.semanticscholar.org/CorpusID:59777418.

[17] *Cross Industry Standard Process for Data Mining - Wikipedia, la enciclopedia libre — es.wikipedia.org*. https://es.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining. [Accessed 21-06-2025].

[18] GeeksforGeeks. *CSV: A Complete Guide to Comma-separated values files*. Ene. de 2024. URL: https://www.geeksforgeeks.org/csv-file-format/.

[19] *pickle — Python object serialization — docs.python.org*. https://docs.python.org/3/library/pickle.html. Accedido el 06-06-2025.

[20] *MySQL — mysql.com*. https://www.mysql.com/. Accedido el 06-06-2025.

[21] *python.org*. https://www.python.org/. Accedido el 06-06-2025.

[22] *pandas - Python Data Analysis Library — pandas.pydata.org*. https://pandas.pydata.org/. Accedido el 06-06-2025.

[23] *Joblib: running Python functions as pipeline jobs - joblib 1.5.1 documentation — joblib.readthedocs.io*. https://joblib.readthedocs.io/en/stable/. Accedido el 06-06-2025.

[24] *SQLAlchemy — sqlalchemy.org*. https://www.sqlalchemy.org/. Accedido el 06-06-2025.

[25] *scikit-learn: machine learning in Python - scikit-learn 1.6.1 documentation — scikit-learn.org*. https://scikit-learn.org/stable/. Accedido el 06-06-2025.

[26] *NLTK :: Natural Language Toolkit — nltk.org*. https://www.nltk.org/. Accedido el 06-06-2025.

[27] *re — Regular expression operations — docs.python.org*. https://docs.python.org/es/3.13/library/re.html. Accedido el 06-06-2025.

[28] *string — Common string operations — docs.python.org*. https://docs.python.org/3/library/string.html. Accedido el 06-06-2025.

[29] *datetime — Basic date and time types — docs.python.org*. https://docs.python.org/3/library/datetime.html. Accedido el 06-06-2025.

[30] *argparse — Parser for command-line options, arguments and subcommands — docs.python.org*. https://docs.python.org/es/3/library/argparse.html. Accedido el 06-06-2025.

[31] *Client Challenge — pypi.org.* https://pypi.org/project/zstandard/. Accedido el 06-06-2025.

[32] *PRAW 7.7.1 documentation — praw.readthedocs.io.* https://praw.readthedocs.io/en/stable/. Accedido el 06-06-2025.

[33] *Yellowbrick: Machine Learning Visualization - Yellowbrick v1.5 documentation — scikit-yb.org.* https://www.scikit-yb.org/en/latest/. Accedido el 06-06-2025.

[34] *logging — Logging facility for Python — docs.python.org.* https://docs.python.org/3/library/logging.html. Accedido el 06-06-2025.

[35] *Client Challenge — pypi.org.* https://pypi.org/project/mysql-connector-python/. Accedido el 12-06-2025.

[36] *Visual Studio Code - Code Editing. Redefined — code.visualstudio.com.* https://code.visualstudio.com/. Accedido el 08-06-2025.

[37] *GitHub · Build and ship software on a single, collaborative platform — github.com.* https://github.com/. Accedido el 08-06-2025.

[38] Google Workspace. *Google Drive: comparte archivos online con almacenamiento seguro en la nube | Google Workspace — workspace.google.com.* https://workspace.google.com/intl/es/products/drive/. Accedido el 08-06-2025.

[39] *Overleaf, Editor de LaTeX online — es.overleaf.com.* https://es.overleaf.com. Accedido el 08-06-2025.

[40] *TeamGantt: The Easiest FREE Gantt Chart Maker Online — teamgantt.com.* https://www.teamgantt.com/. Accedido el 08-06-2025.

[41] *TeamViewer, tu programa de conectividad remota — teamviewer.com.* https://www.teamviewer.com/es. Accedido el 08-06-2025.

[42] *NCRI — pushshift.io.* https://pushshift.io/. Accedido el 08-06-2025.

[43] *¿Qué es el sobreajuste? - Explicación del sobreajuste en machine learning - AWS — aws.amazon.com.* https://aws.amazon.com/es/what-is/overfitting/. Accedido el 09-06-2025.

[44] *Introduction of Holdout Method - GeeksforGeeks — geeksforgeeks.org.* https://www.geeksforgeeks.org/software-engineering/introduction-of-holdout-method/. [Accessed 21-06-2025].

[45] *GridSearchCV — scikit-learn.org.* https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accedido el 10-06-2025.

[46] *3.1. Cross-validation: evaluating estimator performance — scikit-learn.org.* https://scikit-learn.org/stable/modules/cross_validation.html. Accedido el 10-06-2025.

[47] *precision_recall_fscore_support — scikit-learn.org.* https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html. Accedido el 10-06-2025.

[48] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.