

Universidad  
Rey Juan Carlos

Escuela Técnica Superior  
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2023 - 2024

Trabajo Fin de Grado

**AI MENTAL HEALTH: DESARROLLO DE SISTEMA  
DE SOPORTE A LA DECISIÓN PARA LA  
DETECCIÓN DE SEÑALES DE SUICIDIO EN  
TEXTOS**

**Autor** | José Luis Toledano Díaz

**Tutor** | Antonio González Pardo



# Agradecimientos

Empezando esta sección por las personas a las que más debo en todos los años que tengo, quiero agradecer a mis padres y a mi hermano el apoyo constante que me han demostrado siempre. Cada día me han inculcado valores que me definen y moldean para ser mejor persona. No hay espacio suficiente en esta sección ni existen las palabras que puedan comunicar lo que me han dado durante toda mi vida. Le debo todo a mi familia. **Gracias.**

También quiero dar créditos a varios profesionales que me han acompañado en mi educación, pasando muchos de ellos a ser amigos y compañeros. Al profesor José Luis quien me introdujo en el mundo de las matemáticas, un primer paso hacia la ingeniería desde el colegio. A Sol y Julio, dos profesores que supieron ver el potencial escondido de un chico tímido e introvertido, asumiendo papeles más allá de los deberes del profesor. A Fernando por revolucionar mi ser y cambiarme, dándome la oportunidad de abrirme la mente y descubrir un gran potencial. **Gracias.**

En mi etapa universitaria conocí a muchas personas con las que he compartido el camino y compartiré la senda que queda por recorrer. Amigos como Diego, Flavia, Debi, Ismael, Agudo y Marcos me han ayudado a superar esta etapa y juntos hemos compartido muchos momentos especiales. Muchas gracias por estar ahí, espero expresar en este corto espacio todo lo que os debo. **Gracias.**

A mi actual pareja, Andrea, le debo un gran agradecimiento especial por lo buena que es conmigo. Una persona que te escuche y te apoye con la confianza de decir lo que realmente piensa en cada momento es un suceso poco común, por eso, muy preciado. **Gracias.**

Por último, pero no menos importante, quiero expresar mi agradecimiento a mi tutor y guía de este trabajo, Antonio González Pardo, por apoyarme en el proyecto y darme las ideas necesarias para poder desarrollar e ingeniar la herramienta. Además, agradezco a la Universidad Rey Juan Carlos la oportunidad que me ha ofrecido para estudiar y desarrollarme en el campo de la ingeniería del software. **Gracias.**



# Resumen

El presente documento detalla la composición y desarrollo de una aplicación **DSS** (*Decision Support System*) destinada a especialistas en materia de salud mental o a usuarios con inquietudes en el auto-descubrimiento del estado de su psique. Además, permite el desarrollo en investigación de Inteligencia Artificial (AI, por sus siglas en inglés).

Constando de una interfaz de texto mediante consola (p. ej., CMD), la aplicación denominada **AIMentalHealth**, proporciona soporte a usuarios no especializados en la informática, como personal sanitario y a usuarios interesados en el desarrollo; e investigación de Inteligencia Artificial.

El código fuente de este proyecto se distribuye bajo licencia de código abierto Apache 2.0 mediante un repositorio público de GitHub <sup>1</sup>. En dicho repositorio se encuentra documentación adicional respecto al desarrollo y uso de la aplicación. Además, en el mencionado repositorio se almacenan modelos pre-entrenados específicos <sup>2</sup> para el uso de **AIMentalHealth** y el *dataset* original utilizado para el entrenamiento de estos.

## Palabras clave:

- Inteligencia Artificial
- Transformers
- DSS
- Procesamiento Lenguaje Natural
- Análisis de Sentimientos
- Salud Mental

---

<sup>1</sup>Enlace al repositorio: <https://github.com/JLToledano/AIMentalHealthProblems>.

<sup>2</sup>En el caso de los modelos entrenados que sobrepasen el límite de almacenamiento de la plataforma GitHub, se habilitará una carpeta con los ficheros en OneDrive.



# Abstract

The present document details the composition and development of a **DSS** (*Decision Support System*) application aimed at mental health specialists or users interested in self-discovery of their psyche's state. Additionally, it allows for development in Artificial Intelligence (AI) research.

Featuring a text-based console interface (e.g., CMD), the application named **AIMentalHealth** provides support to non-specialized computer users, such as healthcare personnel, and users interested in the development and research of Artificial Intelligence.

The source code of this project is distributed under the Apache 2.0 open-source license through a public GitHub repository <sup>3</sup>. In this repository, additional documentation regarding development and usage of application can be found. Additionally, pre-trained specific models <sup>4</sup> for use of **AIMentalHealth** and original dataset used for their training are stored in mentioned repository.

## Keywords:

- Artificial Intelligence
- Transformers
- DSS
- Natural Language Processing
- Sentiment Analysis
- Mental Health

---

<sup>3</sup>Repository link: <https://github.com/JLToledano/AIMentalHealthProblems>.

<sup>4</sup>In case of trained models that exceed storage limit of GitHub platform, a folder with files will be provided on OneDrive.





# Índice de contenidos

<b>Índice de figuras</b>	<b>XV</b>
<b>Índice de tablas</b>	<b>XVII</b>
<b>Índice de códigos</b>	<b>XIX</b>
<b>Abreviaturas</b>	<b>XXI</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Estructura . . . . .	3
<b>2. Objetivos</b>	<b>4</b>
2.1. Apoyo a especialista sanitario . . . . .	4
2.2. Desarrollo del estudio en IA . . . . .	5
<b>3. Tecnologías, herramientas y metodología</b>	<b>7</b>
3.1. Tecnologías . . . . .	7
3.1.1. Python . . . . .	7
3.1.2. Pandas . . . . .	8
3.1.3. Numpy . . . . .	8
3.1.4. Transfomers . . . . .	9
3.1.5. ScikitLearn . . . . .	9
3.1.6. Rich . . . . .	10
3.2. Herramientas . . . . .	11
3.2.1. Edición de código . . . . .	11
3.2.2. Entorno virtual . . . . .	11
3.2.3. Control de versiones . . . . .	13
3.3. Metodología de trabajo . . . . .	14
<b>4. Descripción del sistema</b>	<b>17</b>
4.1. Motivos del desarrollo . . . . .	17
4.2. Situaciones de uso real . . . . .	18
4.3. Elecciones de diseño . . . . .	19

<b>5. Desarrollo del sistema</b>	<b>21</b>
5.1. Historias de usuario . . . . .	21
5.2. Diseño y arquitectura de la aplicación . . . . .	23
5.3. Implementación . . . . .	25
5.3.1. Aspectos generales . . . . .	25
5.3.2. Historia de usuario 001 . . . . .	25
5.3.3. Historia de usuario 002 . . . . .	26
5.3.4. Historia de usuario 003 . . . . .	27
5.3.5. Historia de usuario 004 . . . . .	28
5.3.6. Historia de usuario 005 . . . . .	30
5.3.7. Historia de usuario 006 . . . . .	32
5.3.8. Historia de usuario 007 . . . . .	33
5.3.9. Historia de usuario 008 . . . . .	34
5.3.10. Historia de usuario 009 . . . . .	35
5.3.11. Historia de usuario 010 . . . . .	36
5.3.12. Historia de usuario 011 . . . . .	37
5.3.13. Historia de usuario 012 . . . . .	37
5.3.14. Historia de usuario 013 . . . . .	38
5.3.15. Historia de usuario 014 . . . . .	38
5.4. Pruebas configuraciones de red neuronal . . . . .	40
<b>6. Conclusiones y trabajos futuros</b>	<b>43</b>
6.1. Conclusiones . . . . .	43
6.2. Trabajos futuros . . . . .	44
<b>Bibliografía</b>	<b>46</b>
<b>Apéndices</b>	<b>49</b>
<b>A. Especificación de requisitos</b>	<b>51</b>
A.1. Introducción . . . . .	51
A.1.1. Propósito . . . . .	51
A.1.2. Alcance . . . . .	51
A.1.3. Definiciones y siglas . . . . .	52
A.1.3.1. Definiciones . . . . .	52
A.1.3.2. Siglas . . . . .	52
A.1.4. Referencias . . . . .	53
A.1.5. Visión global . . . . .	53
A.2. Descripción general . . . . .	53
A.2.1. Perspectiva del producto . . . . .	53
A.2.2. Funciones del producto . . . . .	54
A.2.3. Características del usuario . . . . .	56

A.2.4.	Restricciones . . . . .	57
A.2.5.	Supuestos y dependencias . . . . .	57
A.3.	Requisitos específicos . . . . .	57
A.3.1.	Requisitos funcionales . . . . .	57
A.3.1.1.	Requisito funcional 001 . . . . .	57
A.3.1.2.	Requisito funcional 002 . . . . .	58
A.3.1.3.	Requisito funcional 003 . . . . .	58
A.3.1.4.	Requisito funcional 004 . . . . .	58
A.3.1.5.	Requisito funcional 005 . . . . .	59
A.3.1.6.	Requisito funcional 006 . . . . .	59
A.3.1.7.	Requisito funcional 007 . . . . .	59
A.3.1.8.	Requisito funcional 008 . . . . .	60
A.3.1.9.	Requisito funcional 009 . . . . .	60
A.3.1.10.	Requisito funcional 010 . . . . .	60
A.3.1.11.	Requisito funcional 011 . . . . .	61
A.3.1.12.	Requisito funcional 012 . . . . .	61
A.3.1.13.	Requisito funcional 013 . . . . .	61
A.3.1.14.	Requisito funcional 014 . . . . .	62
A.3.1.15.	Requisito funcional 015 . . . . .	62
A.3.2.	Otros requisitos . . . . .	62
A.3.2.1.	Requisito dominio 001 . . . . .	62
A.3.2.2.	Requisito dominio 002 . . . . .	62
A.3.2.3.	Requisito no funcional 001 . . . . .	63
A.3.2.4.	Requisito no funcional 002 . . . . .	63
A.3.2.5.	Requisito no funcional 003 . . . . .	63
A.3.2.6.	Requisito no funcional 004 . . . . .	64
A.3.2.7.	Requisito no funcional 005 . . . . .	64
A.3.2.8.	Requisito no funcional 006 . . . . .	64
A.3.2.9.	Requisito no funcional 007 . . . . .	64
A.3.2.10.	Requisito no funcional 008 . . . . .	65
<b>B.</b>	<b>Estudio modelos neuronales</b>	<b>67</b>
B.1.	Introducción . . . . .	67
B.2.	Modelos . . . . .	68
B.3.	Entrenamiento . . . . .	68
B.4.	Evaluación . . . . .	68
<b>C.</b>	<b>Evolución del Machine Learning</b>	<b>73</b>
C.1.	Inteligencia Artificial . . . . .	73
C.1.1.	¿Qué es? . . . . .	74
C.1.2.	Machine Learning . . . . .	74
C.1.3.	Paradigmas de aprendizaje . . . . .	75

C.1.4.	Modelo . . . . .	76
C.1.5.	Fundamento matemático. Regresión lineal . . . . .	78
C.1.5.1.	¿Qué es? . . . . .	78
C.1.5.2.	Explicación matemática . . . . .	78
C.1.5.3.	Representación . . . . .	79
C.1.5.4.	Mínimos cuadrados ordinarios . . . . .	80
C.1.5.5.	Teoría matemática sobre funciones . . . . .	81
C.1.5.6.	Descenso del gradiente . . . . .	83
C.2.	Redes neuronales . . . . .	84
C.2.1.	Neurona . . . . .	85
C.2.1.1.	¿Qué es? . . . . .	85
C.2.1.2.	Funcionamiento de la neurona . . . . .	85
C.2.2.	Red neuronal . . . . .	87
C.2.2.1.	Formación . . . . .	87
C.2.2.2.	Funcionamiento red neuronal . . . . .	87
C.2.2.3.	Función de activación . . . . .	88
C.2.3.	Backpropagation . . . . .	90
C.2.3.1.	Historia . . . . .	90
C.2.3.2.	Idea del algoritmo . . . . .	91
C.2.3.3.	Matemática del algoritmo . . . . .	92
C.2.4.	Overfitting y underfitting . . . . .	100
C.2.4.1.	Teoría general . . . . .	100
C.2.4.2.	¿Un modelo bien ajustado siempre generaliza co- rrectamente . . . . .	100
C.2.4.3.	Ejemplos de overfitting . . . . .	102
C.2.4.4.	Detección de overfitting . . . . .	103
C.2.4.5.	Error de planteamiento . . . . .	104
C.2.5.	Métricas . . . . .	105
C.2.5.1.	Tipos de métricas . . . . .	105
C.2.5.2.	Exactitud . . . . .	105
C.2.5.3.	Matriz de confusión . . . . .	106
C.2.5.4.	Curvas ROC . . . . .	108
C.3.	Natural Language Processing . . . . .	109
C.3.1.	¿Qué es Natural Language Processing? . . . . .	109
C.3.1.1.	Codificación . . . . .	110
C.3.1.2.	Compactación . . . . .	110
C.3.2.	Modelos NLP . . . . .	111
C.3.2.1.	Historia . . . . .	111
C.3.2.2.	Análisis de texto . . . . .	112
C.3.2.3.	Redes neuronales recurrentes . . . . .	112
C.3.2.4.	Mecanismos de atención . . . . .	113

C.3.2.5. Transformers . . . . .	115
C.3.3. Clasificación del sentimiento . . . . .	119
C.3.3.1. ¿Qué es? . . . . .	119
C.3.3.2. Evolución de las alternativas . . . . .	119

# Índice de figuras

1.1. Evolución de muertes por suicidio en la ciudad de Madrid.[3]	2
3.1. Posibilidades ofrecidas por la librería Rich [20]	10
3.2. Flujo de trabajo GitFlow.	14
3.3. Tablero Trello del proyecto.	16
5.1. Diagrama secuencia inicio de AIMentalHealth	23
5.2. Diagrama de flujo de AIMentalHealth	24
5.3. AIMentalHealth mensaje de bienvenida	26
5.4. AIMentalHealth datos cargados al inicio	26
5.5. AIMentalHealth menú principal	27
5.6. AIMentalHealth elección incorrecta en el menú principal	27
5.7. AIMentalHealth análisis de mensaje	28
5.8. AIMentalHealth resultado de un frase analizada	28
5.9. AIMentalHealth configuración de división de ficheros por directorios	29
5.10. AIMentalHealth división de ficheros por número indicado por pantalla	29
5.11. AIMentalHealth entrada incorrecta en selección de fichero o directorio	29
5.12. AIMentalHealth selección de despliegue de directorio	30
5.13. AIMentalHealth selección de fichero y carga del modelo	30
5.14. AIMentalHealth porcentaje de datos procesados	31
5.15. AIMentalHealth entrenamiento de red neuronal	31
5.16. AIMentalHealth evaluación de red neuronal	32
5.17. AIMentalHealth selección de tecnología	33
5.18. AIMentalHealth configuración de parámetros	34
5.19. AIMentalHealth confirmación de los cambios en la configuración	34
5.20. AIMentalHealth panel de métricas	35
5.21. AIMentalHealth carga inicial del dataset predeterminado	36
5.22. AIMentalHealth input de nuevo dataset incorrecto	36
5.23. AIMentalHealth carga del nuevo dataset	37
5.24. AIMentalHealth finalización de la ejecución	38
5.25. AIMentalHealth contenido del menú de ayuda opción de ejecución 1	38

5.26. AIMentalHealth contenido del menú de ayuda opción de ejecución 2	38
5.27. AIMentalHealth contenido del menú de ayuda opción de ejecución 3	39
5.28. AIMentalHealth contenido del menú de ayuda opción de ejecución 4	39
5.29. AIMentalHealth contenido del menú de ayuda opción de ejecución 5	39
5.30. AIMentalHealth contenido del menú de ayuda opción de ejecución 6	39
5.31. AIMentalHealth contenido del menú de ayuda opción de ejecución 7	39
 B.1. Información de GPU . . . . .	67
B.2. Información de memoria RAM . . . . .	68
B.3. Información de memoria CPU . . . . .	68
 C.1. Representación gráfica función convexa . . . . .	81
C.2. Representación gráfica función no convexa . . . . .	82
C.3. Zonas con pendiente nula . . . . .	82
C.4. Neurona suma ponderada . . . . .	85
C.5. Pesos asociados a cada dato de entrada . . . . .	86
C.6. Regresión lineal en neurona . . . . .	86
C.7. Sesgo de la neurona . . . . .	87
C.8. Estructura básica red neuronal . . . . .	88
C.9. Función de activación . . . . .	89
C.10. Esquema red neuronal . . . . .	93
C.11. Derivadas parciales red neuronal . . . . .	94
C.12. Funcionamiento matemático de la neurona . . . . .	94
C.13. Gráfica regresión lineal . . . . .	100
C.14. Gráfica regresión lineal con underfitting . . . . .	101
C.15. Gráfica regresión polinomial . . . . .	101
C.16. Reparto conjunto de datos . . . . .	103
C.17. Matriz de confusión . . . . .	106
C.18. Curvas ROC . . . . .	109
C.19. Ejemplo de embedding . . . . .	111
C.20. Funcionamiento red neuronal recurrente . . . . .	113
C.21. Mecanismo de atención . . . . .	115
C.22. Arquitectura transformers . . . . .	116





# Índice de tablas

B.1. Configuraciones modelos de inteligencia artificial . . . . .	69
B.2. Entrenamiento de modelos . . . . .	70
B.3. Evaluación de modelos . . . . .	71
C.1. Ejemplo práctico aprendizaje no supervisado . . . . .	76
C.2. Soluciones aprendizaje no supervisado . . . . .	77
C.3. Clasificación de texto mediante polaridad de palabras . . . . .	119



## Índice de códigos

3.1. Creación entorno Conda. . . . .	12
3.2. Activación entorno Conda. . . . .	12
3.3. Desactivación entorno Conda. . . . .	12
3.4. Listado entornos Conda. . . . .	12
3.5. Instalación de paquetes en Conda. . . . .	12
3.6. Eliminación entorno Conda. . . . .	13
3.7. Exportación e importación de entorno Conda. . . . .	13



# Abreviaturas

- **AI:** Artificial Intelligence.
- **API:** Application Programming Interface.
- **ASCII:** American Standard Code for Information Interchange.
- **AUC:** Area Under the Curve.
- **BSD:** Berkeley Software Distribution.
- **CIAP:** Clasificación Internacional de Atención Primaria.
- **CMD:** Command Prompt.
- **COVID:** Coronavirus Disease 2019.
- **CPU:** Central Processing Unit.
- **DSS:** Decision Support System.
- **ERS:** Elicitation Requirements Specification.
- **FN:** False Negative.
- **FSME:** Fundación Española para la Prevención del Suicidio.
- **FP:** False Positive.
- **GPT:** Generative Pre-trained Transformer.
- **GPU:** Graphics Processing Unit.
- **HU:** Historias de Usuario.
- **IA:** Inteligencia Artificial.
- **IDE:** Integrated Development Environment.
- **KNN:** K-Nearest Neighbors.
- **LLM:** Large Language Models.

- **NLP:** Natural Language Processing.
- **RAM:** Random Access Memory.
- **ROC:** Receiver Operating Characteristic.
- **SO:** Sistema Operativo.
- **TFG:** Trabajo Fin de Grado.
- **TN:** True Negative.
- **TP:** True Positive.
- **TUI:** Text User Interface.
- **XOR:** Exclusive OR.
- **XP:** Extreme Programming.
- **YAML:** YAML Ain't Markup Language.

# 1

## Introducción

En esta sección se abarcan las motivaciones iniciales que propulsaron el desarrollo del proyecto, las necesidades que se pretenden satisfacer y los problemas reales en los que se utilizaría el aplicativo. Además, se incluye la descripción de la estructura y división del presente documento, incluyendo explicaciones del contenido de cada uno de los apartados.

### 1.1. Motivación

En los últimos años, se ha subrayado en la sociedad un problema que afecta a gran parte de la población española. Siendo irrelevante la edad de la persona, desde la pandemia del COVID-19 se ha hecho notable el aumento de casos relacionados con enfermedades de la salud mental. Según la FSME [2], cerca de 4.000 personas se suicidaron en España en el año 2021. Además, más del 25 % de la población asignada a atención primaria presenta algunos de los problemas psicológicos o mentales recogidos en la CIAP-2 (bajo el epígrafe P-Problemas psicológicos).

La salud mental es un tema presente en la sociedad y muy investigado en los últimos años. Estos estudios han demostrado la gran necesidad de especialistas y herramientas dedicadas a este sector. La excesiva demanda de sesiones de atención privada y la poca oferta en el mercado actual provoca una situación de desequilibrio, teniendo como consecuencia listas de espera para el paciente de más de medio año.

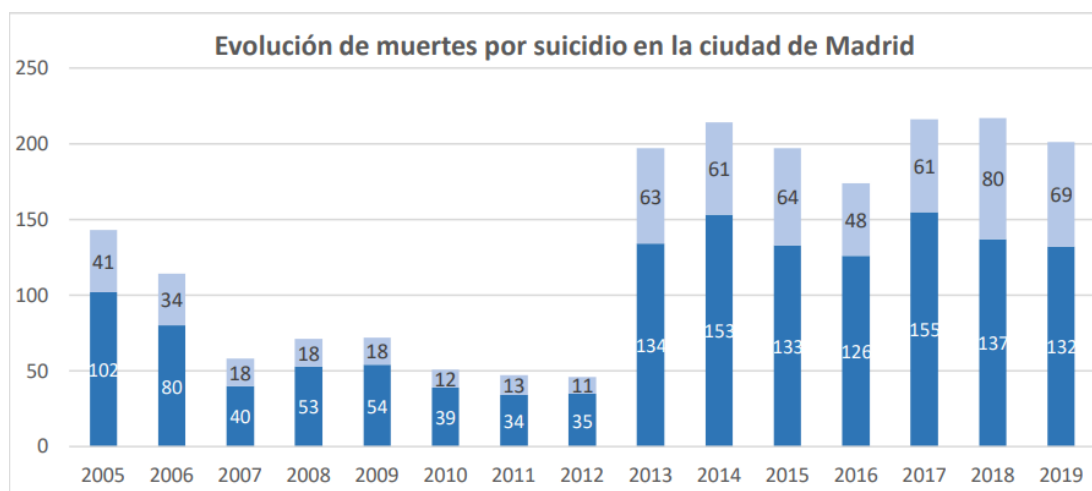


Figura 1.1: Evolución de muertes por suicidio en la ciudad de Madrid.[3]

En otro sector, la informática y el desarrollo de software, se investigan y desarrollan nuevos proyectos dedicados a la evolución de la Inteligencia Artificial. En la última década, han nacido nuevos modelos de redes neuronales, técnicas de entrenamiento y algoritmos de aprendizaje, dando como resultado herramientas dominantes y revolucionarias, teniendo el ejemplo reciente de Chat-GPT4. Consecuencia de esta popularidad; se han creado estudios superiores dedicados al área de Inteligencia Artificial, y el interés demostrado por estudiantes y profesores destaca la necesidad de plataformas y/o herramientas que faciliten el estudio.

Combinando las nuevas tecnologías emergentes de AI junto con las medidas tomadas por el gobierno para enfrentar la problemática del suicidio, es posible multiplicar la eficiencia y efectividad de las estrategias establecidas. Con la implementación de aplicaciones que faciliten o automaticen el diagnóstico, se agiliza el proceso de detección de señales de suicidio, mejorando, por ejemplo, el alcance los objetivos estratégicos 1 y 2 de las *Medidas de prevención del suicidio de la ciudad de Madrid*<sup>1</sup>. En conjunto se consigue que las medidas tomadas lleguen a más población reduciendo así el número de suicidios cometidos y su aumento en los últimos años [ver Figura 1.1].

AIMentalHealth es un DSS destinado a dos posibles usos simultáneos. Su función principal consiste en el análisis de textos escritos para la detección de señales de suicidio, marcando finalmente dicho texto como suicidio o no suicidio. El segundo cometido de la aplicación es servir de herramienta de estudio en el desarrollo, entrenamiento, validación y uso de modelos neuronales basados en la tecnología Transformer.

<sup>1</sup>Disponibles en el *Plan de actuación de prevención del suicidio del ayuntamiento de Madrid 2023-2024*. [4]



## 1.2. Estructura

La estructura del presente documento consiste en un total de seis secciones principales además de los apéndices añadidos y secciones propias de un documento de estas características: agradecimientos, resumen, abstract, índices, abreviaturas y bibliografía. Las secciones principales constituyen la mayor parte del contenido del documento y en ellas se refleja el trabajo completo del TFG.

En la sección 1 se detallan aspectos generales del proyecto desarrollado, explicando las motivaciones que condujeron a la idea desarrollada [subsección 1.1] y la estructura del documento [subsección 1.2]. Posteriormente, en la sección 2 se detallan en profundidad los objetivos concretos que se pretenden cumplir con la aplicación: el apoyo al especialista sanitario [subsección 2.1] y la implementación de una herramienta para impulsar y facilitar el estudio de las AI [subsección 2.2]. Seguidamente, en la sección 3 se detalla cada una de las tecnologías externas utilizadas en la implementación del aplicativo [subsección 3.1], las herramientas utilizadas con este fin [subsección 3.2] y la metodología de trabajo seguida [subsección 3.3].

La siguiente sección constituye uno de los pilares del documento, debido a que detallan los motivos por los cuales se desarrolla esta aplicación y las necesidades que suple [sección 4].

Las dos últimas secciones constituyen la explicación técnica de la propia aplicación [sección 5] y las conclusiones y trabajos futuros para esta [sección 6]. La descripción técnica se divide en subsecciones correspondientes a la definición de las historias de usuario [subsección 5.1], arquitectura de la aplicación [subsección 5.2] y la implementación de cada historia de usuario [subsección 5.3], además de una última subsección destinada a la explicación de las configuraciones probadas para entrenar y evaluar modelos [subsección 5.4]. Por otro lado, la última sección del documento se divide en dos apartados correspondientes a las conclusiones [sección 6.1] y los trabajos futuros [sección 6.2].

El apartado de apéndices lo constituyen tres documentos independientes: la especificación de requisitos [apéndice A], el estudio de modelos neuronales [apéndice B] y la evolución desde la base de la Inteligencia artificial hasta el procesamiento del lenguaje natural [apéndice C]. La especificación técnica de requisitos es un documento redactado formalmente para detallar las características que debe tener la aplicación desarrollada, y el segundo documento consiste en un resumen de los resultados obtenidos por cada entrenamiento y evaluación de modelos, especificando los parámetros de configuración del modelo y el valor obtenido en cada medida. Por último, el tercer documento explica aspectos generales de la Inteligencia Artificial [subsección C.1], continuando con el nacimiento, desarrollo y funcionamiento de las redes neuronales [subsección C.2] y finalizando con la descripción del Análisis de Lenguaje Natural [subsección C.3].

# 2

## Objetivos

Esta sección está dedicada a la explicación de los objetivos principales que se han marcado como metas en el desarrollo de la aplicación AIMentalHealth.

AIMentalHealth abarca dos metas principales: apoyo al especialista sanitario [subsección 2.1] y ser una herramienta de investigación para el campo de la Inteligencia Artificial [subsección 2.2].

### 2.1. Apoyo a especialista sanitario

AIMentalHealth surgió de la idea de querer ayudar a los especialistas sanitarios en área de psiquiatría e inteligencia emocional en sus análisis diarios ofreciendo una herramienta adicional a los *tests* ya existentes. Es interesante el concepto de una aplicación que ofrezca la posibilidad de agilizar el filtrado y análisis de textos escritos por un paciente para reducir los costes en tiempo y esfuerzo que supone un análisis.

Por otra parte, la aplicación está diseñada de tal forma que un paciente puede autoanalizar sus textos sin un especialista presente para después comunicar los resultados. Sin embargo, tal como se refleja en la propia documentación de AIMentalHealth, los resultados mostrados por la aplicación no determinan una sentencia final sin el posterior apoyo de un especialista.

La combinación de ambas opciones da lugar a un abanico de posibilidades para el desarrollo del diagnóstico psiquiátrico, permitiendo al doctor elegir cómo

utilizar la herramienta para adaptarse a las necesidades de cada paciente y al flujo y saturación de trabajo.

### **2.2. Desarrollo del estudio en IA**

Debido al estudio de las redes neuronales complejo por la rápida evolución y el incremento exponencial en la complejidad de los modelos se dificulta la investigación para nuevos desarrolladores, la falta de herramientas simples con las que investigar el entrenamiento y evaluación de modelos simples supone una barrera importante a los nuevos ingenieros interesados en el área de las redes neuronales.

Reutilizando la plataforma creada con el fin de cumplir el primer objetivo, se han expandido los usos de AIMentalHealth para posibilitar la creación de nuevas redes neuronales con distintas configuraciones. De esta forma, se tiene una herramienta útil para entrenar, validar y probar redes neuronales con casos reales de forma unificada y sencilla.

Se estableció este segundo objetivo en la herramienta para ampliar el rango de usuarios a los que les interesaría utilizar la aplicación. Estudiantes de Inteligencia Artificial o iniciados en áreas similares podrían apreciar el valor de la aplicación gracias a su uso amplio e intuitivo para el desarrollo de redes neuronales. De esta forma, se abre una puerta a los iniciados y se reduce la curva de aprendizaje existente para las personas con pocos recursos tecnológicos debido a la oferta de pequeños modelos por parte del aplicativo.



# 3

## Tecnologías, herramientas y metodología

En esta sección, el objetivo es explicar de las tecnologías implicadas en el desarrollo del aplicativo, además de las herramientas y las metodologías utilizadas.

### 3.1. Tecnologías

Al tratarse de una aplicación utilizada por consola en el propio equipo del usuario, las tecnologías utilizadas para su desarrollo consisten en el lenguaje de programación Python y librerías compatibles.

También se han utilizado tecnologías del ámbito del desarrollo de redes neuronales poniendo como ejemplo Transformers.

#### 3.1.1. Python

Python es un lenguaje de programación de alto nivel superando en uso a otros famosos lenguajes de programación como Java [14]. A diferencia de este último, es un lenguaje de programación interpretado; es decir, no necesita de compilación previa a la ejecución del código, además, es un lenguaje de tipado dinámico.

Una propiedad adicional de este lenguaje de programación es el soporte que ofrece a diversos paradigmas de programación, permitiendo al desarrollador decidir con qué paradigma programar, pudiendo usar varios en un mismo progra-

ma. Los paradigmas de programación admitidos por Python son el imperativo, orientado a objetos y el funcional. Gracias a este soporte, se pueden utilizar las características más ventajosas de cada paradigma para solucionar un mismo problema.

Por otro lado, la disponibilidad de una extensa biblioteca estándar, añadiendo un gran número de librerías implementadas con el fin de utilizarlas simultáneamente, permite a los desarrolladores experimentados desarrollar aplicaciones con más sencillez gracias a los recursos existentes y especializados.

Algunas de las características de programación de Python son:

- Los tipos de datos son dinámicos.
- Mezclar tipos incompatibles genera una excepción, por lo que los errores se detectan con facilidad. Esto se puede producir debido al intento de una suma entre una variable *string* y una variable numérica.
- Python contiene características avanzadas de programación, como generadores y comprensiones de listas.

### 3.1.2. Pandas

La biblioteca Pandas [17] se utiliza en Python para la creación y manipulación de datos estructurados en tablas (*dataframes*). Empezando su desarrollo en el año 2008, se tiene como objetivo actual convertir esta librería en la herramienta de manipulación y análisis de datos más flexible disponible en cualquier lenguaje de programación.

Destacando que se desarrolla bajo licencia de código abierto Pandas, se erige en una de las principales opciones ante la manipulación eficiente de datos con indexación integrada. Como añadido, esta herramienta es frecuentemente utilizada para escribir datos construidos en códigos, en ficheros CSV o en las distintos formatos existentes de Microsoft Excel.

Por último, la posibilidad de transformaciones y rotaciones en los *dataframes* junto con operaciones nativas de agrupación, división, filtrado y búsqueda de datos terminan de afianzar la librería como herramienta indispensable en el desarrollo software en Python.

### 3.1.3. Numpy

Publicada bajo los términos de la licencia BSD modificada y creada sobre la base de las bibliotecas previas Numeric y Numarray, la librería Numpy [18] es un proyecto de código abierto que permite la computación numérica con Python.

Esta herramienta incorpora una nueva clase de objetos denominados arrays, consiguiendo grandes ventajas frente a las listas, siendo este último tipo de dato el semejante nativo del lenguaje Python. Un array consiste en una colección ordenada de datos de un mismo tipo. Su procesamiento es más eficiente que el de las listas, permitiendo el ahorro en tiempo de computación en problemas relacionados con vectores y matrices de grandes dimensiones.

### 3.1.4. Transformers

Siendo tecnología esencial para el desarrollo de este trabajo, Transformers [21] es una librería que proporciona herramientas y API para entrenar de forma sencilla modelos preentrenados. Los modelos disponibles son compatibles y admiten la interoperabilidad con las principales tecnologías de desarrollo de inteligencia artificial utilizadas en Python: PyTorch, TensorFlow y JAX.

Gracias al uso de modelos preentrenados, se reduce el costo en tiempo y cómputo necesarios para originar un nuevo modelo sin ninguna base previa. Además, permite a los desarrolladores partir de modelos ya entrenados en alguna tarea general, como puede ser el entendimiento de algún lenguaje o la detección de cierto tipo de objetos, y así dedicar los esfuerzos a la especialización del modelo ante el nuevo problema u objetivo propuesto.

Las principales tareas comunes a las que se destinan los modelos existentes en la plataforma Transformers son:

- Procesamiento del lenguaje natural: modelado del lenguaje, traducción, generación de texto y clasificación de texto.
- Visión Artificial: clasificación de imágenes y detección de objetos.
- Audio: reconocimiento de voz y clasificación de audio.
- Multimodal: extracción de información de documentos escaneados, clasificación de vídeos, respuesta de preguntas en imágenes, entre otros.

### 3.1.5. ScikitLearn

ScikitLearn [19] es una librería utilizada en el área del aprendizaje automático en Python debido a que proporciona diversas herramientas y recursos que facilitan el entrenamiento y evaluación de un modelo.

Entre las herramientas que ofrece, destacan funciones para clasificación, regresión, agrupamiento, reducción de dimensionalidad en el aspecto del modelo, además de funciones de preprocesamiento de datos.

Esta librería está implementada sobre la base de librerías populares en el desarrollo de software en Python como NumPy, SciPy y Matplotlib.



Figura 3.1: Posibilidades ofrecidas por la librería Rich [20]

### 3.1.6. Rich

Rich [20] es un paquete desarrollado para Python y una herramienta que posibilita al desarrollador implementar una TUI compatible con sistemas operativos Linux, macOS y Windows.

Mediante la API ofrecida por esta librería, es posible decorar la consola utilizada con tablas, secciones, colores, barras de progreso; facilitando y añadiendo valor visual a aplicaciones usadas por consola. Como añadido, es posible utilizar este recurso para los mensajes de error y trazas de un aplicativo software estándar para facilitar el trabajo de los desarrolladores.

Observando la imagen 3.1, se puede apreciar el amplio abanico de posibilidades ofertadas por esta herramienta y que aportar valor a un producto software que explote sus opciones.



## 3.2. Herramientas

Para la gestión de dependencias de la aplicación, se ha procedido a utilizar el gestor de entornos virtuales Conda debido a las ventajas que ofrece no instalar las librerías necesarias de forma nativa en el dispositivo utilizado por el usuario. Para la escritura y edición del código fuente que conforma el aplicativo, se ha utilizado el IDE Visual Studio Code.

Por otro lado, se ha utilizado el sistema de control de versiones Git en la plataforma GitHub. Además, complementando el sistema de control de versiones, se ha seguido el flujo de trabajo GitFlow en la creación y gestión de ramas.

### 3.2.1. Edición de código

Utilizando típicamente un IDE para la escritura y edición del código fuente de un producto software, la elección escogida para este proyecto ha sido Visual Studio Code debido a las herramientas y servicios ofrecidos por esta aplicación.

Visual Studio Code [10], de Microsoft, es un editor de código fuente ligero que consiste en una aplicación de escritorio, ofreciendo de forma nativa la instalación de múltiples *plugins*. Esto permite a los desarrolladores adaptar la herramienta a sus necesidades. Se publicó su primera versión en el año 2015 y desde entonces se han mejorado sus funciones nativas e implementado nuevos *plugins* para personalizar el uso de esta herramienta con diferentes lenguajes de programación, control de versiones, calidad de código, entre otros.

La mencionada herramienta está disponible para Windows, macOS y Linux e incluye soporte integrado para JavaScript, TypeScript y Node.js. Por último, es posible descargar extensiones para soportar múltiples lenguajes de programación (por ejemplo Python).

### 3.2.2. Entorno virtual

Un entorno virtual es un espacio dentro de una computadora en la que se descargan los paquetes necesarios con sus respectivas versiones para el desarrollo y ejecución de un proyecto software. La principal ventaja que ofrece es poder tener en un mismo dispositivo el mismo paquete en múltiples versiones para diversos proyectos y sin provocar ningún tipo de conflicto entre ellos.

Para el desarrollo y ejecución del presente proyecto software, se ha escogido como administrador de entorno virtual la herramienta Conda. Es posible utilizar esta herramienta mediante Anaconda o Miniconda, siendo la diferencia entre ambas la instalación simultánea de paquetes por defecto junto con Conda (primer

caso) o solo la descarga de Conda (segundo caso).

Conda es un manejador de paquetes usado en computación científica y ciencia de datos. Está diseñado para proveer de librerías científicas y dependencias en el lenguaje seleccionado, siendo Python en este caso. Es una herramienta de código abierto y es ejecutable en multiplataforma. Como añadido, es posible combinar Conda con sistemas de integración continua como Travis CI para proporcionar soporte a pruebas automatizadas en el proyecto.

A continuación, se especifican los comandos básicos para el uso de esta herramienta en un proyecto Python.

- **Creación entorno virtual:** Creación del entorno virtual sin ninguna librería adicional y especificando la versión del lenguaje Python.

```
1 conda create --name projectname python==pythonversion
```

Código 3.1: Creación entorno Conda.

- **Activar entorno virtual:** Es necesario haber creado el entorno anteriormente.

```
1 conda activate projectname
```

Código 3.2: Activación entorno Conda.

- **Desactivar entorno virtual:** Cerrar entorno virtual activado con anterioridad.

```
1 conda deactivate
```

Código 3.3: Desactivación entorno Conda.

- **Listar entornos:** Mostrar el listado de los entornos Conda existentes.

```
1 conda list
```

Código 3.4: Listado entornos Conda.

- **Instalación de paquetes:** En un entorno Conda activado es posible instalar paquetes con la herramienta pip o con el propio instalador de paquetes de Conda.

```
1 #Usando instalador pip
2 pip install packagename==packageversion
3
4 #Usando instalador Conda
```

```
5 conda install packagename==packageversion
```

Código 3.5: Instalación de paquetes en Conda.

- **Eliminar entorno virtual:** Eliminación de un entorno virtual junto con los paquetes instalados en el mismo.

```
1 conda env remove --name projectname
```

Código 3.6: Eliminación entorno Conda.

- **Exportación/Importación entorno virtual:** Con Conda es posible exportar las características de un entorno virtual ya existente (nombre del entorno, versión de lenguaje de programación y paquetes instalados) a un fichero YAML para poder replicar el entorno en otro dispositivo con este mismo fichero.

```
1 #Exportacion entorno virtual
2 conda env export --name projectname > filename.yml
3
4 #Importacion entorno virtual
5 conda env create -f filename.yml
```

Código 3.7: Exportación e importación de entorno Conda.

### 3.2.3. Control de versiones

El sistema de control de versiones seleccionado y utilizado en el desarrollo de la aplicación ha sido Git, la plataforma de almacenamiento GitHub y el flujo de trabajo GitFlow.

Git [22] es un sistema de control de versiones que permite a los desarrolladores construir un seguimiento de los cambios en un proyecto software. Utilizando una analogía de ramificaciones con una raíz común, este sistema de control de versiones permite el desarrollo en paralelo a varios programadores en múltiples ramas, y los conflictos generados serán resueltos al momento de fusionar dichas ramas. El proyecto software se almacena en la plataforma GitHub, que utiliza Git como sistema de control de versiones, permitiendo el acceso al repositorio en remoto mediante web.

Por último, en la gestión de *commits* y ramas de Git, se ha utilizado la metodología GitFlow [23]. Esta metodología de ramificación se basa en el uso de diferentes tipos de ramas con objetivos específicos. Las ramas básicas existentes son:

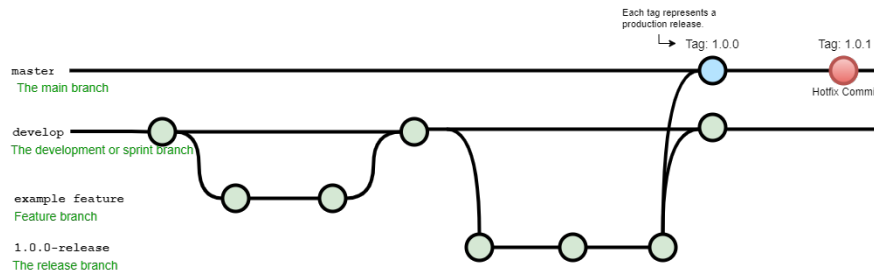


Figura 3.2: Flujo de trabajo GitFlow.

- **Master:** Rama cuyo contenido corresponde a la última versión del proyecto estable.
- **Develop:** Rama cuyo contenido corresponde a los nuevos desarrollos finalizados de los programadores. Se utiliza como paso intermedio a la rama master para integrar nuevas características o correcciones en el proyecto.
- **Feature:** Ramas donde se implementa una nueva funcionalidad o corrección de errores existentes en el proyecto software. Se originan a partir de la rama develop y se vuelven a fusionar con esta al finalizar la tarea.
- **Release:** Ramas utilizadas para preparar una nueva versión del proyecto software marcando con un nuevo *tag*<sup>1</sup> el desarrollo existente. Este tipo de ramas se originan a partir de la rama develop y posteriormente se fusionan en la rama develop y en la rama master.
- **Hotfix:** Ramas cuyo propósito es incorporar la solución a fallos críticos ocurridos en producción. Se crean a partir de la rama master y se fusionan en master y en develop al solucionar el error crítico.

Un ejemplo visual de la organización del control de versiones con GitFlow es apreciable en la figura 3.2. Este flujo se puede llevar a cabo mediante consola de comandos instalando la extensión necesaria para consola o mediante una herramienta de escritorio como GitHub Desktop.

### 3.3. Metodología de trabajo

En los proyectos relacionados con la ingeniería del software, existen múltiples metodologías de trabajo que se pueden seguir en la situación de dirigir y coordinar un equipo o el propio trabajo. Entre las múltiples opciones como la tradicional con la documentación exhaustiva del trabajo o las más vanguardistas denominadas

<sup>1</sup>Referencia a un punto específico en la historia del repositorio, como versiones de lanzamiento o hitos importantes.

metodologías *agile*, se ha escogido estas últimas para el desarrollo de este trabajo de fin de grado.

La motivación principal de escoger una metodología *agile* para el desarrollo ha sido por su enfoque al factor humano y principalmente por su capacidad de respuesta al cambio. Además, estos métodos de trabajo son el futuro y presente de los proyectos software en empresas, por lo que su uso en este trabajo sirve como demostración de la adaptación hecha al futuro laboral.

Concretamente, se ha adaptado el *framework* ágil Scrum, respetando los valores detallados en el Manifiesto Ágil [24]. La adaptación ha consistido en la reducción o eliminación de las dinámicas de roles y trabajos en equipo propuestas debido a que el desarrollo ha sido realizado por una única persona. Sin embargo, se ha utilizado la organización propuesta para las tareas surgidas del proyecto y el trabajo mediante iteraciones incrementales.

Scrum [9] es una metodología ágil desarrollada en la década de 1990 por Jeff Sutherland y Ken Schwaber, pero no es hasta el año 2001 cuando se instaura como componente fundamental en el desarrollo software, debido a la firma del Manifiesto Ágil (en el que participaron ambos autores). La principal característica de esta metodología es la división por roles de los miembros de un equipo (no aplicable en este proyecto) y la organización del trabajo en dos «zonas» principales: *Product Backlog* (lista priorizada de todas las características y tareas del producto) y *Sprint Backlog* (lista de elementos en los que se trabajará en la iteración). El *Sprint Backlog* se divide en tres secciones por las que las tareas a realizar tienen que pasar:

- **TO DO:** Tareas estudiadas y preparadas para ser desarrolladas.
- **DOING:** Tareas en desarrollo.
- **DONE:** Tareas finalizadas.

En el desarrollo del proyecto también se ha utilizado otro *framework agile* denominado Extreme Programming, presentando así los requisitos de la aplicación de una forma cercana al usuario, simple y comprensible.

La pizarra de trabajo se ha implementado con la herramienta Trello [7], como se puede apreciar en la imagen 3.3, se ha estructurado siguiendo la metodología Scrum. Es apreciable en la organización de la pizarra en dos partes: *Product Backlog* y *Sprint Backlog* (este último dividido ya en sus columnas TO DO, DOING y DONE). Como añadido, se tiene una columna de información en la que se detallan aspectos generales del proyecto, como el nombre de este y estándares a seguir.

Por último, se ha utilizado un sistema de estimación para las historias de usuario creadas en el proyecto denominado *planning poker* consistente en asignar,

### 3.3. Metodología de trabajo

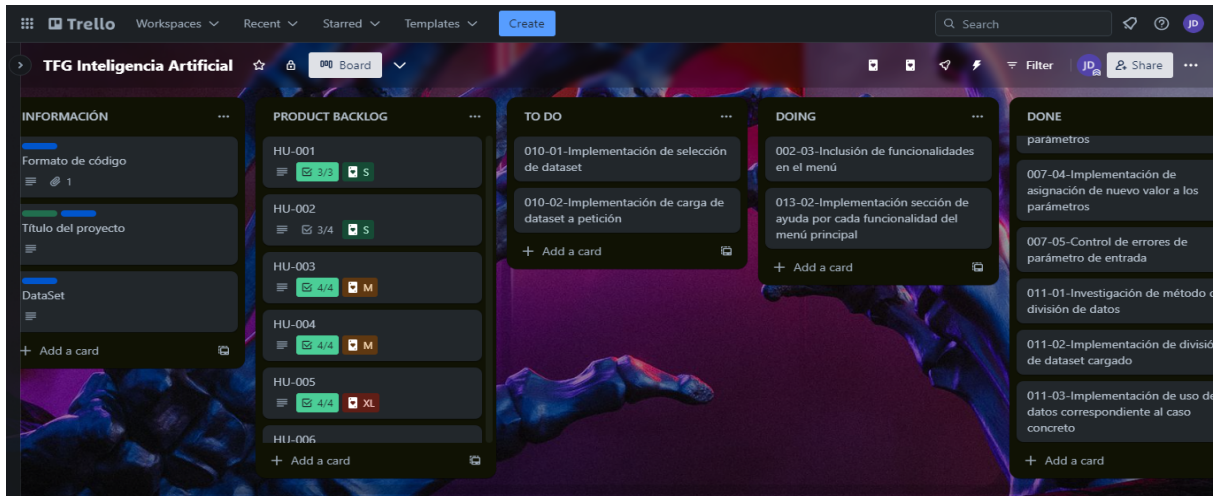


Figura 3.3: Tablero Trello del proyecto.

en este caso, una talla de ropa (XS, S, M, L, XL) a cada historia para así apreciar la cantidad de tiempo y trabajo necesarios para cada una de ellas. El resultado es apreciable en el tablero Trello (imagen 3.3) en la columna de *Product Backlog*, en el centro de cada HU.

# 4

## Descripción del sistema

Esta sección incluye información respecto a los fundamentos seguidos, estudiados y desarrollados para la creación del aplicativo. El estudio de concepto consiste en el por qué de la aplicación, casos reales de posible uso y elecciones de diseño.

### 4.1. Motivos del desarrollo

Reflexionando sobre las principales razones por las que se ha desarrollado el sistema presentado se reflejan las necesidades de la sociedad actual que satisface. Destaca el aumento de las listas de espera para tratamiento psicológico, el aumento de casos de suicidio y la integración mayor de la IA.

El uso de la aplicación por la comunidad especialista en tratamiento psicológico permite el ahorro del tiempo dedicado al análisis del paciente debido a que se pueden filtrar los mensajes escritos por el sujeto en un corto periodo de tiempo. El proceso de lectura por parte del especialista consumiría un tiempo excesivo provocando una demora en la atención, causa del aumento de las listas de espera.

Otra causa que desemboca en la primera motivación del desarrollo es el aumento de casos de suicidio en la sociedad moderna. Los casos no tratados de depresión y enfermedades mentales con síntomas similares pueden desembocar en un alto porcentaje en suicidio o intento de suicidio. La aplicación desarrollada permite a los propios usuarios autodiagnosticarse (en cierto grado debido

a la falta de formación en la mayoría de los casos). Sin embargo, el aplicativo puede ofrecer las primeras señales de un posible trastorno depresivo existente en los mensajes escritos por el usuario en diversas plataformas e introducidos en la herramienta para su análisis. De esta forma el usuario puede observar de forma aproximada si realmente está sufriendo el episodio de depresión.

Por último, una herramienta sencilla que permita la manipulación de parámetros de configuración (hiperparámetros) de los modelos neuronales es llamativa debido al área en el que se enfoca su funcionamiento, la IA. Los ingenieros y en general la sociedad tienen interés por los nuevos sistemas que utilizan Inteligencia Artificial en su desarrollo y la implementación de una herramienta que permita de forma sencilla su uso y estudio es aceptada con deseo.

## 4.2. Situaciones de uso real

La aplicación desarrollada ofrece a sus múltiples perfiles de usuario una extensa variedad de funcionalidades que son posibles de utilizar en diversas situaciones. Estas variaciones difieren en la temática de uso y los objetivos con los que se utiliza la aplicación.

La aplicación se puede emplear de diversas formas en función de los objetivos que persiga el usuario. En la siguiente lista se recogen algunos de los más relevantes:

- **Autoanálisis:** Un usuario en su privacidad puede analizar los mensajes que escribe para saber si el sentimiento demostrado en los escritos corresponde a un posible suicidio. Con este objetivo se aportan modelos preentrenados para evitar al usuario la necesidad de entrenar uno.
- **Diagnóstico:** Un usuario no especializado en informática dedicado al sector del análisis psicológico puede usar la aplicación como herramienta de diagnóstico con sus pacientes. De esta manera se filtran masivamente los mensajes que no muestren señales de suicidio pudiendo centrar el esfuerzo y tiempo en los mensajes relevantes. Además, puede entrenar nuevos modelos para personalizarlos a cada paciente si así lo desea.
- **Estudio de NLP:** Un estudiante de ingeniería del software o de alguna rama del conocimiento similar interesado en la AI, concretamente en el análisis del lenguaje natural, tiene la posibilidad de utilizar la aplicación para estudiar cómo funciona esta técnica y en qué consiste en todas sus fases.
- **Estudio de configuraciones:** La aplicación dispone de la opción de personalizar los hiperparámetros que influyen en el entrenamiento y evaluación



de modelos para posibilitar el estudio de la mejor configuración con un modelo y dataset determinados.

- **Desarrollo de modelos:** Estudiantes e investigadores tienen la opción de investigar con datasets de diferentes temáticas el desarrollo de modelos de análisis de lenguaje natural y realizar clasificación binaria.

### 4.3. Elecciones de diseño

El objetivo de la presente sección es justificar de forma concisa y breve las principales decisiones de diseño relevantes en el producto final desarrollado.

En primer lugar, el "entorno gráfico" escogido para la herramienta ha sido consola de comandos debido a la sensación de seguridad que aporta al usuario. Un usuario con conocimientos en ingeniería informática es capaz de reconocer los riesgos que puede sufrir un dispositivo mediante la manipulación con comandos por parte de terceros. Sin embargo, un usuario no informado tiene la sensación de seguridad al no estar publicando datos fuera del entorno de su ordenador como podría ocurrir en una página web. Sin las adecuadas medidas de seguridad ambos casos son igual de vulnerables pero utilizando consola los usuarios sin conocimiento informático tienen sensación de seguridad beneficiosa para que se empiece a utilizar la aplicación.

Por otro lado, las opciones disponibles en la aplicación están colocadas de tal forma que las opciones más complejas de entender estén en la parte central del listado debido a que un usuario lee en una lista corta el principio y el final primero, donde se sitúan las opciones de análisis de mensaje y ayuda/salir respectivamente.



# 5

## Desarrollo del sistema

La presente sección contiene información detallada del desarrollo informático realizado, demostrando el alcance de los objetivos establecidos. En primer lugar se describen las historias de usuario creadas y seguidas para el desarrollo de la aplicación. Posteriormente se detalla la implementación para cada HU y en último lugar la verificación del software y su despliegue.

El código fuente resultado del desarrollo se encuentra disponible de forma pública en un repositorio GitHub.

### 5.1. Historias de usuario

Partiendo de los objetivos establecidos en la sección 2 y junto con la especificación de requisitos disponible en el apéndice A se procede a definir y detallar las historias de usuario que han desembocado en la creación y desarrollo del aplicativo.

Siguiendo **eXtreme Programming** (XP) y su definición de **historia de usuario** por Kent Beck en la que se indica que una HU consiste en una descripción breve y simple de una característica, funcionalidad o requisito del sistema desde la perspectiva del usuario o cliente; se han estructurado siguiendo el esquema propuesto:

Como *[rol del usuario]*, quiero *[hacer algo]* para *[lograr algún objetivo o beneficio]*.

La codificación de las historias de usuario consiste en un número identificativo de tres dígitos precedido de la partícula HU: **HU-XXX**. Además, en el desarrollo de cada una de las historias de usuario se especifica qué requisitos abarca y qué se ha implementado para cumplirlos.

- **HU-001**: Como usuario, quiero un mensaje o señal para saber que se ha iniciado la ejecución de la aplicación.
- **HU-002**: Como usuario, quiero un menú principal para acceder rápidamente a las funcionalidades.
- **HU-003**: Como usuario no especializado, quiero analizar textos evitando configuraciones para comprender lo que realizo y sus resultados.
- **HU-004**: Como usuario, quiero poder utilizar modelos ya existentes para clasificar los textos.
- **HU-005**: Como usuario especializado, quiero entrenar un modelo y evaluarlo cuando requiera para poder determinar la factibilidad de un modelo.
- **HU-006**: Como usuario especializado, quiero elegir entre varias tecnologías para construir mi modelo.
- **HU-007**: Como usuario especializado, quiero configurar los parámetros del modelo para personalizar los entrenamientos y evaluaciones.
- **HU-008**: Como usuario especializado, quiero saber los resultados de las métricas estándar en AI para estudiar un modelo.
- **HU-009**: Como usuario especializado, quiero que se cargue un dataset pre-determinado para poder entrenar un modelo.
- **HU-010**: Como usuario especializado, quiero cargar un dataset propio para entrenar modelos con mis ejemplos.
- **HU-011**: Como usuario especializado, quiero que los datos se dividan en entrenamiento y evaluación para poder estudiar un modelo.
- **HU-012**: Como usuario, quiero poder apagar la aplicación para finalizar su ejecución sin errores.
- **HU-013**: Como usuario, quiero un menú de ayuda para saber en que consiste cada opción disponible.
- **HU-014**: Como dueño de la aplicación, quiero una advertencia a los usuarios para evitar consecuencias legales derivadas de los resultados de clasificación.

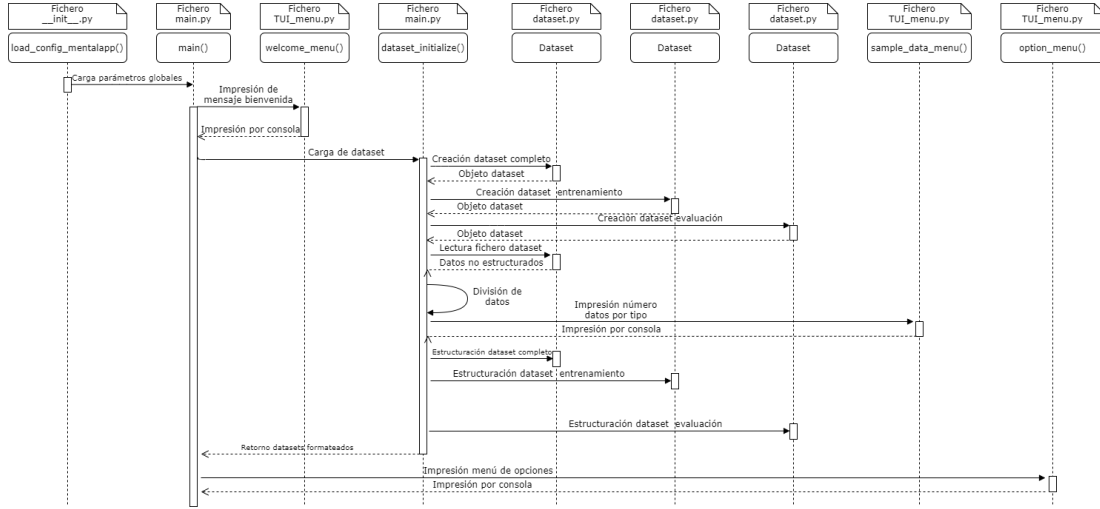


Figura 5.1: Diagrama secuencia inicio de AIMentalHealth

## 5.2. Diseño y arquitectura de la aplicación

La arquitectura de la aplicación AIMentalHealth está dividida en dos grandes secciones correspondientes al componente principal de esta y al conjunto de dependencias utilizadas en la aplicación.

Definiendo en primer lugar la arquitectura de las dependencias de la aplicación, esta arquitectura se rige por el modelo modular; debiéndose a la independencia de los diversos códigos fuente correspondientes a cada librería utilizada (especificadas en la sección 3.1). Mediante un entorno virtual para evitar conflictos con lo existente en el entorno local se importan las librerías y se almacenan en el entorno para, posteriormente, utilizar funciones de estas en la implementación de la aplicación.

La arquitectura de la aplicación principal corresponde a una arquitectura monolítica debido a la facilidad de la implementación en esta primera versión de la aplicación. Además, permite un despliegue sencillo facilitando el uso de los primeros usuarios y simplificando el proceso de creación y diseño.

Con referente al diseño de la aplicación se ha optado por un sistema que cargue las configuraciones iniciales y el dataset por defecto mientras se refleja el menú de bienvenida siendo así un proceso transparente por el usuario [imagen 5.1]. Posteriormente, se accede a un menú principal del cual inician todas las posibles funcionalidades disponibles y que una vez concluidas regresan al mismo menú exceptuando la finalización de la aplicación. Este flujo se ha diseñado con la motivación de facilitar a un usuario poco experimentado en el uso de la CMD o PowerShell la utilización de la aplicación.

La imagen 5.2 refleja el diagrama de flujo de AIMentalHealth.

## 5.2. Diseño y arquitectura de la aplicación

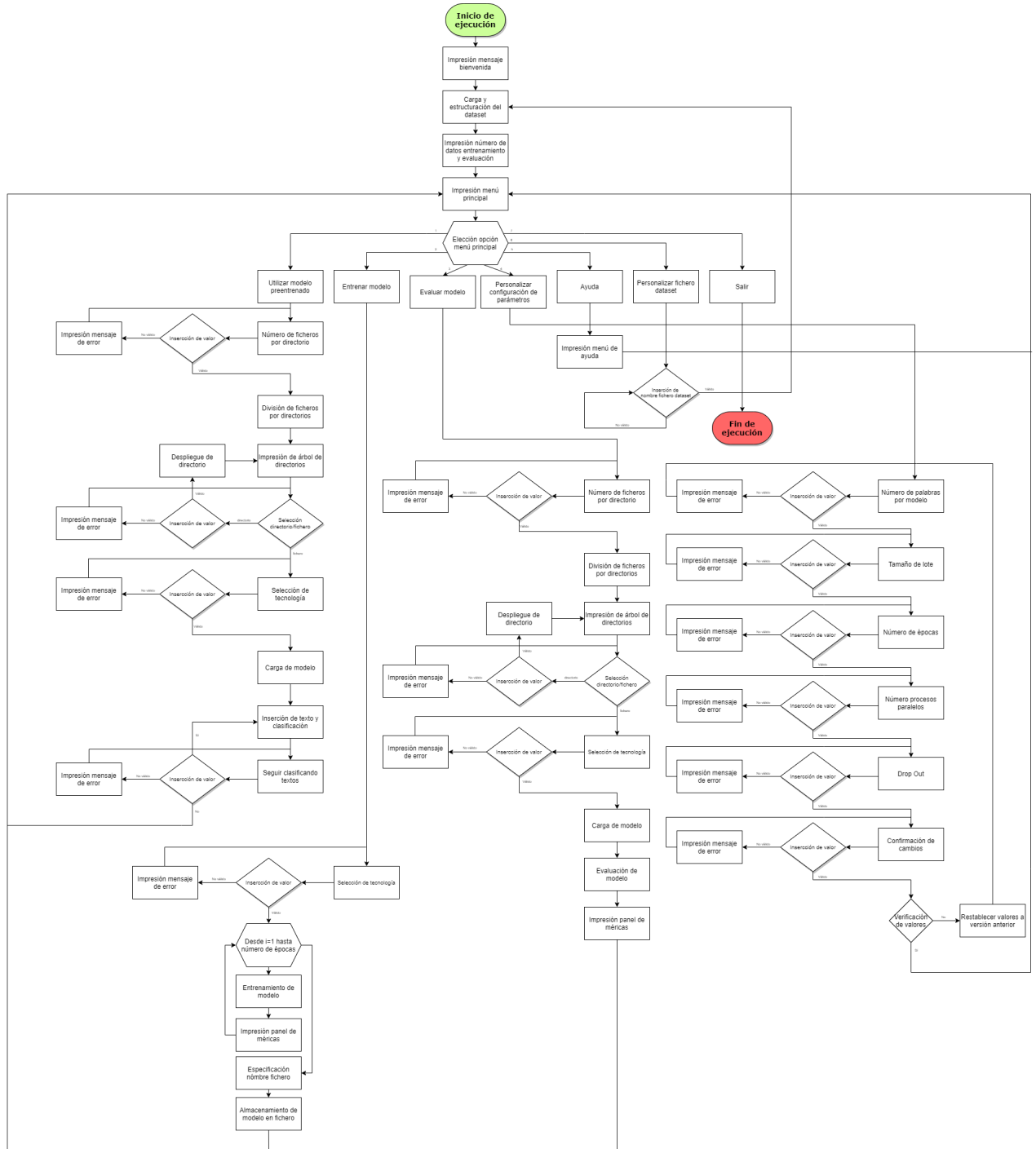


Figura 5.2: Diagrama de flujo de AIMentalHealth

## 5.3. Implementación

El desarrollo de la presente sección es estructurada por las historias de usuario definidas en la sección 5.1. El primer apartado recoge los cimientos generales que permiten el funcionamiento de la aplicación y los siguientes apartados con HU asignada relacionarán el objetivo de esta con los requisitos que abarque además de los detalles técnicos consecuentes.

### 5.3.1. Aspectos generales

La aplicación AIMentalHealth consume para la carga de los parámetros utilizados un fichero `.yaml`<sup>1</sup> permitiendo así la utilización de la configuración como constantes globales. Sin embargo, aún definiendo lo disponible en el fichero como constante; solamente se mantendrá el valor predefinido (inicial) de estos hasta que el usuario desee expresamente realizar un cambio de valor. Este nuevo valor se mantendrá hasta un nuevo cambio o hasta el fin de ejecución de la aplicación ya que en un nuevo inicio los valores cargados serán los predefinidos.

La carga de los parámetros se realiza en formato diccionario, es decir, pares clave-valor en los que invocando la variables con su nombre (clave) se obtiene su magnitud (valor).

### 5.3.2. Historia de usuario 001

**«Como usuario, quiero un mensaje o señal para saber que se ha iniciado la ejecución de la aplicación.»**

Historia de usuario relacionada con los requisitos: **RNF-001**.

Debido a que la interfaz de usuario consiste en una TUI por consola de comandos, el mensaje de bienvenida se ha diseñado con caracteres ASCII evitando así conflictos en sistemas y dispositivos que solamente tengan configurado el conjunto básico de caracteres.

El mensaje de bienvenida se muestra al iniciar la aplicación y se mantiene como última línea de CMD hasta la carga del menú principal, indicando al usuario que se está cargando la aplicación. El contenido consiste en el nombre de la aplicación y un cuadro de texto en el que se da la bienvenida al usuario.[ver imagen 5.3]

---

<sup>1</sup>Formato de texto plano utilizado para representar datos estructurados de forma legible para el ser humano.

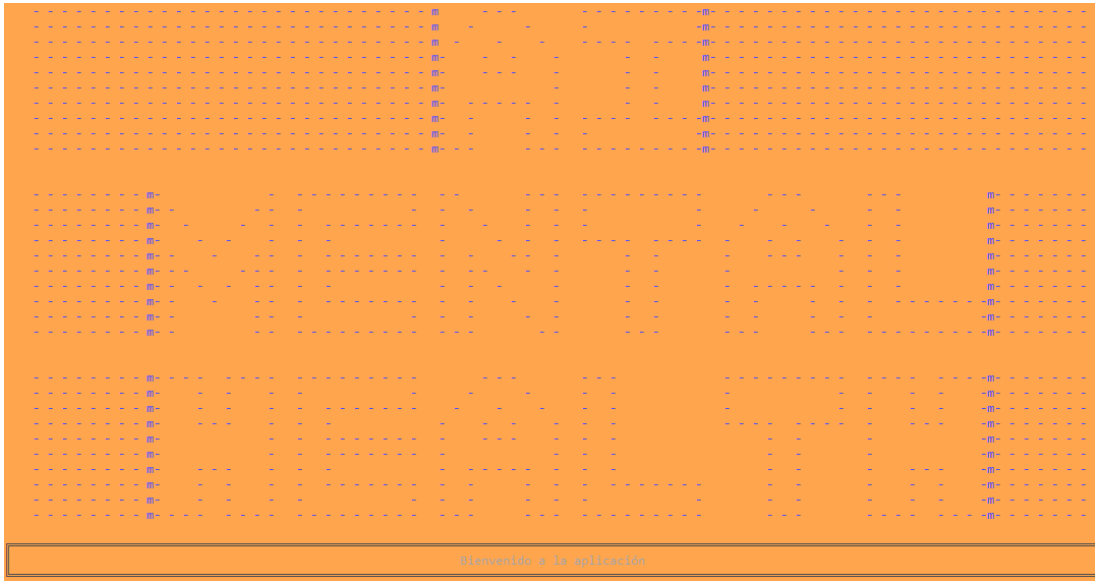


Figura 5.3: AIMentalHealth mensaje de bienvenida

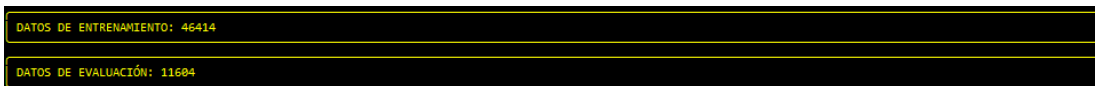


Figura 5.4: AIMentalHealth datos cargados al inicio

### 5.3.3. Historia de usuario 002

«Como usuario, quiero un menú principal para acceder rápidamente a las funcionalidades.»

Historia de usuario relacionada con los requisitos: **RF-008**, **RF-010**, **RF-015**, **RNF-002**, **RNF-004**.

Seguidamente al inicio de la aplicación se cargan los parámetros de configuración con los valores predeterminados y el dataset predeterminado; formateando los datos para que tengan la estructura adecuada y se dividen entre datos de entrenamiento y evaluación siguiendo la proporción 80 % y 20 % respectivamente [imagen 5.4].

Todo el proceso definido es transparente para el usuario y solamente son mostrados por pantalla el número de datos de entrenamiento y evaluación una vez cargados, y en dos cuadros de texto independientes y secuenciales. A continuación se imprime un cuadro de texto indicando que las próximas líneas representan las opciones disponibles y se espera que se escoja una de ellas [imagen 5.5].

Cada opción posee un número asignado resaltado en otro color para destacarlos sobre la totalidad del texto y tras finalizar las opciones se pide al usuario que indique que opción desea. Si la entrada por teclado no es válida (ya sea por ser



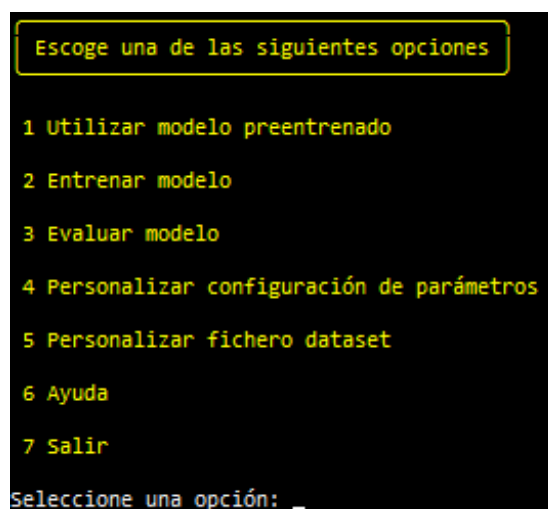


Figura 5.5: AIMentalHealth menú principal

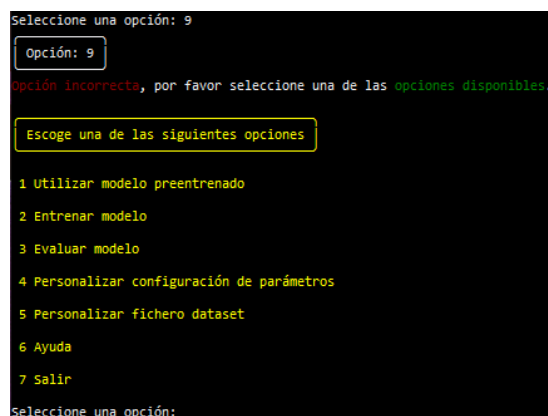


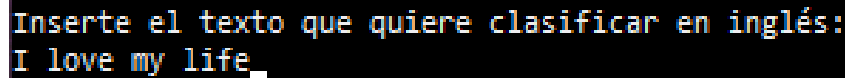
Figura 5.6: AIMentalHealth elección incorrecta en el menú principal

un número no existente en las opciones u otra clase de entrada como un texto) se indica que la opción es incorrecta y que se seleccione una de las disponibles. En esta última casuística se imprime nuevamente las opciones disponibles y la petición de selección al usuario [imagen 5.6].

Por último, si la aplicación sigue en ejecución, se volverá a imprimir nuevamente el menú principal y la petición de selección de funcionalidad al finalizar la ejecución seleccionada con anterioridad.

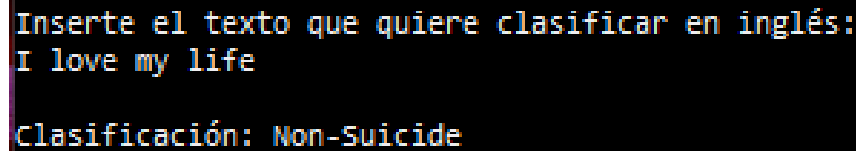
#### 5.3.4. Historia de usuario 003

«Como usuario no especializado, quiero analizar textos evitando configuraciones para comprender lo que realizo y sus resultados.»



```
Inserte el texto que quiere clasificar en inglés:
I love my life_
```

Figura 5.7: AIMentalHealth análisis de mensaje



```
Inserte el texto que quiere clasificar en inglés:
I love my life
Clasificación: Non-Suicide
```

Figura 5.8: AIMentalHealth resultado de un frase analizada

Historia de usuario relacionada con los requisitos: [RF-001](#), [RF-002](#), [RF-015](#)

Incluyendo un acceso para esta funcionalidad en el menú principal [opción 1 de la imagen [5.5](#)], una vez seleccionada por el usuario se escoge un modelo de red neuronal para el análisis. El modelo es cargado desde un fichero .pt cuyo contenido corresponde a las diversas capas de la red con los pesos de las conexiones y sesgos de cada neurona.

Con el modelo ajustado y desplegado con su tecnología asociada se indica al usuario que inserte un texto para procesar detallando que debe ser en inglés. Introduciendo el texto mediante teclado o pegado sobre la CMD se procede a analizar con la red neuronal y se devuelve el resultado [imagen [5.7](#)].

El proceso de análisis es transparente para el usuario y se muestra en una línea contigua la clasificación resultante siendo una de las dos disponibles: Suicide o Non-Suicide. Un ejemplo práctico es apreciable en la imagen [5.8](#).

### 5.3.5. Historia de usuario 004

«Como usuario, quiero poder utilizar modelos ya existentes para clasificar los textos.»

Historia de usuario relacionada con los requisitos: [RF-011](#), [RNF-006](#), [RNF-007](#)

En los diversos puntos de la aplicación en la que se requiere la selección de un modelo de red neuronal pre-entrenado se utiliza la implementación de la división y estructuración de ficheros.

En primer lugar se pide al usuario el número de ficheros que quiere por “directorio” y mostrando el valor predefinido si el input introducido es vacío. En caso de no ser una entrada válida se indica al usuario que inserte un valor válido [imagen [5.9](#)].

```

Por favor indique el número de ficheros por directorio (25): input no válido
Please enter a valid integer number
Por favor indique el número de ficheros por directorio (25):
Directorios
└─ 1. Ficheros del 1 al 11

```

Figura 5.9: AIMentalHealth configuración de división de ficheros por directorios

```

Por favor indique el número de ficheros por directorio (25): 3
Directorios
└─ 1. Ficheros del 1 al 3
└─ 2. Ficheros del 4 al 6
└─ 3. Ficheros del 7 al 9
└─ 4. Ficheros del 10 al 11

```

Figura 5.10: AIMentalHealth división de ficheros por número indicado por pantalla

Seleccionado un número válido, la aplicación busca en el directorio de modelos **ruta\_aplicacion/mentalapp/models/** incluido en el repositorio de código del aplicativo y, ordenando por orden alfabético, se dividen los ficheros en grupos de tantos elementos como haya indicado el usuario. Por cada grupo formado se muestra una línea con origen en la “raíz” que indica qué conjunto numérico de ficheros abarca [imagen 5.10].

Posteriormente, se pide al usuario si desea un directorio o un fichero teniendo que introducir unos de estos dos valores por teclado. En caso de insertar un valor distinto o vacío se muestra un mensaje de error y se exige nuevamente [imagen 5.11].

En el primer caso mostrado, selección de directorio, se pide un número de directorio entre los existentes (formados y mostrados con anterioridad). Una vez seleccionado uno de ellos se volverá a mostrar las líneas anteriores pero en el caso del directorio seleccionado se muestra en un subnivel los ficheros que contiene junto con su número correspondiente (en distinto color). Se vuelve a pedir nuevamente directorio o fichero repitiendo el proceso en caso de seleccionarse la opción directorio [imagen 5.12].

En el camino alterno se pide al usuario el número de fichero que desea, mostrando a continuación un mensaje de error si se ofrece una entrada no válida y

```

¿Qué desea seleccionar? [directorio/fichero]:
Please select one of the available options
¿Qué desea seleccionar? [directorio/fichero]: entrada no válida
Please select one of the available options
¿Qué desea seleccionar? [directorio/fichero]:

```

Figura 5.11: AIMentalHealth entrada incorrecta en selección de fichero o directorio

```
¿Qué desea seleccionar? [directorio/fichero]: directorio
Por favor indique el número de directorio a desplegar (1): 2
Directorios
├─ 1. Ficheros del 1 al 3
├─ 2. Ficheros del 4 al 6
│   ├── 4. 0004_BERT_200_16_1_1_03_25_80.pt
│   ├── 5. 0005_ALBERT_200_16_1_1_03_25_80.pt
│   └── 6. 0006_ROBERTA_200_16_1_1_03_25_80.pt
├─ 3. Ficheros del 7 al 9
└─ 4. Ficheros del 10 al 11

¿Qué desea seleccionar? [directorio/fichero]: _
```

Figura 5.12: AIMentalHealth selección de despliegue de directorio

```
Por favor indique el número de ficheros por directorio (25): 3
Directorios
├─ 1. Ficheros del 1 al 3
├─ 2. Ficheros del 4 al 6
├─ 3. Ficheros del 7 al 9
└─ 4. Ficheros del 10 al 11
```

Figura 5.13: AIMentalHealth selección de fichero y carga del modelo

mostrando la petición de nuevo. Si se selecciona un fichero existente se carga el modelo junto con su tecnología asociada [imagen 5.13].

### 5.3.6. Historia de usuario 005

«Como usuario especializado, quiero entrenar un modelo y evaluarlo cuando requiera para poder determinar la factibilidad de un modelo.»

Historia de usuario relacionada con los requisitos: RF-003, RF-004, RF-005, RF-006, RF-011

Al poder evaluar un modelo existente múltiples veces y no solamente cuando se entrene, la ejecución de estas dos funcionalidades debe de ser independiente. Por tanto, en el menú principal de la aplicación se añaden dos funcionalidades: Entrenar modelo y Evaluar modelo [opciones 2 y 3 de la imagen 5.5 respectivamente].

Si se selecciona entrenar modelo se pide al usuario la elección de una tecnología y comienza el entrenamiento del modelo básico, utilizando los datos cargados al iniciar la aplicación destinados a este propósito. Por cada época de entrenamiento se ajustan los pesos de las conexiones entre neuronas y sesgos por cada lote de datos hasta abarcar el conjunto completo.



Figura 5.14: AIMentalHealth porcentaje de datos procesados



Figura 5.15: AIMentalHealth entrenamiento de red neuronal

En cada época se muestra en tiempo real el porcentaje de datos analizados mediante una barra de progreso junto con el valor en tanto por ciento [imagen 5.14]. Además, al finalizar la época se muestra un panel de métricas con los valores obtenidos.

Finalizando la ejecución del entrenamiento se indica al usuario que inserte mediante teclado un nombre para el fichero que contendrá el modelo entrenado. Dicho fichero será almacenado en la ruta de modelos de la aplicación **ruta\_aplicacion/mentalapp/models/** con extensión .pt<sup>2</sup> [imagen 5.15].

En el caso de selección de evaluación de modelo se pide la selección de un modelo almacenado en fichero con extensión .pt y su tecnología asociada. Posteriormente se evalúa el modelo con los datos cargados con tal propósito y se muestra el panel de métricas resultantes, además del porcentaje de datos analizado en tiempo real [imagen 5.16].

<sup>2</sup>Extensión estándar para ficheros cuyo contenido es la estructura de una red neuronal con los pesos y sesgos.



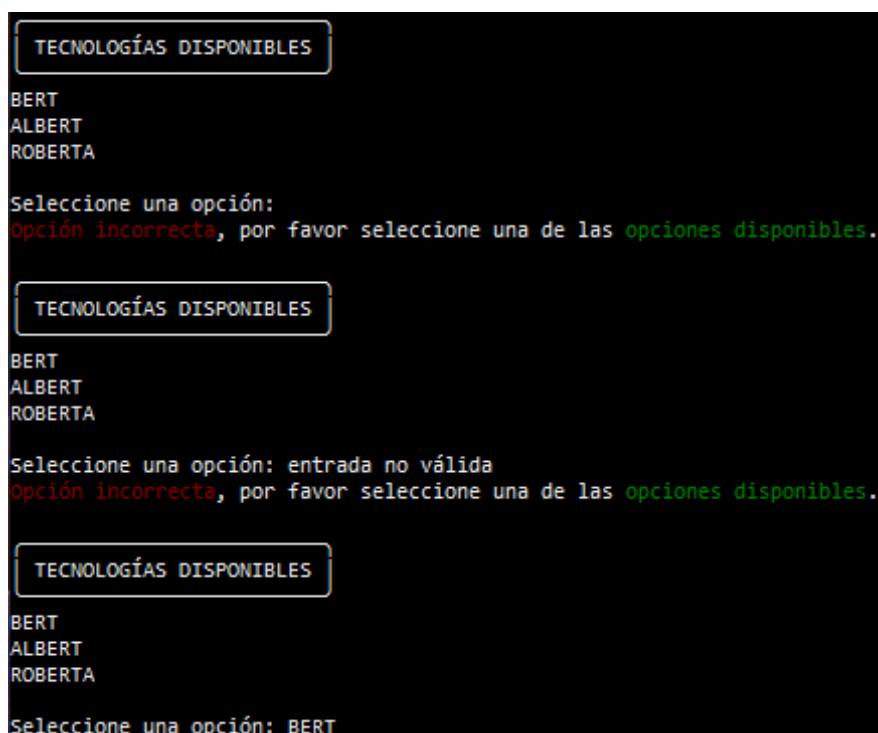


Figura 5.17: AIMentalHealth selección de tecnología

modelo).

### 5.3.8. Historia de usuario 007

«Como usuario especializado, quiero configurar los parámetros del modelo para personalizar los entrenamientos y evaluaciones.»

Historia de usuario relacionada con los requisitos: [RF-012](#), [RF-015](#), [RNF-006](#)

Añadiendo un acceso al menú principal se ha implementado un menú secuencial de configuración de los diversos parámetros [opción 4 de la imagen [5.5](#)]. Al iniciar la ejecución de la aplicación la carga de los parámetros es transparente al usuario para poder utilizar/entrenar/evaluar modelos directamente.

Ejecutando la opción de configuración de parámetros, por cada parámetro configurable se le pedirá al usuario que ingrese un valor, mostrando entre paréntesis el valor actual (correspondiente al predeterminado si aún no se ha modificado). Si el usuario inserta un valor no válido se muestra el mensaje de error y se vuelve a pedir un valor para el parámetro. En el caso de que el usuario inserte un valor válido o vacío, permaneciendo el valor existente en esta última casuística, se repite el proceso con el siguiente parámetro [imagen [5.18](#)].

Al definir el valor de todos los parámetros se muestra un mensaje indicando

```

Opción: 4
Por favor indique el número máximo de palabras que admita el modelo (200): entrada no valida
Please enter a valid integer number
Por favor indique el número máximo de palabras que admita el modelo (200):
Por favor indique el tamaño de lote (16): 4
Por favor indique el número de épocas (iteraciones) (3): 1
Por favor indique el número de procesos paralelos (1): 3
Por favor indique el tanto por uno de drop out para BERT (0.3):

```

Figura 5.18: AIMentalHealth configuración de parámetros

```

¿Estás seguro de los cambios realizados? [y/n]: entrada no válida
Please enter Y or N
¿Estás seguro de los cambios realizados? [y/n]:
Please enter Y or N
¿Estás seguro de los cambios realizados? [y/n]: N
Por favor indique el número máximo de palabras que admita el modelo (200):
Por favor indique el tamaño de lote (4):
Por favor indique el número de épocas (iteraciones) (1):
Por favor indique el número de procesos paralelos (3):
Por favor indique el tanto por uno de drop out para BERT (0.3):
¿Estás seguro de los cambios realizados? [y/n]: y

```

Figura 5.19: AIMentalHealth confirmación de los cambios en la configuración

si se quiere perpetuar los cambios realizados en la configuración. Si la respuesta es afirmativa se regresa al menú principal y en caso contrario, se reinicia la configuración de parámetros [imagen 5.19].

### 5.3.9. Historia de usuario 008

«Como usuario especializado, quiero saber los resultados de las métricas estándar en AI para estudiar un modelo.»

Historia de usuario relacionada con los requisitos: RF-007, RNF-005

Al efectuar un entrenamiento o evaluación de un modelo de red neuronal, por cada etapa (una en el caso de evaluación) se muestra por pantalla un panel con las métricas estándar y los valores medidos durante la ejecución del entrenamiento.

Siendo un proceso transparente para el usuario hasta la finalización de la etapa, la medición se realiza con la cuenta de los distintos casos de la matriz de confusión. Estos casos son:

- **True Negative (TN)**: Casos negativos clasificados como negativos.
- **False Positive (FP)**: Casos negativos clasificados como positivos.
- **False Negative (FN)**: Casos positivos clasificados como negativos.
- **True Positive (TP)**: Casos positivos clasificados como positivos.



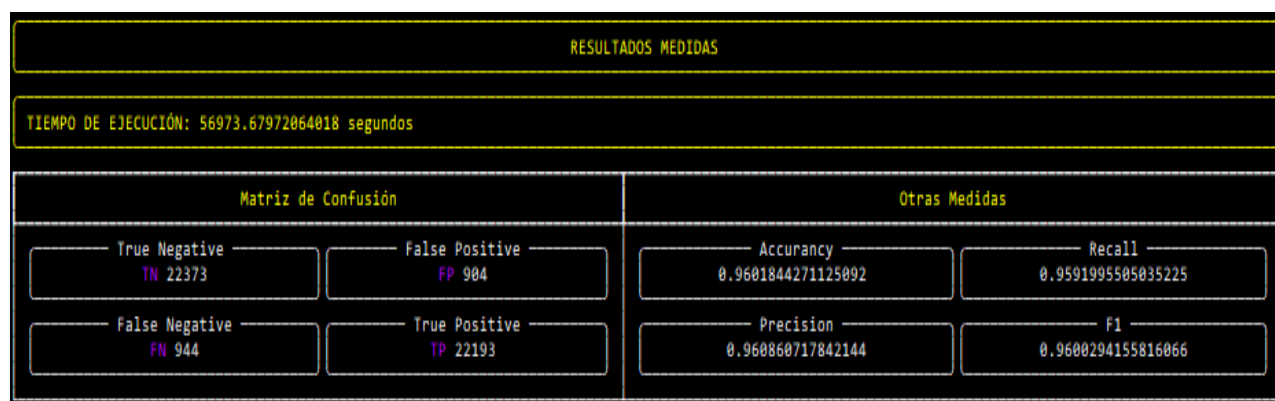


Figura 5.20: AIMentalHealth panel de métricas

Finalizando el entrenamiento/evaluación, con los resultados de las métricas anteriores se calculan:

- **Accuracy**<sup>3</sup>:  $(TP + TN) / (TP + FP + FN + VN)$
- **Recall**<sup>4</sup>:  $TP / (TP + FN)$
- **Precision**<sup>5</sup>:  $TP / (TP + FP)$
- **F1**:  $(2 * Precision * Recall) / (Precision + Recall)$

Por último se imprime por consola el panel de métricas estructurado cuya disposición consiste en una cabecera con el tiempo de ejecución y una tabla dividida en dos partes: lado izquierdo dedicado a la matriz de confusión y lado derecho dedicado a las métricas restantes. Cada métrica posee un cuadro gráfico con título correspondiente al nombre de la medida y su valor en el interior. La estructura en su conjunto es apreciable en la imagen 5.20.

### 5.3.10. Historia de usuario 009

«Como usuario especializado, quiero que se cargue un dataset predeterminado para poder entrenar un modelo.»

Historia de usuario relacionada con los requisitos: RF-008, RF-010, RNF-004

Al inicio de la aplicación se carga el dataset predeterminado de la aplicación disponible en la siguiente ruta: [ruta\\_aplicacion/mentalapp/mod\\_dataset/](#)

<sup>3</sup>Exactitud

<sup>4</sup>Sensibilidad

<sup>5</sup>Precisión

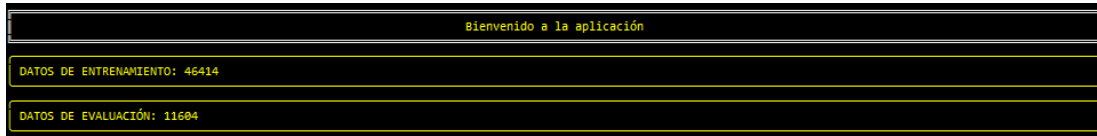


Figura 5.21: AIMentalHealth carga inicial del dataset predeterminado

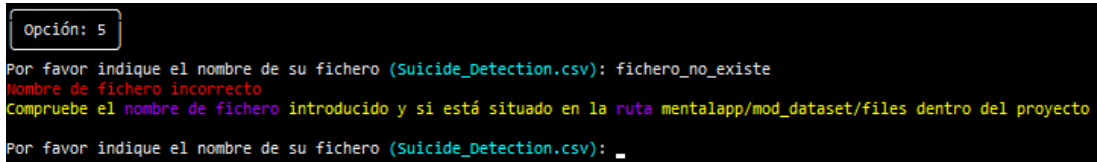


Figura 5.22: AIMentalHealth input de nuevo dataset incorrecto

**files/**. La carga de datos consiste en la lectura de un fichero .csv y por cada línea, aplicando paradigma funcional, se crea un objeto mensaje el cuál consiste en un texto en inglés para clasificar y una clasificación (non-suicide o suicide). Una estructura de datos cuyo contenido son todos los objetos creados constituye el dataset.

Posteriormente el dataset se divide entre datos de entrenamiento y evaluación siguiendo la proporción 80 % y 20 % respectivamente.

Todo el proceso definido es transparente para el usuario y solamente es mostrado por pantalla el número de datos de entrenamiento y evaluación una vez cargados, y en dos cuadros de texto independientes y secuenciales [imagen 5.21].

### 5.3.11. Historia de usuario 010

«Como usuario especializado, quiero cargar un dataset propio para entrenar modelos con mis ejemplos.»

Historia de usuario relacionada con los requisitos: RF-008, RF-009, RF-010, RF-015, RNF-004, RNF-006

Adicionando al menú principal una opción de ejecución para el cambio de dataset se habilita al usuario la opción de cambio de origen de datos de entrenamiento y evaluación [opción 5 de la imagen 5.5].

Al seleccionar la opción se pide al usuario que indique el nombre del fichero que desea cargar como dataset mostrando el valor predeterminado o último, insertado en un cambio anterior, entre paréntesis. En el caso de que el input no sea correcto se vuelve a pedir al usuario el nombre del dataset y se indica por pantalla que el fichero tiene que estar en la siguiente ruta: **ruta\_aplicacion/mentalapp/mod\_dataset/files/** [imagen 5.22].

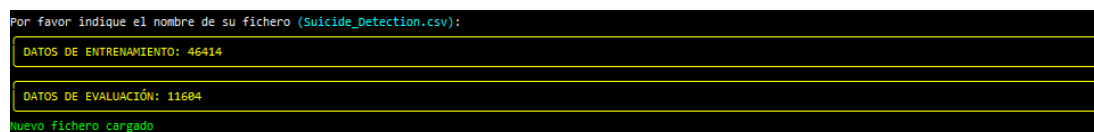


Figura 5.23: AIMentalHealth carga del nuevo dataset

Por otra parte, si el input es correcto o se mantiene el actual se carga el fichero de igual manera que al inicio de la aplicación. Es decir, se estructuran los datos y se dividen en entrenamiento y evaluación con una proporción 80 % – 20 % respectivamente además de mostrar el número de datos dedicados a cada fin por pantalla [imagen 5.23].

### 5.3.12. Historia de usuario 011

«Como usuario especializado, quiero que los datos se dividan en entrenamiento y evaluación para poder estudiar un modelo.»

Historia de usuario relacionada con los requisitos: RF-010

Al cargar un dataset para un evento relacionado con la inteligencia artificial y concretamente con el área de las redes neuronales es imperativo la división de los datos para el correcto funcionamiento de la red resultante. Esto se debe a que se necesita un grupo de datos independientes de control para evaluar un modelo una vez haya sido entrenado con el resto de datos disponibles, además de que la cantidad de datos de entrenamiento sea superior a los datos de evaluación y ambos grupos estén equilibrados, es decir, que posean datos para todos los casos posibles. En este caso en específico consiste en que ambos grupos posean casos de Non-Suicide y Suicide.

En la división del dataset se destina un 80 % de los datos a entrenamiento y un 20 % a evaluación independientemente del número de casos disponibles. Dicha división se realiza de forma aleatoria y equilibrada gracias a la librería **Sklearn** y concretamente se utiliza la función `train_test_split`.

### 5.3.13. Historia de usuario 012

«Como usuario, quiero poder apagar la aplicación para finalizar su ejecución sin errores.»

Historia de usuario relacionada con los requisitos: RF-014

Habilitando una opción de ejecución para finalizar la aplicación en el menú principal [opción 7 de la imagen 5.5] posibilita que el usuario finalice la aplicación

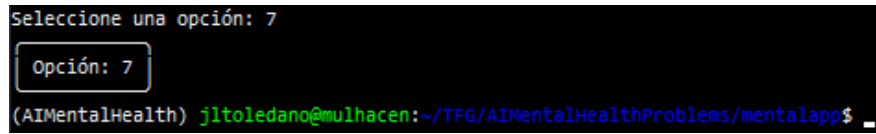


Figura 5.24: AIMentalHealth finalización de la ejecución

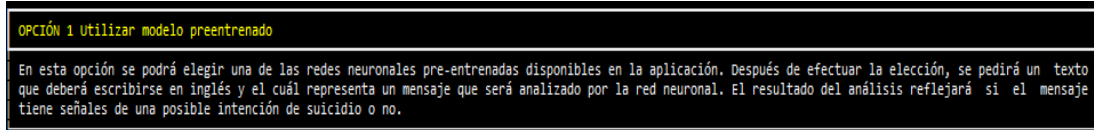


Figura 5.25: AIMentalHealth contenido del menú de ayuda opción de ejecución 1

de forma correcta evitando errores debido a procesos que se estén ejecutando mientras se produce una finalización forzosa [imagen 5.24].

#### 5.3.14. Historia de usuario 013

«Como usuario, quiero un menú de ayuda para saber en que consiste cada opción disponible.»

Historia de usuario relacionada con los requisitos: **RF-013**, **RNF-003**

Habilitando el acceso al menú de ayuda desde el menú principal el usuario puede acceder a la guía de la aplicación [opción 6 de la imagen 5.5]. Al ejecutar esta opción se muestra de forma secuencial un cuadro con explicación al detalle por cada opción disponible en el menú principal.

Cada sección detalla el funcionamiento de una opción de ejecución del menú principal además de la definición de parámetros y sus valores predeterminados en caso de ser necesario. Además se indica la estructura que debe tener un dataset en el caso que se requiera cargar uno nuevo. El contenido de cada sección de ayuda se puede observar en las imágenes 5.25, 5.26, 5.27, 5.28, 5.29, 5.30 y 5.31.

#### 5.3.15. Historia de usuario 014

«Como dueño de la aplicación, quiero una advertencia a los usuarios para evitar

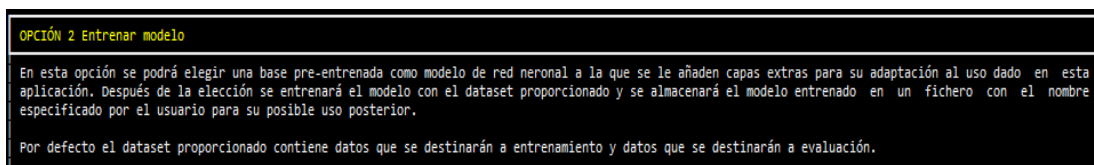


Figura 5.26: AIMentalHealth contenido del menú de ayuda opción de ejecución 2

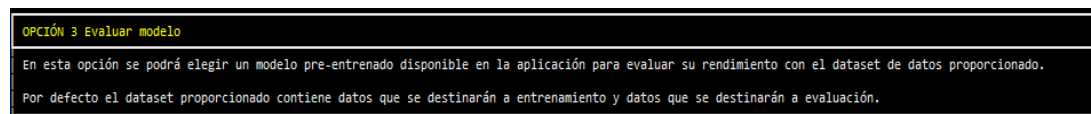


Figura 5.27: AIMentalHealth contenido del menú de ayuda opción de ejecución 3

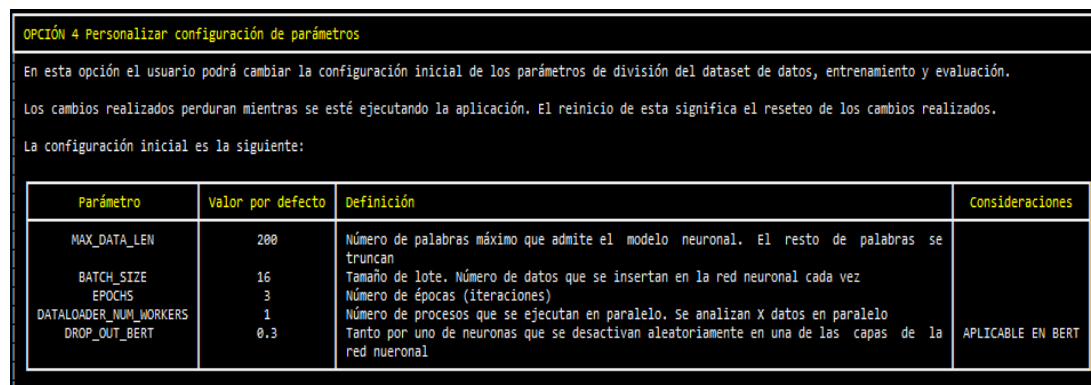


Figura 5.28: AIMentalHealth contenido del menú de ayuda opción de ejecución 4

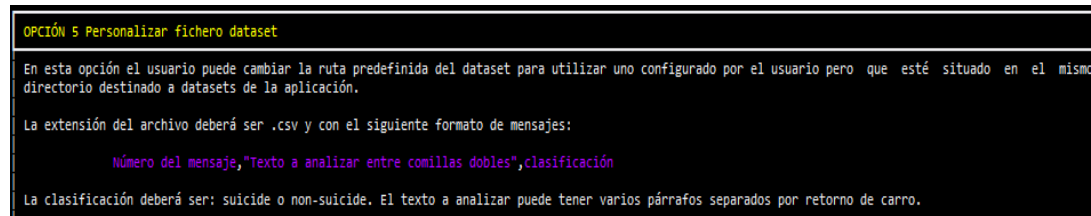


Figura 5.29: AIMentalHealth contenido del menú de ayuda opción de ejecución 5

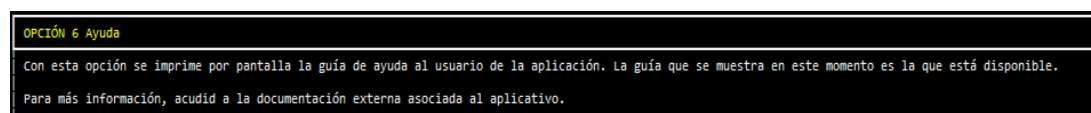


Figura 5.30: AIMentalHealth contenido del menú de ayuda opción de ejecución 6

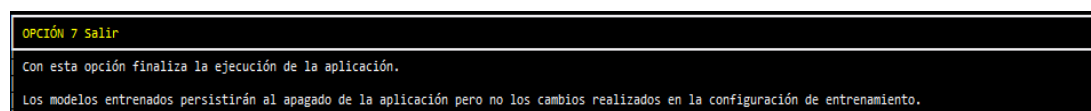


Figura 5.31: AIMentalHealth contenido del menú de ayuda opción de ejecución 7

**consecuencias legales derivadas de los resultados de clasificación.»**

Historia de usuario relacionada con los requisitos: **RNF-008**

Debido a la naturaleza delicada de los datos y sus resultados por su relación con la inteligencia emocional de un usuario es necesario avisar de que los resultados ofrecidos por la aplicación no son definitivos. Es necesario un especialista en el área de la salud relacionado con la psicología y/o salud mental que certifique la validez y diagnostique al paciente según sus conocimientos. [insertar imagen]

Con este aviso se asegura que la aplicación sea utilizada como herramienta de investigación de inteligencia artificial o herramienta auxiliar para un diagnóstico.

## 5.4. Pruebas configuraciones de red neuronal

Las pruebas realizadas en la aplicación han consistido en experimentar con las posibilidades que ofrece respecto al entrenamiento de modelos.

Utilizando la aplicación simulando ser un usuario experimentado en la informática y en el desarrollo de redes neuronales se han entrenado múltiples modelos diferenciando la tecnología y algún parámetro de la configuración. De esta forma se han conseguido múltiples modelos disponibles para su uso en la aplicación y se han evaluado para comprobar cuál es el que ofrece el mejor ratio entre tiempo de entrenamiento y resultado de métricas en la evaluación.

Los modelos almacenados siguen la siguiente lógica de nomenclatura:

**AAAA\_BBBB\_CCCC\_DDDD\_EEEE\_FFFF\_GGGG\_HHHH\_IIII**

Correspondiendo cada sección:

- **AAAA** → Código numérico del modelo.
- **BBBB** → Tecnología utilizada.
- **CCCC** → Longitud en palabras del input.
- **DDDD** → Tamaño de lote.
- **EEEE** → Número de épocas.
- **FFFF** → Número de procesos paralelos.
- **GGGG** → Drop out.

- **HHHH** → Porcentaje de datos sobre el total del dataset.
- **IIII** → Porcentaje de datos destinados a entrenamiento.

Los resultados de las pruebas se pueden observar en el **apéndice B** disponible en este documento. Estos resultados reflejan la aplicación de paradigma concurrente en el entrenamiento de redes neuronales y cómo afectan al tiempo de entrenamiento además de mostrar qué tecnología es mejor para la problemática a la que se destina la aplicación.

Resumiendo el estudio disponible en el apéndice, la elección de la tecnología afecta considerablemente al tiempo de entrenamiento siendo ROBERTA la más costosa en esta medida. Por otra parte, el número de hilos no demuestra una diferencia considerable en el tiempo de entrenamiento ni efecto alguno en el resto de métricas.





# 6

## Conclusiones y trabajos futuros

El objetivo de esta sección es reflejar las conclusiones extraídas del trabajo realizado y los posibles caminos que se pueden seguir para su desarrollo en siguientes etapas.

### 6.1. Conclusiones

AIMentalHealth cumple la función de herramienta de apoyo en investigación de IA y en el análisis psicológico de un paciente. Dependiendo del objetivo perseguido el usuario utilizará ciertas funciones disponibles. Poniendo un ejemplo, un estudiante de IA utilizará en gran medida la opción de configuración de parámetros para personalizar sus entrenamientos.

Sin embargo, esta aplicación no oculta funcionalidades según el tipo de usuario que la utilice por lo que beneficiará en gran medida a personas que estén interesadas en ambos objetivos.

El estudio y desarrollo de inteligencias artificiales está en auge en los últimos años destacando el pico de interés provocado por la empresa OpenAI y la publicación de su herramienta chatGPT al público general, acercando así a nuevos estudiantes (especializados o no) al área del machine learning. Al poder utilizar esta herramienta de forma gratuita la empresa ha provocado que un área de estudio muy concreta haya conseguido llegar al público general independientemente de la profesión o cargo.

AIMentalHealth facilitará el estudio a nuevos estudiantes de inteligencia artificial debido a su fácil uso y a la baja necesidad de recursos. Debido a estas facilidades cabe la posibilidad de que más personas se interesen por este estudio además de permitir a los ya interesados avanzar más rápido en sus estudios. De esta manera se incrementará el número de personas formadas pudiendo abastecer la demanda de especialistas de AI que existe en el mercado actual.

Por otra parte, destacando su importancia desde la reciente pandemia del COVID-19, la inteligencia emocional está presente en la sociedad actual. Problemas que hace unos años se clasificaban como nervios o estrés se diagnostican hoy en día con nombres propios y se personaliza el tratamiento a cada paciente. Desgraciadamente, la saturación del sistema sanitario relacionado con el tratamiento psicológico está muy saturado y muchos de los pacientes no pueden permitirse sesiones privadas aún cuando las necesitan urgentemente. El conjunto de estos factores desemboca en citas para diagnóstico muy tardías y en el deterioro del estado de los pacientes sin tratamiento. Cada año más de estos casos terminan en suicidio.

La herramienta desarrollada en este trabajo permitirá a los especialistas recibir un informe con el análisis de los textos escritos por un paciente ya hecho. Este proceso agiliza el análisis secuencial que tiene que realizar un profesional de la salud por cada paciente pudiendo delegar esta tarea a un personal menos especializado que cree el informe. Se logra que una parte del diagnóstico del paciente se realice de manera más rápida permitiendo descolapsar el sistema sanitario a medio y largo plazo, pudiendo abarcar a más personas en un futuro.

## 6.2. Trabajos futuros

La aplicación desarrollada en este trabajo es una sencilla interfaz de texto desde el punto de vista del usuario. El primer paso del desarrollo ha sido centrado en diseñar una aplicación útil para el ser humano implementado su lógica principal y funcionalidades auxiliares.

Los próximos pasos en el desarrollo de la aplicación deben de estar enfocados en la accesibilidad del producto. Mediante diseñadores y especialistas en áreas como la interacción persona-ordenador se puede diseñar una aplicación de escritorio con las mismas funcionalidades actuales o incluso una app para dispositivos móviles. Además, publicando el producto como página web siguiendo la vía de chatGPT se puede abrir el uso de la aplicación al gran público debido a la ausencia de complicaciones producidas por una instalación.

Por otro lado, si el objetivo futuro seguido es apoyar preferentemente a la comunidad de desarrolladores AI se puede desarrollar widgets con usos específicos y publicarlos en las principales plataformas de distribución de conocimientos

referentes a la inteligencia artificial. Poniendo como ejemplo la plataforma que ha ofrecido los recursos necesarios para el desarrollo de este proyecto, Hugging Face permite la publicación de este tipo de producto software bajo el nombre de *Spaces*.

Un camino paralelo es destinar los futuros esfuerzos de los desarrolladores en hacer pruebas reales con los modelos entrenados junto con especialistas interesados en el proyecto. En el estado actual de la aplicación se puede ejecutar este trabajo sin complicaciones excepto las posibles producidas por la instalación en un inicio. El apoyo de los especialistas en sanidad y el estudio del desempeño del aplicativo en la actividad diaria permitirá refinar diversos aspectos en la estructura o en el flujo de uso.

Por último, los ingenieros del software con interés en AIMentalHealth pueden realizar sprints más pequeños que implementando una interfaz gráfica para la aplicación. El desarrollo se puede enfocar en la ampliación de tecnologías de modelos para ofrecer su uso a los usuarios. Como añadido se pueden realizar nuevas pruebas variando los parámetros de configuración y los modelos para observar qué combinación se adapta mejor a la problemática tratada.

# Bibliografía

- [1] Observatorio del Suicidio en España. (2020). Prevención del suicidio. Recuperado de <https://www.fsme.es/observatorio-del-suicidio-2020/>
- [2] Observatorio del Suicidio en España 2022 (datos definitivos diciembre 2023). (s. f.). Prevención del suicidio. <https://www.fsme.es/observatorio-del-suicidio-2022-definitivo/>
- [3] Madrid Salud. (2022, 8 noviembre). Datos de Madrid - Prevención del suicidio. Prevención del suicidio. <https://prevenciondelsuicidio.es/datos-de-madrid/>
- [4] Junta de Gobierno de la Ciudad de Madrid. (2022). Plan de actuación de prevención del suicidio del ayuntamiento de Madrid 2023-2024.
- [5] IEEE. (1998). IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998). Institute of Electrical and Electronics Engineers.
- [6] GitHub: Let's build from here. (s. f.). GitHub. <https://github.com/>
- [7] Manage your team's projects from anywhere — Trello. (s. f.). <https://trello.com/>
- [8] Manifiesto Ágil. (2001). Manifiesto para el Desarrollo Ágil de Software. Recuperado de <http://agilemanifesto.org/>
- [9] Home. (s. f.). Scrum.org. <https://www.scrum.org/>
- [10] Visual Studio Code - Code editing. Redefined. (2021, 3 noviembre). <https://code.visualstudio.com/>
- [11] Inteligencia Artificial - Un Enfoque Moderno (2ª ed.). Russell, S.; Norvig, P. (2003): Prentice Hall Hispanoamericana
- [12] Inteligencia Artificial - Una Nueva Síntesis. Nilsson, N. (2001): McGraw-Hil
- [13] Shneiderman, B., & Plaisant, C. (s. f.). Diseño de interfaces de usuario: estrategias para una interacción persona-computadora efectiva.

- [14] TIOBE Index - TIOBE. (2022, 3 junio). TIOBE. <https://www.tiobe.com/tiobe-index/>
- [15] Lutz, M. (2018). Learning Python. O'reilly.
- [16] Hugging Face – The AI community building the future. (n.d.). Huggingface.co. <https://huggingface.co>
- [17] API reference — pandas 1.1.4 documentation. (n.d.). Pandas.pydata.org. <https://pandas.pydata.org/docs/reference/index.html>
- [18] NumPy Reference — NumPy v1.23 Manual. (n.d.). Numpy.org. <https://numpy.org/doc/stable/reference/index.html#reference>
- [19] scikit-learn. (2019). scikit-learn: machine learning in Python. Scikit-Learn.org. <https://scikit-learn.org/stable/>
- [20] Introduction — Rich 12.6.0 documentation. (n.d.). Rich.readthedocs.io. <https://rich.readthedocs.io/en/stable/introduction.html>
- [21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv (Cornell University), 30, 5998-6008. <https://arxiv.org/pdf/1706.03762v5>
- [22] Git. (s. f.). <https://git-scm.com/>
- [23] Atlassian. (s. f.). Flujo de trabajo de GitFlow — Atlassian Git Tutorial. <https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>
- [24] Principios del manifiesto ágil. (s. f.). <https://agilemanifesto.org/iso/es/principles.html>



# Apéndice







# Especificación de requisitos

Este documento refleja el resultado del proceso de análisis, extracción y redacción de los requisitos exigidos para el desarrollo del aplicativo.

## **A.1. Introducción**

### **A.1.1. Propósito**

El propósito del presente documento es mostrar el contenido correspondiente al ERS del aplicativo desarrollado. Las principales funciones a cumplir son la definición precisa de las funcionalidades y restricciones del aplicativo además de las características que debe cumplir su estado final.

### **A.1.2. Alcance**

La aplicación denominada AI Mental Health tiene el propósito de servir como herramienta de investigación en el ámbito de la inteligencia artificial y como herramienta de diagnóstico en el área de psicología.

Los desarrolladores informáticos o personas interesadas en el estudio de la Inteligencia Artificial utilizarán esta herramienta con el propósito de experimentar con modelos pre-entrenados en el análisis de lenguaje natural (NPL), concretamente en la detección de señales de suicidio de un texto, variando diferentes

métricas de entrenamiento o el propio dataset del problema si se adapta a las condiciones especificadas en el aplicativo. Además, los investigadores pueden exportar sus modelos entrenados en el formato estándar para poder utilizarlos en herramientas externas que posean.

Por otro lado, personas especializadas en el campo de la psicología pueden consumir los resultados de la investigación ya ofrecidos en la aplicación para aplicar la inteligencia artificial en la detección del suicidio en mensajes de texto de sus pacientes. No obstante, en la propia aplicación se indica que los resultados obtenidos no son definitivos y que siempre deberán de estar respaldados por un especialista para ser válidos en un diagnóstico.

### A.1.3. Definiciones y siglas

#### A.1.3.1. Definiciones

- **Inteligencia Artificial:** Subdisciplina del campo de la informática que busca la creación de máquinas que puedan imitar comportamientos inteligentes.
- **Sistema operativo:** Conjunto de órdenes y programas que controlan los procesos básicos de una computadora y permiten el funcionamiento de otros programas.
- **Modelo:** Construcción conceptual simplificada de una realidad más compleja.

#### A.1.3.2. Siglas

- **AI:** Artificial Intelligence.
- **CMD:** Command Prompt.
- **ERS:** Engineering Requirements Specification.
- **NLP:** Natural Language Processing.
- **IEEE:** Institute of Electrical and Electronics Engineers.
- **SO:** Sistema Operativo.
- **TUI:** Text User Interface.

#### A.1.4. Referencias

- IEEE Std 830-1998, “Recommended Practice for Software Requirements Specifications”.
- “IEEE Recommended Practice for Software Requirements Specifications”.
- Especificación de Requisitos según el estándar de IEEE 830.
- IEEE-STD-830-1998

#### A.1.5. Visión global

El contenido principal del presente documento se estructura en dos secciones principales denominándose **Descripción general** y **Requisitos específicos**.

En la sección **Descripción general** se detallan las funcionalidades de la aplicación, además de las características de los usuarios a los que se destina, dependencias existentes con sistemas externos y limitaciones actuales del producto software.

En la extensa sección **Requisitos específicos** se detallan a nivel necesario para un desarrollador software las especificaciones a cumplir por la aplicación para satisfacer los objetivos y características propuestos.

### A.2. Descripción general

#### A.2.1. Perspectiva del producto

El producto software AI Mental Health consiste en una aplicación ejecutada mediante consola de comandos (CMD) que facilitará una interfaz de texto al usuario para su utilización. Al iniciar la aplicación se desplegará un menú de bienvenida, seguido de un menú que ofrecerá diversas funcionalidades para todo tipo de usuarios siendo el operador quién decida ajustarse a un uso solamente de consumidor del aplicativo o una utilización relacionada con la investigación del análisis del lenguaje natural mediante inteligencia artificial.

Mediante mensajes de texto impresos en la consola de comandos el usuario navegará por las distintas funcionalidades de la aplicación y podrá seguir las instrucciones indicadas para, por ejemplo, establecer los parámetros de configuración del entrenamiento del nuevo modelo.

La aplicación mostrará un mensaje de bienvenida con el nombre de la aplicación (AI Mental Health) seguido de la cantidad de datos cargados para un futuro

entrenamiento y/o evaluación. A continuación, aparece en la CMD el menú principal con las opciones esenciales disponibles en el aplicativo. Dichas opciones serán utilizadas por los diversos tipos de usuario en función de sus necesidades.

Empezando por las funcionalidades generales para todo tipo de usuarios, la aplicación dispone de un menú de ayuda que indica cómo funciona el menú de inicio y todas las opciones disponibles. Además, en el propio menú se establece una opción para finalizar la ejecución de la aplicación, y por tanto, salir de esta y liberando la CMD para otras ejecuciones.

Los usuarios no especializados en el desarrollo software tendrán disponible una opción que ejecutará automáticamente un modelo pre-entrenado a elección por el propio usuario. Se pedirá un texto en inglés por pantalla que una vez introducido mediante el teclado se analizará y se ofrecerá por el mismo dispositivo un mensaje, el cuál indica la clasificación del texto. En esta aplicación existen dos posibles clasificaciones: Suicide y Non-Suicide.

Por otro lado, los usuarios especializados en el área de la inteligencia artificial o con interés de investigación en esta pueden entrenar y evaluar modelos. Ambas funcionalidades se destacan en el menú principal como opciones independientes. Por último, en el menú principal se disponen las opciones de configuración de parámetros para entrenamiento y evaluación; y personalización de la ruta del dataset utilizado.<sup>1</sup>

La aplicación AI Mental Health no dispone de sistema de login al ser ejecutado en la consola local de cada usuario. Debido a este motivo, cualquier usuario puede utilizar todas las opciones disponibles anteriormente detalladas si lo requieren.<sup>2</sup>

Por último, cabe destacar que la aplicación dispone de un algoritmo de división, estructuración y muestra de los ficheros correspondientes a los modelos en los apartados donde se requieren, evitando de este modo la lista secuencial de ficheros estándar de una CMD que obstruye el uso de la aplicación.

### A.2.2. Funciones del producto

El producto es una aplicación relacionada con el área de la psicología y el sector del desarrollo e investigación de la inteligencia artificial, concretamente en el desarrollo de modelos de clasificación de texto con el uso de tecnología Transformers.

Ya iniciada la aplicación y mostrándose el menú principal, se puede seleccionar

---

<sup>1</sup>Es necesario que el nuevo dataset tenga la misma estructura que el disponible de forma predefinida en la aplicación. Se especifica en el menú de ayuda.

<sup>2</sup>La explicación anterior y su división es por motivos aclarativos siguiendo las previsiones de uso general de cada tipo de usuario.

y ejecutar una de las siguientes funciones; volviendo al finalizar al menú inicial excepto en la última funcionalidad:

### 1. Utilizar modelo preentrenado

Seleccionando esta opción, el usuario observa por pantalla un mensaje que le indica cuántos ficheros desea por directorio, pidiendo un número por teclado. Al insertar dicho número aparece por pantalla un árbol de directorios conteniendo cada uno el número de ficheros seleccionado. Continuando la ejecución, el usuario inserta mediante el teclado si desea seleccionar directorio (para desplegar los ficheros contenido en dicho directorio) o un fichero indicando su número de posición disponible en el despliegue mostrado por pantalla.<sup>3</sup>

Después de escoger un fichero se pide al usuario que seleccione una tecnología de entre las disponibles mostradas por pantalla y, en última instancia, dicho usuario insertará por teclado el mensaje que quiere clasificar en inglés (claramente indicada por consola) recibiendo finalmente la clasificación de este.<sup>4</sup>

### 2. Entrenar modelo

Al ejecutar la opción de entrenar modelo se indica al usuario que debe introducir qué tecnología desea para su nuevo modelo y al recibir una opción válida por teclado se procede automáticamente al entrenamiento del nuevo modelo.

Por cada etapa del entrenamiento anteriormente mencionado se indica en todo momento el porcentaje de datos analizados actualizando el dato en tiempo real y, al terminar la etapa, las métricas obtenidas en dicho entrenamiento.

Al finalizar el entrenamiento en su totalidad se pide al usuario un nombre para el fichero generado representante del modelo entrenado y se almacena en formato estándar para su posterior uso.

### 3. Evaluar modelo

El primer paso al elegir esta opción de ejecución es la elección de fichero de modelo preentrenado entre los disponibles en el directorio establecido al igual que en otras opciones disponibles desde el menú principal.

Ya seleccionado el fichero se pide al usuario la tecnología con la que se requiere hacer la evaluación y eligiendo entre las opciones disponibles empieza el proceso de evaluación de forma automática con los datos del dataset reservados para este propósito.

---

<sup>3</sup>Proceso correspondiente al algoritmo de división, estructuración y muestra de los ficheros correspondientes a los modelos.

<sup>4</sup>Suicide o Non-Suicide.

Durante la duración de la evaluación se muestra en tiempo real el porcentaje de datos analizados y al finalizar se muestra un cuadro de métricas idéntico al que se obtiene al entrenar un modelo nuevo.

**4. Personalizar configuración de parámetros**

Seleccionando esta opción en el menú principal se procede a la revisión de cada uno de los parámetros personalizables en la aplicación poniendo el ejemplo del número de etapas de un entrenamiento.

Por cada parámetro se muestra su valor actual, correspondiente al predefinido si no se ha modificado anteriormente en la ejecución actual de la aplicación, y se pregunta al usuario si desea modificarlo; exigiendo un valor válido si se quiere modificar el valor o un salto de línea si se desea mantener el valor actual.

**5. Personalizar fichero dataset**

Similar a la configuración de parámetros, se pide al usuario por pantalla que indique el nuevo nombre del fichero dataset cuyo contenido se especifica en el cuadro correspondiente en la sección de Ayuda. Se muestra el nombre del dataset predefinido y al indicar un nuevo nombre de fichero se cargan los datos dividiéndolos en entrenamiento y evaluación. Finalmente se regresa al menú principal.

**6. Ayuda**

A petición del usuario, en la ejecución de la opción de Ayuda de la aplicación se muestra por CMD un cuadro explicativo por cada una de las opciones disponibles en el menú principal detallando los posibles valores requeridos en caso de ser necesario o el flujo de ejecución percibido por el usuario si se ejecuta dicha opción.

**7. Salir**

Opción que finaliza de forma segura y correcta la ejecución actual de la aplicación liberando la CMD utilizada para el resto de usos requeridos por el usuario.

### **A.2.3. Características del usuario**

La aplicación tiene dos usuarios potenciales en su diseño: el usuario informático y el usuario especialista en el área de la salud. Careciendo de sistema de login, la completitud de las funciones pueden ser utilizadas por todos los usuarios aún se destine las funciones a un tipo de usuario en su planteamiento.

El usuario informático es un usuario datado de conocimientos en el área de la programación y que al usar la aplicación demuestra su interés en el ámbito de la inteligencia artificial. No es necesario que posea conocimiento experto referente

a la inteligencia artificial: comprendiendo las definiciones básicas y teniendo noción superficial de su funcionamiento. Estos conocimientos son suficientes para interpretar los resultados ofrecidos por las métricas disponibles en la aplicación.<sup>5</sup>

El segundo tipo de usuario correspondiente al especialista en el área de salud (concretamente psicología y anexos) utilizará la aplicación como herramienta de análisis, siendo indiferente las métricas de entrenamientos y/o evaluación a excepción de a clasificación ofrecida a un texto introducido por el propio usuario.

#### A.2.4. Restricciones

La aplicación debe de avisar al usuario en manuales y en el propio contenido ejecutable y observable por el usuario que los resultados obtenidos no son válidos sin el respaldo de un experto en la materia del análisis psicológico. Debido a la gravedad de un auto-diagnóstico en el tema tratado en el aplicativo es necesario resaltar el este aviso.

#### A.2.5. Supuestos y dependencias

La aplicación está restringida a los sistemas operativos Windows y Linux además de depender de diversas librerías externas que implementan funcionalidades necesarias para el desarrollo de la aplicación. Debido a esta última razón especificada, la aplicación dependerá de un gestor de entornos virtuales para que el impacto en el entorno local del usuario sea el menor posible y permitiendo, además, la automatización de descarga e instalación de dependencias.

### A.3. Requisitos específicos

#### A.3.1. Requisitos funcionales

##### A.3.1.1. Requisito funcional 001

- **Código:** RF-001
- **Título:** Análisis de texto introducido por teclado
- **Input:** Mensaje de texto en inglés.
- **Procesamiento**

Es necesario la existencia de una funcionalidad que pida al usuario un men-

---

<sup>5</sup>Se establece el pensamiento de que un usuario informático utilizará el aplicativo con fines de investigación.

saje de texto para ser analizado por la inteligencia artificial devolviendo la clasificación deducida.

- **Output:** La aplicación muestra por pantalla el resultado del análisis siendo Suicide o Non-Suicide.

#### A.3.1.2. Requisito funcional 002

- **Código:** RF-002
- **Título:** Implementación diversas tecnologías Transformers
- **Input:** —
- **Procesamiento**  
Implementación de modelo de red neuronal integrando una tecnología Transformers existente relacionada con el análisis de texto. Se debe posibilitar diversas tecnologías para su posible elección por parte del usuario.
- **Output:** Posibilidad de entrenar/evaluar un modelo con diversas tecnologías relacionadas con el análisis de texto y basadas en la tecnología Transformers

#### A.3.1.3. Requisito funcional 003

- **Código:** RF-003
- **Título:** Entrenamiento de red neuronal
- **Input:** Tecnología Transformers y dataset.
- **Procesamiento**  
Dada una tecnología Transformers seleccionada por el usuario y un dataset válido cargado, se entrena un modelo según ciertos parámetros de configuración (número de etapas, tamaño de bloque, ...).
- **Output:** Modelo de red neuronal entrenado.

#### A.3.1.4. Requisito funcional 004

- **Código:** RF-004
- **Título:** Almacenamiento modelo entrenado en fichero
- **Input:** Modelo de red neuronal entrenado.
- **Procesamiento**  
Dado un modelo de red neuronal entrenado por la aplicación se debe almacenar en un fichero con el formato estándar .pt para permitir su posterior uso.
- **Output:** Fichero con extensión .pt.



#### A.3.1.5. Requisito funcional 005

- **Código:** RF-005
- **Título:** Selección nombre fichero modelo entrenado
- **Input:** Fichero con extensión .pt.
- **Procesamiento**  
El usuario tiene que indicar el nombre del fichero que almacena el modelo de red neuronal entrenado y lo especifica utilizando el teclado de la computadora dónde se ejecuta la aplicación.
- **Output:** Fichero con extensión .pt teniendo como nombre el especificado por el usuario.

#### A.3.1.6. Requisito funcional 006

- **Código:** RF-006
- **Título:** Evaluación de modelo de red neuronal existente
- **Input:** Fichero con extensión .pt.
- **Procesamiento**  
Dado un fichero con extensión .pt seleccionado por el usuario se evalúa el modelo almacenado en el fichero según ciertos parámetros de configuración (número de etapas, tamaño de bloque, ...).
- **Output:** —

#### A.3.1.7. Requisito funcional 007

- **Código:** RF-007
- **Título:** Métricas de inteligencia artificial
- **Input:** Proceso de entrenamiento/evaluación de modelo.
- **Procesamiento**  
Por cada etapa completada en un entrenamiento/evaluación de un modelo se debe calcular y mostrar las principales métricas del área de la inteligencia artificial además de otras complementarias como el tiempo de entrenamiento/evaluación, el porcentaje, en tiempo real; de datos analizados en la duración de cada etapa y número de etapa. Las métricas necesarias son las siguientes:
  - Matriz de confusión
  - Exactitud
  - Sensibilidad
  - Precisión
  - F1
- **Output:** Resultado de métricas impresas por consola de comandos.

**A.3.1.8. Requisito funcional 008**

- **Código:** RF-008
- **Título:** Carga de dataset
- **Input:** Fichero con datos y resultados.
- **Procesamiento**

Al inicio de la aplicación se debe cargar y estructurar los datos disponibles en un fichero .csv para su utilización en los modelos. La estructura debe contener por cada elemento un campo texto con el mensaje original y un campo con la clasificación predeterminada del texto (suicide/non-suicide).
- **Output:** Estructura de datos cuyo contenido corresponde a los datos de entrenamiento/evaluación.

**A.3.1.9. Requisito funcional 009**

- **Código:** RF-009
- **Título:** Elección de nuevo dataset
- **Input:** Nombre de fichero.
- **Procesamiento**

Dado un nuevo nombre de fichero dataset indicado por el usuario se debe cargar el nuevo dataset y formar la misma estructura que había con el dataset original con los nuevos datos.
- **Output:** Estructura de datos cuyo contenido corresponde a los datos de entrenamiento/evaluación.

**A.3.1.10. Requisito funcional 010**

- **Código:** RF-010
- **Título:** División de dataset
- **Input:** Estructura de datos con la información del dataset.
- **Procesamiento**

La estructura de datos que almacena la información del dataset se divide en dos secciones representando los datos para entrenamiento y los datos para evaluación. Es necesario dividir los datos de manera aleatoria y además siguiendo el porcentaje estándar por cada sección. El porcentaje estándar en el entrenamiento y evaluación de los modelos de red neuronal indica que se debe destinar el 80 % de los datos a entrenamiento y el 20 % de los datos a evaluación.
- **Output:** Dos estructuras de datos con los respectivos datos destinados a entrenamiento y a evaluación.

#### A.3.1.11. Requisito funcional 011

- **Código:** RF-011
- **Título:** Selección de modelo entrenado
- **Input:** Opción elegida por el usuario (dependiendo del sistema elegido puede ser el nombre del modelo, su posición, ...).
- **Procesamiento**  
El usuario debe elegir el modelo a utilizar entre las opciones disponibles para su uso o la evaluación del modelo. El modelo es seleccionado utilizando el teclado.
- **Output:** Modelo seleccionado.

#### A.3.1.12. Requisito funcional 012

- **Código:** RF-012
- **Título:** Personalización parámetros de configuración
- **Input:** —
- **Procesamiento**  
El usuario debe de tener la opción de personalizar los parámetros de configuración predefinidos en la aplicación. Estos contarán con un valor predeterminado al iniciar la aplicación pero el nuevo valor dado por el usuario debe perdurar hasta la finalización de la ejecución de la aplicación. Los parámetros configurables son:
  - Máxima longitud input
  - Tamaño de lote
  - Número de épocas
  - Número de procesos paralelos
  - Drop out
- **Output:** Parámetros de configuración actualizados.

#### A.3.1.13. Requisito funcional 013

- **Código:** RF-013
- **Título:** Menú de ayuda
- **Input:** —
- **Procesamiento**  
La aplicación debe contener un menú de ayuda integrado y accesible en la propia ejecución de la aplicación en la que se especifique las funcionalidades disponibles, qué hace y cómo funciona cada una de ellas.
- **Output:** Menú de ayuda impreso en consola de comandos.

#### A.3.1.14. Requisito funcional 014

- **Código:** RF-014
- **Título:** Opción fin de ejecución
- **Input:** —
- **Procesamiento**  
El usuario debe de tener la posibilidad de finalizar la ejecución de la aplicación de forma correcta y controlada.
- **Output:** —

#### A.3.1.15. Requisito funcional 015

- **Código:** RF-015
- **Título:** Control de input incorrecto
- **Input:** Valor introducido por teclado.
- **Procesamiento**  
Dada una petición de datos por teclado hecha al usuario se debe comprobar que el valor introducido es correcto acorde a la situación presente. En caso contrario se debe informar por pantalla que el valor introducido no es correcto y requerir un nuevo valor.
- **Output:** —

### A.3.2. Otros requisitos

#### A.3.2.1. Requisito dominio 001

- **Código:** RD-001
- **Título:** Sistemas operativos admitidos
- **Input:** —
- **Procesamiento**  
La aplicación debe de ejecutarse sin inconvenientes en las consolas de comandos de los SO Windows y Linux.
- **Output:** —

#### A.3.2.2. Requisito dominio 002

- **Código:** RD-002
- **Título:** Gestor de dependencias
- **Input:** —
- **Procesamiento**

La aplicación debe ejecutarse en un entorno virtual Conda para evitar conflictos con las versiones de las librerías de las que depende.

- **Output:** —

#### A.3.2.3. Requisito no funcional 001

- **Código:** RNF-001
- **Título:** Mensaje de bienvenida
- **Input:** —
- **Procesamiento**

Al iniciar la aplicación es necesario imprimir un mensaje por consola con el nombre de la aplicación (AIMentalHealth) formado por caracteres ASCII. Posteriormente, se muestran los datos cargados.

- **Output:** Mensaje de bienvenida impreso por consola.

#### A.3.2.4. Requisito no funcional 002

- **Código:** RNF-002
- **Título:** Menú minimalista y secuencial
- **Input:** —
- **Procesamiento**

Debido al entorno de ejecución de la aplicación es necesario implementar un diseño de menú que utilice recuadros, saltos de línea y la alineación para diseñar una visión de menú y opciones similar a la disponible en una aplicación de escritorio estándar.

- **Output:** Vistas de menú estructuradas.

#### A.3.2.5. Requisito no funcional 003

- **Código:** RNF-003
- **Título:** Estructuración de menú de ayuda
- **Input:** —
- **Procesamiento**

El menú de ayuda tiene que ser estructurado por funcionalidad disponible en el menú principal, incluidas las opciones menú de ayuda y fin de ejecución para explicar su contenido. Por cada sección se debe tener un recuadro con el título de la funcionalidad centrado y posteriormente una descripción de la funcionalidad correspondiente. En el caso de tener parámetros o métricas se debe especificar cuáles son las que están disponibles. Es necesario un diseño modulado visualmente para una mejor comprensión visual por parte del usuario al visualizar la consola de comandos.

- **Output:** Instrucciones de la aplicación estructuradas y modularizadas.

**A.3.2.6. Requisito no funcional 004**

- **Código:** RNF-004
- **Título:** Muestra de datos cargados
- **Input:** —
- **Procesamiento**

Al cargar el dataset al iniciar la aplicación o cuando se cambie el dataset se debe imprimir el número de datos destinados a entrenamiento y a evaluación respectivamente en dos recuadros.
- **Output:** Número de datos cargados para entrenamiento y evaluación.

**A.3.2.7. Requisito no funcional 005**

- **Código:** RNF-005
- **Título:** Panel de métricas de entrenamiento/evaluación
- **Input:** —
- **Procesamiento**

Dada una situación de entrenamiento o de evaluación de un modelo se debe mostrar por CMD un panel dividido en secciones y formando tablas para que se pueda saber el valor de una métrica en un golpe visual. Se debe mostrar en la parte de arriba el tiempo de duración de la etapa y debajo dos secciones en horizontal. La sección de la izquierda debe corresponder a la matriz de confusión en formato tabla y la sección derecha las métricas restantes.
- **Output:** Panel de métricas estructurado y modularizado.

**A.3.2.8. Requisito no funcional 006**

- **Código:** RNF-006
- **Título:** Muestra valor parámetro
- **Input:** —
- **Procesamiento**

En la situación en la que se pida al usuario cambiar el valor de un parámetro se deberá mostrar junto a la petición correspondiente el valor actual de dicho parámetro entre paréntesis.
- **Output:** Valor actual entre paréntesis del parámetro.

**A.3.2.9. Requisito no funcional 007**

- **Código:** RNF-007
- **Título:** Muestra ficheros de modelos entrenados

- **Input:** Lista de ficheros entrenados.

- **Procesamiento**

En la situación en la que se requiera la elección de un modelo entrenado se deben mostrar los ficheros existentes simulando una estructura de directorios. Para ello se necesita que el usuario especifique cuántos ficheros quiere por directorio y posteriormente que indique si quiere desplegar un directorio para ver su contenido o seleccionar un fichero (por su posición). Además, los ficheros se deben ordenar por orden alfabético.

- **Output:** Simulación de estructura de directorios por pantalla.

#### A.3.2.10. Requisito no funcional 008

- **Código:** RNF-008

- **Título:** Mensaje de advertencia

- **Input:** —

- **Procesamiento**

La aplicación debe de contener un mensaje de advertencia relacionado con los fines educativos de la aplicación. Es decir, indicar que las clasificaciones obtenidas por la aplicación carecen de valor fuera del área de investigación excepto si se corroboran con un especialista en inteligencia emocional.

Dicho mensaje tiene que estar visible para el usuario en la ejecución del programa (v.g. menú de ayuda).

- **Output:** Mensaje de advertencia por pantalla.





# B

## Estudio modelos neuronales

Documento en el que se reflejan las métricas obtenidas en las pruebas realizadas en la aplicación AIMentalHealth.

### B.1. Introducción

El propósito del presente documento es persistir los resultados obtenidos en las investigaciones oficiales realizadas mediante la aplicación AIMentalHealth.

Teniendo como objetivo conseguir la mejor configuración de modelo para el problema de clasificación presentado por la aplicación se han realizado diversos estudios creando modelos con variaciones en sus parámetros de creación, entrenamiento y evaluación.

Las pruebas se han realizado en una máquina Linux con las siguientes características:

- **GPU:** Información extraída con el mandato `lspci | grep -i vga`. Ver Imagen [B.1](#)
- **RAM:** Información extraída con el mandato `free -h`. Ver Imagen [B.2](#)

```
00:02.0 VGA compatible controller: Intel Corporation HD Graphics 6000 (rev 09)
```

Figura B.1: Información de GPU

	total	used	free	shared	buff/cache	available
Mem:	7,6Gi	218Mi	7,0Gi	1,0Mi	408Mi	7,2Gi
Swap:	4,0Gi	0B	4,0Gi			

Figura B.2: Información de memoria RAM

```

Architecture: x86_64
CPU op-mode(s): 32-bit; 64-bit
Address sizes: 30 bits physical, 48 bits virtual
Byte Order: Little Endian
CPU(s): 4
On-line CPU(s) list: 0-3
Vendor ID: GenuineIntel
Model name: Intel(R) Core(TM) i5-5250U CPU @ 1.60GHz
CPU family: 6
Model: 61
Thread(s) per core: 2
Core(s) per socket: 2
Socket(s): 1
Stepping: 4
CPU(s) scaling MHz: 43%
CPU max MHz: 2700,0000
CPU min MHz: 500,0000
BogoMIPS: 3192.45
Flags: fpu vme de ppe tsc nr pae mce cmf apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_
performance pbs bts rep_good nopl stompology nonstop tsc cpuid aperfperf pni pclmulqdq dtes64 monitor ds_cpl vmx est ts2 sse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vmni flex
priority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap intel_pt xsaveopt dtherm ida arat pln pts md_clear flush_lid
Virtualization features: VT-x
Caches (sum of all):
L1d: 64 KiB (2 instances)
L1i: 64 KiB (2 instances)
L2: 512 KiB (2 instances)
L3: 3 MiB (1 instance)
NUMA:
NUMA node(s): 1
NUMA node0 CPU(s): 0-3
Vulnerabilities:
Itlb multihit: KVM: Mitigation: VMX disabled
L1tf: Mitigation: PTE Inversion; VMX conditional cache flushes, SMT vulnerable
Mds: Mitigation: Clear CPU buffers; SMT vulnerable
Meltdown: Mitigation: PTI
Mmio stale data: Unknown: No mitigations
Retbleed: Not affected
Spec store bypass: Mitigation: Speculative Store Bypass disabled via prctl
Spectre v1: Mitigation: usercopy/swapgs barriers and __user pointer sanitization
Spectre v2: Mitigation: Retpolines, IBPB conditional, IBRS_FW, STIBP conditional, RSB filling, PBSRB-eIBRS Not affected
Srbds: Mitigation: Microcode
Tsx async abort: Not affected

```

Figura B.3: Información de memoria CPU

- CPU: Información extraída con el mandato `lscpu`. Ver Imagen [B.3](#)

## B.2. Modelos

Modelos existentes con su configuración básica identificados con un código único. Ver Tabla [B.1](#)

## B.3. Entrenamiento

Entrenamiento de modelo identificado por el código representativo de este. Todos los entrenamientos han sido realizados con los mismos datos y se muestran los valores de las métricas ofrecidos por la aplicación. Ver Tabla [B.2](#)

## B.4. Evaluación

Evaluación de modelo identificado por el código representativo de este. Todos las evaluaciones han sido realizadas con los mismos datos y se muestran los valores de las métricas ofrecidos por la aplicación. Ver Tabla [B.3](#)

Código	Modelo	Porcentaje datos	Tokenizador	Longitud input	(Nº palabras)	Tamaño de lote	Épocas	Workers	Drop out
0001	BERT	25	BERT	200		16	1	5	0.3
0002	ALBERT	25	ALBERT	200		16	1	5	0.3
0003	ROBERTA	25	ROBERTA	200		16	1	5	0.3
0004	BERT	25	BERT	200		16	1	1	0.3
0005	ALBERT	25	ALBERT	200		16	1	1	0.3
0006	ROBERTA	25	ROBERTA	200		16	1	1	0.3
0007	BERT	25	BERT	200		16	1	3	0.3
0008	ALBERT	25	ALBERT	200		16	1	3	0.3
0009	ROBERTA	25	ROBERTA	200		16	1	3	0.3

Tabla B.1: Configuraciones modelos de inteligencia artificial

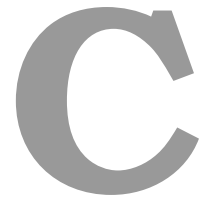
Código	Porcentaje datos	Número datos	Tiempo (segundos)	Matriz de confusión							
				True Negative	False Positive	False Negative	True Positive	Exactitud	Sensibilidad	Precisión	F1
0001	80	46414	55288.00633	22373	904	944	22193	0.96018	0.95920	0.96086	0.96003
0002	80	46414	46716.91757	22761	516	635	22502	0.97520	0.97255	0.97758	0.97506
0003	80	46414	81841.53777	22796	481	575	22562	0.97725	0.97515	0.97913	0.97713
0004	80	46414	51279.79382	22373	904	944	22193	0.96018	0.95920	0.96086	0.96003
0005	80	46414	46136.06444	22761	516	635	22502	0.97520	0.97255	0.97758	0.97506
0006	80	46414	85017.14165	22796	481	575	22562	0.97725	0.97515	0.97913	0.97713
0007	80	46414	55154.40477	22373	904	944	22193	0.96018	0.95920	0.96086	0.96003
0008	80	46414	46183.44849	22761	516	635	22502	0.97520	0.97255	0.97758	0.97506
0009	80	46414	95501.43697	22796	481	575	22562	0.97725	0.97515	0.97913	0.97713

Tabla B.2: Entrenamiento de modelos

Código	Porcentaje datos	Número datos	Tiempo (segundos)	Matriz de confusión						Exactitud	Sensibilidad	Precisión	F1
				True Negative	False Positive	False Negative	True Positive						
0001	20	11604	3550,59255	5653	149	160	5642	0,97337	0,97242	0,97427	0,97335		
0002	20	11604	4066,12965	5735	67	118	5684	0,98406	0,97966	0,98835	0,98399		
0003	20	11604	3538,71513	5764	38	75	5726	0,99026	0,98707	0,99341	0,99023		
0004	20	11604	3504,78262	5653	149	160	5642	0,97337	0,97242	0,97427	0,97335		
0005	20	11604	4033,52740	5735	67	118	5684	0,98406	0,97966	0,98835	0,98399		
0006	20	11604	3527,45223	5764	38	75	5726	0,99026	0,98707	0,99341	0,99023		
0007	20	11604	3556,91044	5653	149	160	5642	0,97337	0,97242	0,97427	0,97335		
0008	20	11604	4022,00663	5735	67	118	5684	0,98406	0,97966	0,98835	0,98399		
0009	20	11604	3539,71039	5764	38	75	5726	0,99026	0,98707	0,99341	0,99023		

Tabla B.3: Evaluación de modelos





# Evolución del Machine Learning

Este documento tiene como objetivo definir y desarrollar los pilares esenciales referentes a la evolución y desarrollo del Machine Learning.

Esta sección incluye información respecto a los fundamentos teóricos seguidos, estudiados y desarrollados para la creación del aplicativo. El estudio de concepto comienza desde la base de qué es la Inteligencia Artificial, continuando con la definición de los componentes de una red neuronal

## C.1. Inteligencia Artificial

Subsección destinada a la explicación del fundamento teórico de qué es la Inteligencia Artificial junto con algunas de sus subcategorías. Además se detalla la categoría del Machine Learning debido a que es el área desarrollada en este proyecto.

Adicionalmente se especifican los distintos paradigmas de aprendizaje que se pueden utilizar en el Machine Learning junto con el concepto de modelo aplicado a la AI y su fundamento matemático.

### C.1.1. ¿Qué es?

La Inteligencia Artificial es la subdisciplina del campo de la informática que busca la creación de máquinas que puedan imitar comportamientos inteligentes.

Los comportamientos inteligentes pueden ser muy diversos: conducir, analizar patrones, reconocer voces, ... En ciertas áreas las inteligencias artificiales tienen mejor rendimiento y resultados que el ser humano. Sin embargo, son incapaces de realizar tareas diversas.

Existen diversas clasificaciones en la inteligencia artificial, siendo una de estas en función de las tareas que puede cumplir:

- **Inteligencia Artificial débil:** Sistemas que únicamente pueden cumplir un conjunto muy limitado de tareas.
- **Inteligencia Artificial fuerte:** Sistemas capaces de aplicarse a una gran variedad de problemas y dominios diferentes.

Por último, se puede dividir la Inteligencia Artificial en múltiples campos o subcategorías:

- **Robótica:** Categoría centrada en la capacidad de moverse y adaptarse al entorno.
- **Natural Language Processing:** Categoría centrada en la capacidad de entender el lenguaje.
- **Voz:** Categoría centrada en la capacidad de poder hablar. Se estudia la conversión de voz a texto y de texto a voz.
- **Machine Learning:** También conocida como aprendizaje automático, categoría que busca como dotar a las máquinas de capacidad de aprendizaje. Se entiende como aprendizaje la generalización del conocimiento a partir de un conjunto de experiencias.

### C.1.2. Machine Learning

El Machine Learning es una disciplina perteneciente al área de la Inteligencia Artificial. Es un componente nuclear debido a que se relaciona con el resto de disciplinas porque estas pueden ser aprendidas en vez de programadas. Se centra en desarrollar algoritmos y técnicas que permiten a los ordenadores aprender y mejorar automáticamente a partir de datos, sin necesidad de ser programadas explícitamente para realizar tareas.



Mediante el uso de modelos matemáticos y algoritmos se analizan datos y se extraen patrones y relaciones. Algunas de las técnicas utilizadas para diversos tipos de aplicaciones son:

- Árboles de decisión.
- Modelos de regresión.
- Modelos de clasificación.
- Técnicas de clusterización.
- Redes neuronales.

Una parte importante e igualmente interesante del Machine Learning son las redes neuronales debido a la capacidad que tienen de aprender de forma jerarquizada. La información se aprende por niveles donde las primeras capas aprenden conceptos muy concretos y en las capas posteriores se utiliza la información aprendida en las capas anteriores para aprender conceptos más abstractos. No existe límite en el número de capas que se pueden utilizar.

Debido al incremento en el número de capas en las redes neuronales se incrementa la complejidad de los algoritmos. Por este motivo se nombra a los citados algoritmos como algoritmos de **deep learning**.

### C.1.3. Paradigmas de aprendizaje

Los paradigmas de aprendizaje son mecanismos que permiten procesar toda aquella información nueva percibida para terminar por transformarla en conocimiento.

Los algoritmos y técnicas utilizados en Machine Learning se pueden clasificar en tres grandes grupos dependiendo del paradigma de aprendizaje que apliquen.

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje reforzado

El **aprendizaje supervisado** se refiere al tipo de aprendizaje que consiste en descubrir la relación existente entre unas variables de entrada y sus correspondientes variables de salida. El aprendizaje surge de enseñar a estos algoritmos cuál es el resultado que se quiere obtener para un determinado valor. Después de repetir el proceso con muchos casos de ejemplo y si esta fase de entrenamiento es













Identifica lenguajes de los siguientes símbolos			
			
			
			

Tabla C.1: Ejemplo práctico aprendizaje no supervisado

satisfactoria, el algoritmo será capaz de dar un resultado correcto incluso cuando procese valores que no haya conocido antes.

El **aprendizaje no supervisado** es el paradigma consistente en producir conocimientos únicamente de los datos que se le proporcionan como entradas, sin necesidad de explicar al sistema qué resultado se quiere obtener. Las tablas C.1 y C.2 son un ejemplo de funcionamiento de este paradigma.

La dificultad de los algoritmos no supervisados es que no tienen ningún ejemplo de respuesta con el que saber si el algoritmo está actuando correctamente. La primera ventaja que se puede deducir de este tipo de aprendizaje es que los conjuntos de datos para entrenar son menos costosos de conseguir (no es necesario tener "la solución").

El **aprendizaje reforzado** se basa en el concepto de aprendizaje a través de la interacción y retroalimentación con el entorno. Un agente aprende a través de ensayo y error, realizando acciones en un entorno y recibiendo retroalimentación en forma de recompensas o castigos. El objetivo del agente es aprender una política óptima, es decir, una estrategia que maximice las recompensas a largo plazo.

El agente toma decisiones en función de su estado actual y de la información disponible, utilizando una función de valor que evalúa las acciones posibles y elige la acción con mayor valor esperado.

#### C.1.4. Modelo

Un modelo es una construcción conceptual simplificada de una realidad más compleja. A través de esta construcción se es capaz de entender dicha realidad y poder utilizarla a favor.

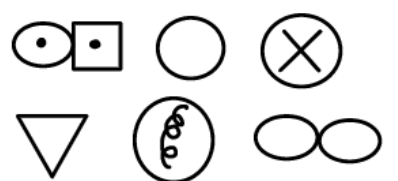
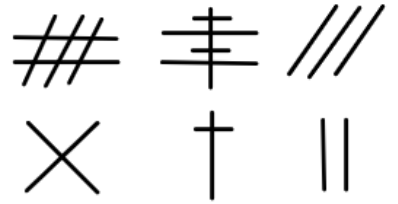



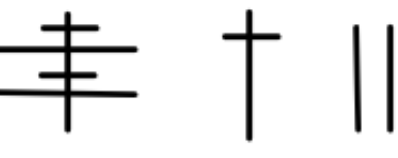
Primera opción	
¿Lenguaje poligonal?	¿Lenguaje lineal?
	
Segunda opción	
¿Lenguaje poligonal relleno?	¿Lenguaje lineal diagonal?
	
¿Lenguaje poligonal vacío?	¿Lenguaje lineal recto?
	

Tabla C.2: Soluciones aprendizaje no supervisado

Un mapa sería un tipo de modelo ya que permite representar de manera simplificada en un plano bidimensional el mundo tridimensional, eliminando detalles innecesarios como texturas o artefactos del entorno. Cuando se hace uso de un diagrama ya sea de los más simples o de los más complejos, también se está haciendo uso de modelos.

Un modelo busca el equilibrio entre aproximarse a representar correctamente la realidad y ser simple para poder utilizarlo.

En la construcción de modelos se puede utilizar la probabilidad creando así modelos probabilísticos. Estos modelos comprimen en base a probabilidades mucha de la variabilidad de la realidad, facilitando la gestión de la información recibida del entorno. Un ejemplo real de aplicación de este tipo de construcciones es el cerebro humano habilitando capacidades como conceptualización, predicción, generalización, razonamiento o aprendizaje. Estas características son las buscadas en el campo del Machine Learning.

En la construcción de un modelo intervienen 3 elementos principales:

1. Datos:

Toma de contacto con la realidad, debido que a partir de estos se extrae toda la información para construir el modelo. Una característica importante es

que los datos son multidimensionales y debido a la gran cantidad de dimensiones que intervienen comúnmente es necesario utilizar las matemáticas para la comprensión de los datos.

## 2. Parámetros:

Valores modificables en el modelo. Aportan flexibilidad en el ajuste del modelo a los datos.

La tarea del Machine Learning es encontrar aquellos algoritmos capaces de aprender los valores óptimos de los parámetros a partir de los datos.

## 3. Error:

Es necesario definir una función de error que indique como el modelo se ajusta (o no) a los datos. Normalmente, cuando se usa algoritmos de aprendizaje supervisado, esta función de error se computa a partir de los datos de salida suministrados y en el caso del aprendizaje no supervisado otras medidas son computadas en base a los datos de entrada.

La señal de error tiene gran importancia debido a que a partir de esta serán ajustados los parámetros del modelo en un proceso que se denomina optimización, conocido popularmente como entrenamiento o ajuste del modelo.

# C.1.5. Fundamento matemático. Regresión lineal

## C.1.5.1. ¿Qué es?

La **regresión lineal** es un método estadístico en el que se busca la relación entre una variable dependiente y una o varias variables independientes teniendo como precepto la existencia de una relación lineal entre las variables.

El principal objetivo de la regresión lineal es entornar la línea recta mejor ajustada a los datos para utilizarse con el propósito de predecir el valor de la variable dependiente. Esta línea se denomina **recta de regresión**.

## C.1.5.2. Explicación matemática

Matemáticamente, en un espacio bidimensional, una recta se define como:

$$y = w_0 + w_1x$$

$w_0 \rightarrow$  Término independiente

$w_1 \rightarrow$  Pendiente

$x \rightarrow$  Variable de entrada

$y \rightarrow$  Variable de salida

Este caso corresponde a la estructura de un modelo de regresión lineal simple debido a que se tiene una única variable de entrada. Si se introducen múltiples variables de entrada, se tiene un modelo de regresión lineal múltiple.

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots$$

El objetivo es encontrar el mejor plano que se ajuste a la nube de datos multidimensional.

### C.1.5.3. Representación

El método más sencillo utilizado para la representación de la combinación lineal de variables por cada dato es de forma vectorial, es decir, generar una matriz.

$$y_1 = w_0 + w_1x_{1,1} + w_2x_{1,2} + w_3x_{1,3} + \dots$$

$$y_2 = w_0 + w_1x_{2,1} + w_2x_{2,2} + w_3x_{2,3} + \dots$$

$$y_3 = w_0 + w_1x_{3,1} + w_2x_{3,2} + w_3x_{3,3} + \dots$$

$$y_4 = w_0 + w_1x_{4,1} + w_2x_{4,2} + w_3x_{4,3} + \dots$$

$$y_5 = w_0 + w_1x_{5,1} + w_2x_{5,2} + w_3x_{5,3} + \dots$$

Cada columna representa una característica de los datos de entrada y cada fila representa cada una de las mediciones que se tienen en el conjunto de datos. La matriz generada sería la matriz  $\mathbf{X}$ .

En cada una de las ecuaciones se tiene la variable  $y$  que se quiere modelar. Agrupando el total de las apariciones se forma un vector de elementos llamado  $\mathbf{Y}$ .

Siguiendo el mismo procedimiento con los factores se genera el vector de parámetros denominado  $\mathbf{W}$ .

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \dots \\ 1 & x_{21} & x_{22} & x_{23} & \dots \\ 1 & x_{31} & x_{32} & x_{33} & \dots \\ 1 & x_{41} & x_{42} & x_{43} & \dots \\ 1 & x_{51} & x_{52} & x_{53} & \dots \end{bmatrix} Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} W^T = [w_0 \quad w_1 \quad w_2 \quad w_3]$$

Se pueden reducir las ecuaciones a una simple ecuación vectorial.

$$Y = XW$$

El uso de las matrices consigue que los entrenamientos sean más eficiente debido a que la GPU está especializada en el procesamiento de matrices.<sup>1</sup>

#### C.1.5.4. Mínimos cuadrados ordinarios

Se quiere encontrar la recta que mejor represente a los datos y se quiere conseguir de forma automática.

Es necesario utilizar el concepto de error. Dibujando una recta y seleccionando un punto de los datos se puede apreciar el valor de *y predicho* por la recta y el valor de *y real* del dato. La distancia entre el valor predicho y el valor real corresponde al **error** del modelo.

Debido a la existencia de más datos (puntos en la gráfica), una opción para la función de coste del modelo es la media de las distancias de los puntos a la recta.

Esta primera aproximación es demasiado simple para conseguir un entrenamiento satisfactorio del modelo. Por tanto, se puede utilizar una evolución de esta función de coste que consiste elevar al cuadrado los errores. Conceptualmente, elevando al cuadrado las distancias (errores) se agrava la penalización de los puntos más cercanos a la recta y a su vez se minimiza el impacto producido por los puntos cercanos a la recta. Esta función de coste se denomina **error cuadrático medio**.

---

<sup>1</sup>En las próximas secciones se utiliza la regresión lineal simple para simplificar las explicaciones pero son aplicables a la regresión lineal múltiple.

$$\text{Error cuadrático medio} \rightarrow \text{media}((y_r - y_e)^2)$$

Expresando la función de coste con vectores y derivando la función igualada a 0 se consigue la expresión matemática que asegura el mínimo valor de error.

$$\text{Error cuadrático medio vectorial} \rightarrow (Y - X \cdot W)^T \cdot (Y - X \cdot W)$$

$$(Y - X \cdot W)^T \cdot (Y - X \cdot W)$$

$$Y^T \cdot Y - W^T \cdot X^T \cdot Y - Y^T \cdot X \cdot W + W^T \cdot X^T \cdot X \cdot W$$

$$-2 \cdot X^T \cdot Y + 2 \cdot X^T \cdot X \cdot W = 0$$

$$\text{Mínimo error cuadrático medio} \rightarrow W = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

Aplicando este método se calcula la transpuesta de una matriz y la inversa de una operación de matrices. Cálculos ineficientes en un ordenador.

#### C.1.5.5. Teoría matemática sobre funciones

Una propiedad de las funciones convexas [figura C.1] indica que de encontrar un punto mínimo se asegura que dicho punto es el mínimo global de la función (punto mínimo). En el caso de las funciones cóncavas se puede invertir la función y aplicar la misma propiedad que en la función convexa.

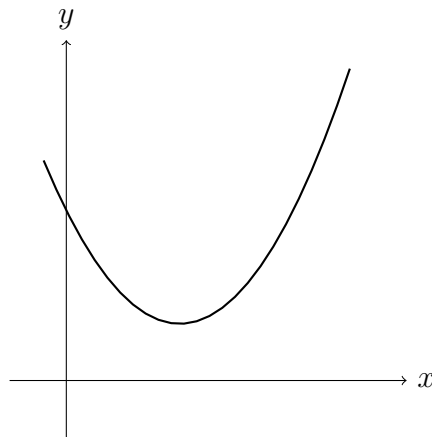


Figura C.1: Representación gráfica función convexa

Por otro lado, existen funciones no convexas [figura C.2] cuya forma es problemática debido a que no cumplen la propiedad anteriormente especificada de las funciones convexas. Es decir, se puede encontrar un punto mínimo que no corresponda al mínimo global de la función.

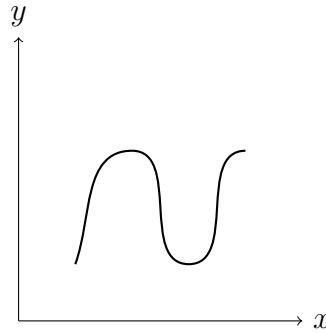


Figura C.2: Representación gráfica función no convexa

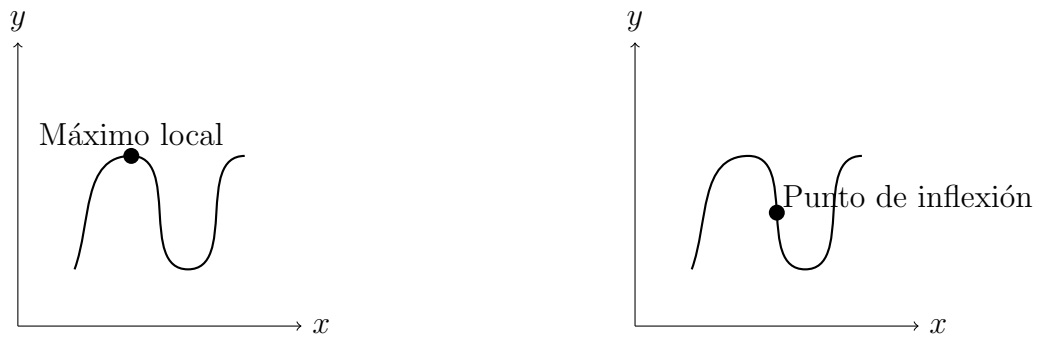


Figura C.3: Zonas con pendiente nula

El problema surge al aplicar el método de búsqueda de punto mínimo en la gráfica.

1. **Derivar la función:** La función derivada indica la pendiente de la función.
2. **Igualar a cero:** Resolver dicha ecuación para así encontrar el punto donde la pendiente es nula.

$$f'(x) = 0$$

Es posible hallar múltiples puntos mínimos y por tanto múltiples ecuaciones que resolver. Además existen más zonas cuyo valor de pendiente es nulo [figura C.3].

Por tanto, queda demostrado que un sistema de ecuaciones es ineficiente de resolver.



### C.1.5.6. Descenso del gradiente

Dado un modelo independientemente del tipo se tienen parámetros y si se modifica su valor es posible variar el error del modelo.

La función de descenso del gradiente indica cuál es el error para cada una de las combinaciones de los parámetros y matemáticamente se corresponde con una función  $f'(x)$ .

La información proporcionada por el cálculo de la derivada es aprovechada para encontrar mínimos locales debido a que la derivada ofrece información de la pendiente de la función. En primer lugar se evalúa la inclinación para encontrar la mayor pendiente. Posteriormente se avanza una distancia en esa dirección para detenerse y finalmente repetir el proceso en la nueva posición.

La lógica del algoritmo del gradiente comprende los siguientes pasos:

1. Localización de la mayor pendiente en la posición actual.
2. Avance en la dirección establecida.
3. Detención en la nueva posición y repetición del proceso.

Para traducir esta intuición al modelo matemático se supone que la función de costes comprende una superficie tridimensional en la que los ejes  $x$ ,  $z$  son los dos parámetros y la  $y$  el error del modelo. Se utilizan dos parámetros para poder visualizar el caso de estudio en tres dimensiones, sin embargo, en Machine Learning se trabaja con modelos de miles de parámetros.

Al iniciar el entrenamiento los parámetros se inicializan con un valor aleatorio (equivalente a iniciar en un punto al azar de la superficie). Se evalúa la posición seleccionada correspondiendo al cálculo de la derivada de la función en dicho punto. Al ser la función multidimensional, es necesario calcular derivadas parciales para cada uno de los parámetros y cada valor indica cuál es la pendiente en el eje de dicho parámetro.

Conjuntamente todas las direcciones (derivadas parciales) conforman un vector que indica la dirección hacia la que la pendiente asciende. Este vector se denomina **gradiente**.

$$\begin{bmatrix} \frac{\partial \text{error}}{\partial \theta_1} \\ \frac{\partial \text{error}}{\partial \theta_2} \end{bmatrix} = \nabla f$$

El objetivo principal consiste en descender y se puede utilizar este vector para el sentido opuesto. Es decir, el gradiente indica cómo se tendría que actualizar los parámetros para ascender. Por ende, el siguiente paso es restar el valor del gradiente.

$$\theta := \theta - \nabla f$$

Se tendría como resultado un nuevo conjunto de parámetros y por tanto se estaría situado en un nuevo lugar de la función, donde se repetirá el proceso múltiples veces hasta llegar a una zona donde moverse ya no suponga una variación notable del coste. Es decir, la pendiente es próxima a nula y lo más probable es que se esté situado en un mínimo local. Con esto se ha minimizado el coste del modelo.

Completando el algoritmo se incorpora un parámetro denominado **ratio de aprendizaje**. El ratio de aprendizaje define cuánto afecta el gradiente a la actualización de los parámetros en cada iteración, es decir, cuánto se progresa en cada paso.

$$\begin{array}{l} \text{Repetir hasta convergencia} \\ \theta := \theta - \alpha \cdot \nabla f \end{array}$$

Si el valor del ratio de aprendizaje es ínfimo se avanzará hacia un punto de mínimo coste con un gran coste de iteraciones. Como resultando se obtendría un algoritmo ineficiente y se podría provocar el estancamiento en un punto local ineficiente.

Si el valor del ratio de aprendizaje es elevado, en cada iteración se avanza demasiado y el punto es incapaz de introducirse correctamente dentro de la zona de mínimo coste siendo imposible para el algoritmo converger en dicho punto. El efecto producido sería un bucle infinito de proceso de optimización.

La correcta configuración del ratio de aprendizaje es fundamental para poder hacer que el algoritmo trabaje correctamente. Existen diferentes técnicas que sirven para ajustar este parámetro de forma dinámica al igual que mejoras en las fórmulas vistas para que el movimiento del punto sea más eficiente.

## C.2. Redes neuronales

En esta sección el objetivo es reflejar un primer acercamiento al área de las redes neuronales explicando su estructura desde la base hasta las métricas comunes utilizadas en este campo para la medición del rendimiento; abarcando además los algoritmos utilizados en el entrenamiento y evaluación de los modelos.

## C.2.1. Neurona

### C.2.1.1. ¿Qué es?

Una neurona es una unidad básica de procesamiento ubicada dentro de una red neuronal.

Similar a una neurona biológica, estas neuronas tienen conexiones de entrada a través de los que reciben estímulos externos. Con estos valores la neurona realizará un cálculo interno y generará un valor de salida [figura C.4]. Una neurona es en esencia una denominación alternativa de una función matemática.

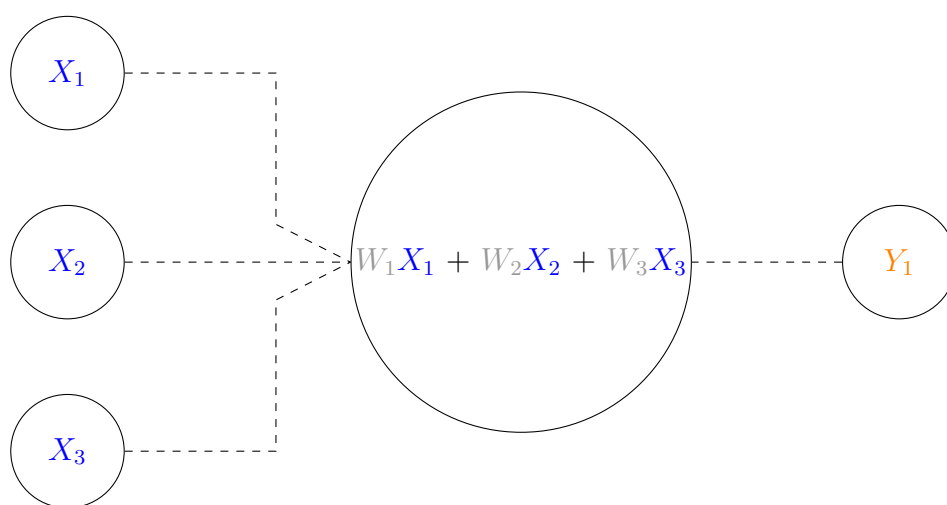


Figura C.4: Neurona suma ponderada

### C.2.1.2. Funcionamiento de la neurona

Internamente la neurona utiliza todos los valores de entrada para realizar una suma ponderada de estos. La ponderación de cada una de las entradas es definida por el peso asignado a cada conexión de entrada [figura C.5]. Es decir, cada conexión que llega a la neurona tendrá asociado un valor que servirá para definir con qué intensidad afecta cada variable de entrada a la neurona.

Estos pesos son los parámetros del modelo y serán los valores que se podrán ajustar para que la red neuronal pueda operar.

Se puede decir que internamente una neurona realiza un modelo de regresión lineal. Se tienen variables de entrada que definen una recta o un hiper plano. Se puede variar la inclinación utilizando los parámetros.

En el modelo de regresión lineal se tenía un término independiente utilizado para el desplazamiento vertical de las rectas. La neurona también posee el mismo

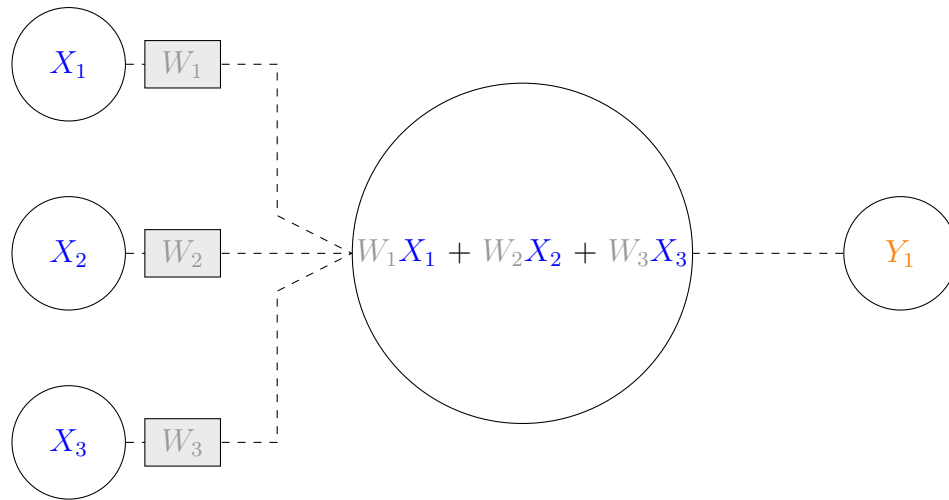


Figura C.5: Pesos asociados a cada dato de entrada

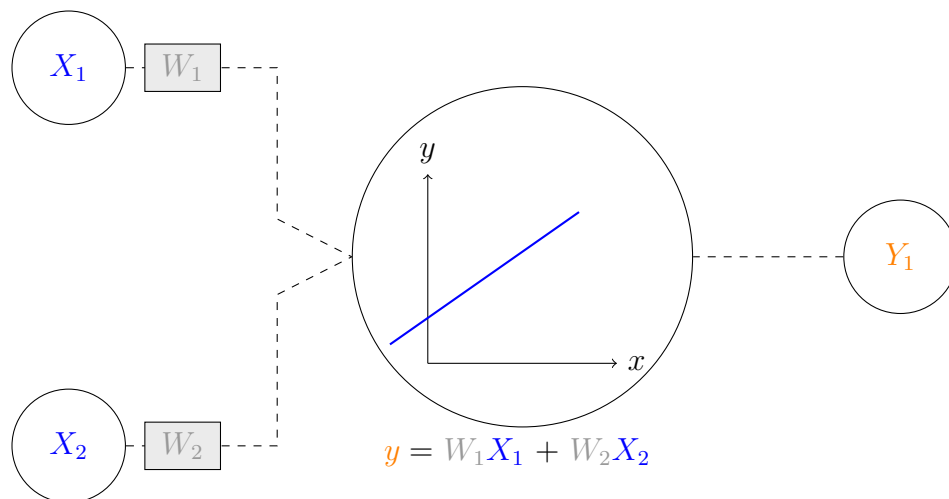


Figura C.6: Regresión lineal en neurona

término que ofrece el control para mover la función. A este valor se le denomina **sesgo** (bias) y se representa como otra conexión a la neurona pero en el que la variable siempre está asignada a 1 [figura C.7]. El valor de entrada se puede controlar manipulando el valor del parámetro de sesgo. De esta forma la neurona actuaría exactamente como el modelo de regresión lineal.

Existen limitaciones para una sola neurona como implementar una puerta XOR. Esta limitación de las neuronas de poder resolver el problema de la puerta XOR se conoce desde 1969 e ilustra la necesidad de combinar varias neuronas para construir modelos más complejos.

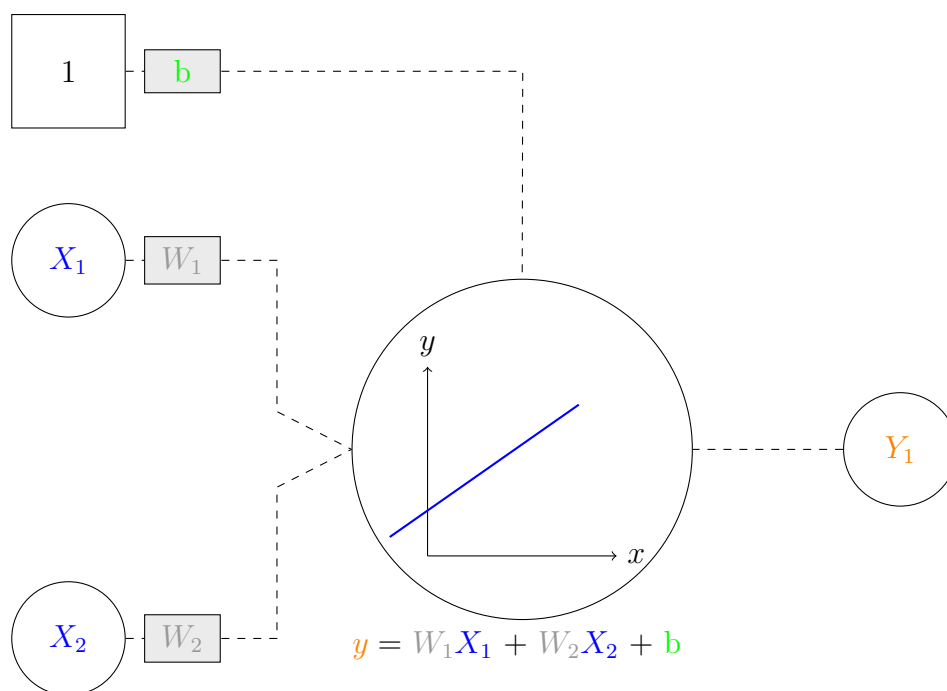


Figura C.7: Sesgo de la neurona

## C.2.2. Red neuronal

### C.2.2.1. Formación

La formación de una red neuronal se organiza desde dos perspectivas. La primera perspectiva consiste en posicionar en la misma columna denominada **capa**. Dos neuronas que se encuentran en la misma capa recibirán la misma información de entrada de la capa anterior y los cálculos que realicen pasarán a la capa siguiente.

La primera capa dónde están las variables de entrada se denomina **capa de entrada** y la última capa recibe el nombre de **capa de salida** [figura C.8]. Las capas intermedias con denominadas **capas ocultas**.

### C.2.2.2. Funcionamiento red neuronal

Cuanto más capas se añaden más complejo puede ser el conocimiento que se elabora. Esta profundidad en la cantidad de capas es lo que da nombre al aprendizaje profundo (**Deep Learning**).

Para alcanzar este aprendizaje profundo se requiere conectar múltiples neuronas de forma secuencial y lo que hace cada una de las neuronas es un problema de regresión lineal. Es decir, planteándolo de manera matemática, se concatenan

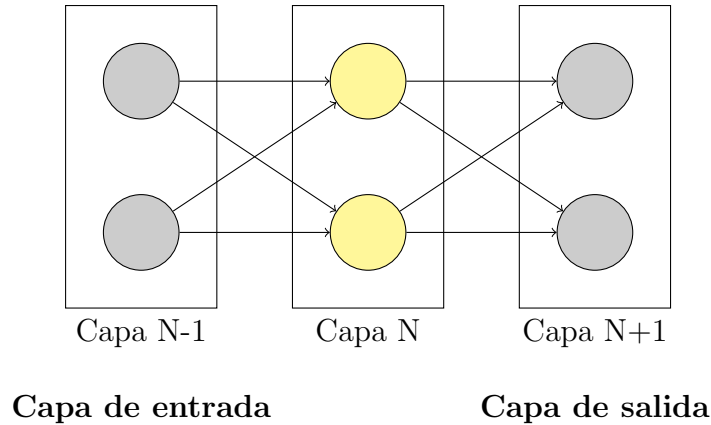


Figura C.8: Estructura básica red neuronal

diferentes operaciones de regresión lineal. El problema reside en que matemáticamente se puede comprobar que el efecto de sumar muchas operaciones de regresión lineal, es decir, sumar diversas líneas rectas; equivale a solamente haber hecho una única operación obteniendo como resultado una nueva línea recta. Con el planteamiento actual de la red se tiene como efecto hacer que la totalidad de la estructura que se quería conseguir colapse hasta ser equivalente a tener una única neurona.

Para conseguir que la red no colapse se necesita que esta suma dé como resultado algo diferente a una línea recta. Para esto se necesita que cada una de estas líneas sufra alguna manipulación no lineal que las distorsione. Con este objetivo se utilizan las funciones de activación.

### C.2.2.3. Función de activación

El funcionamiento actual de la neurona es calcular como valor de salida una suma ponderada de las entradas. El siguiente paso es pasar dicho valor de salida por la función de activación. El efecto de la función de activación sobre el resultado es distorsionar el valor de salida añadiéndole deformaciones no lineales para que así se pueda encadenar de forma efectiva la computación de varias neuronas [figura C.9].

Existen múltiples funciones de activación:

- **Función de activación escalonada**

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$$

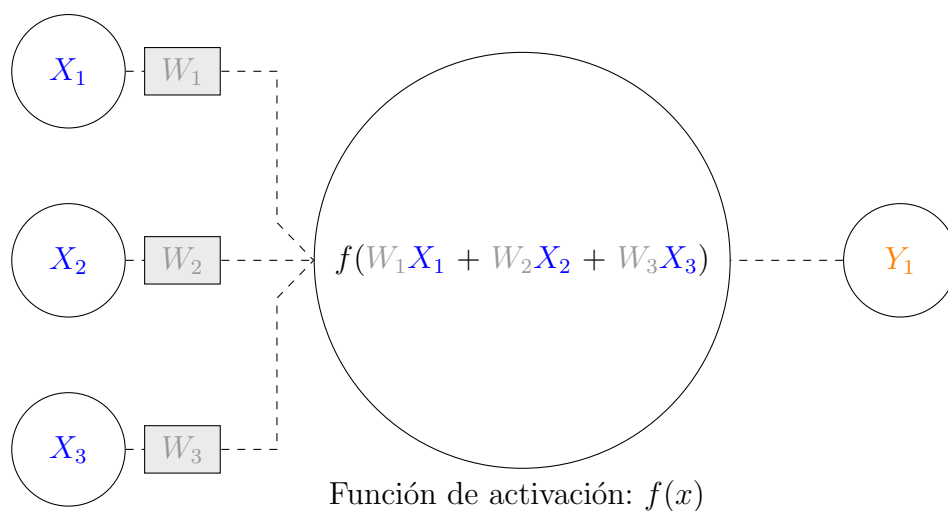


Figura C.9: Función de activación

Para un valor de entrada mayor al umbral el output es 1 y 0 si es inferior o igual a 0. Es denominada escalonada porque el cambio de valor se produce instantáneamente y no de forma gradual produciendo así un escalón, algo que no favorece el aprendizaje. Esta función de activación no posee mucho valor práctico.

- **Función de activación sigmoide**

$$f(x) = \frac{1}{1 + e^x}$$

La distorsión producida hace que los valores elevados se saturan en 1 y los valores pequeños se saturan en 0. Con esta función no solo se consigue añadir la deformación buscada si no que además sirve.

- **Función de activación tangente hiperbólica**

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Forma similar a la función de activación sigmoide pero cuyo rango varía de -1 a 1.

- **Función de activación unidad rectificada lineal**

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x > 0 \end{cases}$$

Se comporta como una función lineal cuando es positiva y constante a cero cuando el valor de entrada es negativo.

Cada una de estas funciones además de aportar la no linealidad que se busca también ofrecen diferentes beneficios dependiendo de cuando se utilicen.

### C.2.3. Backpropagation

#### C.2.3.1. Historia

En el año 1986 se publicó el artículo científico que cambiaría el curso de las redes neuronales. Cuando solo se trabaja con una única neurona, esta puede considerarse un tipo muy simple de red neuronal y es de hecho la precursora de las redes neuronales actuales. Este tipo de red se conoce desde los años 50 como **perceptrón**.

En los primeros años de vida del perceptrón se creyó que este sería el algoritmo que resolvería una gran variedad de problemas aunque pronto se hizo evidente algunas de sus importantes limitaciones como el caso de la simulación de la puerta lógica XOR. Esta arquitectura de una única neurona sólo es capaz de resolver problemas lineales siendo necesario combinar más neuronas para poder abarcar problemas no lineales más complejos. El problema reside en que el algoritmo de aprendizaje automático que se venía utilizando para entrenar el perceptrón no era extensible a otros tipos de redes más complejas. Se querían redes neuronales pero no se conocía como entrenarlas.

Fue tal la repercusión de este problema que tras publicarse el libro *Perceptrons* de Minsky donde se mostraba matemáticamente las importantes limitaciones del perceptrón. Hubo un corte repentino en la financiación de proyectos de inteligencia artificial y más específicamente en aquellos relacionados con los sistemas de redes neuronales. Un periodo de más de 15 años conocido como el invierno de la inteligencia artificial.

Con la publicación de un trabajo firmado por Ramón Heart, Hilton y Williams y basado en otros avances en diferenciación automática se mostraría experimentalmente como usando un nuevo algoritmo de aprendizaje se podría conseguir que una red neuronal autoajustara sus parámetros para así aprender una representación interna de la información que estaba procesando. El nombre de este algoritmo es **backpropagation**.



### C.2.3.2. Idea del algoritmo

Utilizando el descenso del gradiente se consigue una estrategia perfecta para ajustar los parámetros de (en su momento) una regresión lineal. La estrategia utilizada es evaluar el error del modelo en un punto de inicio y calcular las derivadas parciales en dicho punto. Con esto se tenía un vector de direcciones que indicaba la pendiente de la función hacia donde el error se incrementaba lo que se llamaba gradiente y con eso se movía en dirección contraria, teniendo una forma por la cual, interactivamente, se puede reducir el error del modelo.

$$\begin{bmatrix} \frac{\partial \text{error}}{\partial \theta_1} \\ \frac{\partial \text{error}}{\partial \theta_2} \end{bmatrix} = \nabla f$$

$$\theta := \theta - \nabla f$$

Para aplicar el descenso del gradiente se necesita el gradiente ( $\nabla f$ ) siendo este el vector que contiene las pendientes para cada una de las dimensiones de  $f$ . Para la regresión simple se tienen solo dos parámetros que afectan directamente al resultado del modelo con lo cual calcular el gradiente para cada uno de los parámetros. Consiste en responder a la pregunta:

*¿Cómo varía el coste ante un cambio del parámetro  $W$ ?*

Esta pregunta que matemáticamente se responde con las derivadas parciales. Derivada parcial de la función de coste respecto a cada uno de los parámetros:

$$\frac{\partial C}{\partial W}$$

Sin embargo, cuando se trabaja con redes neuronales es más complejo. El **gradiente** es el mismo concepto, como varía el coste se varía uno de los parámetros. En una red neuronal la forma en la que variar un parámetro puede afectar al resultado final y por tanto al coste de la red no es intuitivo. Se forma una cadena de responsabilidades que provoca que la derivada de cómo varía el coste cuando se varía algunos parámetros sea más compleja de calcular. Este valor es ofrecido por el algoritmo de **backpropagation**.

Se utiliza el descenso del gradiente para optimizar la función de coste haciendo uso de la técnica de backpropagation para calcular el vector de gradiente dentro de la complejidad de la arquitectura de la red neuronal.

Para analizar un resultado de error, una estrategia inteligente a seguir sería la de analizar toda la cadena de responsabilidades que ha afectado al resultado. Si se

encuentra una neurona que haya tenido influencia en el error obtenido, entonces se deberá de responsabilizar a dicha neurona con parte de ese error.

Este análisis de cuánta responsabilidad tiene cada neurona tiene sentido hacerlo hacia atrás, desde la señal de error hacia las primeras capas. Esto tiene sentido hacerlo porque en una red neuronal el error de las capas anteriores depende directamente del error de las capas posteriores.

Existe una lógica que se puede explotar para determinar qué parte de culpa tiene cada neurona en el resultado final mediante la retropropagación de errores. Siguiendo esta estrategia de ir hacia atrás también se está trabajando de forma eficiente.

Empezando a analizar desde la señal de error, la primera parada sería la última capa. En función de cuánto se haya implicado cada neurona en generar el resultado final se puede responsabilizar a cada una de estas neuronas de un porcentaje del error, repartirlo; y será lo que se utiliza para calcular cuánto hay que modificar cada parámetro en dicha neurona. Lo destacable es que una vez se han imputado los errores a las neuronas de dicha capa se puede proceder a repetir el mismo proceso de antes como si este fuera el error final de la red. Es decir, asumiendo que esta es la nueva última capa.

Así aplicar **backpropagation** es operar siempre de forma recursiva capa tras capa moviendo el error hacia atrás. Con este razonamiento cuando se llegue a la primera capa se habrá obtenido cuál es el error para cada neurona y para cada uno de sus parámetros solamente propagando una única vez el error hacia atrás. Esto hace de este algoritmo una estrategia muy eficiente.

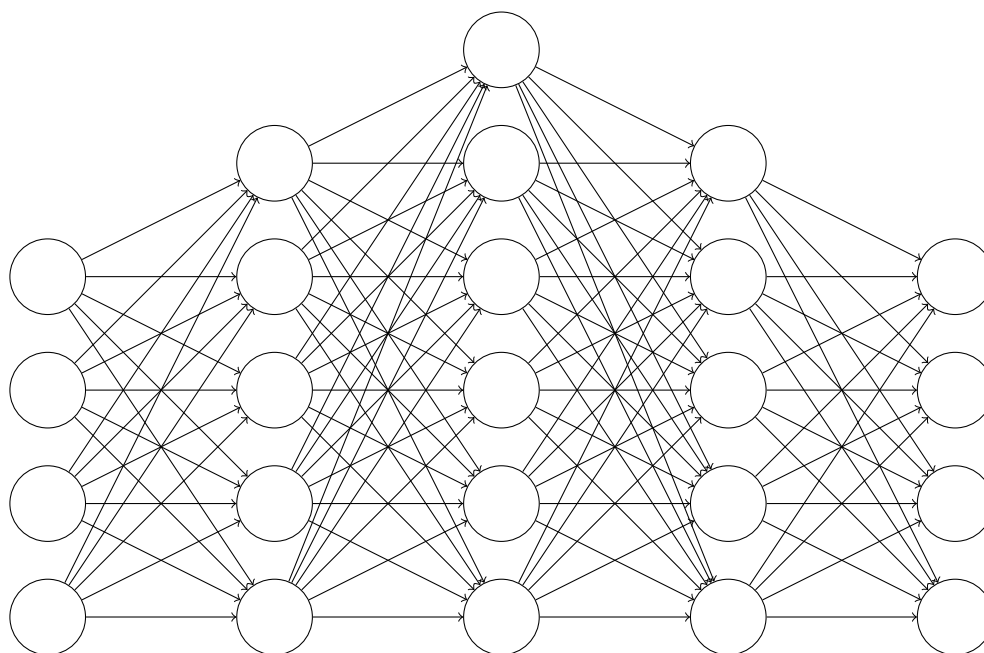
Antes de existir backpropagation el cálculo de la responsabilidad de cada neurona en el resultado se hacía siguiendo una estrategia de fuerza bruta (perturbación aleatoria), preguntándose para cada neurona cómo variaba el coste cuando se introducía algún pequeño cambio. En la práctica este procedimiento requería computar muchas veces pasos hacia adelante en la red siendo así muy ineficiente de entrenar. Por el contrario lo propuesto en el algoritmo de backpropagation permite obtener todos esos errores con un único pase hacia atrás.

Esos errores son los que usarán para calcular las derivadas parciales de cada parámetro de la red conformando así el vector gradiente. Esto es lo que necesita el algoritmo del descenso del gradiente para minimizar el error y por tanto entrenar a la red.

### C.2.3.3. Matemática del algoritmo

Una red neuronal recién creada tiene sus parámetros inicializados de forma aleatoria significando eso que el resultado que se obtiene para un input cualquiera es aleatorio. Cuando se compare con el valor real probablemente la predicción

haya sido nefasta y que la función de coste le asignara un error muy elevado. Se usa dicho error para entrenar a la red [figura C.10].



$$X \rightarrow \text{Red neuronal} \rightarrow Y_p \longrightarrow \text{ERROR}$$

Figura C.10: Esquema red neuronal

Lo que se quiere calcular para cada parámetro dentro de la red neuronal es la derivada parcial del coste respecto a cada uno de los parámetros de la red. En una red neuronal hay dos tipos de parámetros: los **pesos  $\mathbf{W}$**  y el **término de sesgo  $\mathbf{b}$** . Esto significa que en realidad se tendrá que calcular dos tipos diferentes de derivadas parciales: una respecto al parámetro  **$\mathbf{W}$**  y otra respecto al parámetro  **$\mathbf{b}$** .

Se empieza a trabajar hacia atrás. Comenzando a calcular la derivada de los parámetros de la última capa. Para mejor comprensión se marca con un superíndice el número de capa a la que pertenece el parámetro. Si la red neuronal tiene  **$L$**  capas entonces las derivadas que se quieren calcular para los parámetros de la última capa son las visibles en la figura C.11

Para calcular esta derivada es importante analizar cuál es el camino que conecta el valor del parámetro y el coste final. En la última capa este camino no es muy largo aunque sí tiene varios pasos.

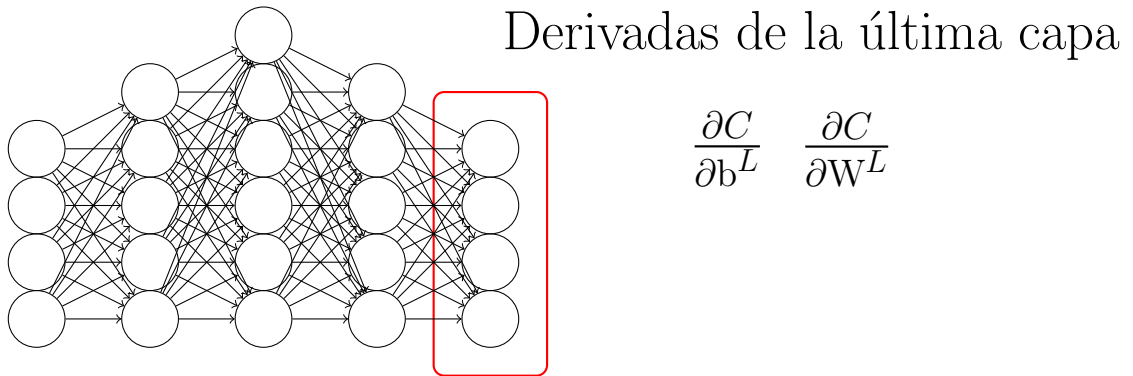


Figura C.11: Derivadas parciales red neuronal

Recordando el funcionamiento de la neurona el parámetro  $\mathbf{W}$  participa en una suma ponderada. Se va a referir a esta suma como  $\mathbf{Z}$  que posteriormente sería pasada por la función de activación  $\mathbf{a}$  y el resultado de las activaciones de la neurona en la última capa conformaría el resultado. Por último, sería evaluado por la función de coste  $\mathbf{C}$  para determinar así el error de la red [figura ??].

Lo descrito anteriormente donde el resultado de una función pasado por otra y nuevamente por una tercera función es denominado composición de funciones y, para calcular la derivada de una composición de funciones, es necesario aplicar la herramienta de cálculo matemático llamado Chain Rule (Regla de la Cadena). **Chain Rule** indica que para calcular la derivada de una composición de funciones

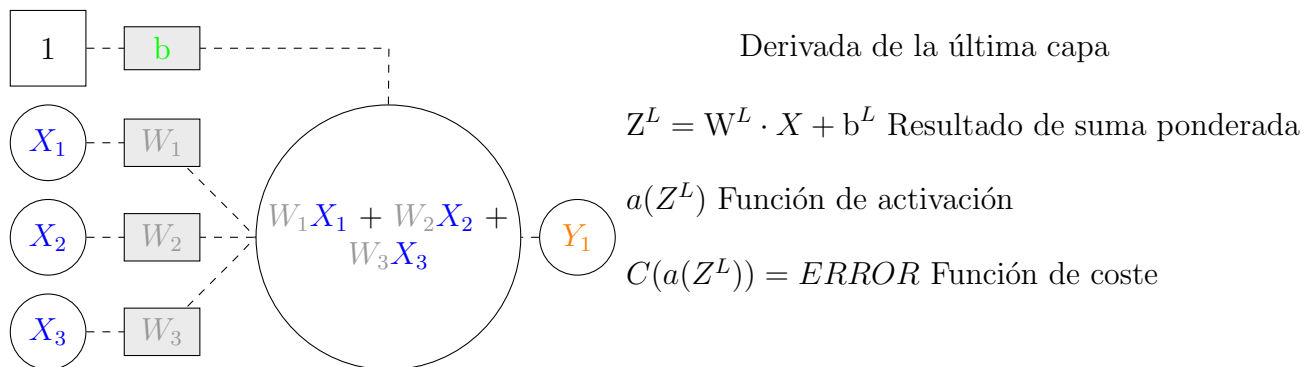


Figura C.12: Funcionamiento matemático de la neurona

simplemente se tienen que multiplicar cada una de las derivadas intermedias.

La derivada del coste respecto a  $\mathbf{W}$  y del coste respecto a  $\mathbf{b}$  influyen a través de la composición anotada con anterioridad. Aplicando la herramienta matemática para la resolución se necesitan las derivadas intermedias.

$$Z^L = W^L \cdot a^{L-1} + b^L \quad C(a^L(Z^L))$$

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial W^L}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial b^L}$$

Estas derivadas son fáciles de calcular:

### Derivada del coste respecto a la activación

$$\boxed{\frac{\partial C}{\partial a^L}}$$

Como varía el coste de la red cuando se varía el output (la activación de las neuronas) en la última capa. En la última capa, las activaciones de las neuronas son la salida de la red.

Realmente el cálculo consiste en la derivada de la función de coste con respecto al output de la red neuronal. Teniendo la siguiente función de coste:

Función de coste: error cuadrático medio

$$C(a_j^L) = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$

La derivada de la función de coste respecto al output de la red es la siguiente:

$$\frac{\partial C}{\partial a_j^L} = (a_j^L - y_j)$$

### Derivada del coste respecto a Z

$$\frac{\partial a^L}{\partial Z^L}$$

Como varía el output cuando se varía la suma ponderada de una neurona en la última capa. Lo único que separa a  $\mathbf{Z}$  con la activación de la neurona es la función de activación.

Realmente el cálculo consiste en la derivada de la función de activación. Teniendo en cuenta la siguiente función de activación:

Función de activación: Sigmoide

$$a^L(Z^L) = \frac{1}{1+e^{-Z^L}}$$

La derivada de la activación con respecto a  $\mathbf{Z}$  es la siguiente:

$$\frac{\partial a^L}{\partial Z^L} = a^L(Z^L) \cdot (1 - a^L(Z^L))$$

### Derivada de $\mathbf{Z}$ con respecto a $\mathbf{W}$ y derivada de $\mathbf{Z}$ con respecto de $\mathbf{b}$

$$\frac{\partial Z^L}{\partial W^L} \frac{\partial Z^L}{\partial b^L}$$

Como varía la suma ponderada  $\mathbf{Z}$  con respecto a una variación de los parámetros. La red neuronal tiene dos tipos de parámetros que son  $\mathbf{W}$  (los pesos) y  $\mathbf{b}$ , el término de sesgo.

Teniendo la siguiente suma ponderada:

$$Z^L = \sum_i a_i^{L-1} \cdot W_i^L + b^L$$

La derivada de la suma ponderada con respecto al término de sesgo es 1 porque el término de sesgo es independiente provocando una derivada constante (1).

$$\frac{\partial Z^L}{\partial b^L} = 1$$

La derivada de la suma ponderada con respecto a  $\mathbf{W}$  es el valor de entrada a la neurona que conecta esa conexión para la cuál el parámetro hace referencia. En este caso, los valores de entrada de la neurona  $a^L$  se corresponde con el output (la salida) de las neuronas de la capa anterior, la capa  $\mathbf{L} - 1$ .

$$\frac{\partial Z^L}{\partial W^L} = a_i^{L-1}$$

Finalmente la solución buscada para los parámetros de la última capa se calcula computando las siguientes fórmulas donde cada una de las derivadas parciales calculadas se multiplican unas con otras.

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial W^L}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial b^L}$$

$$\frac{\partial a^L}{\partial Z^L} = a^L(Z^L) \cdot (1 - a^L(Z^L)) \quad \frac{\partial C}{\partial a_j^L} = (a_j^L - y_j) \quad \frac{\partial Z^L}{\partial b^L} = 1 \quad \frac{\partial Z^L}{\partial W^L} = a_i^{L-1}$$

Usando la intuición en el próximo bloque señalado representa cómo varía el error en función del valor de  $\mathbf{Z}$  que es la suma ponderada calculada dentro de la neurona.

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial a^L} \cdot \underbrace{\frac{\partial a^L}{\partial Z^L}}_{\frac{\partial C}{\partial Z^L}} \cdot \frac{\partial Z^L}{\partial W^L}$$

Es decir, lo que indica esta derivada es en qué grado se modifica el error (el coste) cuando se produce un pequeño cambio en la suma de la neurona. Si esta derivada es elevada significa que ante un pequeño cambio en el valor de la neurona este se verá reflejado en el resultado final. Por el contrario si la derivada es pequeña no importa cómo se varía el valor de la suma ya que éste no afectará significativamente al error de la red.

La siguiente derivada es la que indica qué responsabilidad tiene la neurona en el resultado final y por tanto en el error:

$$\frac{\partial C}{\partial Z^L}$$

Si la neurona es una parte responsable del error final entonces se debería utilizar esta información para reducir parte de ese error. Por esta lógica es por lo que comúnmente se suele interpretar esta derivada como el error imputado a esta neurona. El citado error de la neurona se representa con el símbolo  $\delta^L$ .

Error imputado a la neurona

$$\frac{\partial C}{\partial Z^L} = \delta^L$$

Por tanto, para simplificar se puede reestructurar la expresión inicial en función del error de las neuronas de la capa  $\mathbf{L}$ . Por un lado se tiene que la derivada

del coste respecto al término de sesgo  $\mathbf{b}$  es igual al error de las neuronas y, por otra parte, la derivada del coste respecto a los pesos  $\mathbf{W}$  es igual al error de las neuronas multiplicado por la activación de la capa previa.

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial W^L} \rightarrow \frac{\partial C}{\partial W^L} = \delta^L \cdot \frac{\partial Z^L}{\partial W^L} \rightarrow \frac{\partial C}{\partial W^L} = \delta^L \cdot a_i^{L-1}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial b^L} \rightarrow \frac{\partial C}{\partial b^L} = \delta^L \cdot \frac{\partial Z^L}{\partial b^L} \rightarrow \frac{\partial C}{\partial b^L} = \delta^L \cdot a_i^{L-1}$$

Se han deducido tres expresiones diferentes que permiten obtener las derivadas parciales buscadas para la última capa. Una que indica cómo calcular el error de las neuronas en la última capa y otras dos para cada una de las derivadas parciales.

$$\delta^L = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$$

$$\frac{\partial C}{\partial b^L} = \delta^L \cdot 1$$

$$\frac{\partial C}{\partial W^L} = \delta^L \cdot a_i^{L-1}$$

Si se quiere calcular los parámetros de la capa anterior, la capa  $L - 1$ , aplicando el mismo razonamiento de Chain Rule la expresión se prolonga hasta convertirse en lo siguiente (se tiene el coste, la activación de la última capa, la suma ponderada, la transformación a la capa anterior y la función de activación):

$$\frac{\partial C}{\partial W^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial Z^{L-1}} \cdot \frac{\partial Z^{L-1}}{\partial W^{L-1}}$$

$$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial Z^{L-1}} \cdot \frac{\partial Z^{L-1}}{\partial b^{L-1}}$$

Se aplica la Chain Rule a la siguiente composición:

$$C(a^L(W^L \cdot a^{L-1}(W^{L-1} \cdot a^{L-2} + b^{L-1}) + b^L))$$

Como se está retrocediendo en las capas, partes de la ecuación están ya calculadas:



	<div style="background-color: #808080; color: white; padding: 2px; font-size: 0.8em; margin-bottom: 5px;">Error de la capa L <math>\delta^L</math></div> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <math display="block">\frac{\partial C}{\partial W^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}</math> </div>	$\frac{\partial Z^L}{\partial a^{L-1}}$	<div style="background-color: #808080; color: white; padding: 2px; font-size: 0.8em; margin-bottom: 5px;">Derivada función activación</div> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <math display="block">\frac{\partial a^{L-1}}{\partial Z^{L-1}}</math> </div>	$\frac{\partial Z^{L-1}}{\partial W^{L-1}} \rightarrow a^{L-2} \text{ Activación de la capa previa}$
	$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$	$\frac{\partial Z^L}{\partial a^{L-1}}$	$\frac{\partial a^{L-1}}{\partial Z^{L-1}}$	$\frac{\partial Z^{L-1}}{\partial b^{L-1}} \rightarrow 1$

Solo falta calcular la siguiente derivada:

$$\frac{\partial Z^L}{\partial a^{L-1}}$$

Como varía la suma ponderada de una capa cuando se varía el output de una neurona en la capa previa. Es la matriz de parámetros  $\mathbf{W}$  que conecta ambas capas ( $W^L$ ).

Realmente su efecto es mover el error de una capa a la capa anterior, distribuyendo el error en función de cuáles son las ponderaciones de las conexiones.

Con esto ya se tendrá nuevamente una expresión a partir de la cual obtenga las derivadas parciales buscadas.

Nuevamente el bloque señalado se convierte en la siguiente derivada que vuelve a representar el error de las neuronas de esta capa.

	<div style="background-color: #ff8c00; color: white; padding: 5px; font-size: 0.8em; margin-bottom: 10px;"> <math display="block">\frac{\partial C}{\partial Z^{L-1}} = \delta^{L-1}</math> </div> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="background-color: #808080; color: white; padding: 2px; font-size: 0.8em; margin-bottom: 5px;"><math>\delta^L</math></div> <div style="background-color: #0000ff; color: white; padding: 2px; font-size: 0.8em; margin-bottom: 5px;"><math>W^L</math></div> <div style="background-color: #808080; color: white; padding: 2px; font-size: 0.8em; margin-bottom: 5px;">Derivada función activación</div> </div> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <math display="block">\frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}</math> </div> <div style="border: 1px solid #0000ff; padding: 10px; margin-bottom: 10px;"> <math display="block">\frac{\partial Z^L}{\partial a^{L-1}}</math> </div> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <math display="block">\frac{\partial a^{L-1}}{\partial Z^{L-1}}</math> </div> </div>	$\frac{\partial Z^{L-1}}{\partial W^{L-1}} \rightarrow a^{L-2} \text{ Activación capa previa}$		
	$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$	$\frac{\partial Z^L}{\partial a^{L-1}}$	$\frac{\partial a^{L-1}}{\partial Z^{L-1}}$	$\frac{\partial Z^{L-1}}{\partial b^{L-1}} \rightarrow 1$

En esto reside la eficiencia del algoritmo de backpropagation ya que el reciente cálculo es extensible al conjunto de capas restantes de la red aplicando la misma lógica.

1. Cómputo del error de la última capa.

$$\delta^L = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$$

2. Se retropropaga el error a la capa anterior.

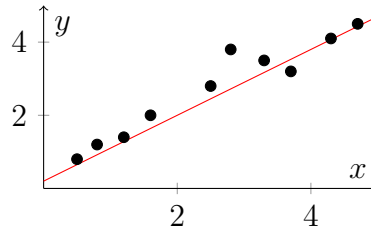


Figura C.13: Gráfica regresión lineal

$$\delta^{L-1} = W^L \cdot \delta^L \cdot \frac{\partial a^{L-1}}{\partial Z^{L-1}}$$

3. Se calculan las derivadas de la capa usando el error.

$$\frac{\partial C}{\partial b^{L-1}} = \delta^{L-1} \quad \frac{\partial C}{\partial W^{L-1}} = \delta^{L-1} \cdot a^{L-2}$$

Así sucesivamente recorriendo todas las capas de la red hasta el final. Gracias a este método, con un único pase se calculan todos los errores y las derivadas parciales de la red haciendo solamente uso de cuatro expresiones.

## C.2.4. Overfitting y underfitting

### C.2.4.1. Teoría general

La capacidad de un modelo de generalizar es la componente más perseguida en el campo del Machine Learning debido a que es irrelevante el hecho de acertar un resultado previamente estudiado en su entrenamiento. Lo que se quiere es que a partir de una serie de ejemplos el modelo sea capaz de realizar predicciones y clasificar correctamente casos nunca antes vistos durante el entrenamiento.

Esto es apreciable con el modelo de regresión lineal de forma superficial. Partiendo de un conjunto de muestras iniciales que dan datos de una relación entre dos variables X Y, lo que se quiere es inferir a partir de esos datos un modelo. En ese caso se restringía sólo una línea recta para que una vez ajustado el modelo a los datos sirviera para hacer predicciones incluso de datos no vistos anteriormente [figura C.13].

### C.2.4.2. ¿Un modelo bien ajustado siempre generaliza correctamente

La respuesta a esta pregunta conduce a la definición de los dos términos principales en esta sección: **overfitting** y **underfitting**.

La respuesta es no, un modelo bien ajustado no implica que ese mismo modelo generalice correctamente. Si se quiere ajustar un modelo utilizando el modelo de

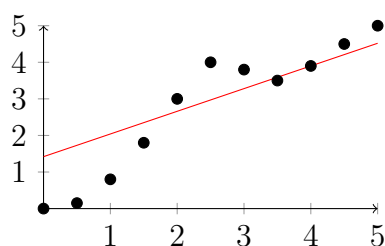


Figura C.14: Gráfica regresión lineal con underfitting

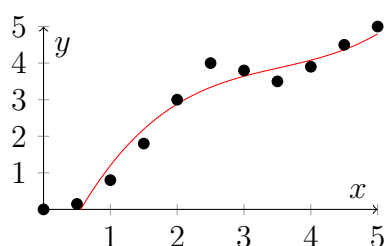


Figura C.15: Gráfica regresión polinomial

regresión anterior con la recta se daría la hipótesis de que la relación entre ambas variables es lineal. No es suficiente puesto que la recta no es capaz de adaptarse a la curva natural de los datos. Como consecuencia el modelo no tendrá la capacidad de hacer buenas predicciones.

Por tanto se indica que se tiene el modelo mal ajustado, que el ajuste es deficiente o usando el término común en inglés; que el modelo sufre de **underfitting** (subajuste). Lo sucedido se explica indicando que el modelo es una recta y no posee la flexibilidad suficiente para poder adaptarse a la nube de puntos de los datos [figura C.14].

Por otro lado, si una ecuación que define una línea recta se le aumenta el grado del polinomio que la define se consigue que la línea se curve. Además el grado del polinomio determinará el número de puntos críticos que podrá tener la curva, es decir, cuánto se podrá curvar. Cuanto mayor es el grado del polinomio más flexible es la curva. Esta teoría se puede utilizar en el modelo de regresión lineal simple introduciendo las variables elevadas a un exponente en la ecuación.

En este punto se pueden utilizar las ecuaciones del mínimo cuadrado ordinario del descenso del gradiente para ajustar la línea curva a los datos. Este modelo se conoce como el modelo de regresión polinomial [figura C.15].

En este **caso práctico** con un polinomio de grado 2 se mejora el modelo y se reduce el error. Sin embargo, se puede mejorar el modelo subiendo el polinomio a un grado 3 y conseguir la curva suficiente para que se ajuste perfectamente a los datos.

Aparentemente el modelo ajustado se comporta de forma más satisfactoria frente al modelo que sufría **underfitting**. El modelo mal ajustado era un modelo rígido cuya naturaleza no puede adaptarse a los datos y por tanto los errores de las predicciones para datos nunca vistos eran mayores. Puede dar la sensación que cuanto más flexibilidad se da a la curva mejor se ajusta el modelo y menor es el error obtenido.

Sin embargo, si se comienza a elevar el grado del modelo de regresión polinomial la curva cada vez se torna más flexible y se ajusta demasiado a los datos. Lo que se está haciendo es modelar el ruido de los datos. Esto supone un problema debido a que se reduce la capacidad de generalización del modelo y valores nuevos provocan errores elevados.

Por tanto, indicar que un modelo bien ajustado siempre va a generalizar bien es falso y este problema identificado se denomina **overfitting** (sobreajuste). Como su nombre indica es exactamente el caso opuesto a lo que sucedía con el **underfitting**. Conceptualmente el modelo se especializa en resolver los casos para los que ha sido entrenado, simulando una "memorización" de las soluciones.

Si por el contrario el modelo sufre **underfitting** es porque no tiene la capacidad suficiente para resolver el entrenamiento planteado.

La clave es encontrar aquel modelo que esté en el plano intermedio. Un modelo capaz de aprender en el entrenamiento pero que también sea capaz de generalizar su conocimiento.

#### C.2.4.3. Ejemplos de overfitting

El **overfitting** no es un problema exclusivo de un único modelo. Este es un problema general en todos los ámbitos del machine learning. Un modelo de regresión lineal puede tener overfitting pero también puede ocurrir en un problema de clasificación logística. Por ejemplo, un modelo logístico con la frontera de decisión trazada muy flexible buscará complacer todos los casos del entrenamiento, Cuando recibe nuevos datos la frontera de decisión no generalizaría.

Una situación similar se produce en un algoritmo como el KNN (**N** vecinos cercanos) donde se analiza los **K** puntos más cercanos para decidir cómo clasificar un elemento. Si el valor de **K** es muy pequeño la frontera de decisión será muy ruidosa y se tendrá overfitting. Ocurre igual en los problemas de clusterización como el K-MEANS si se opta por un elevado número de clusters. También puede suceder en un problema de *computer vision* que memorice cual es la respuesta asociada a una determinada imagen o que el modelo generador de textos realmente base la generación de sus frases en repetir palabra por palabra párrafos contenidos en los ejemplos de entrenamiento.

Por último, en el caso del aprendizaje reforzado, un agente que no aprende a

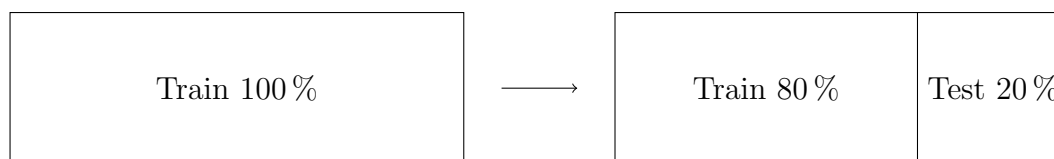


Figura C.16: Reparto conjunto de datos

reaccionar de manera inteligente ante los diferentes elementos que puedan surgir durante su partida sino que tras mucho ensayo y error memorice cual es la combinación de elecciones que le conducen a la victoria. Esto es correcto para ganar en ese nivel pero cuando se sitúe el agente en otro nivel será completamente incapaz de resolverlo puesto que su conocimiento no es generalizable.

Lo que se quiere conseguir son modelos que sean robustos ante nuevos ejemplos. Modelos que generalicen correctamente y por tanto hay que detectar si el modelo tiene overfitting.

#### C.2.4.4. Detección de overfitting

Hasta este punto se ha identificado el overfitting de manera visual al observar que la línea de regresión o la frontera decisión eran tan flexibles que se estaban adaptando en exceso a los ejemplos de entrenamiento y que estas no eran muy útiles cuando se añadían nuevos datos. Esto es observable en espacios de trabajo de 2 o 3 dimensiones pero en machine learning se pueden tener millones de dimensiones imposibles de procesar.

Facilitando la explicación, se desarrolla un caso práctico para ejemplarizar el conocimiento teórico.

Problema a resolver:

$$X : \text{Datos del paciente} \rightarrow Y : \text{Éxito trasplante}$$

Queriendo evaluar el modelo con datos diferentes a los de entrenamiento, el adecuado proceder consiste en inicialmente dividir los datos disponibles en dos grupos. De forma habitual un 80 % de los datos para el entrenamiento y un 20 % para las pruebas [figura C.16]. De esta forma se puede evaluar la capacidad de generalización del modelo.

Con los mismos datos que se tenían inicialmente se puede entrenar al modelo y evaluar su rendimiento con datos que el modelo nunca ha estudiado. Ahora es posible evaluar la capacidad de generación del modelo.

Cuanto más tiempo se entrene el modelo menor será el error. Esta evaluación se puede hacer simultáneamente con los datos de validación. Como es de esperar

del entrenamiento, el error del entrenamiento será menor al error de validación. Es normal que el modelo funcione mejor con los datos que sí conoce. Según se entrena al modelo ambos errores seguirán descendiendo. En este caso el descenso del error de validación es una señal positiva ya que indica que el modelo está aprendiendo a predecir incluso casos que nunca se han visto. Está generalizando.

Sin embargo, si se continua con el entrenamiento se llega a un punto en el que el error de entrenamiento sigue bajando pero el error de validación aumenta. Esto significa que la capacidad de generalizar se deteriora. En el punto donde empieza subir el error de validación se entra en una zona de overfitting, donde el modelo se ajusta en exceso a los datos del modelo y no a los de evaluación.

La estrategia de reservar parte de los datos para evaluar se denomina **hold-out**. Permite tratar en paralelo dos métricas cuya divergencia significa que el modelo está sufriendo el overfitting.

#### C.2.4.5. Error de planteamiento

Es de gran importancia asegurarse que los datos de ambos grupos (train & test en inglés) tengan la misma naturaleza. Los datos pueden estar ordenados siguiendo una lógica dentro del problema, provocando que el error de entrenamiento disminuya pero el error de evaluación, por el contrario, aumente. Esto sucede porque las estrategias aprendidas por el modelo durante el entrenamiento no son aplicables a los datos de evaluación al no ser el mismo tipo de datos.

Esta problemática se resuelve ordenando los datos disponibles de forma aleatoria antes de dividirlos, teniendo ambos grupos el mismo tipo de datos. Este desordenamiento aleatorio de los datos no siempre es beneficioso y depende de la naturaleza del problema.

La estrategia es asegurarse cuando se trabaje con datos y se quiera preparar un dataset de validación, que los datos están idénticamente distribuidos en ambos grupos. Además, asegurarse de que no existan dependencias entre datos que puedan hacer que el overfitting se infiltre en el conjunto de pruebas.

La conclusión más destacable es que en el mundo del machine learning la parte más compleja no es implementar una red neuronal, si no que lo difícil es saber aplicar una metodología correcta en el tratamiento de los datos para que las conclusiones obtenidas no sean erróneas.

## C.2.5. Métricas

### C.2.5.1. Tipos de métricas

En el machine learning existen diversas métricas que aportan valiosa información a los desarrolladores e investigadores. Estas se clasifican en dos grandes grupos:

**Métricas de clasificación:** Evalúan y cuantifican el rendimiento de un modelo de clasificación en función de sus predicciones y los valores reales de las etiquetas de clase.

- **Exactitud (accuracy)**
- **Logarithmic loss**
- **Área bajo la curva ROC (AUC-ROC)**
- **Matriz de confusión (confusion matrix)**
  - ▷ **Precisión (precision)**
  - ▷ **Sensibilidad (recall)**
  - ▷ **Especificidad (recall)**
  - ▷ **F1-Score**

**Métricas de regresión:** Evalúan y miden la calidad de las predicciones de un modelo de regresión, el cual busca predecir valores numéricos continuos en lugar de etiquetas de clase.

- **Error absoluto medio (mean absolute error)**
- **Error cuadrático medio (mean square error)**
- **Coeficiente de determinación  $R^2$**

### C.2.5.2. Exactitud

Denominada accuracy en inglés, la exactitud corresponde al número de predicciones correctas divididas por el número total de predicciones realizadas.

Es la métrica más utilizada y requiere que la cantidad de datos de cada clase sea similar. No se debe utilizar esta métrica si hay muy pocos ejemplos de una clase, en ese caso es mejor utilizar la matriz de confusión.

A continuación se ejemplifica los problemas existentes con esta métrica:

Total de población  $\rightarrow 10000$   
 Tienen enfermedad  $\rightarrow 10$  (Sanos 9990)

función TRAMPOSO(persona): devuelve False
--

$$\text{EXACTITUD} = 9990 / 10000 = 0.999$$

¡EXACTITUD DEL 99.9%!  $\rightarrow$  Medida falseada. Hay que calcular la medida por cada clase al estar tan balanceadas.

$$\text{No enfermos: } 9990 / 9990 = 1.0$$

$$\text{Enfermos: } 0/10 = 0.0$$

Se aciertan todos los sanos y falla todos los enfermos.

### C.2.5.3. Matriz de confusión

		Realidad (ejemplos)	
		1	0
Predicción	1	Verdadero positivo (True positive) VP o TP	Falso positivo (False positive) FP
	0	Falso negativo (False negative) FN	Verdadero negativo (True negative) VN o TN

Figura C.17: Matriz de confusión



La matriz de confusión [ver figura C.17] es una herramienta utilizada en el campo del aprendizaje automático para evaluar el rendimiento. Su potencial reside en la fácil comprensión y estructuración de las diferentes comparativas realizadas entre los datos reales y las predicciones del modelo.

De la matriz de confusión se pueden extraer múltiples métricas como las siguientes:

- Exactitud

$$\frac{VP + VN}{VP + FP + FN + VN}$$

- Precisión

$$\frac{VP}{VP + FP} \quad \frac{VN}{VN + FN}$$

Número de aciertos en las predicciones realizadas. Se calcula la precisión por separado para los valores positivos y los valores negativos.

- Sensibilidad

$$\frac{VP}{VP + FN}$$

Número de aciertos en el total de positivos reales.

- Especificidad

$$\frac{VN}{FP + VN}$$

Número de aciertos en el total de negativos reales.

- F1-Score

$$\frac{2 \cdot \text{Precisión} \cdot \text{Sensibilidad (o Especificidad)}}{\text{Precisión} + \text{Sensibilidad (o Especificidad)}}$$

El F1-Score combina la precisión y la sensibilidad (o especificidad) dependiendo de cuál sea la clase minoritaria. Se utiliza como una métrica común en machine learning cuando se busca encontrar un equilibrio entre la precisión y la exhaustividad del modelo.

Dependiendo de la problemática presente, la matriz de confusión se debe de interpretar de diferentes formas o se debe bucar un resultado concreto:

$$Normal \begin{cases} 1 & \text{si } h_0(x) \geq 0,50 \\ 0 & \text{si } h_0(x) < 0,50 \end{cases}$$

$$\text{Evitar FP: Estar muy seguro} \begin{cases} 1 & \text{si } h_0(x) \geq 0,75 \text{ Alta precisión} \\ 0 & \text{si } h_0(x) < 0,75 \text{ Baja sensibilidad/especificidad} \end{cases}$$

$$\text{Evitar FN: No perder ninguno} \begin{cases} 1 & \text{si } h_0(x) \geq 0,25 \text{ Baja precisión} \\ 0 & \text{si } h_0(x) < 0,25 \text{ Alta sensibilidad/especificidad} \end{cases}$$

#### C.2.5.4. Curvas ROC

Las curvas ROC son una representación gráfica utilizada en la evaluación de modelos. Estas curvas muestran la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos (1 - especificidad) a medida que se varía el umbral de clasificación del modelo.

La curva ROC se crea trazando la sensibilidad en el eje vertical y la tasa de falsos positivos en el eje horizontal para diferentes valores del umbral. Cada punto en la curva ROC representa un umbral específico, y el área bajo la curva (AUC) se utiliza como medida de la calidad global del modelo. Cuanto mayor sea el AUC, mejor será el rendimiento del modelo en la clasificación.

La interpretación de la curva ROC se realiza observando su forma y la distancia al punto de referencia (0,1). Un modelo con una curva ROC cercana al punto superior izquierdo del gráfico indica un mejor rendimiento, ya que tiene una alta sensibilidad y baja tasa de falsos positivos. En cambio, una curva ROC que se aproxima a la línea diagonal (que representa el rendimiento aleatorio) indica un modelo con un rendimiento insatisfactorio.

Los posibles casos que pueden reflejar las curvas ROC se representan gráficamente en la figura [C.18](#).

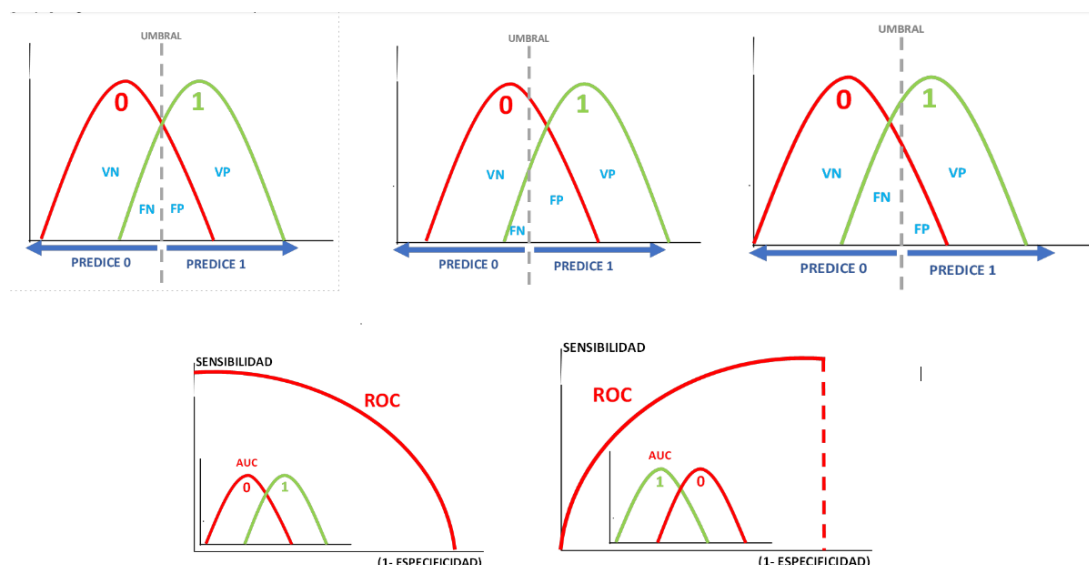


Figura C.18: Curvas ROC

## C.3. Natural Language Processing

El objetivo es profundizar en los modelos especializados en el análisis de lenguaje natural, explicando qué es, en qué consiste el proceso y tecnologías actuales utilizadas para este propósito.

### C.3.1. ¿Qué es Natural Language Processing?

El Procesamiento de Lenguaje Natural o más conocido como NLP por sus siglas en inglés es una tecnología de *machine learning* que permite el entendimiento del lenguaje humano por parte de las máquinas además de la manipulación y comprensión de este. Tecnología de gran actualidad en el mundo empresarial y personal debido al auge de la información digitalizada en formato audiovisual y la necesidad de procesar automáticamente dicha información.

El primer paso necesario para el procesamiento del lenguaje natural es escoger una estrategia de codificación del lenguaje cuya complejidad inicial consiste en seleccionar la unidad mínima de texto que se identificará con un identificador numérico. Al comienzo se utilizaba el carácter como unidad mínima utilizando el código ASCII para posteriormente evolucionar la estrategia hasta utilizar como unidad mínima la palabra. Finalmente en los últimos años se ha utilizado en tecnologías como GPT-3 la sub-palabra para asociar identificadores al lenguaje natural.

### C.3.1.1. Codificación

El proceso de codificación consiste en asignar a cada uno de los bloques que conforman la secuencia de datos (caracteres, palabras o subpalabras) un identificador numérico comprensible para la máquina. Cada bloque, independientemente de la estrategia seguida, se denomina **token** y el proceso de división en unidades se denomina **tokenización**.

En primera instancia se puede seguir como método de tokenización la asociación de cada palabra distinta encontrada con una misma etiqueta. Sin embargo, la principal problemática de esta estrategia reside en la imposibilidad de crear asociaciones con sentido entre los tokens debido a que la asignación del identificador es aleatoria o secuencial sin tener en cuenta el token en específico.

Un método alternativo de tokenización es asignar un vector con tantas posiciones como tokens tenga el problema. Cada posición del vector representa una palabra del vocabulario y se marca con un 1 la posición que indique la palabra que se está representando. El resto de posiciones del vector se marcan con 0. Este método denominado **one-hot encoding** sufre la problemática de la cantidad de elementos por vector siendo la mayoría de las posiciones 0 y la equidistancia entre los vectores, es decir, la imposibilidad de crear asociaciones por agrupación.

### C.3.1.2. Compactación

El proceso de compactación es utilizado con la finalidad de conseguir numéricamente asociaciones entre tokens, reduciendo la dimensionalidad de los datos del problema presente.

El método de trabajo habitual de una inteligencia artificial es comprimir datos y ordenarlos de manera inteligente, y así solucionar de la mejor forma la tarea a resolver. La nueva representación de los datos que aprende la red dependerá del tipo de tarea que quiera resolver, encontrando aquella codificación de los datos de entrada que mejor le permita acercarse a la solución óptima.

El proceso ocurrido en la primera capa de la red neuronal en el que necesariamente se tiene que aprender a comprimir vectores dados por one-hot encoding a una representación más compacta que le sirva para resolver su tarea es lo que se conoce como compactación (**embedding**) [figura C.19]. Es preferible contar con un embedding pre-entrenado ya existente debido a que parte del conocimiento ya aprendido podrá ser transferido a la nueva tarea (**transfer learning**).

En el año 2013 se publicó **Word2vec**, un sistema de embedding pre-entrenado y es uno de los sistemas más utilizados hasta la fecha. **Word2vec** transforma vectores de un vocabulario de tamaño 10.000 posiciones a vectores de tamaño 300 posiciones. El funcionamiento esencial de este sistema consiste en una pequeña

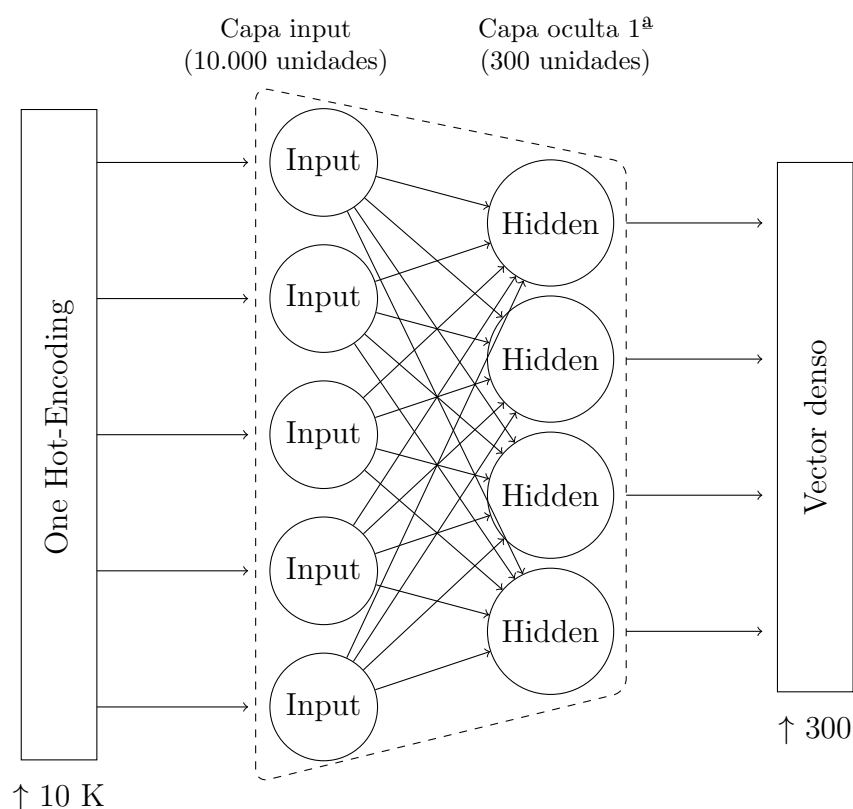


Figura C.19: Ejemplo de embedding

red neuronal entrenada la cual recibe 10.000 posiciones de entrada y retorna 300 posiciones por su capa de salida. El procesamiento de los datos de entrada es satisfactorio gracias a las capas ocultas.

Para poder apreciar las relaciones existentes entre palabras halladas por el sistema Word2vec o similares es necesario reducir nuevamente la dimensionalidad de los datos a tres dimensiones aplicando un algoritmo de reducción de la dimensionalidad. Esta última transformación se debe a que el ser humano no es capaz de analizar gráficamente ni visualizar más de tres dimensiones.

## C.3.2. Modelos NLP

### C.3.2.1. Historia

En los comienzos se utilizaban arquitecturas de redes neuronales sencillas, como las redes neuronales multi-capas, para construir los primeros modelos que aprendieron a resolver tareas básicas. Posteriormente se adaptaron las redes neuronales a los diferentes tipos de datos que se usaban: redes neuronales convolucionales para entender datos espaciales como imágenes, redes neuronales recurrentes

para manipular datos secuenciales como textos... redes neuronales que no sólo eran utilizadas para aprender a analizar patrones, sino que también eran capaces de generar patrones similares.

En 2017 se transformó la concepción de los límites que puede alcanzar la inteligencia artificial y lo que puede llegar a hacer. En el estudio *Attention is all you need* aparece el concepto y desarrollo de Transformers. Algunos ejemplos actuales de tecnología Transformer son:

- Alpha Fold 2: Para analizar secuencias de datos genómicos.
- Auto Pilot: Utilizado en el sistema de conducción de Tesla
- GPT3: Para la modelización y generación de texto. Uno de los modelos con más potencial es Megatron-Turing.

#### C.3.2.2. Análisis de texto

Dada la naturaleza secuencial de una frase donde cada palabra ocupa una posición en el tiempo una tras otra, la estrategia utilizada en el campo del deep learning era usar una red neuronal estándar. Como input, la red neuronal recibía la primera palabra e internamente esta palabra se procesaba multiplicándose capa tras capa con los parámetros aprendidos de la red.

Posteriormente se analizará la siguiente palabra y así sucesivamente hasta finalizar el texto ofrecido como entrada a la red. Internamente, cada tecnología ofrece una alternativa a la cuestión del procesamiento y aprendizaje interno.

#### C.3.2.3. Redes neuronales recurrentes

La peculiaridad existente en una red neuronal recurrente consiste en que la información que ha sido procesada por la red será agregada a la nueva información introducida en el siguiente paso de la secuencia.

Realizado el proceso descrito de encadenar el output de la red con el input del próximo pase y permitiendo que analice todas las palabras, se acabará en un punto donde toda la información de la secuencia habrá sido procesada y analizada [figura C.20].

El concepto de conectar el procesamiento del output anterior con el input del procesamiento actual es lo que nombra a este tipo de redes, redes neuronales recurrentes.

El principal problema en este tipo de redes es precisamente su método de funcionamiento en cadenas de texto con muchas palabras. Al repetir el proceso

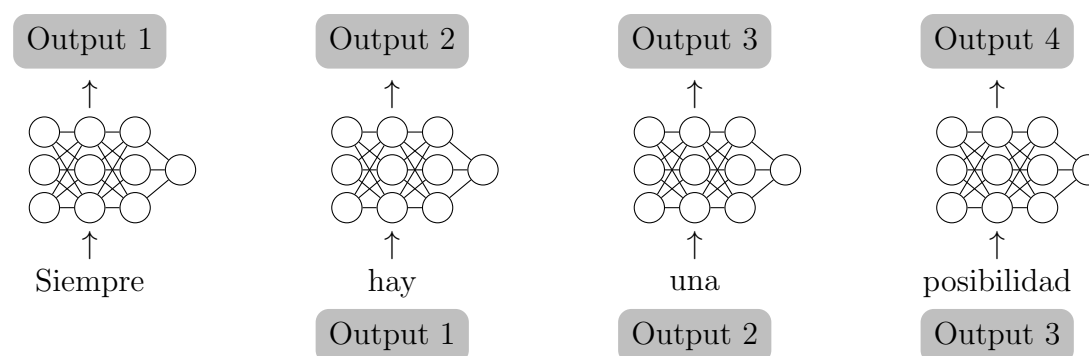


Figura C.20: Funcionamiento red neuronal recurrente

de nutrir el input con el output anterior durante muchos pasos, el peso que tienen durante el entrenamiento las primeras palabras respecto a las últimas agregadas es menor. Para solventarlo existen nuevos procedimientos como los mecanismos de atención.

#### C.3.2.4. Mecanismos de atención

El problema identificado en la sección anterior es conocido como falta de memoria. Al analizar las palabras en orden secuencial se dificulta en exceso la búsqueda de relaciones en oraciones extensas.

La flor que florece en la adversidad es la más rara y hermosa de todas.

↑ Relación difícil de identificar ↑

Para solucionar los problemas relacionados con la falta de memoria se utilizan unos mecanismos denominados **mecanismos de atención**.

Las palabras en realidad están representadas como vectores numéricos, vectores multidimensionales que capturan gran parte de la información semántica y sintáctica de la palabra que representan y con los que se puede operar matemáticamente.

Respecto a los vectores, si la dirección en el espacio multidimensional es muy parecida en varios vectores, esto representarán conceptos cuyas palabras también sean parecidas y se alejarán de aquellas palabras sin relación. Es posible calcular matemáticamente el ángulo formado por los vectores para así estudiar cuál es la similitud entre palabras, frases o documentos.

El objetivo es que por cada una de las palabras de la frase, sin importar la distancia que hay entre ellas en el texto, se pueda estudiar cuál es la relación con cada una de las otras palabras de la frase. Es decir, se está buscando cuál es la relación de todas las palabras con todas las palabras.

Para ello se entrenan a dos redes neuronales diferentes para que con estas palabras dadas como input, aprendan a generar dos vectores distintos:

- **Vector identificador:** Vector utilizado para identificar las propiedades interesantes que caracterizan dicha palabra.
- **Vector búsqueda:** Vector utilizado para describir aquellas propiedades interesantes que esta palabra está buscando.

La relación entre ambos vectores se puede entender mediante la metáfora de la llave y la cerradura, equivaliendo el vector identificador al **vector llave** y el vector búsqueda al **vector cerradura**. Sin embargo, en el paper original donde se desarrolló esta tecnología se denomina **vector query** a la llave y **vector key** a la cerradura.

Se puede identificar si dos vectores tienen una dirección similar con una operación matemática denominada **producto escalar** o dot product. Si el resultado es cercano a 1 la dirección de los vectores query y key es similar, es decir, llave y cerradura encajan.

Dado el vector query de una palabra es posible calcular cuál es la compatibilidad que tiene con el resto de palabras. Esto se consigue haciendo el producto escalar entre el vector query de la palabra y los vectores key del resto de palabras. El vector resultante de agrupar todos los resultados de los productos escalares se denomina **vector de atención** debido a que refleja qué importancia le asigna un modelo de inteligencia artificial al resto de palabras. Si se comparan estos vectores para cada palabra de una frase se forma la **matriz de atención**.

En una matriz de atención se refleja la importancia que cada palabra está asignando al resto de la frase. Muestra qué parte de la información dada como input está prestando atención la inteligencia artificial para dictaminar sus decisiones.

Utilizando una red neuronal para procesar cada una de las palabras de la frase se consigue el equivalente numérico de la palabra. Este vector se denomina en el paper original como **vector valor**.

El uso dado a cada atención calculada será el de computar una suma ponderada donde se utiliza la atención prestada a cada palabra como factor de mezcla de cada uno de los vectores valor [ver imagen [C.21](#)].

Con este mecanismo se ha hallado una manera de poder contextualizar a cada una de las palabras de la frase con cualquier otra palabra que se pueda encontrar a cualquier distancia.



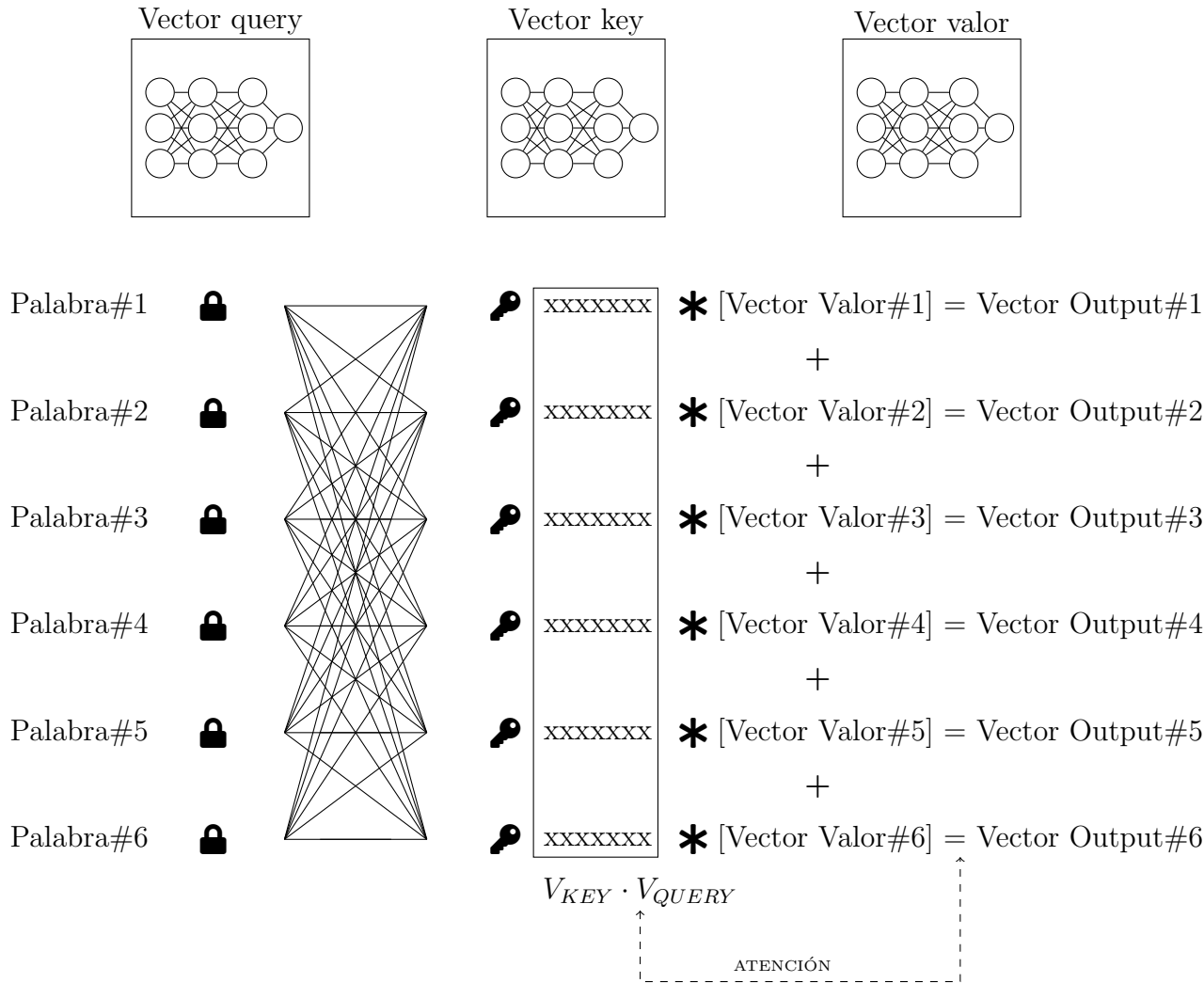


Figura C.21: Mecanismo de atención

### C.3.2.5. Transformers

Transformers es una arquitectura de red neuronal [imagen C.22] utilizada en el área de procesamiento del lenguaje natural. Desarrollada en el artículo «*Attention is All You Need*», esta arquitectura destaca por el uso de mecanismos de atención como el detallado en la sección anterior.

Transformers consta de múltiples secciones entre las que destacan las capas de alimentación hacia delante (feed-forward) utilizando cada una múltiples cabezas de atención (multi-head attention) para la recogida de diferentes tipos de información contextual.

Permitiendo capturar relaciones y dependencias contextuales dado un texto, la arquitectura Transformers y variantes más específicas cómo BERT, GPT o RoBERTa demuestran una gran eficiencia en tareas de procesamiento del lenguaje

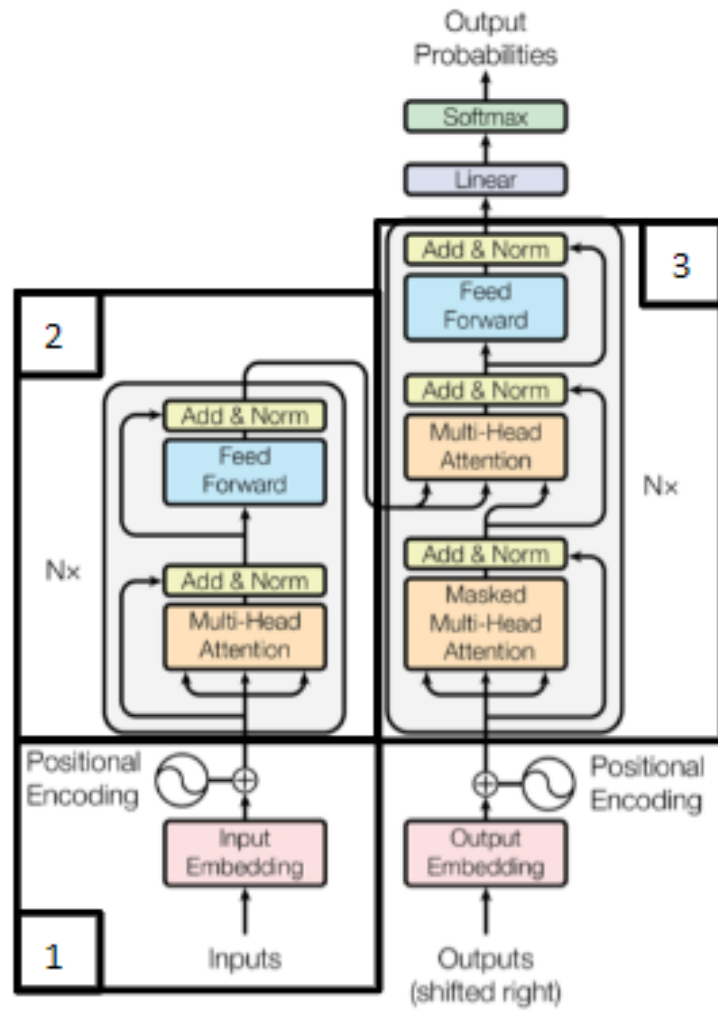


Figura C.22: Arquitetura transformers

natural (clasificación de texto, análisis de sentimientos, traducción automática, ...).

La primera sección de la arquitectura [sección 1 de la imagen C.22] corresponde a la entrada, es decir, donde se introduce el input. La frase a representar no se introduce en su estado original si no que se reconvierten las palabras en vectores numéricos.

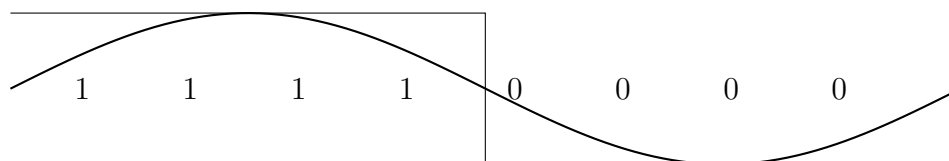
Utilizando una red neuronal recurrente, cada palabra es procesada una tras otra. Sin embargo, Transformer analiza el conjunto completo de palabras simultáneamente. Debido a esto es necesario establecer alguna técnica de identificación para detectar el orden de las palabras.

Una primera posibilidad consiste en sumar otro vector al vector identificador de la palabra para así marcar la posición de esta. Esta técnica se denomina **información posicional** pero el problema de este método consiste en que la información posicional añadida con este formato (posición absoluta) haría poco relevante la información del vector original en posiciones altas.

Otra alternativa es seleccionar los componentes de los vectores posicionales y dividirlos entre el número de palabras del input consiguiendo normalizar las componentes obteniendo valores entre 0 y 1. El principal inconveniente de este método es que no se puede saber la posición exacta debido al número variable de palabras por frase. En una secuencia de 3 palabras tendría una codificación posicional para la segunda palabra de 0.66 y, por otro lado, en una secuencia de 6 palabras tendría una codificación posicional para la cuarta palabra de 0.66. Por lo tanto, no se puede saber la verdadera posición.

La codificación ideal es utilizar un vector posicional que contenga la posición absoluta de la palabra y sumarlo al vector identificador de esta. Sin embargo, la posición absoluta hay que codificarla en binario para no cometer el error anteriormente especificado en posiciones altas. El funcionamiento de la codificación ideales apreciable en el siguiente ejemplo:





### C.3.3. Clasificación del sentimiento

#### C.3.3.1. ¿Qué es?

Un problema de análisis de sentimiento de un texto dado como input es en realidad un problema específico de clasificación de texto en el que se indica si el texto es positivo o negativo.

#### C.3.3.2. Evolución de las alternativas

La primera alternativa utilizada con el propósito de analizar el sentimiento de un texto dado consiste en analizar la polaridad de las palabras que conforman dicho texto. Buscando un vocabulario predefinido que indique si una palabra es positiva o no, se puede clasificar una frase haciendo el sumatorio de las puntuaciones de cada palabra. Librerías como Textblob para el lenguaje de programación Python ofrecen este servicio. Sin embargo, el principal problema de este método es que al analizar frases escritas con palabras positivas pero en las que se utilice la ironía o el doble sentido serán erróneamente clasificadas. Este efecto es apreciable en los ejemplos de la tabla C.3.

Texto a analizar	Tipo de frase	Clasificación
Me encanta esta película	Positiva	Positiva
Odio esta película	Negativa	Negativa
Ver esta película ha sido perfecta para perder el tiempo	Negativa	Positiva

Tabla C.3: Clasificación de texto mediante polaridad de palabras

En el año 2017 cambió este sistema debido a las nuevas tecnologías utilizadas en el NLP (Transformers) pudiendo escoger un modelo pre-entrenado relacionado con la tarea que se quiere realizar y especializarlo. Ese proceso se denomina **fine tuning** de modelos pre-entrenados. Además, esta tendencia de entrenamiento de modelos evitó a los desarrolladores dedicar tiempo y recursos a etiquetar los resultados manualmente. El aprendizaje supervisado (**supervised learning**) evoluciona al aprendizaje auto-supervisado (**self-supervised learning**).

Hasta la fecha el camino seguido ha sido aumentar de magnitud los datasets de entrenamiento empezando en el año 2017 la era de los enormes modelos del

lenguaje (**LLM**). Favoreciendo el crecimiento, las grandes asociaciones que entrenaron estos modelos los liberaron para poder utilizarlos de manera gratuita destacando la plataforma Hugging Face en la recopilación de estos modelos.

La empresa OpenAI entrenó sus modelos generativos pre-entrenados basados en Transformers (modelos GPTs). En el año 2023 se entrena el modelo GPT-4, modelo de lenguaje más grande entrenado hasta la fecha con 45 gigabytes de datos de entrenamiento frente a los 17 gigabytes de datos de su antecesor GPT-3.

Estos modelos masivos aprendieron gran cantidad de tareas debido a su tamaño aun habiendo sido entrenados con ejercicios de auto-completar texto. Pueden traducir textos, ejercer de chatbot, análisis de sentimientos en textos, diversas clasificaciones, etc.