



Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES
Y MULTIMEDIA

Curso Académico 2021/2022

Trabajo Fin de Grado

IMPLEMENTACIÓN DE FUNCIONALIDADES EN
LEARNINGML: RECONOCIMIENTO DE
CONJUNTOS DE DATOS (DATASETS)

Autor : Ignacio Rueda Rodríguez

Tutor : Dr. Gregorio Robles Martínez

Co-tutor : Juan David Rodríguez García

Trabajo Fin de Grado

Implementación de Funcionalidades en LearningML: Reconocimiento de
Conjuntos de Datos (Datasets)

Autor : Ignacio Rueda Rodríguez

Tutor : Dr. Gregorio Robles Martínez

Co-tutor : Juan David Rodríguez García

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 2022, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2022

*Dedicado a
mi familia, abuelos y amigos*

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

El objetivo de este proyecto consiste en añadir la funcionalidad de reconocimiento de conjuntos de datos a la aplicación web LearningML, que ya reconoce imágenes y textos.

LearningML es una aplicación web que tiene como objetivo ayudar a los profesores de secundaria a explicar de una manera sencilla y visual el machine learning y para fomentar las habilidades de pensamiento computacional en alumnos de entre 10 y 16 años.

LearningML está creada en angular, que es un entorno de trabajo que utiliza HTML y CSS para la creación de aplicaciones web y utiliza TypeScript para implementar la funcionalidad de las aplicaciones web. Para llevar a cabo este proyecto ha sido necesario crear un nuevo servicio y un nuevo componente además de modificar algunos componentes y servicios, y crear distintas funciones pero siguiendo la dinámica de lo ya creado.

Summary

The goal of this project consists of adding dataset recognition functionality to the LearningML web application, which already recognises images and text.

LearningML is a web application that aims to help secondary school teachers explain machine learning in a simple and visual way and to foster computational thinking skills in students aged 10-16.

LearningML is based on Angular, which is a framework that uses HTML and CSS to create web applications and uses TypeScript to implement the functionality of the web application. To carry out this project it has been necessary to create a new service and a new component, as well as to modify some components and services, and to create different functions but following the dynamics of what has already been created.

Índice general

1. Introducción	1
1.1. Estructura de la memoria	2
2. Objetivos	5
2.1. Objetivo general	5
2.2. Objetivos específicos	5
2.3. Planificación temporal	6
3. Estado del arte	7
3.1. Inteligencia Artificial	7
3.2. Machine Learning	8
3.3. LearningML	9
3.4. Angular	9
3.4.1. Angular CLI	12
3.4.2. MVC	12
3.5. HTML5	13
3.6. CSS3	14
3.7. TypeScript	15
3.8. Node.js	15
3.9. TensorFlow	16
3.10. JSON	17
3.11. Conjunto de datos	17
4. Diseño e implementación	19
4.1. Arquitectura de LearningML	19

4.2.	Cambios en la arquitectura de LearningML	22
4.3.	Cambios en la interfaz de LearningML	23
4.4.	Creación de conjuntos de datos	25
4.4.1.	Atributos	25
4.4.2.	Clases	26
4.4.3.	Muestras	26
4.5.	Extracción de características	27
5.	Experimentos, validación y resultados	31
5.1.	Conjunto de datos numéricos	31
5.1.1.	Redes neuronales	33
5.1.2.	KNN	37
5.2.	Conjunto de datos numéricos y categóricos	41
5.2.1.	Redes neuronales	42
5.2.2.	KNN	47
6.	Conclusiones	53
6.1.	Consecución de objetivos	53
6.2.	Aplicación de lo aprendido	53
6.3.	Lecciones aprendidas	54
6.4.	Trabajos futuros	55
	Bibliografía	57

Índice de figuras

4.1. Arquitectura general de LearningML. En rojo los servicios, en verde los componentes, en amarillo las funciones y en azul las variables.	20
4.2. Arquitectura nueva de LearningML. En rojo los servicios, en verde los componentes, en amarillo las funciones y en azul las variables.	22
4.3. Página principal de LearningML.	24
4.4. Sección de entrenamiento de ml-model.	24
4.5. Componente ml-test-dataset-model.	25
4.6. Aspecto de labelsWithData.	27
4.7. Muestra con números y cadena de caracteres	29
5.1. Muestras con datos numéricos para el entrenamiento	32
5.2. Configuración redes neuronales	34
5.3. Gráfica de evolución de aprendizaje redes neuronales con datos numéricos . . .	34
5.4. Matriz de confusión redes neuronales con datos numéricos. 25 % ejemplos de validación	35
5.5. Muestra de test de la clase iris setosa con redes neuronales	36
5.6. Muestra de test de la clase iris virginica con redes neuronales	36
5.7. Muestra de test de la clase iris versicolor con redes neuronales	37
5.8. Matriz de confusión redes neuronales con datos numéricos. 75 % ejemplos de validación	38
5.9. Configuración KNN	38
5.10. Matriz de confusión KNN con datos numéricos. 25 % ejemplos de validación .	39
5.11. Muestra de test de la clase iris setosa con KNN	40
5.12. Muestra de test de la clase iris virginica con KNN	40

5.13. Muestra de test de la clase iris versicolor con KNN	41
5.14. Matriz de confusión KNN con datos numéricos. 75 % ejemplos de validación .	42
5.15. Muestras con datos numéricos para el entrenamiento	43
5.16. Gráfica de evolución de aprendizaje redes neuronales con datos numéricos y categoricos	44
5.17. Matriz de confusión redes neuronales con datos numéricos y categoricos. 25 % ejemplos de validación	45
5.18. Muestra de test de la clase good con redes neuronales	45
5.19. Muestra de test de la clase vgood con redes neuronales	46
5.20. Matriz de confusión redes neuronales con datos numéricos y categoricos. 75 % ejemplos de validación	47
5.21. Matriz de confusión KNN con datos numéricos y categoricos. 25 % ejemplos de validación	48
5.22. Muestra de test de la clase good con KNN	49
5.23. Muestra de test de la clase vgood con KNN	50
5.24. Matriz de confusión KNN con datos numéricos y categoricos. 75 % ejemplos de validación	50

Capítulo 1

Introducción

Actualmente se generan una cantidad inmensa de datos digitales, tantos que se han multiplicado por más de treinta en la última década, pasando de dos zetabytes en 2010 a 64 zetabytes en 2020 y esta cantidad seguirá aumentando de forma exponencial [1]. El problema es el procesamiento de los datos ya que debido a su complejidad los humanos no somos capaces de extraer información útil de tal cantidad de datos en un tiempo aceptable, para ello se utiliza la inteligencia artificial. El machine learning es una de las ramas de la inteligencia artificial que se utiliza para el procesamiento y análisis de los datos, ya que es capaz de analizar una gran cantidad de datos de manera eficiente y transformarlos en información más sencilla, legible y fácil de analizar por los humanos además de aportar muchas utilidades a partir de los datos que se generan, como puede ser anuncios personalizados en internet [14].

Debido al aumento del uso de la inteligencia artificial y del aprendizaje automático Juan David Rodríguez García decide crear la aplicación web LearningML dado que es un apasionado de la programación y al ser docente cree que la introducción de contenidos de inteligencia artificial en la escuela a través de proyectos prácticos es el camino a seguir para educar ciudadanos conscientes y críticos, para despertar vocaciones entre los jóvenes y para fomentar las habilidades de pensamiento computacional. Por eso fue evaluada por estudiantes de 10 a 16 años mostrando su utilidad para aprender conceptos de inteligencia artificial y machine learning [10].

LearningML¹ es una aplicación web dirigida al aprendizaje automático supervisado, una de las técnicas de IA más exitosas que se encuentra en la base de casi todas las aplicaciones actuales de IA y con la que se pretende cubrir la falta de herramientas para la enseñanza práctica de la

¹<https://learningml.org/editor/>

Inteligencia Artificial. Es una herramienta que permite la enseñanza del aprendizaje automático de una forma sencilla haciendo que el reconocimiento de imágenes o el reconocimiento de textos se pueda hacer y observar de una manera sencilla y fácil de ver. Hace uso de los algoritmos de redes neuronales y KNN, permite descargar el modelo creado en un archivo JSON para poder usar ese modelo en otro momento y no tener que crear de nuevo el modelo, tiene la opción de registrarse para poder guardar los modelos en la nube y permite crear un modelo en scratch y utilizarlo en LearningML.

Este proyecto consiste en la integración de un nuevo tipo de reconocimiento: el reconocimiento de conjuntos de datos. Los conjuntos de datos es una de las formas de almacenar todos los datos que se generan y por eso creo que es interesante que se disponga de este tipo de reconocimiento.

1.1. Estructura de la memoria

La memoria de este trabajo se divide en seis capítulos, resumidos a continuación:

- **Capítulo 1: Introducción.** Se explica el contexto en el que se ha desarrollado, se hace una breve explicación de qué es, para que se usa y cual es el objetivo de la aplicación web LearningML, y la estructura de la memoria.
- **Capítulo 2: Objetivos.** Se describen tanto el objetivo principal como los objetivos específicos y la planificación temporal del proyecto.
- **Capítulo 3: Estado del arte.** Se explican cada una de las tecnologías usadas en el desarrollo del proyecto
- **Capítulo 4: Diseño e implementación.** Se explica detalladamente la arquitectura de LearningML y los cambios realizados sobre la arquitectura, así como el desarrollo e implementación de la nueva funcionalidad para reconocer conjuntos de datos
- **Capítulo 5: Experimentos, validación y resultados.** Se comprueba si funciona correctamente la nueva funcionalidad en los dos algoritmos disponibles y con los dos tipos de datos que puede tener un conjunto de datos.

- **Capítulo 6: Conclusiones.** Se exponen las conclusiones finales del proyecto, objetivos alcanzados y las lecciones aprendidas. Por último, se presentan posibles trabajos futuros que se podrían desarrollar para mejorar la aplicación web

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo de este trabajo fin de grado consiste en implementar el reconocimiento de conjuntos de datos en la aplicación web LearningML siguiendo la dinámica de los otros algoritmos de reconocimiento ya existentes y tratando de hacerlo de la forma más intuitiva y sencilla posible para el usuario.

2.2. Objetivos específicos

Para llevar a cabo la realización del objetivo general se han planteado los siguientes objetivos específicos:

- Analizar la arquitectura de LearningML para que la implementación de la nueva funcionalidad siga la dinámica de lo ya creado.
- Añadir a la interfaz un apartado para reconocer conjuntos de datos.
- Comprender que son y como se pueden crear conjuntos de datos para saber como tienen que ser las muestras y crear el conjunto de datos. Esto es algo que también tiene que tener claro el usuario.
- Entender como funciona la extracción de características para poder extraer las características de los conjuntos de datos.

2.3. Planificación temporal

La idea de empezar con el TFG la tuve en abril de 2021 pero no tenía claro de que podía hacerlo. En mayo me puse en contacto con el Dr. Gregorio Robles y me comentó un poco el proyecto que estaba llevando a cabo Juan David Rodríguez. Me llamó la atención el proyecto y me propuso una lista de funcionalidades que Juan David quería realizar y la que más me llamo la atención fue el reconocimiento de conjuntos de datos ya que hoy en día se usan mucho y creo que son una parte importante del machine learning. Una vez le confirmé a Gregorio que me interesaba añadir esa funcionalidad, me puso en contacto con Juan David para que me explicase el objetivo que perseguía con este proyecto y me comentase lo interesado que estaba en añadir esa funcionalidad. Después de hablar con Juan David me mandó un tutorial de Angular y durante un mes aproximadamente estuve aprendiendo a usar angular y comprender TypeScript. Una vez ya tenía claro como funcionaban los lenguajes tuve una reunión con Juan David a mediados de junio en la cual me explicó como funcionaba LearningML. Después vino el verano en el cual lo dejé apartado y a finales de agosto me volví a ver el tutorial de Angular para recordar el lenguaje ya que no me acordaba. En septiembre empecé con la implementación de la nueva funcionalidad y me surgieron varias dudas por lo que contacté con Juan David pero no fue hasta mediados de octubre que pudo atenderme debido a que el es profesor y con el comienzo del colegio estaba muy liado. Una vez me resolvió las dudas comencé con la implementación en la cual avancé bastante, modifiqué la arquitectura añadiendo el componente y servicio que eran necesarios y modifiqué la interfaz pero a finales de noviembre volví a tener dudas respecto a los tensores para la extracción de características y con los datos categóricos, dudas que me resolvió y a mediados de diciembre tenía la nueva funcionalidad implementada y quedé con Juan David para enseñárselo pero me comento que le gustaría cambiar el diseño de como mostrar los atributos y como solicitarlos. Después de las navidades a mediados de enero realicé esos cambios, se los enseñé y dimos por finalizada la implementación de la nueva funcionalidad, para llevar a cabo la implementación le dediqué unas 6 horas diarias y cuatro días a la semana. Por último quedaba crear la memoria y para ello me volví a poner en contacto con Gregorio y me pasó la plantilla con los apartados a rellenar. Con la memoria comencé el 2 de febrero dedicándole unas 5-6 horas diarias, seis días a la semana y finalmente el de febrero terminé.

Capítulo 3

Estado del arte

3.1. Inteligencia Artificial

La IA (*Inteligencia Artificial*) [14] se puede definir como la habilidad de los ordenadores para hacer actividades que normalmente requieren inteligencia humana. Aunque de forma más técnica se puede definir como la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones como si se tratase de un humano.

El uso de la IA cada vez es mayor, ya que nos ayuda a beneficiarnos de mejoras significativas y disfrutar de una mayor eficiencia en casi todos los ámbitos de la vida. Pero debemos estar atentos para prevenir y analizar las posibles desventajas directas o indirectas que pueda generar el crecimiento del uso de la IA. La IA se puede aplicar a una inmensidad de situaciones, algunas de ellas son las siguientes:

- Cambiará la forma de hacer negocios debido a que las empresas que busquen entender y aplicar estas herramientas de forma rápida y eficaz obtendrán ventajas competitivas. Por ejemplo, mejoras del desempeño de la estrategia algorítmica comercial o detección y clasificación de objetos.
- La IA será capaz de ofrecernos sugerencias y predicciones relacionadas con temas importantes de nuestra vida, esto supondrá un impacto en áreas como la salud, el bienestar, la educación, el trabajo y las relaciones interpersonales. Por ejemplo, el procesamiento eficiente y escalable de datos de pacientes implicará que la atención médica sea más efectiva y eficiente.

- Permitirá que las máquinas y los robots realicen tareas que los humanos consideran difíciles, aburridas o peligrosas. Además que las máquinas no necesitan descansar y pueden analizar grandes volúmenes de información a la vez y con un porcentaje de error menor que los humanos.

3.2. Machine Learning

El machine learning o aprendizaje automático [14] es uno de los enfoques fundamentales de la inteligencia artificial. Trata un elemento de la informática en el que los ordenadores o las máquinas tienen la capacidad de aprender sin estar programados para ello. El aprendizaje automático usa algoritmos para aprender de diferentes patrones de datos, es decir, es un conjunto de algoritmos con los que se construyen modelos de predicción y clasificación a partir de conjuntos de datos. Un ejemplo es la personalización de los sitios de medios sociales como Facebook o los resultados del motor de búsqueda de Google. Otro ejemplo son los filtros de Spam en el correo electrónico. Aprenden en base a unos patrones que tipo de mensajes son correo basura y cuales no y toman la decisión de clasificarlos como tal o no.

En el aprendizaje automático se diferencian tres tipos de aprendizaje:

- **Aprendizaje supervisado:** es el humano quien tiene que suministrar datos etiquetados y organizados para indicar cómo tendría que ser categorizada la nueva información o entrada. Por ejemplo, enseñar previamente al algoritmo fotos donde aparezca un perro para que luego pueda identificar imágenes similares. Es el tipo de aprendizaje utilizado en LearningML.
- **Aprendizaje no supervisado:** es el algoritmo el que tiene que encontrar la forma de asignar a los datos una etiqueta sin información previa, de manera que no precisa de la intervención de un humano.
- **Aprendizaje por refuerzo:** el algoritmo aprende de la experiencia. Cuando el algoritmo acierta al clasificar una entrada recibe un refuerzo positivo y de esta forma mejora con el uso.

3.3. LearningML

LearningML [10] es una aplicación web creada por Juan David Rodríguez García en 2020, pero fue en 2019 cuando al autor decidió crear LearningML al participar en la confección de un recurso educativo¹ sobre la enseñanza del aprendizaje automático en la escuela. La idea de crear LearningML surge porque la mayoría de las plataformas educativas de programación existentes carecen de algunas características necesarias para desarrollar proyectos completos de IA.

La aplicación web consiste en facilitar la enseñanza de contenidos sobre el aprendizaje automático y el fomento del pensamiento computacional a alumnos de edades comprendidas entre los 10 y los 16 años. Para comprobar si era una herramienta válida para enseñar, aprender machine learning y de uso sencillo se llevó a cabo una investigación² que resultó satisfactoria. En la actualidad, es una herramienta que está siendo utilizada en ambientes educativos reales.

Hoy en día LearningML dispone de dos funcionalidades básicas, reconocimiento de textos y reconocimiento de imágenes. Además cuenta con una amplia traducción a distintos idiomas.

3.4. Angular

Angular [11, 3] nació en 2010 con el nombre de AngularJS pero en el año 2016 pasó a llamarse Angular, en su versión 2.0 y ha ido evolucionando e implantando mejoras. La última versión es la 13 publicada en noviembre de 2021.

Angular es un *framework* de código abierto desarrollado por Google que mediante el uso de TypeScript y HTML permite crear aplicaciones de una sola página, denominadas SPA (*Single Page Application*).

Las ventajas que tiene Angular es que mantiene la aplicación más ordenada, simplifica el código y evita escribir código repetitivo gracias a que sigue un modelo MVC (*Modelo-Vista-Controlador*), a la vez que posibilita que las modificaciones y las actualizaciones de las aplicaciones sean rápidas y sencillas. Además separa el *frontend* y el *backend* en la aplicación.

La principal ventaja que tienen las SPA es que tiene una alta velocidad de carga entre las vistas que tiene la aplicación web, ya que cuando hay un cambio de vista no se recarga la página si no que las vistas se cargan de forma rápida, dinámica y reactiva. Esto se debe a que solamente

¹https://code.intef.es/prop_didacticas/inteligencia-artificial-en-el-aula-con-scratch

²<https://web.learningml.org/investigacion-sobre-learningml/>

hay una petición inicial al servidor y una respuesta HTML del mismo, luego funciona mediante routing.

Como cualquier *framework*, Angular tiene una serie de librerías que se pueden importar para facilitar el desarrollo del código, o para solucionar problemas concretos. Las librerías amplían las funcionalidades.

La arquitectura de una aplicación Angular está basada en cuatro clases distintas, que se identifican a través de decoradores. Estos decoradores definen su tipo y proporcionan metadatos que le indican a Angular cómo usarlos. Las cuatro clases que utiliza son:

- **Módulos:** declaran un contexto de compilación para un conjunto de componentes. Los módulos juegan un papel fundamental en la estructuración de las aplicaciones Angular. Es donde se definen o declaran los componentes, las directivas y los servicios que conforman la aplicación. De manera que representa una agrupación lógica de lo que podríamos llamar áreas funcionales de una aplicación. También definen las rutas que establecen las vistas de la aplicación y las dependencias con otros módulos, es decir, que módulos necesita importar y que componentes o directivas exporta.

Cualquier aplicación de Angular tiene un módulo raíz, llamado `AppModule`, que proporciona el mecanismo de arranque que inicia la aplicación, pero no es el único, normalmente una aplicación contiene varios módulos funcionales, además de los propios del *framework*.

Los módulos de Angular pueden importar o exportar funcionalidades de otros módulos, esto hace que cada módulo sea independiente. La organización en distintos módulos ayuda a gestionar el desarrollo de aplicaciones complejas y a la reutilización de código.

Para definir una clase como módulo se utiliza el decorador `@NgModule`.

- **Componentes:** son los bloques de construcción que componen una aplicación, contienen la lógica y los datos de la aplicación. Cada componente tiene asociada una plantilla HTML. Las plantillas son las vistas que conforman la interfaz de usuario y se cargan cuando se cambia o modifica la URL.

Una aplicación de Angular tiene al menos el componente raíz, llamado `AppComponent`, que conecta una jerarquía de componentes con el modelo de objeto del documento (DOM) de la página.

Los componentes pueden tener uno o varios subcomponentes, estos pueden relacionarse entre sí de dos formas. Mediante eventos para actualizar datos cuando el usuario interactúa con la interfaz gráfica y mediante las propiedades o interpolación para enviar datos a las plantillas que conforman las vistas. Esto hace que los cambios en el DOM pueden modificar los datos de la aplicación y los datos de la aplicación pueden modificar las plantillas.

Para definir una clase como componente se utiliza el decorador `@NgComponent`.

- **Directivas:** son clases que agregan comportamiento adicional a los elementos en sus aplicaciones Angular para modificar de alguna manera el HTML. Antes de que se muestre una vista, Angular evalúa las directivas y resuelve la sintaxis vinculante en la plantilla para modificar los elementos HTML y el DOM.

Hay tres tipos de directivas:

- * Directivas de componente: los componentes son un tipo de directiva.
- * Directivas de atributos: modifican el comportamiento o la apariencia de un elemento, componente o directiva.
- * Directivas estructurales: modifican la apariencia agregando y eliminando elementos del DOM.

Se pueden crear directivas o Angular dispone de directivas integradas que se pueden usar para administrar formularios, listas, estilos y lo que ven los usuarios. Algunos ejemplos de directivas de atributos integradas son `NgClass`, `NgStyle` y `NgModel`, y de directivas estructurales `NgIf`, `NgFor` y `NgSwitch`.

- **Servicios:** proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Son clases con un propósito limitado y bien definido, deben hacer algo específico y de uso general. Se usan para cambiar datos entre componentes, obtener datos del servidor, validar la entrada del usuario o cualquier servicio que se repita en distintas componentes.

Angular distingue los componentes de los servicios para aumentar la modularidad y la reutilización. Al separar la funcionalidad relacionada con la vista de un componente de

otros tipos de procesamiento, puede hacer que sus clases de componentes sean sencillas y eficientes.

Para definir una clase como servicio se utiliza el decorador `@Injectable`, ya que proporciona los metadatos que permiten inyectar otros proveedores como dependencias en su clase.

3.4.1. Angular CLI

Angular CLI (*Command Line Interface*) [12] es una herramienta de interfaz de línea de comandos desarrollada por Angular que se utiliza para inicializar, desarrollar, montar y mantener aplicaciones de Angular directamente desde un shell de comandos.

Es una herramienta que facilita el inicio de una aplicación de Angular, porque con una instrucción `ngnew`, crea una carpeta de trabajo y genera el esqueleto de aplicación. También tiene instrucciones que permiten crear nuevas clases de forma sencilla, creando los distintos archivos que forman un módulo, componente, servicio o una directiva, además de incluirlos en los archivos necesarios y crear las dependencias. Las herramientas predefinidas más destacadas son el servidor web, el compilador y el sistema de testing.

3.4.2. MVC

Para el desarrollo de una aplicación web Angular utiliza el patrón MVC (*Modelo-Vista-Controlador*) [2] que se utiliza para separar en tres componentes los datos, la metodología y la interfaz gráfica de una aplicación, lo que permite modificar cada uno de ellos sin tener que modificar los demás. La arquitectura MVC está formada por tres componentes, que son:

- **Modelo:** es el encargado de la manipulación, gestión y actualización de los datos. En el caso de que haya una base de datos es donde se realizan las consultas, búsquedas o actualizaciones.
- **Vista:** es la representación gráfica de los datos proporcionados por el controlador. Ni el controlador ni el modelo se preocupan de cómo se verán los datos, toda la parte del diseño de la interfaz es responsabilidad de la vista.

- **Controlador:** es el componente principal, se encarga de gestionar las instrucciones que se reciben del usuario, atenderlas y procesarlas. Después realiza las consultas al modelo y una vez se hayan obtenido dichos datos, se envía a la vista para producir una salida como respuesta a el evento.

Las ventajas que aporta esta arquitectura son: facilita el mantenimiento, permite la reutilización de componentes y mejora la escalabilidad, ya que separa cada tipo de lógica.

3.5. HTML5

HTML (*HyperText Markup Language*) [8, 9, 13] es un lenguaje de marcado que se utiliza para definir la estructura y el contenido de una página Web, y es el componente más básico de una web. Se escribe en texto plano y tiene una estructura de árbol de elementos y textos. La estructura básica de un documento HTML se compone de los siguientes elementos: `<!DOCTYPE>`, `<html>`, `<head>` y `<body>`. HTML provee los elementos estructurales y se complementa con CSS para modificar la apariencia de la pagina y con JavaScript para proveer dinamismo y construir aplicaciones web completamente funcionales.

La idea de desarrollar HTML5 nació en 2004, cuando el consorcio W3C (*World Wide Web Consortium*) que está a cargo del estándar decidió dejar de evolucionar HTML, por lo que la asociación WHATWG (*Web HyperText Application Technology Working Group*), formada por Apple, Opera y Mozilla, decidió crear HTML5 y publicó el primer borrador de HTML5. En el año 2007 W3C formó un grupo de trabajo autorizado para trabajar con WHATWG en el desarrollo de HTML5, pero en 2011 se separaron debido a que tenían distintos objetivos, W3C quería publicar una versión terminada, mientras que WHATWG quería seguir trabajando en una constante evolución de HTML5. En 2019, WHATWG y W3C firmaron un acuerdo para colaborar en una única versión de HTML en el futuro.

Fue creado con la intención de hacerlo más eficiente que las versiones anteriores, mantener la compatibilidad con versiones anteriores y facilitar el desarrollo web compatible con distintos navegadores. Las principales novedades de HTML5 son:

- Incorpora nuevas etiquetas que permiten una mejor estructuración de los documentos HTML, ya que antes solo estaba el `<div>` para definir secciones. Ahora se pueden uti-

lizar también `<header>` para la cabecera, `<section>` para la información principal, `<nav>` para la barra de navegación y `<footer>` para el pie, entre otros.

- Permite incorporar elementos multimedia para reproducir audios y videos desde el propio navegador, con las etiquetas `<audio>` y `<video>`.
- Incluye numerosas APIs, algunas de ellas son:
 - * Forms: incluye mejoras en los formularios para personalizar todos los aspectos de procesamiento y validación.
 - * Canvas: permite dibujar, presentar gráficos en pantalla, animar y procesar imágenes y texto.
 - * Geolocation: permite a los desarrolladores determinar la ubicación física real del usuario y e mostrar e interactuar con un mapa de Google Maps.
 - * Drag and drop: permite arrastrar un elemento desde un lugar y luego soltarlo en otro.
 - * Web Storage: es una mejora de las cookies y permite el almacenamiento local en el lado del cliente, con `localStorage` cuando la información tiene que estar disponible solo durante la sesión y `sessionStorage` cuando tiene que ser preservada todo el tiempo que el usuario desee

3.6. CSS3

CSS3 (*Cascading Style Sheets*) [13] es la última versión de CSS. Es un lenguaje de diseño gráfico que permite definir y crear el aspecto visual de los documentos HTML. Es un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML, por lo que se separa el estilo del documento de la estructura y el contenido de este. Esta separación permite que varios documentos HTML tengan la misma hoja de estilo, simplificando el trabajo de los desarrolladores, o que un mismo documento HTML pueda tener distintos estilos, permitiendo variaciones en la visualización para diferentes dispositivos y tamaños de pantalla.

El consorcio W3C es el encargado de definir y mantener las especificaciones de este lenguaje. Las principales novedades de CSS3 son: esquinas redondeadas, sombras en cajas y textos,

elegir la fuente del texto, gradientes, opacidad y transparencia de los colores, incluir transiciones, entre otras.

3.7. TypeScript

TypeScript [7] es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, es decir, amplía JavaScript con una nueva sintaxis que añade tipos estáticos y objetos basados en clases. Nació en 2012 como solución para el desarrollo de aplicaciones robustas y de gran tamaño, ya que con JavaScript era muy complicado debido a su escasa escalabilidad. Su uso está permitido tanto en el lado del cliente como en el lado del servidor con Node.js

Los programas de JavaScript son compatibles en TypeScript, por lo que se puede integrar en proyectos ya existentes porque a través de un compilador de TypeScript se traducen a código JavaScript original. Además tiene las herramientas de JavaScript ES6 y JavaScript ES7, de esta manera se mantiene actualizado con las últimas mejoras de JavaScript. Los cambios más importantes respecto a JavaScript son:

- Evita errores en tiempo de ejecución, ya que incorpora el tipado estático, es decir, al crear variables se puede añadir el tipo de dato.
- Es más funcional, ya que al añadir objetos basados en clases hace que la programación orientada a objetos sea más sencilla

En la actualidad se encuentra en la versión 2.0 que introduce varias características, entre ellas la capacidad de evitar la asignación de variables con un valor nulo.

3.8. Node.js

Node.js [5] fue creado por Ryan Dahl en 2009. Es un entorno en tiempo de ejecución de JavaScript orientado a eventos asíncronos, diseñado para construir aplicaciones de red escalables, como por ejemplo, servidores web. Está influenciado por sistemas como Event Machine de Ruby y Twisted de Python.

Con Node.js el código JavaScript, no se ejecuta en un navegador, sino en el servidor. Utiliza el motor V8 de Google para interpretar y ejecutar el código JavaScript, suministrando un entorno de ejecución en el lado del servidor. Del mismo modo que hace el navegador cuando se utiliza JavaScript en el lado del cliente.

Al estar orientado a eventos asíncronos puede soportar una gran cantidad de conexiones. Esto lo diferencia del modelo de concurrencia más común, en el que se emplean hilos del Sistema Operativo que usan un hilo por cada conexión, lo que provoca que si se tienen muchas conexiones son necesarios muchos servidores. Node.js utiliza un único hilo de ejecución y un bucle de eventos asíncrono, tratando las nuevas conexiones como un nuevo evento dentro del bucle y cada conexión recibe una pequeña asignación de espacio de memoria dinámico, sin tener que crear un hilo de ejecución, evitando que se produzcan bloqueos en el flujo de trabajo.

Node.js utiliza un único hilo de ejecución y un bucle de eventos asíncrono, tratando las nuevas conexiones como un nuevo evento dentro del bucle y cada conexión recibe una pequeña asignación de espacio de memoria dinámico, sin tener que crear un hilo de ejecución, de esta manera evita que se produzcan bloqueos en el flujo de trabajo y permite soportar una gran cantidad de conexiones. Esto lo diferencia del modelo de concurrencia más común, en el que se emplean hilos del Sistema Operativo que usan un hilo por cada conexión, lo que provoca que si se tienen muchas conexiones son necesarios muchos servidores.

3.9. TensorFlow

TensorFlow [6] es el sistema de aprendizaje automático de segunda generación de Google Brain, liberado como software de código abierto en 2015, por lo que el acceso a esta herramienta es libre y es posible editarla en función de las necesidades.

Es una plataforma para construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, parecidos al aprendizaje y razonamiento usados por los humanos. Tiene un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite a los desarrolladores crear e implementar fácilmente aplicaciones basadas en ML.

3.10. JSON

JSON (*JavaScript Object Notation*) [4] es un formato ligero de intercambio de datos. Es fácil de leer y escribir para los humanos, mientras que para las máquinas es sencillo de interpretar y generar. A pesar de que está basado en un subconjunto del lenguaje de programación JavaScript, es un formato de texto completamente independiente del lenguaje. JSON está construido por dos estructuras:

- Objeto: colección de pares de nombre/valor. Comienza con una llave de apertura y termina con una llave de cierre, cada nombre es seguido de dos puntos y los pares nombre/valor están separados por una coma.
- Array: lista ordenada de valores. Comienza con un corchete izquierdo y termina con un corchete derecho, los valores se separan por comas.

Estas estructuras pueden anidarse y los valores pueden ser:

- Cadena de caracteres. Se ponen entre comillas dobles.
- Números. Pueden contener parte fraccional separada por un punto y se permiten números negativos.
- Booleanos. True y false.
- Null. Representa el valor nulo.

3.11. Conjunto de datos

Un conjunto de datos o dataset es una colección de datos tabulados, es decir, en forma de tabla. Las columnas de la tabla son los atributos, que son el nombre de las características de los datos. Las filas de la tabla son las muestras con los datos de los atributos, los datos pueden ser numéricos y categóricos, es decir, cadenas de caracteres. Las muestras se agrupan en clases, una clase tiene muestras con valores similares por lo que las muestras definen como es una clase. Un ejemplo de un conjunto de datos muy utilizado en el aprendizaje automático es el *conjunto de datos flor iris*³, está compuesto de tres clases que son tres especies de la flor iris (iris setosa, iris

³https://es.wikipedia.org/wiki/Conjunto_de_datos_flor_iris

virginica e iris versicolor), cuatro atributos (largo de sépalo, ancho de sépalo, largo de pétalo y ancho de pétalo) y 50 muestras de cada clase que definen como son cada flor.

Capítulo 4

Diseño e implementación

4.1. Arquitectura de LearningML

La arquitectura de LearningML está construida con Angular, por lo que está compuesta por módulos, componentes y servicios, de manera que es sencillo realizar cambios o añadir funcionalidades.

En la figura 4.1 se puede ver la arquitectura general de LearningML, primero está el servicio `labeled-data-manager`, es de uso general y lo importan la mayoría de componentes y algunos servicios, ya que se encarga de guardar algunos datos que necesitan los servicios y componentes. Luego está el componente `ml-home` que se corresponde con la página principal y es donde se selecciona que tipo de reconocimiento se desea realizar, reconocimiento de textos o de imágenes. Una vez se ha seleccionado se pasa al componente `ml-model` mediante routing con el modelo `app-routing` que funciona mediante rutas, por lo que además de cambiar de componente también se cambia de página. En función del reconocimiento escogido se mostrarán unos elementos u otros del HTML de `ml-model`.

En el componente `ml-model` se importan los servicios `feature-extraction` y `ml-algorithm`, que a su vez el primero importa dos servicios: `feature-extractor-text` y `feature-extractor-image` y el segundo importa otros dos servicios: `ml-algorithm-knn` y `ml-algorithm-neural-networks`.

El componente `ml-model` es el constructor del modelo y está dividido en tres secciones, la primera es donde se introducen los datos de entrenamiento, primero se introducen las distintas clases a las que puede pertenecer una entrada y una vez se ha introducido al menos una clase,

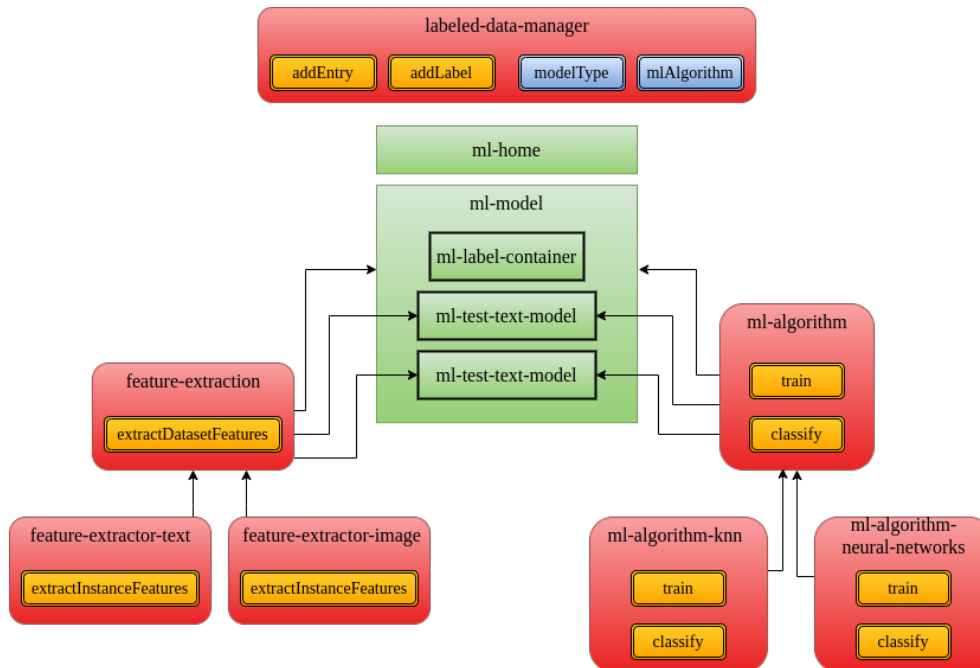


Figura 4.1: Arquitectura general de LearningML. En rojo los servicios, en verde los componentes, en amarillo las funciones y en azul las variables.

entra en funcionamiento el componente `ml-label-container` que es donde se recogen las entradas de cada clase con los datos que se utilizan en la segunda sección, la parte de aprendizaje.

En la parte de aprendizaje es donde se utiliza la función `train` que es donde realmente se realiza la construcción del modelo y se entrena, primero se realiza la extracción de características mediante el servicio `feature-extraction` y este es el que elige dependiendo del tipo de dato cual de los dos servicios usar para la extracción de características. Después llama a `ml-algorithm` y este servicio decide que algoritmo utilizar para construir y entrenar el modelo para el reconocimiento. Por defecto utiliza el algoritmo `ml-algorithm-neural-networks`, pero se puede seleccionar el modo avanzado en el cual se puede elegir entre los dos y modificar los parámetros del algoritmo.

Cuando el modelo ya está entrenado se pasa a la tercera sección, la parte de testeo, que es donde se realizan las pruebas de reconocimiento. En función del reconocimiento escogido se hace uso del componente `ml-test-text-model` o del componente `ml-test-image-model`. Primero llaman a `feature-extraction` y después a `ml-algorithm`, igual que `ml-model`.

En la figura 4.1 también se puede observar que contiene principalmente cada servicio nom-

brado anteriormente. Estos servicios tienen un desempeño específico:

- **labeled-data-manager:** es el encargado de guardar datos, estos datos son características que definen el modelo. Guarda las etiquetas, es decir, las clases a las que puede pertenecer una entrada, esto se hace con la función `addLabel` que guarda las etiquetas en un array en la variable `labels` y en un objeto Map en la variable `labelsWithData`. Guarda los datos de entrada, ya sean textos o imágenes, con la función `addEntry` que los guarda en la variable `labelsWithData` junto con la clase en la que ha sido añadida la entrada para así tener cada etiqueta con sus datos. Guarda el tipo de modelo que se va a construir, es decir, el tipo de reconocimiento seleccionado, en la variable `modelType`. También guarda el tipo de algoritmo en la variable `mlAlgorithm`. Además de guardar estos datos también borra etiquetas o entradas y se encarga de guardar o cargar los datos de un JSON tanto del ordenador como de la nube si el usuario está registrado.
- **feature-extraction:** es el encargado de llamar a los otros dos servicios en función de si se quieren extraer características de un texto o de una imagen mediante la función `extractDatasetFeatures`.
 - * **feature-extractor-text:** es el encargado de extraer las características de los textos, es decir, convertir un texto en un tensor. La función que lo convierte en un tensor es `extractInstanceFeatures`
 - * **feature-extractor-image:** es el encargado de extraer las características de las imágenes, es decir, convertir una imagen en un tensor. La función que lo convierte en un tensor es `extractInstanceFeatures` al igual que en el caso de los textos.
- **ml-algorithm:** es el encargado de llamar a los otros dos servicios en función del algoritmo que se utilice. Tiene dos funciones principales, `train` y `classify` con las que llama a los otros servicios para entrenar o clasificar.
 - * **ml-algorithm-knn:** se usa en caso de que el algoritmo elegido sea K-KNN. Tiene dos funciones que son las que llama el servicio `ml-algorithm`, `train` para entrenar o `classify` para clasificar.
 - * **ml-algorithm-neural-networks:** tiene las mismas funciones que K-KNN pero en este caso entrena o evalúa utilizando el algoritmo de redes neuronales.

4.2. Cambios en la arquitectura de LearningML

Realizar cambios en la arquitectura no fue complicado, ya que la aplicación está hecha para poder realizar cambios e implementar nuevas funcionalidades. Solo hay que entender como esta estructurada y que es lo que hace cada parte.

Para añadir la nueva funcionalidad tenía dos opciones en cuanto a la construcción del modelo: crear un nuevo componente o añadir la funcionalidad en el componente `ml-model`, me decidí por la segunda para seguir con la misma estructura, ya que no había un componente diferente para construir el modelo de textos y otro para imágenes, por lo que me parecía más lógico seguir con la misma dinámica.

Los cambios en la arquitectura se pueden observar en la figura 4.2. Se ha añadido el componente `ml-test-dataset-model` para realizar el testeo cuando el tipo de reconocimiento es de conjuntos de datos, tiene la misma funcionalidad que cuando son textos o imágenes pero adaptado al tipo de datos que tiene un conjunto de datos y sigue utilizando e importando los servicios `feature-extraction` y `ml-algorithm`, para indicar como son ese tipo de datos se añadieron en el fichero `interfaces`.

También se ha añadido el servicio `feature-extractor-dataset` para extraer las carac-

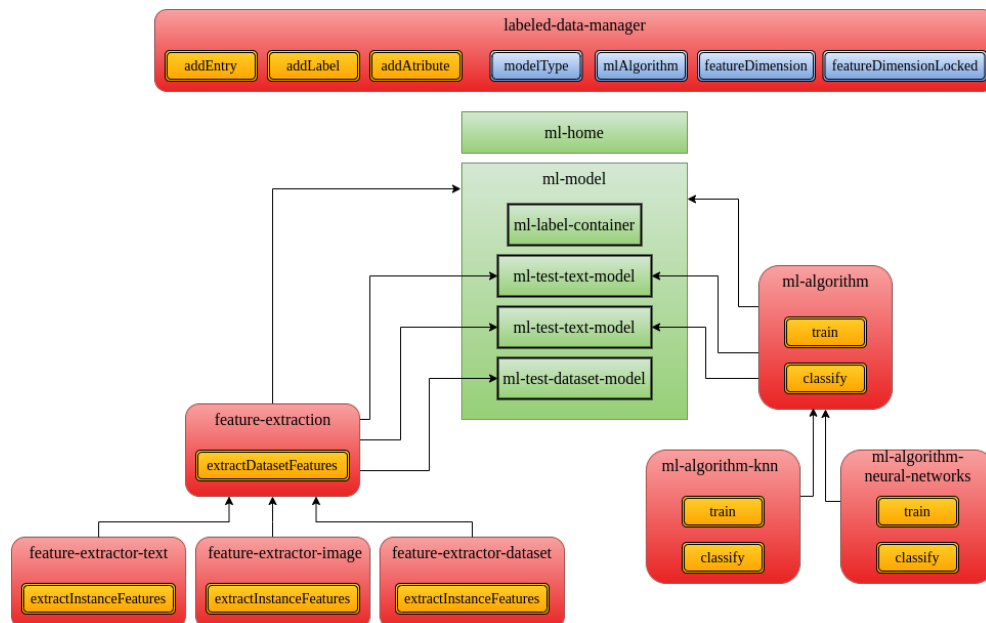


Figura 4.2: Arquitectura nueva de LearningML. En rojo los servicios, en verde los componentes, en amarillo las funciones y en azul las variables.

terísticas de los conjuntos de datos, es decir, convertir un conjunto de datos en un tensor. La función que lo convierte en un tensor es `extractInstanceFeatures` al igual que en el caso de textos e imágenes.

Además en el servicio `labeled-data-manager` se ha añadido la función `addAttribute` para guardar los atributos del conjunto de datos en un array en la variable `attributes`. Se han añadido variables para manejar cambios que se producen en el modelo como son `featureDimensionLocked` y `addAttributeLocked`, son variables booleanas en las cuales cuando se cumple una condición cambian de valor y la condición son cambios en el modelo. También ha añadido la variable `featureDimension` en la que se guarda el número de atributos que va a tener el modelo.

Como este servicio también se encarga de cargar los datos de un JSON se ha añadido el caso en las funciones `load` y `loadFromServer` para cuando el `modelType` es `dataset`. Para cuando se guardan los datos se ha creado un nuevo objeto para cuando se genere el JSON, ya que ahora además de tener `type` con el tipo de modelo y `data` con las clases y sus entradas, también tiene `attributes` con los atributos. Para esto se ha creado la función `getDataObject` que es llamada por la función `serializeModel` que a su vez es llamada por las funciones `save` y `saveOnServer`.

4.3. Cambios en la interfaz de LearningML

Los cambios realizados en la interfaz de usuario son:

- En el componente `ml-home` se ha añadido un `container` para el conjunto de datos como se puede ver en la figura 4.3.
- En el componente `ml-model` se han realizado cambios en dos de las tres secciones:
 - * Sección de entrenamiento: hay que crear el conjunto de datos, por lo que se añade un `div` para cuando el `modelType` es `dataset` porque ya no se introducen las clases lo primero. Ahora hay que indicar el número de atributos que tiene el dataset, mediante un input. A continuación hay que añadir los atributos, mediante un botón. Si el número de atributos es correcto se muestran debajo en un desplegable. Una vez se tiene definido como va a ser el dataset, es decir, se tienen añadidos los atributos,

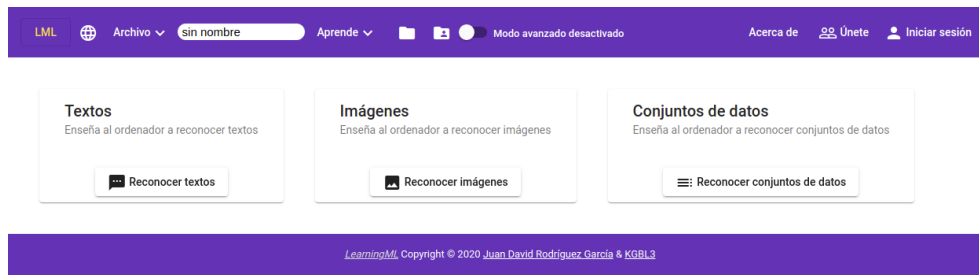


Figura 4.3: Página principal de LearningML.



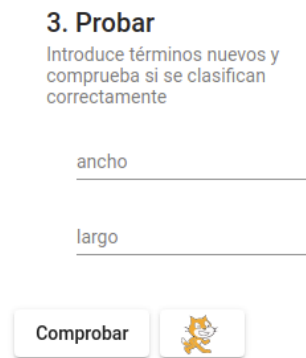
Figura 4.4: Sección de entrenamiento de `ml-model`.

aparece el botón para añadir una clase a las que puede pertenecer una entrada, esta parte está igual que antes. En la figura 4.4 se puede observar como queda la sección de entrenamiento una vez introducidos los atributos.

También se añade una ventana emergente para cuando hay que añadir los atributos, ya que es diferente a cuando hay que añadir una clase. El fichero se llama `ml-add-attribute-dialog`.

* Sección de testeo: se ha añadido la etiqueta con el selector del componente `ml-test-dataset-model`.

- En el componente `ml-test-dataset-model` se muestran tantos inputs como número de atributos se haya indicado y cada uno corresponde con un atributo para introducir



3. Probar
Introduce términos nuevos y
comprueba si se clasifican
correctamente

ancho

largo

Comprobar

Figura 4.5: Componente `ml-test-dataset-model`.

los valores de cada atributo y comprobar si se clasifican correctamente. Se puede observar en la figura 4.5.

4.4. Creación de conjuntos de datos

Los conjuntos de datos se pueden crear o se pueden importar en formato JSON. En esta sección se explica como crearlo desde `LearningML`. Se crea en el componente `ml-model` en la sección de entrenamiento y en el componente `ml-label-container`. Al ser una colección de datos tabulados el conjunto se divide: en atributos, en ejemplares o muestras y en clases.

4.4.1. Atributos

Los atributos es lo primero que se le solicita al usuario, ya que define como va a ser el dataset. Este proceso se realiza en el componente `ml-model`. Primero hay que indicar el número de atributos que tiene el dataset y se guarda en la variable `featureDimension` de `labeled-data-manager`, por defecto tiene dos atributos. Después al pulsar el botón para añadir los atributos, este llama a la función `addAttribute` que deshabilita el input de introducción de número de atributos, esto es posible gracias a la variable `featureDimensionLocked` de `labeled-data-manager`. A continuación abre la ventana emergente `ml-add-attribute-dialog` para escribir los atributos separados por comas. Si el número de atributos introducidos coincide con los de la variable `featureDimension` y todos tienen asignado un valor, es decir, se ha escrito antes y después de cada coma, se mandan los atributos a la fun-

ción `addAttribute` de `labeled-data-manager` para guardarlos y poder mostrarlos debajo en un desplegable para tener presente cuales son las características que tienen que tener los datos de entrada y para bloquear el botón para añadir atributos gracias a la variable `addAttributeLocked`. En el caso de no introducir los atributos correctamente, estos no se guardan y sigue activo el botón para añadir atributos hasta que se introduzcan correctamente.

El input para indicar el número de atributos y el botón para añadir atributos se bloquean porque todo el dataset tiene que tener la misma forma, es decir, todas las entradas deben de tener el mismo número de atributos, por lo que si ya se han añadido entradas y luego se modifica el número de atributos las entradas no van a coincidir en el número de atributos y esto no puede pasar.

4.4.2. Clases

Una vez se han añadido los atributos ya se pueden añadir las clases del conjunto de datos, en este caso el botón no se bloquea en ningún momento, por lo que no es necesario crear todas las clases del conjunto de datos antes de introducir las entradas de cada clase sino que en cualquier momento se puede añadir una clase.

Para añadir clases no se ha añadido nada, funciona como antes. El botón para añadir una nueva clase llama a la función `addLabel` que abre una ventana emergente para escribir la clase y luego manda el valor introducido a la función `addLabel` de `labeled-data-manager` para guardarlo en las variables `labels` y `labelsWithData`. La variable `labelsWithData` está compuesta de pares de datos clave-valor `{label, data}`, cuando se añade una clase se añade un par de datos clave-valor, por lo que hay tantos pares de datos como clases tenga el conjunto de datos.

4.4.3. Muestras

Las muestras de cada clase se añaden en el componente `ml-label-container`, para ello se ha creado la función `addRowToDataset` porque la forma que tienen las entradas es diferente, añade una fila con los valores de los atributos en la clase que se quiere añadir. Primero abre una ventana emergente para añadir los valores separados por comas y si el número de valores introducidos coincide con el número de atributos de la variable `featureDimension`

```
▼ Map(2) {'cuadrado' => Array(3), 'rectángulo' => Array(3)}
  ▼ [[Entries]]
    ▼ 0: {"cuadrado" => Array(3)}
      key: "cuadrado"
      ▶ value: (3) ['1,1', '2,2', '3,3']
    ▼ 1: {"rectángulo" => Array(3)}
      key: "rectángulo"
      ▶ value: (3) ['1,2', '2,4', '1,3']
```

Figura 4.6: Aspecto de `labelsWithData`.

de `labeled-data-manager`, todos tienen asignado un valor, es decir, se ha escrito antes y después de cada coma, y la muestra no está repetida, es decir, no contiene los mismos valores y en el mismo orden que otra ya introducida en esa clase, escribe la muestra en la interfaz de usuario y manda la muestra a la función `addEntry` de `labeled-data-manager` que guarda la muestra en la variable `labelsWithData` en la clase que ha sido añadida. Como `labelsWithData` está compuesta de pares de datos clave-valor `{label, data}` busca la clase en la que se quiere añadir la muestra y añade el string de datos de la muestra en el array `data`. En la figura 4.6 se puede observar el aspecto que tiene `labelsWithData`, este ejemplo consta de dos pares de datos clave-valor, uno por cada clase y las muestras que pertenecen a cada clase.

4.5. Extracción de características

Una vez se tiene el conjunto de datos creado se procede al entrenamiento del modelo pero para ello hay que hacer la extracción de características, es decir, convertir las muestras de las diferentes clases guardadas en la variable `labelsWithData` en tensores.

Cuando se pulsa el botón aprender, este llama a la función `train` de `ml-model` que crea la variable `featurePromise` en la cual se llama a la función `extractDatasetFeatures` del servicio `featureExtraction` de forma asíncrona. Al tratarse de programación asíncrona, se utiliza la función `then` cuya función consiste en esperar a que la función asíncrona devuelva un valor para después hacer uso de ese valor. En la función `extractDatasetFeatures` se crea el `feature-extractor-dataset` ya que el `modelType` es `dataset` y lo guarda en la variable `featureExtractor`, se crea llamando a la función `create` de `feature-extractor-dataset` a la cual se le pasa como parámetro la variable `labelsWithData` para que el servicio `feature-extractor-dataset` guarde los datos de `labelsWithData`. Después llama

a `extractDatasetFeatures` de `featureExtractor`.

A continuación la función `extractDatasetFeatures` de `feature-extractor-dataset` llama a la función `buildDataset` de forma asíncrona, esta función construye el objeto `dataset` con los datos con los que pueden trabajar los algoritmos y que tiene la siguiente estructura:

```
dataset = {
    data: [],
    labels: [],
    dataArray: []
};
```

`labels` es un array que contiene el índice que le corresponde a la clase de cada muestra, `data` es un array que contiene los valores de cada muestra en forma de tensor y `dataArray` es un array que contiene los valores de cada muestra en forma de array con valores numéricos.

La construcción del objeto `dataset` se hace mediante dos bucles, el primero itera sobre las clases y el segundo sobre las muestras de cada clase. Para cada muestra llama a la función `addLabeledSample` que recibe como parámetro un objeto con la muestra y la clase a la que pertenece la muestra, esta función se encarga de añadir las características extraídas de cada muestra al objeto `dataset`. Primero comprueba si la clase está dentro de las clases conocidas por el modelo y si no está la añade. Después llama a la función `extractInstanceFeatures` de forma asíncrona y que recibe como parámetro la muestra para extraer las características, pero para ello tiene que llamar a la función `parseSample` porque los datos de la muestra pueden ser números o una cadena de caracteres. La función `parseSample` recibe como parámetro la muestra que se separa por comas y cada valor de la muestra se pasa a *float*, si es un número se guarda en la variable `sampleNum` y si no es un número se comprueba si el valor de la muestra está en la variable `sampleString` que es un objeto Map clave-valor siendo la clave el valor de la muestra y el valor una id. Si el valor de la muestra no está en la variable `sampleString` se guarda en esa variable el valor de la muestra y en la id un número empezando por cero y sumando uno cada vez que se añada una muestra a la variable. Después se guarda la id de la cadena de caracteres en la variable `sampleNum`, de esta forma se asigna un número a cada cadena de caracteres y cuando se repite una cadena de caracteres tiene el mismo número asignado que se le asignó a esa cadena de caracteres la primera vez que se guardó en la variable `sampleString`.

```

▶ Map(2) {'rojo' => 0, 'azul' => 1}
▶ (4) [0, 2, 1, 3]

```

Figura 4.7: Muestra con números y cadena de caracteres

En la figura 4.7 se puede ver un ejemplo de una muestra con números y cadenas de caracteres, la muestra es 'rojo,2,azul,3' y se observa como asigna un número a una cadena de caracteres, en este caso asigna el cero a 'rojo' y 1 a 'azul' y en la variable `sampleNum` se guarda como [0,2,1,3]. Cuando ya están guardados todos los valores de la muestra en la variable `sampleNum` devuelve un tensor de una dimensión a la función `extractInstanceFeatures` que a su vez devuelve las características extraídas en forma de tensor de una dimensión a la función `addLabeledSample` que añade el tensor a `dataset`, el tensor convertido a array a `dataArray` de `dataset` y el índice que le corresponde a la clase de la muestra a `labels` de `dataset`.

Cuando ya se han añadido todas las muestras de todas las clases al objeto `dataset`, la función `extractDatasetFeatures` devuelve un objeto con la siguiente estructura `{text_labels, tensor_labels, tensor_inputs}` a la función `extractDatasetFeatures` de `featureExtraction` que lo guarda en la variable `extractedFeatures` y se lo devuelve a la función `train` de `ml-model` que lo guarda en la variable `features` para pasárselo a como parámetro a la función `train` de los algoritmos y entrenen con los datos del objeto.

Cuando hay que extraer las características de los datos en la parte de testeo, el componente `ml-test-dataset-model` llama a la función `extractInstanceFeatures` de `feature-extraction` de forma asíncrona y que recibe como parámetro la muestra introducida en la parte de test. Después esta función llama a `extractInstanceFeatures` de `feature-extractor-dataset` que le pasa como parámetro la muestra y llama a `parseSample` igual que en la parte de entrenamiento y luego devuelve la muestra en forma de tensor de una dimensión a la función `extractInstanceFeatures` de `feature-extraction` que a su vez devuelve el resultado a `ml-test-dataset-model` que lo guarda en la variable `inputTensor` para pasárselo como parámetro a la función `classify` de los algoritmos y clasifiquen la muestra de forma asíncrona. Cuando ya ha clasificado la muestra se guarda la predicción en la variable `prediction` que es un array que contiene tantos arrays como clases tenga el modelo y cada array contiene el nombre de la clase y la probabilidad de que esa

muestra pertenezca a esa clase, el primer array es el de mayor probabilidad y es el que se coge para escribir que es esa la clase a la que pertenece la muestra. También se muestra debajo los porcentajes que tiene cada clase de que la muestra le pertenezca.

Capítulo 5

Experimentos, validación y resultados

En esta sección se realizan algunas comprobaciones para ver si los conjuntos de datos se clasifican correctamente. Para ello se usan dos conjuntos de datos, uno con datos numéricos y otro con datos numéricos y datos categóricos para comprobar si se guardan bien los conjuntos de datos y se extraen correctamente las características. Cabe destacar que todas las clases deben tener un número de muestras similar y cuanto mayor sea el conjunto de datos mejor serán los resultados.

5.1. Conjunto de datos numéricos

Para comprobar la clasificación de un conjunto de datos numéricos he utilizado el conjunto de datos de flor iris que se encuentra en el archivo `iris.json` de la carpeta `datasets`. Este dataset está compuesto de tres clases (iris setosa, iris virginica e iris versicolor) y en cada clase se han añadido 48, 49 y 50 muestras respectivamente. Los atributos de las muestras son longitud de sépalo, ancho de sépalo, longitud de pétalo y ancho de pétalo que son las características que diferencian una flor de otra. En la figura 5.1 se pueden observar las muestras de las distintas clases.

La creación del modelo se hace por defecto con el algoritmo de redes neuronales pero para hacer las comprobaciones es mejor hacerlo con el modo avanzado activo, de esta forma podemos probar la validez con el algoritmo de redes neuronales y con el algoritmo de KNN.

Los atributos son:

- Longitud sépalo en cm
- Ancho sépalo en cm
- Longitud pétalo en cm
- Ancho pétalo en cm

+ Añadir nueva clase

Iris-virginica (49)

5.9,3.0,5.1,1.8

6.2,3.4,5.4,2.3

6.5,3.0,5.2,2.0

6.3,2.5,5.0,1.9

6.7,3.0,5.2,2.3

+ -

Iris-versicolor (50)

5.7,2.8,4.1,1.3

5.1,2.5,3.0,1.1

6.2,2.9,4.3,1.3

5.7,2.9,4.2,1.3

5.7,3.0,4.2,1.2

+ -

Iris-setosa (48)

5.0,3.3,1.4,0.2

5.3,3.7,1.5,0.2

4.6,3.2,1.4,0.2

5.1,3.8,1.6,0.2

4.8,3.0,1.4,0.3

+ -

Figura 5.1: Muestras con datos numéricos para el entrenamiento

5.1.1. Redes neuronales

Se elige el clasificador de redes neuronales en el desplegable, se seleccionan los parámetros que en este caso lo dejamos como viene y se escoge el porcentaje de muestras que se usan como ejemplos de validación que en este caso seleccionamos un 25 %, la configuración se puede observar en la figura 5.2.

Hay dos formas de comprobar la validez del modelo con el algoritmo de redes neuronales:

- Con la gráfica de evolución de aprendizaje que muestra la evolución del modelo en cuanto a acierto y error a lo largo de las épocas. En la figura 5.3 se puede ver que cuantas más épocas pasan más aciertos hay y al final de la gráfica el acierto es casi 1, esto sucede porque este algoritmo aprende de sí mismo. Por eso según van aumentando las épocas también aumenta el porcentaje de acierto y por consiguiente baja el porcentaje de fallo. Este resultado indica que el aprendizaje se está realizando correctamente.
- Con la matriz de confusión que nos indica en forma de tabla los datos que predice y los que son en realidad, es decir, el número de aciertos y de fallos que ha tenido pero no de todas las muestras sino del 25 % que se han usado como ejemplo de validación. En la figura 5.4 se puede ver un ejemplo de la matriz de confusión que en este caso tiene un 100 % de acierto, ya que de nueve muestras que predice que son de iris setosa las nueve lo son en realidad, de dieciséis muestras que predice que son iris versicolor las dieciséis son en realidad y de once muestras que predice que son iris virginica las once lo son en realidad. Pero no siempre sale el 100 % depende de las muestras que escoja de ejemplo de validación pero habiendo comprobado bastantes veces el fallo es muy bajo, el acierto suele ser superior al 90 % por lo que se puede considerar un porcentaje de acierto óptimo.

Con los datos obtenidos en la gráfica de evolución de aprendizaje y en la matriz de confusión se puede decir que para este conjunto de datos numéricos el algoritmo de redes neuronales es óptimo para su reconocimiento y el modelo se crea correctamente, esto implica que las muestras se están recogiendo de la forma correcta y la extracción de características es correcta porque los algoritmos ya estaban creados y funcionaban correctamente pero para ello había que pasarles las características extraídas de las muestras de la forma correcta.

Una vez se ha comprobado la validez del modelo, se realizan pruebas en la aplicación con

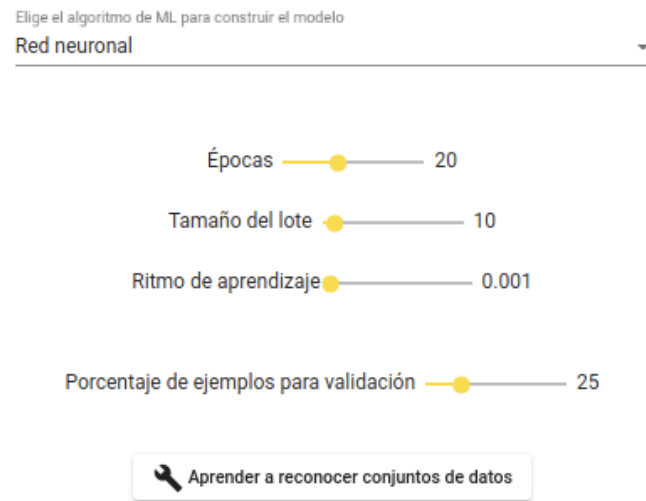


Figura 5.2: Configuración redes neuronales



Figura 5.3: Gráfica de evolución de aprendizaje redes neuronales con datos numéricos

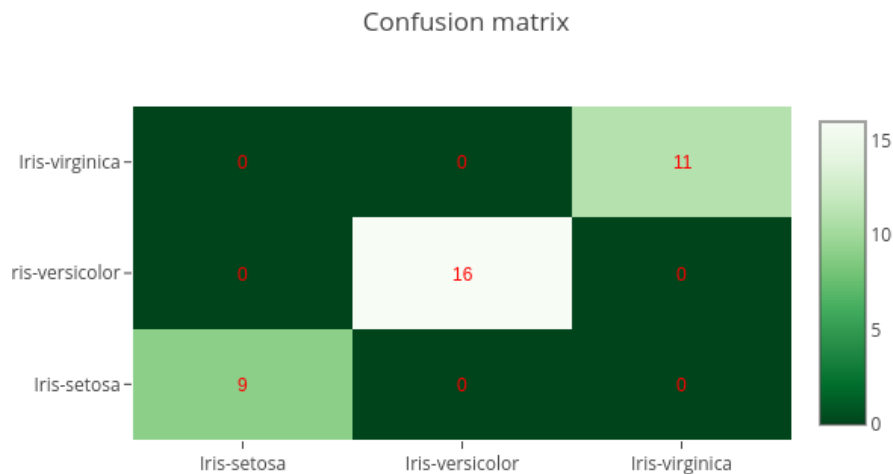


Figura 5.4: Matriz de confusión redes neuronales con datos numéricos. 25 % ejemplos de validación

ejemplos de muestras en la parte de testeo para comprobar si realmente clasifica correctamente la muestra introducida.

Para realizar las pruebas se escogen tres muestras aleatorias del conjunto de datos, una de cada clase:

- Se escoge una muestra de la clase iris setosa, se introduce en la parte de testeo y como resultado se ve que clase se le asigna. En la figura 5.5 se puede observar el porcentaje de que la muestra evaluada pertenezca a una clase u otra, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 99.5 % pertenece a la clase iris setosa.
- Se escoge una muestra de la clase iris virginica, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 96.55 % pertenece a la clase iris virginica como se puede observar en la figura 5.6.
- Se escoge una muestra de la clase iris versicolor, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 96.71 % pertenece a la clase iris versicolor como se puede observar en la figura 5.7.

3. Probar


Introduce términos nuevos y comprueba si se clasifican correctamente

Longitud sépalo en cm
4.6

Ancho sépalo en cm
3.2

Longitud pétalo en cm
1.4

Ancho pétalo en cm
0.2

Comprobar 

Estoy prácticamente segura de que pertenece a la clase Iris-setosa

- Iris-setosa (99.50 %)
- Iris-versicolor (0.50 %)
- Iris-virginica (0.00 %)

Figura 5.5: Muestra de test de la clase iris setosa con redes neuronales

3. Probar


Introduce términos nuevos y comprueba si se clasifican correctamente

Longitud sépalo en cm
6.2

Ancho sépalo en cm
3.4

Longitud pétalo en cm
5.4

Ancho pétalo en cm
2.3

Comprobar 

Estoy prácticamente segura de que pertenece a la clase Iris-virginica

- Iris-virginica (96.55 %)
- Iris-versicolor (3.45 %)
- Iris-setosa (0.00 %)

Figura 5.6: Muestra de test de la clase iris virginica con redes neuronales


3. Probar
Introduce términos nuevos y comprueba si se clasifican correctamente

Longitud sépalo en cm
5.7

Ancho sépalo en cm
2.6

Longitud pétalo en cm
3.5

Ancho pétalo en cm
1

Comprobar 

Estoy prácticamente segura de que pertenece a la clase Iris-versicolor

- Iris-versicolor (96.71 %)
- Iris-setosa (3.05 %)
- Iris-virginica (0.25 %)

Figura 5.7: Muestra de test de la clase iris versicolor con redes neuronales

Se realizan además varias comprobaciones más con muestras de las tres clases y las clasifica correctamente. Pero en la figura 5.6 que corresponde con la clase iris virginica se puede ver que un pequeño porcentaje lo asigna a la clase iris setosa y en la figura 5.7 que corresponde con la clase iris setosa se puede ver que un pequeño porcentaje lo asigna a la clase iris virginica, esto sucede porque estas dos clases tienen características similares y en algún caso puede tener un error de clasificación, esto se puede comprobar viendo la matriz de confusión de la figura 5.8 en la que el porcentaje de ejemplos de validación se aumenta al 75 % para comprobar la validez con la mayoría de muestras. Aun así el porcentaje de acierto es muy alto, en este caso es del 96.36 % ya que falla en la clasificación de cuatro muestras de un total de 110 muestras. Con esto se podría decir que el modelo funciona correctamente.

5.1.2. KNN

Se elige el clasificador de KNN en el desplegable, se selecciona el número de vecinos que en este caso lo dejamos como viene y se escoge el porcentaje de muestras que se usan como ejemplos de validación que en este caso seleccionamos un 25 %, la configuración se puede observar en la figura 5.9.

Con este algoritmo se puede comprobar la validez del modelo con la matriz de confusión. En la figura 5.10 se puede ver un ejemplo de la matriz de confusión que en este caso tiene un

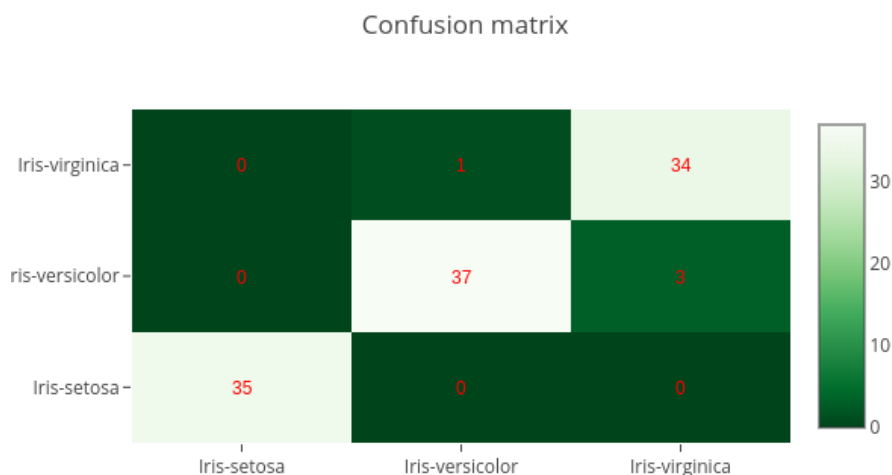


Figura 5.8: Matriz de confusión redes neuronales con datos numéricos. 75 % ejemplos de validación

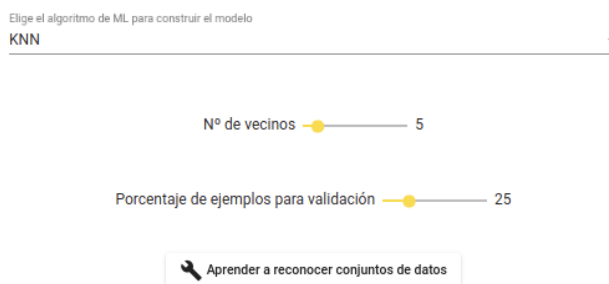


Figura 5.9: Configuración KNN

94.44 % de acierto ya que falla en la clasificación de dos muestras de un total de 36 muestras. Se comprueba la matriz de confusión varias veces para que escoja diferentes muestras y el porcentaje de acierto es alto, suele ser superior al 90 % por lo que se puede considerar un porcentaje de acierto óptimo. El uso de este conjunto de datos no es común para el algoritmo KNN pero con los resultados de la matriz de confusión se puede decir que su uso es óptimo y que las muestras se están recogiendo de la forma correcta.

Una vez se ha comprobado la validez del modelo, se realizan pruebas en la aplicación con ejemplos de muestras en la parte de testeo para comprobar si realmente clasifica correctamente la muestra introducida, al igual que se hizo con redes neuronales.

Para realizar las pruebas se escogen tres muestras aleatorias del conjunto de datos, una de

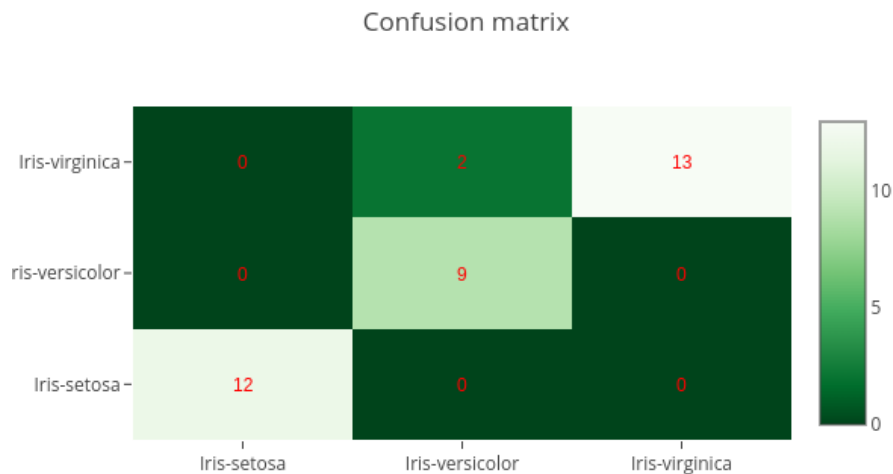


Figura 5.10: Matriz de confusión KNN con datos numéricos. 25 % ejemplos de validación

cada clase:

- Se escoge una muestra de la clase iris setosa, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 100 % pertenece a la clase iris setosa como se puede observar en la figura 5.11.
- Se escoge una muestra de la clase iris virginica, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 80 % pertenece a la clase iris virginica como se puede observar en la figura 5.12.
- Se escoge una muestra de la clase iris versicolor, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 80 % pertenece a la clase iris versicolor como se puede observar en la figura 5.13.

En este caso los porcentajes son múltiplos de 20 debido a que el parámetro K (número de vecinos) tiene el valor 5, por lo que la muestra tendrá 1,2,3,4 o 5 vecinos de una clase que se corresponden con el 20, 40, 60, 80 o 100 por ciento respectivamente. Este conjunto de datos no está pensado para su uso con este algoritmo porque como las clases iris virginica e iris versicolor tienen características similares se colocan cerca las dos clases en la distribución que hace KNN de las muestras y al escoger los vecinos más cercanos puede haber un fallo de asignación de la

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Longitud sépalo en cm

4.5

Ancho sépalo en cm

2.3

Longitud pétalo en cm

1.3

Ancho pétalo en cm

0.3

Comprobar



Estoy prácticamente segura de que pertenece a la clase Iris-setosa

- Iris-setosa (100.00 %)
- Iris-versicolor (0.00 %)
- Iris-virginica (0.00 %)

Figura 5.11: Muestra de test de la clase iris setosa con KNN

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Longitud sépalo en cm

6

Ancho sépalo en cm

3

Longitud pétalo en cm

4.8

Ancho pétalo en cm

1.5

Comprobar



Probablemente pertenezca a la clase Iris-versicolor

- Iris-versicolor (80.00 %)
- Iris-virginica (20.00 %)
- Iris-setosa (0.00 %)

Figura 5.12: Muestra de test de la clase iris virginica con KNN


3. Probar
Introduce términos nuevos y comprueba si se clasifican correctamente

Longitud sépalo en cm
6.2

Ancho sépalo en cm
2.2

Longitud pétalo en cm
4.5

Ancho pétalo en cm
1.5



Probablemente pertenezca a la clase Iris-versicolor

- Iris-versicolor (80.00 %)
- Iris-virginica (20.00 %)
- Iris-setosa (0.00 %)

Figura 5.13: Muestra de test de la clase iris versicolor con KNN

clase que es realmente. Aun así se realizan varias comprobaciones más con muestras de las tres clases y las clasifica correctamente exceptuando algún fallo con estas dos clases. Esto se puede comprobar en la matriz de confusión de la figura 5.14 en la que el porcentaje de ejemplos de validación se aumenta al 75 % para comprobar la validez con la mayoría de muestras. Aun así el porcentaje de acierto es muy alto, en este caso es del 94.5 % ya que falla en la clasificación de seis muestras de un total de 110 muestras. Con esto se podría decir que el modelo funciona correctamente.

Con estos resultados obtenidos se puede decir que el modelo se crea correctamente con conjuntos de datos numéricos y que el conjunto de datos flor iris es válido para los dos tipos de algoritmos.

5.2. Conjunto de datos numéricos y categóricos

Para comprobar la clasificación de un conjunto de datos numéricos y categóricos he utilizado el conjunto de datos de evaluación del coche que se encuentra en el archivo `car.json` de la carpeta `datasets`. Este dataset está compuesto de dos clases (`good` y `vgood`). La clase `good`

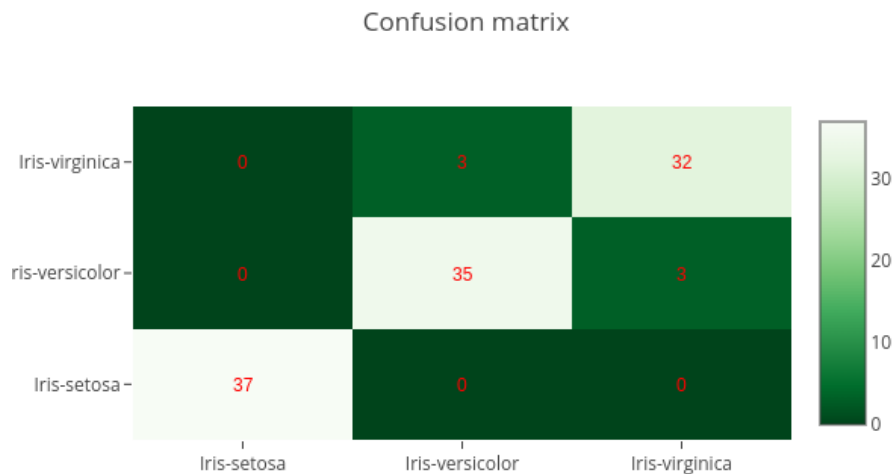


Figura 5.14: Matriz de confusión KNN con datos numéricos. 75 % ejemplos de validación

tiene 69 muestras y la clase vgood tiene 65 muestras. Los atributos de las muestras son precio de compra, precio de mantenimiento, número de puertas, capacidad de personas, tamaño del maletero y seguridad estimada. En la figura 5.15 se pueden observar las muestras de las distintas clases.

Al igual que con el conjunto de datos numéricos se prueba la validez con el algoritmo de redes neuronales y con el algoritmo de KNN.

5.2.1. Redes neuronales

La configuración es igual que en el caso de datos numéricos, figura 5.2. Hay dos formas de comprobar la validez del modelo con el algoritmo de redes neuronales:

- Con la gráfica de evolución de aprendizaje. En la figura 5.3 se puede ver que cuantas más épocas pasan más aciertos hay y al final de la gráfica el acierto es casi 1 como pasa con el conjunto de datos numéricos. Este resultado indica que el aprendizaje se está realizando correctamente.
- Con la matriz de confusión. En la figura 5.4 se puede ver un ejemplo de la matriz de confusión que en este caso tiene un 97 % de acierto, ya que falla en la clasificación de una muestra de un total de 33 muestras. Se comprueba la matriz de confusión varias veces

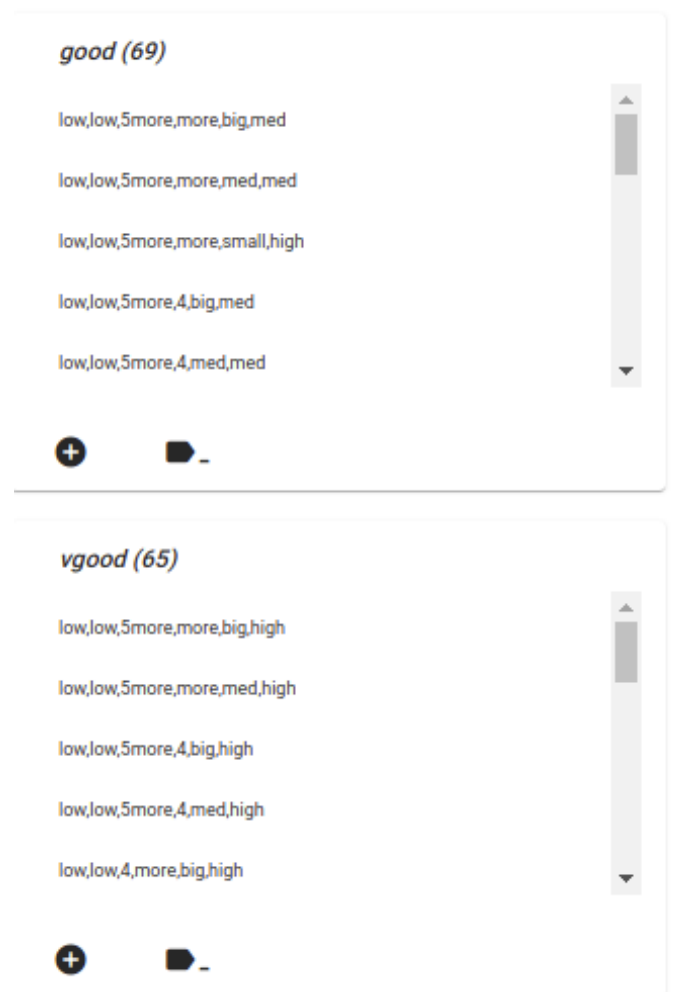


Figura 5.15: Muestras con datos numéricos para el entrenamiento

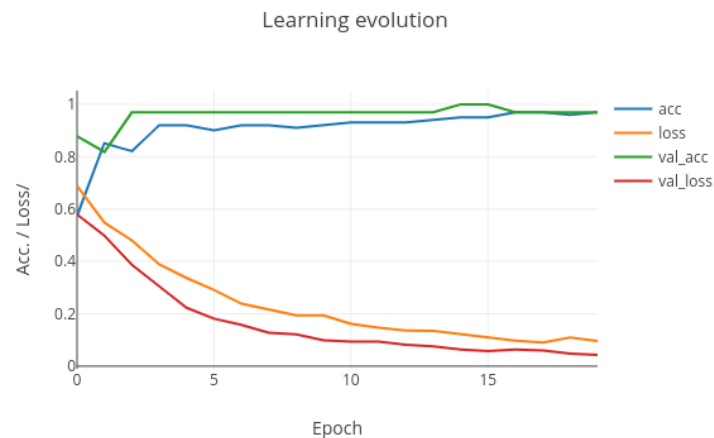


Figura 5.16: Gráfica de evolución de aprendizaje redes neuronales con datos numéricos y categóricos

y el fallo es muy bajo, el acierto suele ser superior al 90 % como pasa con el conjunto de datos numéricos, por lo que se puede considerar un porcentaje de acierto óptimo.

Con los datos obtenidos en la gráfica de evolución de aprendizaje y en la matriz de confusión se puede decir que el modelo se crea correctamente, esto implica que las muestras se recogen de forma correcta y que la extracción de características es correcta. Además el algoritmo de redes neuronales es óptimo para el reconocimiento de este conjunto de datos.

Una vez se ha comprobado la validez del modelo, se realizan pruebas en la aplicación para comprobar si realmente clasifica correctamente la muestra introducida. Para ello se escogen dos muestras aleatorias del conjunto de datos, una de cada clase:

- Se escoge una muestra de la clase good, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 86.54 % pertenece a la clase good como se puede observar en la figura 5.18.
- Se escoge una muestra de la clase vgood, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 95.13 % pertenece a la clase vgood como se puede observar en la figura 5.19.

Además se realizan varias comprobaciones más con muestras de las dos clases y las clasifica correctamente. En algún caso puede fallar ya que las características son similares entre las dos

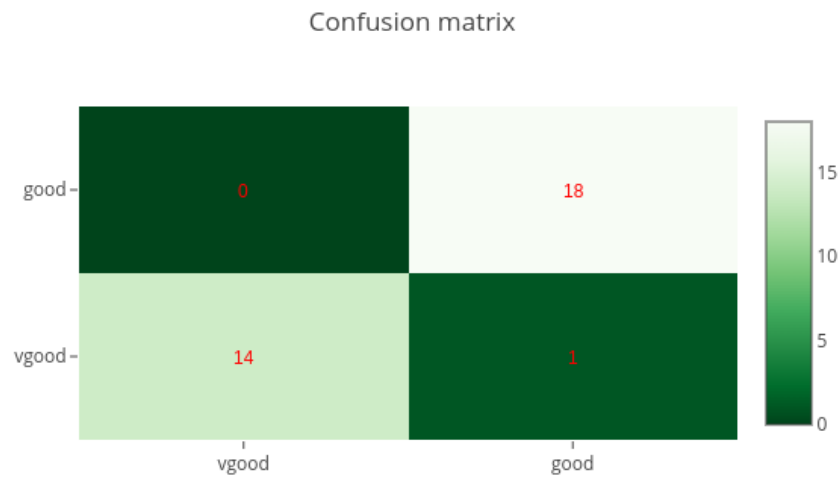


Figura 5.17: Matriz de confusión redes neuronales con datos numéricos y categóricos. 25 % ejemplos de validación

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Precio de compra
med


Precio del mantenimiento
low

Número de puertas
2

Capacidad de personas
4

Tamaño del maletero
med

Seguridad estimada
high

Comprobar 

Probablemente pertenezca a la clase good

- good (86.54 %)
- vgood (13.46 %)

Figura 5.18: Muestra de test de la clase good con redes neuronales

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Precio de compra

low

Precio del mantenimiento

high

Número de puertas

4

Capacidad de personas

4

Tamaño del maletero

med

Seguridad estimada

high

Comprobar



Estoy prácticamente segura de que pertenece a la clase vgood

- vgood (95.16 %)
- good (4.84 %)

Figura 5.19: Muestra de test de la clase vgood con redes neuronales

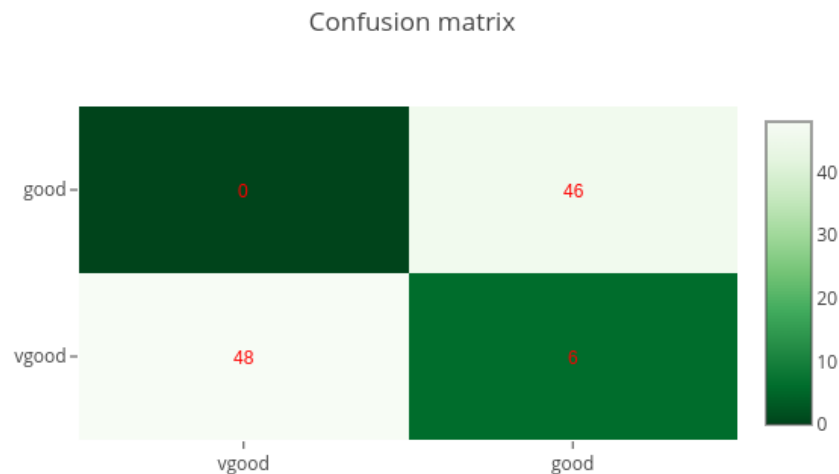


Figura 5.20: Matriz de confusión redes neuronales con datos numéricos y categóricos. 75 % ejemplos de validación

clases pero el porcentaje de acierto es alto como se puede comprobar en la matriz de confusión de la figura 5.20 en la que el porcentaje de ejemplos de validación se aumenta al 75 % para comprobar la validez con la mayoría de muestras, en este caso el porcentaje de acierto es del 91 %. Con estos resultados se podría decir que el modelo funciona correctamente.

5.2.2. KNN

La configuración es igual que en el caso de datos numéricos, figura 5.9. Con este algoritmo se puede comprobar la validez del modelo con la matriz de confusión. En la figura 5.21 se puede ver un ejemplo de la matriz de confusión que en este caso tiene un 93.93 % de acierto ya que falla en la clasificación de dos muestras de un total de 33 muestras. Se comprueba la matriz de confusión varias veces para que escoja diferentes muestras y el porcentaje de acierto es alto, suele ser superior al 80 % por lo que se puede considerar un porcentaje de acierto óptimo. Se podría decir que el modelo se crea correctamente.

Una vez se ha comprobado la validez del modelo, se realizan pruebas en la aplicación con ejemplos de muestras en la parte de testeo para comprobar si realmente clasifica correctamente la muestra introducida, al igual que se hizo con redes neuronales.

Para realizar las pruebas se escogen dos muestras aleatorias del conjunto de datos, una de

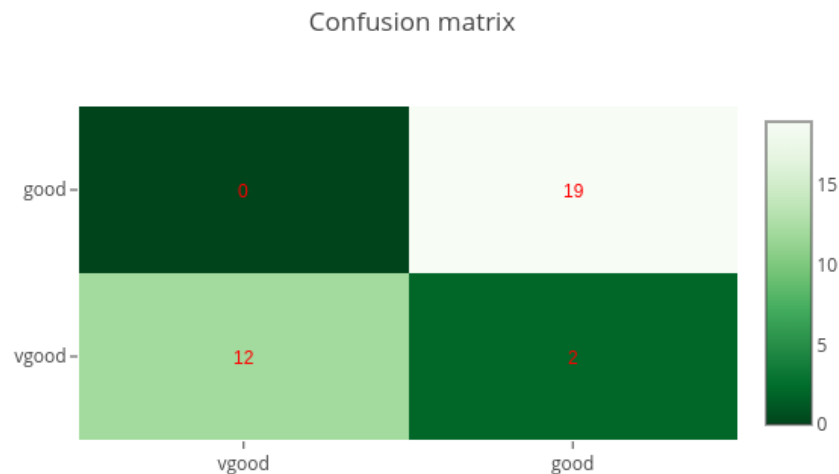


Figura 5.21: Matriz de confusión KNN con datos numéricos y categóricos. 25 % ejemplos de validación

cada clase:

- Se escoge una muestra de la clase good, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 100 % pertenece a la clase good como se puede observar en la figura 5.22.
- Se escoge una muestra de la clase vgood, en este caso asigna correctamente la muestra a la clase que le corresponde ya que indica que al 100 % pertenece a la clase iris virginica como se puede observar en la figura 5.12.

Además se realizan varias comprobaciones más con muestras de las dos clases y las clasifica correctamente exceptuando algún fallo. Esto se puede comprobar en la matriz de confusión de la figura 5.24 en la que el porcentaje de ejemplos de validación se aumenta al 75 % para comprobar la validez con la mayoría de muestras. Aun así el porcentaje de acierto es aceptable, en este caso es del 69 % ya que falla en la clasificación de 31 muestras de un total de 100 muestras. Con esto se podría decir que el modelo funciona correctamente pero el uso de este algoritmo no es el más adecuado.

Con estos resultados obtenidos se puede decir que el modelo se crea correctamente con conjuntos de datos numéricos y categóricos y que el conjunto de datos de evaluación de co-

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Precio de compra
low


Precio del mantenimiento
med

Número de puertas
5more

Capacidad de personas
4

Tamaño del maletero
big

Seguridad estimada
med

Comprobar 

Estoy prácticamente segura de que pertenece a la clase good

- good (100.00 %)
- vgood (0.00 %)

Figura 5.22: Muestra de test de la clase good con KNN

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Precio de compra
med

Precio del mantenimiento
med

Número de puertas
3

Capacidad de personas
more

Tamaño del maletero
med

Seguridad estimada
high

Comprobar 

Estoy prácticamente segura de que pertenece a la clase vgood

- **vgood (100.00 %)**
- good (0.00 %)

Figura 5.23: Muestra de test de la clase vgood con KNN

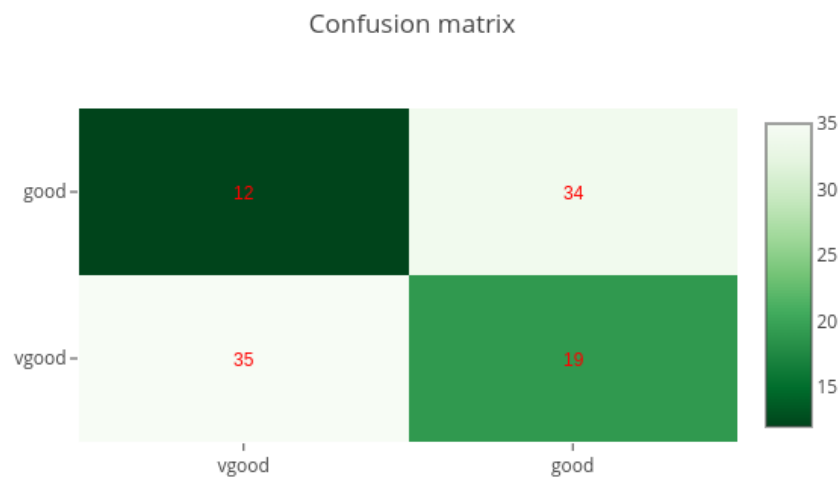


Figura 5.24: Matriz de confusión KNN con datos numéricos y categóricos. 75 % ejemplos de validación

che es válido para los dos tipos de algoritmos pero es mejor usarlo con el algoritmo de redes neuronales.

Como resultado final se puede decir que la creación de modelos de conjuntos de datos es válida , es decir, las muestras se recogen correctamente y la extracción de características también es correcta ya sea con datos numéricos o categóricos, y que dependiendo de como sean los conjuntos de datos funcionará mejor un algoritmo u otro.

Capítulo 6

Conclusiones

Considero que el resultado del trabajo ha sido bueno y creo que la implementación de esta funcionalidad era necesario para LearningML ya que los conjuntos de datos están a la orden del día y los alumnos deben conocer que son y la utilidad que pueden tener, en esto coincide Juan David Rodríguez y quedó muy satisfecho con el trabajo realizado.

6.1. Consecución de objetivos

Haciendo referencia al capítulo 2, el objetivo principal del TFG era implementar la nueva funcionalidad para reconocer conjuntos de datos. Este objetivo se ha cumplido como se esperaba.

En cuanto a los objetivos específicos, también se han cumplido. La arquitectura de LearningML la he entendido y he seguido la dinámica de lo ya creado. Se ha modificado la interfaz correctamente y la creación de conjuntos de datos se realiza correctamente y de una forma sencilla e intuitiva. Por último la extracción de características se realiza correctamente y los resultados obtenidos son buenos.

6.2. Aplicación de lo aprendido

Para la realización del proyecto, se han usado los conocimientos de varias asignaturas cursadas durante el grado como son las siguientes:

- **Construcción de Servicios y Aplicaciones Audiovisuales en Internet (CSAAI).** Esta

asignatura ha sido de gran ayuda para tener conocimientos sobre el desarrollo web con lenguajes como JavaScript, HTML5 y CSS.

- **Laboratorio de Tecnologías Audiovisuales en la Web (LTAW).** Esta asignatura ha sido de gran ayuda para tener conocimientos sobre el desarrollo de aplicaciones web ya que aprendes a usar Node.js y plantillas.
- **Informática II.** Es una asignatura en la que aprendes un lenguaje complicado y altamente tipado como Ada, en el que aprendes a como llevar a cabo un programa de forma eficiente y limpia.
- **Tratamiento Digital de la Imagen y Tratamiento Digital del Sonido.** Esta asignatura ha sido de gran ayuda para comprender como funcionan los algoritmos utilizados en LearningML.
- **Protocolos para la Transmisión de Audio y Vídeo por Internet (PTAVI).** Esta asignatura es la que realmente hizo que cogiera el gusto a programar y en la que entendí como funcionaba la programación.

6.3. Lecciones aprendidas

Con la implementación de la nueva funcionalidad he ampliado y mejorado los conocimientos adquiridos durante la carrera en el desarrollo de aplicaciones web además de aprender otros nuevos como son:

1. Angular. Al principio tuve que dedicar bastante tiempo a entender su funcionamiento pero me ha parecido muy útil ya que facilita mucho el desarrollo web y de una forma muy organizada sobre todo para aplicaciones grandes debido a su arquitectura de componentes, servicios y módulos permite separar las funcionalidades y que se repita el código. Gracias a este framework añadir una nueva funcionalidad ha sido más sencillo y me alegro de haber aprendido a usar este framework porque tiene un gran potencial.
2. Programación asíncrona. Antes de empezar el proyecto tenía una noción básica de que era la programación asíncrona pero no sabía para que podría usarse ni como implementarla.

Una vez realizado el proyecto creo que es muy útil e indispensable en ciertas acciones de una aplicación web porque que no se bloquee es algo que valora mucho el usuario.

3. TypeScript. No me resultó difícil comprender este lenguaje debido a que está basado en JavaScript pero con cambios que son sencillos de entender. Es un lenguaje que me parece más útil que JavaScript ya que tiene sus cosas buenas y fortalece sus debilidades.

6.4. Trabajos futuros

Las mejoras que se pueden añadir a la aplicación web LearningML pueden ser:

- Añadir gráficos que representen la evolución de generación del modelo (entrenamiento), como los de the teachable machine.
- Desarrollar una versión offline del editor de LML y de Scratch.
- Añadir una nueva funcionalidad para el reconocimiento de la postura de la mano o de poses.
- Incluir otros algoritmos de aprendizaje como puede ser las máquinas de vectores de soporte.

Bibliografía

[1] Cantidad de datos.

<https://es.statista.com/grafico/26031/volumen-estimado-de-datos-digitales-creados-o-replicados-en-todo-el-mundo/>.

[2] Características mvc.

<https://seopromarketing.online/que-es-mvc-ventajas.>

[3] Curso de angular.

https://youtu.be/fXpMiweCC_o.

[4] Pagina de json.

<http://www.json.org/json-es.html>.

[5] Pagina de node.js.

<https://nodejs.org/es>.

[6] Pagina de tensorflow.

<https://www.tensorflow.org>.

[7] Pagina de typescript.

<https://www.typescriptlang.org>.

[8] Pagina sobre html.

https://www.w3schools.com/html/html_intro.asp.

[9] Pagina sobre html.

<https://html.spec.whatwg.org/multipage>.

[10] Página de angular.

<https://web.learningml.org>.

[11] Página de angular.

<https://angular.io>.

[12] Página de angular cli.

<https://cli.angular.io>.

[13] J. D. Gauchat. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo, 2012.

[14] L. Rouhiainen. Inteligencia artificial. *Madrid: Alienta Editorial*, 2018.