ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA BIOMÉDICA

TRABAJO FIN DE GRADO

EVALUACIÓN DE LAS PRESTACIONES DE DIFERENTES TÉCNICAS DE FUSIÓN DE CLASIFICADORES

Autor: Ismael Lucas Cobo

Tutor: Cristina Soguero Ruíz

Co-tutor: Nazila Esmaeili

Curso académico 2022/2023

A mis padres, amigos, hermanos y a toda mi familia, solo puedo decir gracias por ser quienes sois y por el apoyo brindado y saludos a mis abuelos, que en paz descansen, por confiar en mí.

Agradecimientos

Primero de todo, agradecer a Medics GMBH y en concreto, a mi tutora de prácticas Nazila Esmaeili por ser la primera empresa en la que he estado y en la cual desarrollé este proyecto desde cero, gracias a su apoyo y guía, además de ser los proveedores de una de las bases de datos utilizadas en el proyecto.

Gracias a la Universidad Rey Juan Carlos por acogerme y a todos los profesores que me han enseñado y ayudado en el camino a convertirme en un ingeniero biomédico.

Este documento ha sido realizado gracias al apoyo de Cristina Soguero Ruiz. Muchas gracias por prestarme tu tiempo para poder desarrollar mi trabajo correctamente.

Muchas gracias a todas las personas que he conocido durante este viaje en mi vida, que es la universidad. Gracias compañeros y amigos por acogerme a mí y ser parte de este viaje y ayudarme y reír conmigo durante este arduo y tedioso camino. En especial, agradecer a José, Andrés y Carlos por siempre estar ahí y divertiros, jugar y trabajar conmigo para atravesar este viaje, siempre seréis parte de mi vida y no creo que pueda olvidar todos los momentos que hemos pasado.

Mi mayor gracias es para mis padres, que siempre habéis confiado en mí y en mi potencial para estudiar y aprender, y siempre habéis estado ahí apoyándome en lo que sea que necesitara y creyendo en que lo podía hacer. Sois los mejores padres que nadie en el mundo podría querer o tener.

Y por último, gracias a mis amigos de toda la vida Javier, Roberto y Aaron por siempre estar conmigo y apoyarmo en todo.

Resumen

El aprendizaje automático es un subcampo de las ciencias de computación y rama de la Inteligencia Artificial. Se usa para llevar a cabo tareas de clasificación o regresión de sucesos u objetos, para crear tanto herramientas de apoyo a la toma de decisión. Lleva desarrollándose desde los años 50, pero en la actualidad, gracias al desarrollo de tecnologías con mayor capacidad de cálculo, velocidad y almacenamiento, se han podido desarrollar programas con mayor complejidad, creando algoritmos más precisos y que pueden procesar una mayor cantidad de datos y datos más complejos, como son las imágenes. Cada clasificador tiene sus ventajas y desventajas propias, y al final, se tiene que decidir que clasificador se deberá utilizar en la tarea a desarrollar. Por esto se pierden algunas ventajas que aportan los diferentes algoritmos. Para intentar aprovechar las diferentes ventajas de los diferentes clasificadores, se han desarrollado los métodos de fusión de clasificadores, los cuales unen en un solo algoritmo diferentes clasificadores.

Este proyecto es una introducción a los sistemas de fusión de clasificación, en el cual se hará una presentación y explicación de que son los sistemas de fusión de clasificación y su utilidad. Se desarrollarán dos métodos, el primero será el método por votación, proceso por el cual la salida final del sistema será la elegida por la mayoría de clasificadores. El segundo método será un perceptrón multicapa (siglas MLP en inglés), que es un sistema de clasificación, que tiene forma de red neuronal, es decir, que al llegar los datos de entrada al bloque de datos inicial , la neurona de entrada, y pasarán al siguiente bloque con una modificación de valor, que se determina por el parámetro definido que tiene cada neurona. La salida final del sistema llegará a la neurona de salida, como el dato transformado tras atravesar todas las múltiples capas de bloques de neuronas existentes en el MLP.

Para el desarrollo del proyecto se usarán dos bases de datos diferentes, una primera base de datos de MNIST, librería de Python formada por imágenes de números escritos a mano; y una segunda base con imágenes de CE-NBI de tumores benignos o malignos en la garganta, transformadas a 26 variables numéricas. Se usarán 7 clasificadores: k vecinos más cercanos, árboles de decisión, bosques aleatorios, regresión polinómica, regresión multilínea y regresión logística y máquinas vector soporte.

Una conclusión obtenida es que los métodos de fusión de clasificadores con los que se ha trabajado, aun siendo métodos sencillos, mejoran el rendimiento en la clasificación de los datos del sistema y son una rama muy amplia a investigar para mejorar las prestaciones del aprendizaje automático.

Índice general

Agradecimientos

Resumen

Índice de figuras V

Índice de cuadros VII

Lista de acrónimos y abreviaturas X

1. Introducción y objetivos 1

1.1. Contexto y motivación 1

1.2. Objetivos 7

1.3. Metodología 8

1.4. Estructura de la memoria 9

2. Conceptos previos 11

2.1. Conceptos de aprendizaje automático 11

2.1.1. Aprendizaje supervisado 12

2.1.2. Aprendizaje no supervisado 13

2.2. Sistemas de fusión de clasficadores 13

2.2.1. Métodos de fusión de etiquetas 14

2.2.2. Métodos de ranking 15
2.2.3. Métodos de salidas blandas 16

3. Métodos **17**
3.1. Etapas de Diseño 17
3.1.1. Normalización de las Características 17
3.1.2. Selección de características con chi cuadrado 18
3.2. Métodos de Aprendizaje Automático 19
3.2.1. Regresión Polinómica 19
3.2.2. Regresión Logística 19
3.2.3. K vecinos más cercanos 20
3.2.4. Árbol de Clasificación 21
3.2.5. Random Forest 22
3.2.6. Máquinas de Vector Soporte 23
3.3. Métodos de fusión de clasificadores 24
3.3.1. Métodos de votación 25
3.3.2. Perceptrón Multicapa 25
3.4. Figuras de mérito 27
3.4.1. Matriz de confusión 27

4. Base de datos y análisis descriptivo **29**
4.1. Base de datos de MNIST 29
4.2. Base de datos de extracción de características 30
4.2.1. Indicadores basados en dirección 31
4.2.2. Indicadores de curvatura 32
4.2.3. Características 33

5. Experimentos y resultados **41**
5.1. Evaluación 41

5.2. Google colab 43
5.3. Resultados de la base de datos de MNIST 43
5.3.1. Discusión de la base de datos de MNIST 45
5.4. Resultados de la base de datos sobre extracción de características. 46
5.4.1. Discusión base de datos extracción de características 48

6. Conclusiones y líneas futuras **51**
6.1. Conclusiones 51
6.2. Líneas futuras 52

A. Código **55**

Bibliografía **57**

Índice de figuras

1.1. Gráfica de rendimiento de los clasificadores [1]. 5
1.2. Sistema de fusión en serie (a) y en paralelo (b). 7

3.1. Distribución normal [2]. 18
3.2. Generalización [3]. 21
3.3. Árbol de decisión [4]. 22
3.4. Truco de *kernel* [5]. 24
3.5. Esquema del método de fusión de clasificadores. 25
3.6. Esquema de la neurona artificial [6]. 26
3.7. Esquema de MLP. 27

4.1. Muestra de las imágenes contenidas en la base de dados *mnist*. 29
4.2. Distribución de clases *mnist* [7]. 30
4.3. Gradiente de una imagen ejemplo [8]. 31
4.4. Construcción de los indicadores ANG y DIS. 33
4.5. Indicadores HGD, RIA, ANG, DIS y CUR de la imagen [9]. 33
4.6. (a) imagen CE-NBI, (b) representación 3D de la imagen, (c) perfil de la etapa. [10]. 36
4.7. Esquema de extracción de Potencia y energía de la imagen. 37
4.8. Gráficas de cada atributo de la base de datos. 39

Índice de cuadros

3.1. Matriz de confusión para clasificación binaria. 28

5.1. Matriz de confusión KNN. 44

5.2. Matriz de confusión RF. 44

5.3. Matriz de confusión SVM. 44

5.4. Matriz de confusión método de votación. 45

5.5. Tabla de *precission*, *recall* y f1 score de los clasificadores con los datos raw. . . 46

5.6. Tabla de *precission*, *recall* y f1 score de los clasificadores con los datos filtrados con chi2. 46

5.7. Tabla de *precission*, *recall* y f1 score de los clasificadores con los datos normalizados. 47

5.8. Tabla de *precission*, *recall* y f1 score del método de votación para los datos *raw*. 47

5.9. Tabla de *precission*, *recall* y f1 score del método de votación para los datos normalizados. 47

5.10. Tabla de *precission*, *recall* y f1 score del MLP para los datos *raw*. 48

5.11. Tabla de *precission*, *recall* y f1 score del MLP para los datos normalizados. . . 48

Lista de acrónimos y abreviaturas

ANG Indicador de ángulo

ANN Redes neuronales artificiales

CE-NBI Endoscopia de contacto con imagen de banda estrecha

chi2 Método de chi cuadrado

CUR Indicador de curvatura

DCS Selección de clasificadores dinámicos

DIS Indicador de distancia

DT *Decission Tree*

ECV Enfermedad cardiovascular

Fa Fuerza de resistencia al aire

Fg Fuerza de gravedad

Fr Fuerza de resistencia al giro

GMM *Gaussian Mixture Model*

HGD Dirección del gradiente del histograma

IA Inteligencia Artificial

KNN *K-Nearest Neighbors*

LR Regresión logística

MAE Mean Absolute Error

MLR Regresión Lineal Múltiple

ML Machine Learning

PR Regresión Polinómica

RF *Random Forest*

RIA Media de rotación de la imagen

RMSE *Root Mean Square Error*

SVM *Support Vector Machines*

TFG Trabajo Fin de Grado

Capítulo 1

Introducción y objetivos

Este capítulo constituye un preámbulo al problema de los métodos clásicos de ML, incluyendo sus ventajas y desventajas, además de presentar los métodos de fusión como solución a las desventajas. A continuación, se definiran los objetivos del presente Trabajo de Fin de Grado (TFG), la correspondiente metodología seguida y una breve estructura de la memoria.

1.1. Contexto y motivación

El aprendizaje hace referencia a toda situación en la que el aprendiz incrementa su conocimiento o sus habilidades a partir de la introducción en su entorno y percepción de nuevos estímulos que nunca antes había percibido, siendo el aprendizaje una herramientas para responder de manera adecuada a este nuevo estímulo y almacenar la experiencia para reaccionar de manera correcta y corregida la próxima vez que una situación similar se produzca frente al aprendiz. El método para llevar a cabo el proceso de aprendizaje es mediante inferencias a la información, es decir, usar el razonamiento lógico y la evidencia para construir una representación apropiada del nuevo estímulo o del proceso [11]. Por otra parte, la automatización hace referencia a la creación de aplicaciones de tecnología con el mínimo de interacción humana. El aprendizaje automático es por tanto el proceso por el que una máquina tiene que funcionar como un organismo autónomo, siendo capaz de realizar una tarea concreta repetidamente, pero haciendo un aprendizaje constante para poder llevar a cabo la tarea de manera correcta frente a cualquier tipo de alteración en las condiciones o datos de entrada al sistema, pudiendo así hacer frente a problemas dinámicos, es decir, problemas en los que las condiciones de entrada son variables, no constantes.

El aprendizaje automático es un componente de la ciencia de datos y una rama de la inteligencia artificial en la que se crean algoritmos de clasificación y regresión [12]. Los algoritmos de aprendizaje automático usan métodos estadísticos para estimar la salida prevista de una entrada de datos para un problema dinámico. En aprendizaje automático, la máquina debe funcionar como un organismo autónomo, es decir, que debe tener la capacidad de realizar una misma tarea de varias maneras y adaptándose a las circunstancias. La máquina debe generar un sistema de toma de decisiones que debe poder obtener la resolución frente a un problema y adaptarse a condiciones variables de entrada. Se puede entender al aprendizaje automático, o *machine learning* (ML) en inglés, como el proceso de aprendizaje que lleva a cabo una máquina para, a partir de unos datos de entrada, obtener la salida del sistema correcta o mejor aproximada [13].

El origen del ML se le suele atribuir al psicólogo Frank Rosenblatt, de la universidad de Cornell [14]. Desarrolló un grupo de máquinas para que reconocieran las letras del alfabeto, creando un sistema de decisión, guiándose por los recientes trabajos sobre como funcionaba el sistema nervioso humano, creando el perceptrón, una máquina que a partir de señales analógicas y discretas incluidas en un sistema parametrizado, generaba una salida. [15] Este trabajo fue el prototipo de las redes neuronales modernas. Su trabajo se desarrolló entre 1957 y 1960 y fue el primero en crear un modelo matemático para el perceptrón. Así fue como en los años 60, apareció el ML y se inició una batalla de teorías entre 2 grupos de investigadores para el desarrollo y la evolución del ML:

1. Acercamiento determinista: teoría basada en la división de los datos en 2 grupos, A y B, separados en 2 rangos numéricos cerrados. Bajo esta rama, cada nueva entrada al sistema solo podrá pertenecer al grupo A o al grupo B, dependiendo de su valor. Los grupos son vectores completamente separados y se pueden visualizar en el hiperplano. Uno de los trabajos propuestos en esta rama es el trabajo de Vapnik y Chervonenkis en 1964, donde para elegir el vector correspondiente al hiperplano que divide los grupos, se creaba un plano adicional de soporte para calcular la convergencia. Este trabajo se usó como referencia para desarrollar las máquinas de vector soporte (SVM) [16].

2. Acercamiento estadístico: en esta teoría, se definían modelos matemáticos basados en la probabilidad de ocurrencia, no asegurando que un nuevo dato sea de un grupo u otro, sino que tiene una probabilidad de pertenecer a cada grupo, pero siempre con la inclusión de un grado de incertidumbre. En esta área destaca el sistema de minimización de riesgo desarrollado por Yakov Tsypkin, basado en el cálculo de la media de la función de coste, que es la diferencia entre el valor determinado y el valor real; y la búsqueda del mínimo a través del gradiente, vector calculado como la tasa de variación media de una función [17].

Tras el desarrollo de algunos trabajos, en los años 70 hubo una crisis en el desarrollo de nuevas teorías en ML, pues se empezó a desarrollar esquemas de representación para asegurar la adquisición de nuevos conocimientos tras el proceso de aprendizaje. Así se desarrollaron jerarquías para organizar los conceptos, que es la agrupación conceptual, es decir, clasificar un conjunto de observaciones y caracterizar las clases obtenidas a partir de su concepto, para identificar los diferentes grupos en unos datos [11]. El trabajo más destacado es el de P. H. Winston sobre el aprendizaje basado en similitudes, cuyo objetivo era que la máquina aprendiera la descripción de un concepto tras recibir varias entradas de ese tipo [18].

Durante los años 80 y 90 se siguieron desarrollando trabajos en las diferentes áreas de interés, como el desarrollo de redes neuronales convolucionales de Kunihiko Fukushima [19] o el desarrollo de una nueva versión de SVM, llamado redes de soporte de vectores, por Cortes & Vapnik.

Pero la época dorada del ML inicia con el siglo XXI, gracias a 3 elementos clave:

- Big data: conjunto de datos extremadamente grandes y complejos, que no se pueden analizar con enfoques simplistas. Los datos no paraban de aumentar en una escala exponencial, necesitando nuevas técnicas del manejo de los datos.
- Reducción del coste computacional y la memoria: la creación de la tecnología *MapReduce* de *Google* y *Hadoop*, permitió procesar fácilmente los datos y la creación y desarrollo de GPUs permitió la evolución del ML, junto con el abaratamiento de los costes de la RAM.
- Desarrollo de algoritmos de *Deep Learning*: todos los conceptos desarrollados anteriormente, se pueden hacer realidad, creando las redes neuronales computacionales y aplicando diferentes algoritmos con una facilidad no disponible anteriormente, impulsando el desarrollo de nuevos sistemas y aplicaciones al mundo real.

La creación de un modelo de ML supone un proceso de 4 fases [10]:

1. Entrada de datos: se obtienen los datos de los ejemplos a introducir para el entrenamiento del modelo. Esta fase consiste en la obtención de un número grande de datos y ejemplos que se usarán para entrenar el modelo. Los datos pueden obtenerse de bases de datos públicas ya creadas o con la creación de una base de datos nueva en la que se integren nuevos ejemplos constantemente para crear una base de datos propia y dinámica.
2. Extracción de características: las características describen propiedades para representar de manera correcta a los datos de entrada. La identificación de patrones y relaciones

dentro de una base de datos mejora la creación del modelo de aprendizaje. Esta fase tiene como objetivo mantener en los datos información discriminatoria y eliminar los datos irrelevantes. La construcción de los modelos depende en gran medida de los datos de entrada, por ello este paso es muy importante para construir un modelo verdaderamente útil para clasificar.

3. Construcción del modelo: los datos entran al clasificador vacío y sin parámetros. Para cada clasificador, los datos se introducen o se modifican acorde a las necesidades propias del clasificador. Por ejemplo: los modelos de árboles de decisión dividen los datos en partes homogéneas creando estructuras de árbol y el modelo SVM crea un hiperespacio entre los datos para separar las clases. Se suelen aplicar métodos de validación cruzada sobre los datos, que consiste en la división de los datos en set de entrenamiento y set test, para verificar el correcto entrenamiento del modelo.

4. Elección del modelo: tras entrenar los diferentes modelos, se evalúa la eficiencia de los modelos a partir de métricas como el error cuadrático medio en regresión, o ratios de predicción como la precisión o el F1-score. Tras analizar el rendimiento obtenido por cada modelo, se elige el modelo que mejor se adapte a las necesidades del usuario.

En la actualidad, el ML es una de las herramientas más usadas en todo el mundo y junto al desarrollo de la IA, son elementos clave para el desarrollo futuro de la humanidad. Sus aplicaciones no tienen límite en la vida humana, puede ser usado en cualquier campo existente: economía, construcción, agricultura, salud.

Por ejemplo, para el sector de la salud, tiene impacto en muchas áreas dentro de los hospitales: creación de modelos de predicción patológica a partir de datos del paciente [20]. Gran demanda en el área de oncología para la mejora del 15-25 % de pronósticos de los pacientes y se crean modelos para identificar cardiopatías y sus factores de riesgo. Se crean modelos para el screening y diagnóstico de los pacientes como herramienta de apoyo a la decisión o métodos de agilización del sistema de la salud con la integración de sistemas de salud por datos, que reducen la carga clínica y el coste del sistema sanitario [21]. También se crean modelos precisos de tratamiento exclusivos para cada paciente y se investiga en el uso de ML para descubrir nuevos tratamientos y fármacos para actuales y futuras enfermedades [22]. Con ML se pueden crear modelos del ciclo de vida y modelos de expansión de diferentes enfermedades víricas, bacterianas o fúngicas, determinando modelos matemáticos de tiempo que tarda en propagarse, mortalidad y morbilidad. Se pueden diseñar planes de distribución de vacunas. Dos ejemplos de proyectos de ML en medicina son:

- Modelo de *Deep Learning* en mamografía: trabajo de desarrollo de una red neuronal para la creación de un modelo de predicción de cáncer de mama. Se evaluaron mamografías de una gran variedad de mujeres entre 40 y 80 años y de diferentes etnias. Así, se creó un modelo cuyas predicciones fueron similares a las dadas por el grupo de expertos, pero mucho más rápida, demostrando que el modelo creado era eficaz y agilizaba el proceso de detección de cáncer, siendo un buen modelo de apoyo en el diagnóstico del cáncer de mama [23].

- Técnicas de ML en medicina cardiovascular: Trabajo de desarrollo de clasificadores para obtener el diagnóstico de enfermedades cardiovasculares (ECV) a partir de unos atributos como la edad, sexo, presión arterial, frecuencia cardíaca. Con el desarrollo de 3 clasificadores, árbol de decisión, regresión logística y SVM, estimaban si la probabilidad de tener una ECV era mayor al 50 %. Observando los resultados obtenidos en la figura 1.1, la evaluación del proyecto dio una predicción positiva del padecimiento o no de la enfermedad de los pacientes, demostrando su utilidad para el diagnóstico de ECV [1].

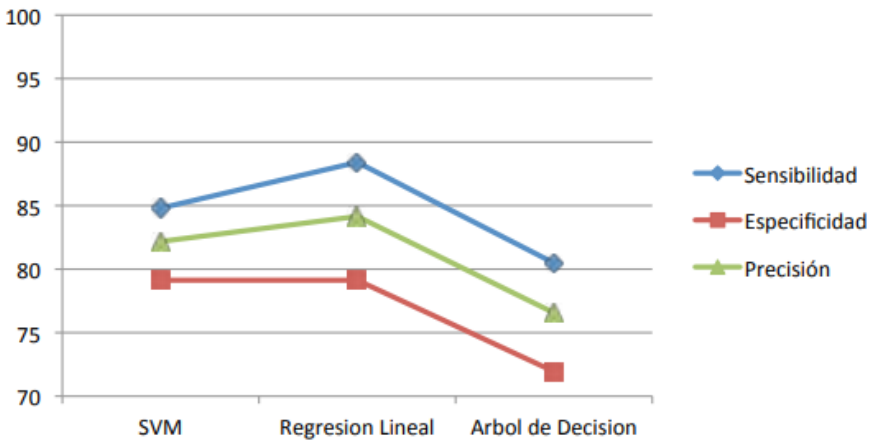

Figura 1.1: Gráfica de rendimiento de los clasificadores [1].

En el mundo de las finanzas, también se usa el ML. Es muy usado para la creación de modelos de predicción de la evolución del mercado de valores, para inversiones a largo y corto plazo. También se puede hacer el cálculo del valor de las viviendas a futuro o modelos de evolución del precio de criptoactivos [24]. Se puede crear un algoritmo de ML para realizar un análisis de la sostenibilidad de un negocio a partir de un análisis de los datos macroeconómicos. Un trabajo ejemplo es el siguiente, en el que se obtuvo una predicción de la inflación y el ratio de cambio de la moneda a través del análisis de la base de datos macroeconómicos de Pakistán desde enero de 1989 hasta diciembre de 2020. El análisis se hizo con 4 clasificadores distintos:

K vecinos más cercanos, regresión polinómica, SVM y redes neuronales artificiales (ANN), evaluados por el MAE, medida que indica la media de error de predicción en valor absoluto, y con RMSE, que es la raíz del error absoluto. Tras el entrenamiento y la fase de test, se obtuvo que el clasificador con menor error a la hora de predecir el ratio de cambio era el SVM y para predecir la inflación, el mejor modelo era el ANN, obteniendo así un modelo de predicción de ratio de cambio e inflación [25].

También se crean algoritmos de recomendación de productos y servicios a usuarios a través de tecnologías como las cookies, por las cuales tu historial de búsqueda, me gusta en redes sociales, tiempo de uso de aplicaciones e historial de voz del micrófono, se pueden hacer anuncios personalizados a usuarios de los artículos en los que tienen o pueden tener un alto interés en adquirir.

En construcción se usa para la evaluación de los posibles materiales a utilizar, dependiendo de las condiciones geológicas, económicas y atmosféricas, además del uso que va a tener el edificio a construir. También se crean modelos de eficiencia energética y económica de plantas, oficinas u hogares a través del análisis de datos de utilización con el tiempo y el correcto uso de las instalaciones.

En general, el ML tiene una gran utilidad para el mundo moderno, pero en todos los casos, siempre se tiene que elegir qué clasificador o método de clasificación o regresión se tiene que utilizar para crear un algoritmo. El problema que presenta elegir solo un clasificador para que lleve a cabo una tarea de clasificación o regresión entre todos los clasificadores probados, es que el conjunto de patrones mal clasificados no se superpone para todos los clasificadores. Esto supone que la elección de un único clasificador lleve a perder rendimiento a la hora de clasificar un grupo de datos frente a otros. Para solucionar el problema se desarrollaron los sistemas de fusión de clasificadores. Estos métodos se basan en el desarrollo de algoritmos matemáticos cuya entrada son un conjunto de clasificadores entrenados.

Los sistemas de fusión de clasificadores son una modalidad de clasificación cuyo inicio de desarrollo data de los años 90 [26]. La combinación de clasificadores se puede hacer con sistemas en serie, donde los resultados obtenidos por cada clasificador, son la entrada del siguiente, como se observa en la figura 1.2 (a). Los sistemas en serie tienen problemas por la incapacidad de corregir errores de los clasificadores anteriores. Otro sistema posible son los sistemas paralelos, donde todos los clasificadores trabajan con la misma entrada y los resultados de todos se fusionan para obtener una decisión en consenso, viendo su estructura en la figura 1.2 (b). Los sistemas en paralelo son los que mejor desempeño tienen. También se pueden crear sistemas jerárquicos, que usan métodos en paralelo y cascada, combinados para obtener un funcionamiento

óptimo [27].

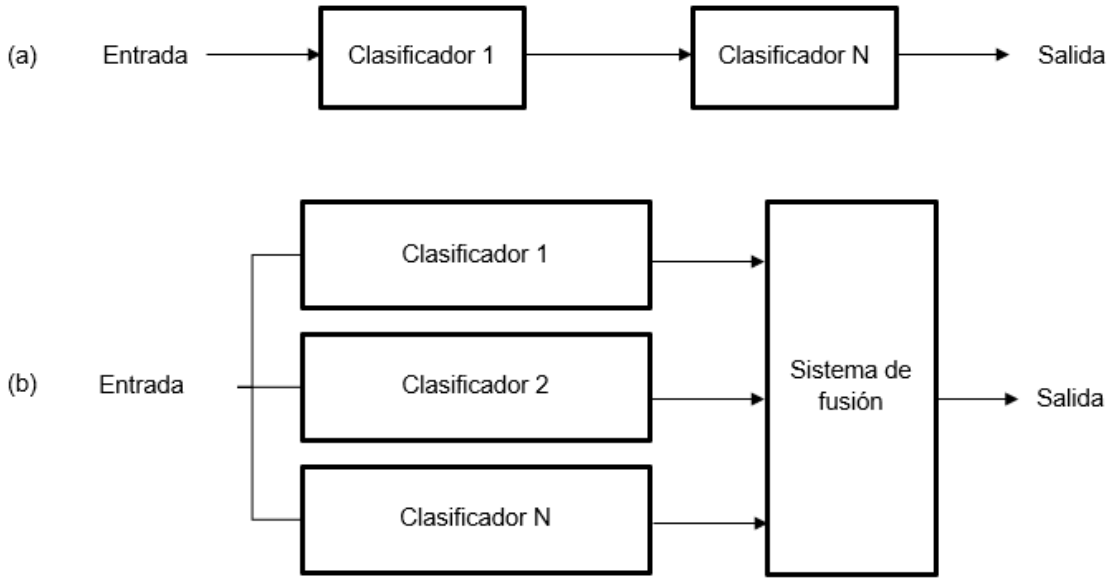

Figura 1.2: Sistema de fusión en serie (a) y en paralelo (b).

Hay 2 maneras de trabajar con los clasificadores en los sistemas de fusión de clasificadores. El primer grupo opera con los clasificadores y pone énfasis en desarrollar la estructura del clasificador. En este grupo no se trabaja con la salida hasta encontrar el proceso de combinación que da el mejor clasificador o grupo de clasificadores para poder operar con la salida. El segundo grupo opera con la salida de los clasificadores y calcula la combinación de salidas de los clasificadores [28].

1.2. Objetivos

El proyecto es un acercamiento a qué son y para qué sirven los sistemas de fusión de clasificadores y ver qué aportan con respecto a elegir clasificadores individuales para hacer tareas de regresión o clasificación. Se trabajará con un modelo paralelo de fusión de clasificadores, es decir, que las salidas de todos los clasificadores o grupos de clasificadores entrarán en el sistema de fusión para obtener la salida definitiva según el método usado.

Se entrenarán 7 métodos de clasificación de ML conocidos, para observar su rendimiento, problemas y ventajas de cada uno, sobre 2 bases de datos diferentes. Se evaluará el rendimiento obtenido por los métodos de fusión y se compararán con los 7 diferentes modelos de clasificación, para ver las ventajas que ofrecen los sistemas de fusión.

Se busca aprender a crear sistemas de fusión de clasificadores, evaluarlos, comprender su funcionamiento y la utilidad de usarlos con respecto a los métodos de clasificación individuales. Para el desarrollo del TFG se usarán los siguientes métodos de fusión de clasificadores existentes.

- Método de fusión por votación. Sistema de votación sencillo, que elige por mayoría de voto la salida estimada del algoritmo de clasificación. Puede modificar el valor del voto de cada clasificador.
- Multiperceptrón multicapa. Uso de una red neuronal artificial como sistema de fusión de las salidas de cada clasificador. Está formado por múltiples capas, construido de forma que la resolución del problema no es linealmente separable, es decir, que puede aprender y hacer tareas más complejas que las de un perceptrón simple.

1.3. Metodología

Para el desarrollo del TFG se han seguido los siguientes pasos:

- Analizar y transformar una base de datos de prueba contenida en el módulo de keras de *Tensorflow*, para su tratamiento e introducción correcta a los algoritmos de ML.
- Analizar y transformar la base de datos con la que se trabajará más en profundidad, formada por imágenes parametrizadas, para facilitar el análisis con algoritmos de ML.
- Crear, ajustar y entrenar 7 diferentes algoritmos de ML y determinar lo bien que actúan con las diferentes bases de datos y sus fallos.
- Crear un sistema de fusión por votación para diferentes grupos de clasificadores y analizar su viabilidad como evolución de los algoritmos de ML, valorando la mejora o empeoramiento del rendimiento y los recursos que necesita.
- Crear un MLP para diferentes grupos de clasificadores y analizar su viabilidad como evolución de los algoritmos de ML, valorando la mejora o empeoramiento del rendimiento y los recursos que necesita.

1.4. Estructura de la memoria

A continuación se describe el contenido incluido en cada capítulo de la memoria:

- **Capítulo 1: Introducción y objetivos**. Está formado por 4 subsecciones. En la primera sección se presenta que es ML, su historia, una pequeña descripción de como funcionan, y se presentan los sistemas de fusión de clasificadores. En la segunda y tercera sección se presentan los objetivos y la metodología a seguir. Por último, está la sección de la estructura de la memoria.

- **Capítulo 2: Conceptos previos**. En esta sección se introducirá más en profundidad el concepto de ML, junto con los métodos de entrenamiento y evaluación de algoritmos utilizados. Se presenta el método de validación cruzada y métricas como la precisión o la f1-score. Después, se explicará que son los sistemas de fusión de clasificación y los diferentes tipos que existen.

- **Capítulo 3: Métodos**. Se dará una explicación detallada de los 7 diferentes clasificadores que se usarán en el trabajo y de los 2 métodos de fusión de clasificadores.

- **Capítulo 4: Base de datos y análisis descriptivo**. Se dará una visión detallada de las 2 bases de datos usadas en este proyecto, haciendo una descripción analítica y una explicación de su construcción. Además, se hablará de la diferente manipulación a las que se someterán los datos para una mejora en la clasificación.

- **Capítulo 5: Evaluación, Experimentos y resultados**. Los resultados de los diferentes clasificadores y algoritmos de fusión se muestran en esta sección.

- **Capítulo 6: Conclusiones y líneas futuras**. Se discutirán las conclusiones y líneas futuras del proyecto.

Capítulo 2

Conceptos previos

En este capítulo se introducen una serie de conceptos fundamentales que ayudan a entender el marco sobre el que se desarrolla este TFG.

2.1. Conceptos de aprendizaje automático

Actualmente el ML crea un sistema de aprendizaje con los datos de una muestra de una base de datos, para crear un llevar a cabo una tarea de clasificación o regresión de datos subsecuentes de la misma fuente [29]. Una base de datos se puede definir como el conjunto de vectores con información que definen a un objeto o concepto [30]. Cada vector está formado por n atributos que describen al objeto lo más cercano posible. Por ejemplo, En el caso de un melón, algunos de sus atributos serían: color, sabor, semilla, olor, sonido, etc. Cada atributo tiene asignado un valor para cada objeto en el espacio de muestra. El algoritmo de ML debe de poder aprender a discernir entre los diferentes objetos de muestra, a partir del análisis de todos los atributos, o los necesarios, para poder generar un sistema de parámetros que diferencie entre los diferentes tipos de objetos que entran al sistema, y así, de manera autónoma, pueda diferenciar los distintos objetos [10]. El proceso usado en este trabajo para crear un sistema de ML, se basa en la división de los datos en 2 grupos [31]:

1. **Grupo de entrenamiento**: es la primera fase y se basa en el entrenamiento del algoritmo de ML **vacío**, es decir, que no tiene definido ningún hiperparámetro o regla de decisión. Usando los datos iniciales de entrenamiento, se desarrollará el algoritmo de ML para que identifique patrones y empiece a modificar sus hiperparámetros para lograr la identificación distintiva de las diferentes clases o valores respecto a las entradas de entrenamiento.

2. **Grupo de test**: segunda fase que consiste en la introducción al sistema entrenado de datos completamente **nuevos** y diferentes de los de entrenamiento para, a partir de diferentes figuras de mérito como la precisión o la sensibilidad, evaluar el rendimiento del algoritmo de ML. Las figuras de mérito se explicarán más en profundidad en el capítulo 3 de este trabajo.

Los modelos de aprendizaje se dividen principalmente en dos tipos importantes según la forma en la que la máquina desarrolla las habilidades de aprendizaje: **aprendizaje supervisado** y **no supervisado** [32].

2.1.1. Aprendizaje supervisado

En el aprendizaje supervisado, el algoritmo creado para que la máquina aprenda produce una **función** que establece una correspondencia entre las entradas y las salidas deseadas del sistema. Las entradas al sistema están completamente **etiquetadas** y tienen una salida definida. Es un algoritmo basado en ejemplos, generando conocimiento a partir de la introducción de series de ejemplos y contraejemplos. En este tipo de aprendizaje el sistema trata de etiquetar los vectores usando diferentes clases de ejemplos anteriores. Algunos ejemplos de métodos de aprendizaje supervisado son redes neuronales artificiales o árboles de decisión [33]. Existen 2 tipos de aprendizaje supervisado:

- **Clasificación**: el algoritmo **etiqueta** las clases de los datos, es decir, que separa los datos en 2 o más **grupos** distintos pero concretos. Por ejemplo, si los objetos son frutas, si entra una manzana la etiqueta como manzana y si entra una pera, la etiqueta como pera o no manzana. El número de etiquetas de los algoritmos de clasificación son numerables, finitas.
- **Regresión**: el algoritmo obtiene un **resultado numérico** a partir de cada entrada. El resultado tendrá un rango de valores posibles definido, y no una etiqueta. Un ejemplo sería predecir el precio de venta de una vivienda. El valor obtenido está definido en un rango de valores, pero el número de posibles resultados es infinito. Es decir, el número de etiquetas posibles es **infinito**.

2.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado está basado en un modelo construido con ejemplos que solo incluyen entradas al sistema, no se tiene información sobre las categorías. El sistema realiza un aprendizaje por **observación y descubrimiento**, analizando las diferentes entradas y determinando características comunes. Tras este análisis, el algoritmo agrupa los datos en **conceptos** [34]. Algunos ejemplos son: la técnica de agrupamiento de *Clústers* o el algoritmo de K-medias.

Un algoritmo de k medias funciona con la selección de k agrupaciones existentes para los datos. Cada agrupación tiene un centroide, el valor más representativo del conjunto, y se utiliza la distancia euclídea con k valores ya categorizados para asignar a cada entrada su grupo. El algoritmo realiza un número t de iteraciones para determinar los centroides óptimos para cada *clúster* [35].

2.2. Sistemas de fusión de clasficadores

Los sistemas de fusión de clasificadores son métodos de ML que trabajan con **grupos de clasificadores** diferentes, que buscan la fusión óptima de diferentes clasificadores para crear un algoritmo conjunto definitivo. Los sistemas de fusión de clasificadores pueden trabajar sobre los grupos de clasificadores y/o sobre la salida.

Los métodos para trabajar con solo los clasificadores son:

- **Selección dinámica de clasificadores** (DCS): busca extraer el **mejor** clasificador en vez de un grupo de diferentes clasificadores. Intenta determinar el clasificador que mejor clasifica la salida, resultando en la elección de un solo clasificador. Hace una partición de la muestra de entrada. Se usan distintos métodos como top N, agrupamiento de características o *local accuracy* [36]. En *local accuracy*: se define un set de particiones de las entradas y se eligen el mejor clasificador para cada set. Tras esto, se introduce una muestra desconocida a cada partición y la salida del mejor clasificador para esa partición es el elegido. Dependen de los datos de entrenamiento, pero pierden información al elegir al mejor localmente. Pero, al hacer DCS secuencialmente excluyendo al mejor elegido anteriormente, se obtiene un ranking de clasificadores y de clase.

- **Clasificador de estructura y agrupamiento**: para los clasificadores de estructura, se organizan los clasificadores en **paralelo**, y por separado se obtienen sus salidas como la

entrada a una **función de combinación**. La segunda aproximación se basa en organizar los clasificadores por **grupos** y aplicar distintos métodos de fusión a cada grupo. Se crea una estructura en múltiples fases. En cada fase se aplican distintos métodos de fusión para cada grupo de clasificadores y se puede aplicar DCS a cada grupo. Hay muchas opciones de diseño, pero se debe buscar un buen nivel de diversidad de clasificadores, datos de entrenamiento y métodos en cada grupo. Se produce una mejora si la incertidumbre de la información se reduce.

- **Mezcla jerárquica de expertos**: Se organizan los clasificadores en grupos de clasificadores para crear una estructura parecida a un **árbol**. Cada hoja es una red experta, a la que dada un vector de entrada x, intenta resolver el problema de aprendizaje supervisado localmente. Las salidas de elementos del mismo nodo se combinan y reparten por la red de entrada y el total de las salidas del nodo se da como una **combinación convexa**, definida como la suma de elementos no negativos cuyo resultado es 1. Las redes de expertos se entrenan para aumentar la probabilidad a posteriori y tras eso, un número de algoritmos de aprendizaje se aplican para afinar el modelo. No se puede aplicar en bases de datos muy grandes, pues se aumenta la complejidad del árbol. Ejemplo de métodos de fusión, cuya fuerza viene de la estructura de los clasificadores. Representa una técnica de aprendizaje supervisado basada en la división para mejorar [37].

Los métodos que trabajan sobre las salidas son: métodos de fusión de etiquetas, métodos de ranking y métodos de salidas blandas.

2.2.1. Métodos de fusión de etiquetas

Método que consiste en la fusión de las salidas de un grupo de clasificadores en un sistema de unión. Este sistema trata a cada salida como una etiqueta. Las etiquetas determinarán la salida definitiva del sistema en función de varios criterios de selección [38]. Si disponemos de datos de entrenamiento podemos mejorar la salida de estos clasificadores a grupos operando en ranking por clases o medidas difusas, que indica que un dato no pertenece de manera exclusiva a un grupo, sino que puede pertenecer a varios, pero con un nivel diferente de probabilidad.

- **Métodos por votación**: se lleva a cabo una votación con las salidas de los diferentes clasificadores para elegir la salida definitiva del sistema. Se aplican a múltiples sistemas de clasificadores, asumiendo que cada clasificador da una sola etiqueta de clase como salida y no disponemos de datos de entrenamiento. Hay muchas aproximaciones para

combinar la información. Por conveniencia, la salida de los clasificadores genera el vector de decisión $d = [d1, d2, \ldots, dn]^T$, donde $di \in \{c1, c2, \ldots, cm, r\}$, siendo ci la etiqueta para la clase i y r el rechazo de asignar a la muestra de entrada a cualquier clase. La función binaria es:

$$B_j(c_j) = \begin{cases} 1 & \text{si } d_j = c_j \\ 0 & \text{si } d_j \neq c_j \end{cases}$$

definiendo la regla general de voto como:

$$B(d) = \begin{cases} c_i & \forall_{t \in \{1,\ldots,m\}} \sum_{j=1}^{n} B_j(c_t) \leq \sum_{j=1}^{n} B_j(c_i) \geq am + k(d) \\ r & \text{resto} \end{cases}$$

siendo α un parámetro que controla el valor del voto de cada clasificador y k(d) una función que da restricciones de voto adicionales. La norma más conservadora es la dada por k(d)=0 y α=1, significando que la clase se elige cuando todos los clasificadores generan la misma salida. Esta regla se aligera reduciendo α, creando la función del voto mayoritario cuando α=0.5. La función k(d) es un nivel de abyección a la clase más seleccionada y hace referencia al score de la segunda clase.

2.2.2. Métodos de ranking

Se combinan múltiples clasificadores y se le da a cada clasificador un valor de verdad en el grupo de clasificadores. Se tienen en cuenta 2 criterios: El tamaño del conjunto de clases y la probabilidad de contener la clase verdadera en el conjunto reducido [39]. Se busca un intercambio entre minimizar el conjunto y maximizar la probabilidad de incluir la clase verdadera. Algunos métodos son:

1. **Intersección de vecinos**: en este algoritmo se hace la interseccion de las salidas de los clasificadores y selecciona el valor de salida según se acerque a un grupo u otro de salidas. Primero se determinan las vecindades de todos los clasificadores por rangos de clases, dividiendo, en el peor caso, en el set de entrenamiento. El menor rango dado por cualquier clasificador es el umbral y solo se usan las clases por encima de este.

2. **Unión de vecinos**: se hace la unión de pequeños grupos de clasificadores y para cada grupo se obtiene la salida por consenso, para luego llevar a cabo un método de interseccion entre todos los grupos.

2.2.3. Métodos de salidas blandas

Es el grupo más extenso y producen salidas blandas, que son salidas de valores reales entre 0 y 1. El valor obtenido aporta probabilidad, posibilidad, necesidad, creencia y plausibilidad [40]. Estos modelos buscan reducir el nivel de incertidumbre maximizando la evidencia con medidas apropiadas. Algunos métodos son:

1. **Métodos bayesianos**: se aplican siempre que la salida de los clasificadores esté expresada en probabilidad. La combinación efectiva de las probabilidades es una probabilidad, mayor que la probabilidad obtenida por el mejor clasificador [41]. Uno de los métodos usados es la media bayesiana simple: si las salidas de los clasificadores son probabilidades de que una entrada x venga de una clase $C_i, i = 1, ..., m : P(x \in \frac{C_i}{x})$, se puede calcular la probabilidad media a posteriori de todos los clasificadores con:

$$P_E(X \in \frac{C_i}{x}) = \frac{1}{K} \sum_{k=1}^{K} P_k(X \in \frac{C_i}{x}) \text{ donde } i = 1, ..., m.$$

   Para obtener la probabilidad a posteriori de los clasificadores se hace la siguiente operación: $Pk(X \in \frac{C_i}{x}) = \frac{k_i}{knn}$ donde ki es el número de muestras de la clase Ci de todas las muestras prototipo de knn.

2. **Redes neuronales artificiales**: la entrada se introduce a un sistema con forma de red neuronal. Este sistema consta de diferentes capas: capas de entrada, a las que llegan las salidas de los clasificadores, capas ocultas, un conjunto de 1 o más capas de datos, a las que se introducen los datos modificados de la capa anterior, y la capa de salida, que da el resultado de la neurona artificial. Cada capa tiene asignada unos pesos que modifican el valor de entrada, y los pesos deben calcularse con entrenamiento para que obtengan el resultado correcto con la entrada.

Capítulo 3

Métodos

3.1. Etapas de Diseño

El primer paso del TFG es la **creación, entrenamiento y evaluación** de 6 clasificadores distintos. Realizaremos una tarea de aprendizaje **supervisado** con todos los clasificadores, puesto que las 2 bases de datos a utilizar en este TFG son etiquetadas. Para preparar los datos, se ha procedido a hacer la división de cada base de datos en subconjunto de *train* y subconjunto de *test*, con relación 80/20, es decir, 80 % de los datos son de entrenamiento y el 20 % restante son de test. Se debe realizar una división totalmente aleatoria e indiscriminada.

- **Fase de entrenamiento**: en esta fase se emplea el subconjunto de *train*. Estos datos se utilizan para **entrenar** el modelo, y que el algoritmo sea capaz de encontrar patrones entre los datos que le permitan hacer una adecuada generalización.
- **Fase de test**: se emplea el subconjunto de *test*. Estos datos son utilizados para **evaluar** el desempeño del modelo creado. Este desempeño es medido a partir de la utilización de varias figuras de mérito.

3.1.1. Normalización de las Características

La normalización es un método de **optimización** de datos, que consiste en reasignar el valor de todos los datos en un **rango** numérico determinado. Es usado mucho en ML pues suele mejorar el rendimiento de los clasificadores, puesto que, algunos clasificadores tienden a dar más relevancia a atributos cuyos valores son muy elevados comparados con el resto de

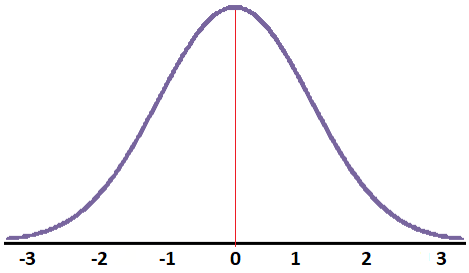

Figura 3.1: Distribución normal [2].

atributos. El objetivo de la normalización es que todos los datos estén distribuidos en un rango de valores determinado, para evitar que ocurra el **dominio por rango**. Así, se les da el mismo peso a todos los atributos y permite al algoritmo evaluarlos correctamente. Se suelen usar en algoritmos que usan distancias, como es K vecinos más cercanos [42].

Para este TFG se ha usado un método de **estandarización**, por el cual se distribuyen los datos como si fuera una función **normal**. Una función normal es una distribución equivalente de los datos entre los valores 1 y -1, solo teniendo desde un cinco hasta un uno por ciento de los datos fuera de este rango [2], como la distribución observada en la figura 3.1. Se obtiene usando la media μ y la varianza σ de cada atributo y se obtiene su valor con la siguiente fórmula:

$$V_i = \frac{V_{0i} \times \mu}{\sigma} \tag{3.1}$$

Para hacer la normalización, se utilizará el módulo *preprocessing* de *sklearn*, usando las funciones de *scale* y *StandardScaler*.

3.1.2. Selección de características con chi cuadrado

Es un método de optimización usado para seleccionar las **características más relevantes** en una base de datos. Es un test que mide como un modelo se compara con los datos observados. Se usa para observar si una variable independiente **afecta** en el resultado de la variable dependiente, observando la distribución de los datos según los datos de salida. Una variable afectará a la salida sí lo modificación de su valor provoca que la salida varíe en una medida proporcional al cambio sufrido por la variable independiente de referencia. Puede ser fuerte (1 o 0.9), haciendo que la variable determine fuertemente el valor de la variable dependiente; o un valor medio (0.6 a 0.4), pues comparte con otras variables la fuerza para determinar la salida. Si el valor es bajo (0.2 o 0.1), es que la variación de la variable apenas afecta al resultado [43]. El valor de medida se suele medir entre 1 y -1. Los valores negativos indican que un aumento en el valor de la

variable x, provoca un decremento en el valor de la variable dependiente y. Se usará el módulo *feature_selection* de *sklearn*, que contiene la función *sklearnSelectKBest* para seleccionar las 3 mejores variables según el método de chi cuadrado.

3.2. Métodos de Aprendizaje Automático

En el presente TFG, los modelos construidos han entrenados con todas las características presentes en cada una de las bases de datos.

3.2.1. Regresión Polinómica

La Regresión Polinómica (PR) es un método paramétrico lineal de clasificación, modelo extensión de MLR que se construye creando un **polinomio** de grado k. Es un modelo construido con la fórmula del MLR, donde cada ecuación contiene el atributo y su coeficiente de correlación respectivo a cada grado del atributo, junto con la pendiente y el error.

$$y = m + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_i x_i^k + e \tag{3.2}$$

Es un modelo con un funcionamiento similar al MLR, pero la PR da una mejor aproximación entre las variables independientes y la dependiente y puede ajustarse a muchos más datos. El problema es que son muy **sensibles a valores externos** [44].

Para crear el modelo PR se usa la función *PolynomialFeatures* de *sklearn*.

3.2.2. Regresión Logística

La Regresión Logística (LR) es un modelo de clasificación lineal y paramétrico, que separa las clases en diferentes grupos, correspondiente a las difernetes salidas existentes en nuestra base de datos. La separación más comun en este modelo es en las **etiquetas** clase 0 y clase 1. La etiqueta se estima con la **combinación lineal** de los parámetros de cada variable multiplicados por sus pesos y genera una salida con valor de probabilidad de pertenencia a cada clase, que es un valor comprendido entre 0 y 1 (en caso de ser un modelo binario) [45].

El problema que suele sufrir es **sobre ajuste de los datos de entrenamiento**, sobre todo en datos con muchas dimensiones o escasos. No obtiene relaciones complejas y requiere de

relación lineal entre variables y salida. Es fácil de implementar y entrenar y es rápido para nuevos registros y clasificar por importancia [46].

$$\ln P(Y=1) = B_0 + B_1X_1 + ... + B_nX_n$$

B0 es el interceptor y Bi es el coeficiente de regresión, el cambio en las probabilidades en caso de que X aumente en 1. A esta probabilidad se le hace la inversa del logaritmo y así se obtiene la probabilidad que tiene cada entrada.

Para crear el modelo LR se usa la función *LogisticRegression* de *sklearn*.

3.2.3. K vecinos más cercanos

K vecinos más cercanos (KNN) es un método de clasificación que usa un conjunto de ejemplos **etiquetados** para determinar la salida. Para determinar a que conjunto pertenece cada ejemplo, se evalua que conjunto de clases están más **cerca**. Se conoce la clase a la que pertenece cada ejemplo, siendo un algoritmo de aprendizaje supervisado. **No es paramétrico**, no hace suposiciones sobre la estructura de los datos y usa la distancia de un punto nuevo xi a cada punto del conjunto de entrenamiento para etiquetarlo. Se usa la **distancia euclídea**:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

donde n es el número total de observaciones del conjunto. Para etiquetar la clase, se escoge por votación de mayoría entre las k clases más cercanas.

Para crear el algoritmo, se debe definir primero el hiperparámetro k, que es el número de vecinos. Valores bajos de k hace genera una frontera de decisión flexible, provocando un sobre ajuste en el modelo por la alta varianza. Valores altos de k hace que el modelo creado tenga poca flexibilidad y lleva al sub ajuste del modelo por la baja varianza [3]. Ejemplos de los posibles ajustes del modelo se observan en la figura 3.2, observando un sobreajuste en la figura de la izquierda, un subajuste en la figura de la derecha y un ejemplo generalización en el centro.

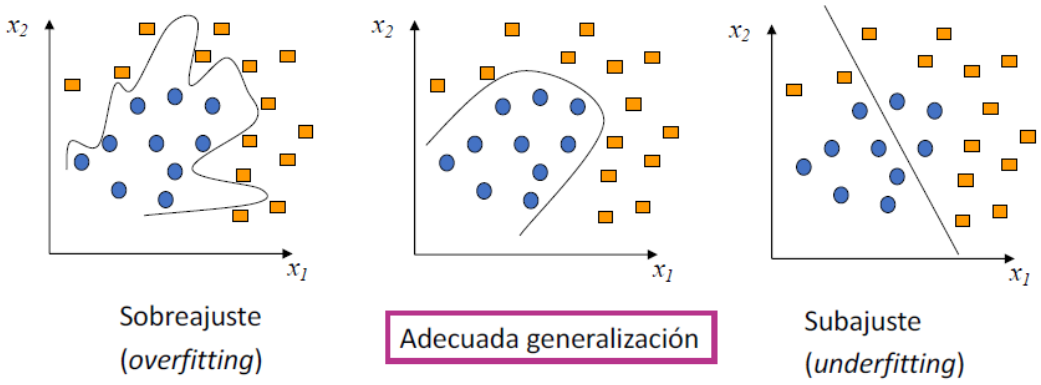

Figura 3.2: Generalización [3].

Es un buen modelo para clasificar **sistemas sencillos de datos**, pero trabaja mal con muchas **dimensiones o sistemas dinámicos**, teniendo una gran **dependencia** en el valor de **k**. Además, al ser un modelo que trabaja con distancia, la normalización suele ser muy necesaria para eliminar el sesgo del rango.

Para crear el modelo KNN, usaremos la función *KneighborsClassifier* de *sklearn*.

3.2.4. Árbol de Clasificación

Un Arbol de Clasificación (DT) es un método supervisado de clasificación no paramétrica y no lineal basado en la creacion de una estructura tipo **árbol** con ramas y hojas como nodos de selección. El recorrido para predecir la clase de una entrada inicia en el **nodo raíz** del árbol. El algoritmo **compara** los valores del atributo raíz para seleccionar la rama del árbol a la que salta en el siguiente paso, y repite el proceso pasando por los sub nodos hasta alcanzar un **nodo hoja**. Se pueden definir los elementos de un DT como:

- Nodo raíz: inicio del DT. Representado por la variable más relevante del sistema.
- Nodo secundario: Son el resto de nodos que no son raíz y son puntos de toma de decisión.
- Ramas: uniones entre nodos secundarios que contienen las condiciones a cumplir para alcanzar el siguiente nodo.
- Nodo hoja: nodo de salida final, que indica la decisión tomada y no se puede segregar más.

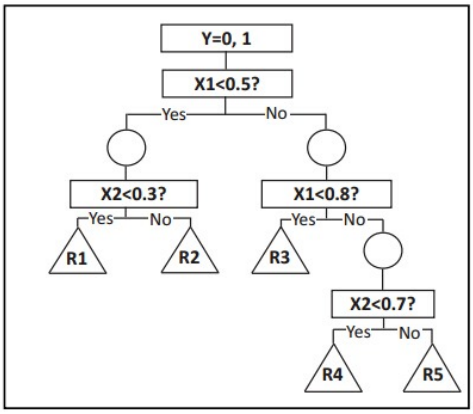

Figura 3.3: Árbol de decisión [4].

A la hora de construir un DT, se empieza en el nodo raíz, el cual definimos como S, que contiene el conjunto de datos completo. A continuación, se busca el mejor atributo en el conjunto de datos utilizando un enfoque estadístico, como es el **índice de Gini**, que es una medida de impureza utilizada para seleccionar las características que mejor dividen a los datos en 2 o más grupos; y tras seleccionar el atributo y determinar las divisiones que hace en el conjunto de datos, se crean nuevas ramas en el árbol. Por tanto, divide el conjunto S en subconjuntos que contienen valores posibles para los mejores atributos y se genera el nodo del árbol de decisión. Finalmente, crea recursivamente nuevos nodos de decisión utilizando los subconjuntos del conjunto de datos creado anteriormente. Continúa con este proceso hasta llegar a una etapa en la que no pueda clasificar más los nodos y llame al nodo final como nodo hoja [4]. En la figura 3.3 se puede observar una construcción ejemplo de un arbol de decisión.

Es un sistema jerárquico parecido a un **diagrama de flujo**, donde cada nodo tiene una regla de decisión interna que ha sido calculada en la fase de entrenamiento. Es un algoritmo fácil de entender y simplifica la relación entre variables, pero el modelo se sobre ajusta o sub ajusta fácilmente y **variables correladas** tienden a controlar el árbol. Se requiere una adecuada selección de los hiperparámetros: número mínimo de muestras por hoja y profundidad máxima [47].

Usaremos la función *DecisionTreeClassifier* de *sklearn* para construir el árbol.

3.2.5. Random Forest

Un *Random Forest* (RF) es un modelo no paramétrico de clasificación y regresión, muy popular en métodos de ensamblado, en el que se usa a muchos áboles aprendices para mejorar

el rendimiento con respecto a un solo individuo [48]. Es un clasificador definido como una **colección de clasificadores** con estructura de árbol, que tienen la misma distribución y cada árbol **vota** por la mejor clase para la entrada.

Cada árbol depende de un **vector independiente** del resto que usa un conjunto aleatorio de la muestra diferente al de cada árbol. Para crear el árbol, iniciando con un conjunto de diferentes DT, se muestrea un subconjunto aleatorio de variables de las características originales y se usa para ajustar un DT, creando un bosque de árboles diferentes. Las predicciones finales de cada árbol se mezclan y se hace votación. [49]

Este modelo mejora los problemas de sobre ajuste de los DT al tener tantos árboles diferentes. El problema con los RF es que un gran número de árboles hace al algoritmo **lento e inefectivo** en clasificaciones en **tiempo real**, siendo malo para predecir datos nuevos [50].

Para crear el modelo de RF se usará la función *RandomForestClassifier* de *sklearn*.

3.2.6. Máquinas de Vector Soporte

La Máquina de Vector Soporte (SVM) es un método de aprendizaje supervisado, lineal y paramétrico usado para clasificación. SVM crea un **hiperplano de separación** entre las diferentes clases. Un hiperplano es un subespacio plano que no tiene que pasar por el origen con una **dimensión menor** al plano original, es decir, que un hiperplano de un espacio de 1 dimensión (una línea recta) es un punto, que divide a la recta en 2, y en un espacio de 3 dimensiones, es un plano en 2 dimensiones que divide el espacio en 2. Se define por la ecuación lineal:

$$a_1x_1 + ... + a_nx_n + b = 0$$

Donde las variables a_1 no tienen por qué ser 0 todas y si b = 0, el hiperplano pasa por el origen. Cada sección se puede describir con las desigualdades [51]:

$$a_1x_1 + ... + a_nx_n + b \leq 0$$

$$a_1x_1 + ... + a_nx_n + b \geq 0$$

La búsqueda del mejor hiperplano se hace con el **método de *kernel***, que es la creación del hiperplano óptimo, que crea la superficie de decisión para separar las clases, ejemplificado en la figura 3.4. Los modelos SVM son buenos, con muchas variables, tienen **buena memoria** y son **versátiles**. El problema que tiene es que no dan estimaciones de probabilidad directa, sino que

se usa validación cruzada y si las funciones son más que las muestras, se debe tener cuidado con la función de *kernel* y la regularización para hacer bien el modelo [5].

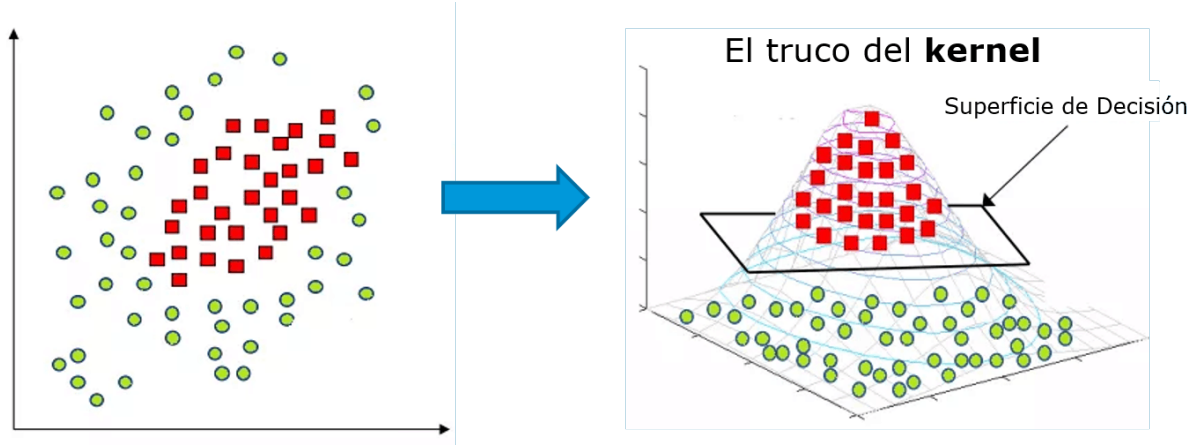

Figura 3.4: Truco de *kernel* [5].

Para crear el modelo SVM se usa la función *SVMLinear_medland*

Cada clasificador necesita declarar sus propios parámetros: KNN necesita un valor óptimo de k para evitar sobre ajustes, DT necesita una profundidad máxima y matices de parada de ramas, RF necesita un número de árboles generados con su máxima profundidad, MLR el valor de cada coeficiente, PR el valor de cada coeficiente y el grado óptimo, LR el interceptor y cada coeficiente de regresión y para SVM el grado de tolerancia y el coeficiente. Encontrar cada parámetro es un proceso tedioso y aburrido, pero para obtenerlo podemos usar la función *GridSearchCV* de *sklearn*.

La función *GridSearchCV* es parte del paquete *model_selection* de la librería *sklearn* de *Python*, cuyo objetivo es hacer una búsqueda exhaustiva de los parámetros que mejor se ajusten a un modelo de regresión o clasificación dado [52].

3.3. Métodos de fusión de clasificadores

Tras entrenar los clasificadores y obtener sus respectivas salidas, cada salida entra en el sistema de fusión de clasificadores, que tendrá una forma de actuar u otra para predecir la salida final de cada entrada, funcionando como un circuito con una entrada, dos nodos y una salida. El modelo a crear, seguirá la estructura reprersentada en la figura 3.5 de modelo en paralelo.

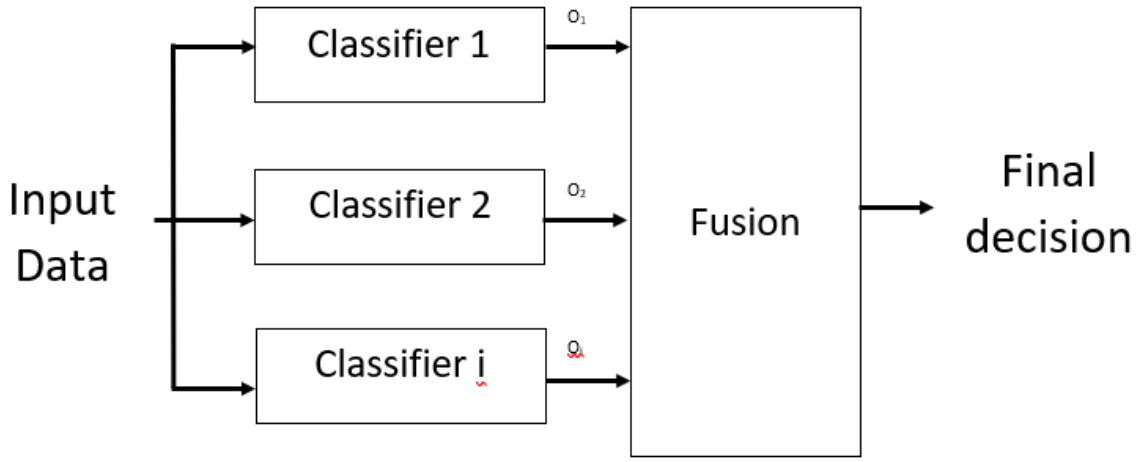

Figura 3.5: Esquema del método de fusión de clasificadores.

3.3.1. Métodos de votación

Este modelo funciona suponiendo que cada clasificador proporciona **una única etiqueta** de clase como salida. El modelo crea un vector de decisión que contiene la salida para cada clasificador. Después de crear el vector de decisión, se crea una regla de decisión. En casos binarios, la decisión final será 1 o 0 dependiendo del **voto mayoritario** del vector de decisión y un parámetro α. Sí $\alpha = 1$, la clase se elige cuando todos los clasificadores producen el mismo resultado. Este parámetro se puede bajar a 0,5 para elegir la **predicción de clase mayoritaria** [33]. Para crear el modelo, se realizará una función hecha manualmente para contar usando la función *Counter* de *collections* y se construirá la matriz de confusión y se obtendrán los valores de precisión, efectividad y el *f1-score*.

3.3.2. Perceptrón Multicapa

El segundo método consiste en la introducción de todas o algunas salidas de los clasificadores a la entrada de un perceptrón multicapa (MLP), que analizará las entradas y llevará a cabo operaciones de **cálculo por pesos** en diferentes capas hasta obtener una salida definida, que será la estimación final a los datos de entrada. Se usa un sistema binario de salida.

El MLP es un modelo que crea una **red neuronal artificial multicapa**, esto significa que, a parte de las neuronas de entrada y salida básicas de una neurona artificial, incorpora una o varias **capas ocultas** entre la entrada y la salida. Todas las capas están **conectadas** siempre con una capa anterior y una posterior, menos las neuronas de entrada y salida y por esta estructura puede

resolver **problemas no linealmente separables**, el cual es un problema que no tiene unas rectas o hiperplanos definibles para separar las muestras; gracias a su capacidad de **aprendizaje** [53]. Una red neuronal artificial es un método de clasificación de ***Deep learning*** que es un esquema de computación distribuida inspirado en la estructura del sistema nervioso humano. La arquitectura de una red neuronal consta de múltiples procesadores con un algoritmo de ajuste de pesos para obtener la salida esperada, basandose en las muestras de entrenamiento anteriores que se usaron para ajustar los pesos de cada neurona artificial.

La estructura de una neurona está basada en el concepto abstracto creado por McCulloch y Pitts en 1943 sobre el modelo de una neurona artificial [6], mostrado en la figura 3.6.

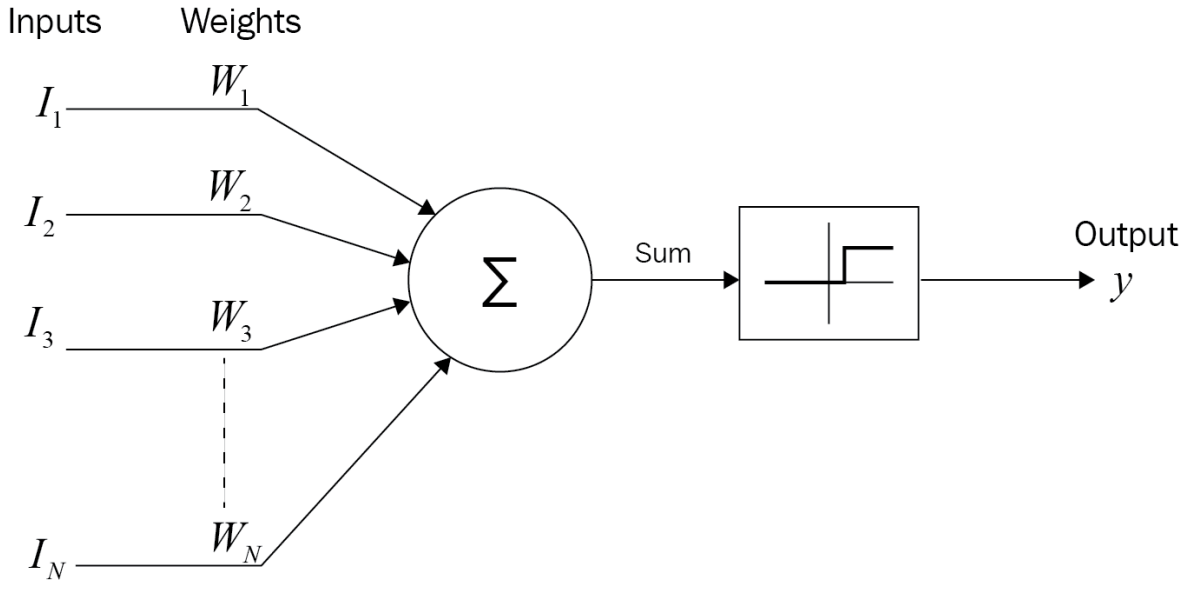

Figura 3.6: Esquema de la neurona artificial [6].

El modelo tiene un **vector de pesos** w = (w1, w2, . . . , wn) para las n entradas al modelo, que entran al sistema y se combinan **linealmente**. La activación de la salida dependerá del valor obtenido y esto generará una salida. En el ejemplo de que sea binaria, se verá de la siguiente forma [54]:

$$y(z) = \begin{cases} 1 \, z \geq 0 \\ 0 \, z < 0 \end{cases} \tag{3.3}$$

El MLP sigue un esquema de capas, como el representado en la figura 3.7, teniendo:

- Capa de entrada: Primera fase del MLP, donde llegan por primera vez al sistema las entradas y son modificadas por los pesos de cada neurona.

- Capas ocultas: Capa seguida de la capa de entrada u otra capa oculta anterior, que realiza un nuevo ajuste de las salidas de la capa anterior. Puede ser 1, 2 o varias o no existir.
- Capa de salida: Capa final con el umbral de decisión para obtener la salida del sistema.

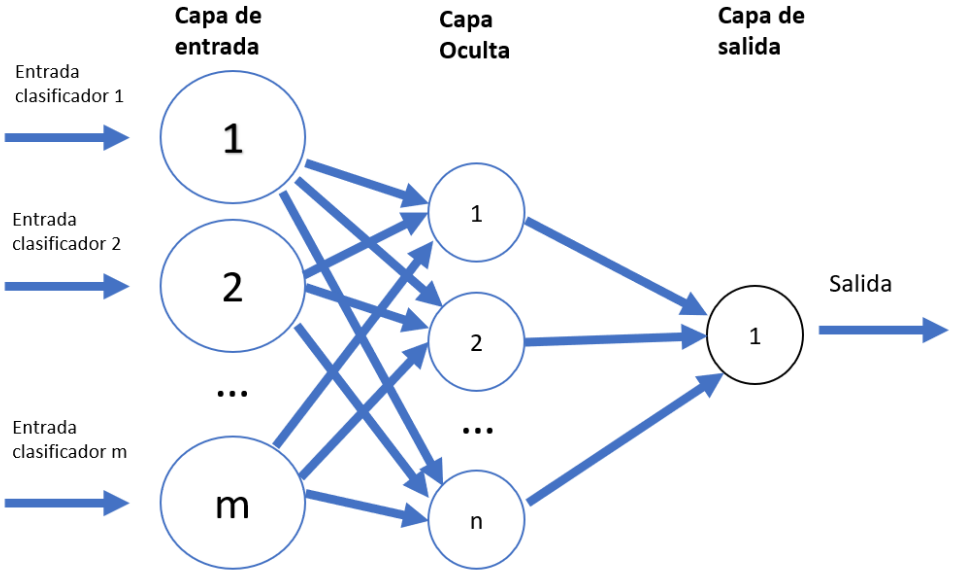

Figura 3.7: Esquema de MLP.

El número de capas ocultas y neuronas de cada capa oculta define la complejidad del modelo, la cual suele estar asociada a una mejor clasificación del modelo, pero siempre necesita de más **recursos** para funcionar [55].

3.4. Figuras de mérito

Para concluir este capítulo, se hace una presentación de las distintas figuras de mérito utilizadas para evaluar las prestaciones de los clasificadores diseñados en el Capítulo 5.

3.4.1. Matriz de confusión

La matriz de confusión, también conocida como matriz de error o matriz de contingencia, es una matriz construida a partir de la imagen de referencia con N filas clasificadas en M clases. En las columnas se ordenan las clases reales y sobre las filas las unidades estimadas. Los valores

en diagonal son interpretaciones correctas y cualquier valor fuera de estas, es un valor mal estimado [56]. Esta matriz se puede observar en el Cuadro 3.1.

		Valor Real	
		Clase=1	Clase=0
Valor Clasificación	Clase=1	Verdaderos positivos (VP)	Falsos positivos (FP)
	Clase=0	Falsos negativos (FN)	Verdaderos negativos (VN)

Cuadro 3.1: Matriz de confusión para clasificación binaria.

Precisión

La precisión (en inglés *accuracy*) se define como la diferencia entre la medida o estimación con respecto a un valor conocido de una precisión mayor. En ML se usa como la proporción de casos correctamente clasificados con relación al número total muestras. Está definido por la siguiente ecuación:

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

Sensibilidad

La sensibilidad (en inglés *recall*) es la proporción de casos correctamente clasificados en relación con el número total de muestras del conjunto evaluado, no del total de casos. Se define por la siguiente ecuación:

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

f1-score

Es la combinación de precisión y sensibilidad en un solo valor. Con el F1 se facilita comparar varias soluciones y el rendimiento combinado de precisión. Se construye con la media armónica de precisión y sensibilidad. Definido en la siguiente ecuación:

$$F_1 = 2 \times \frac{precision + recall}{precision \times recall}$$

Capítulo 4

Base de datos y análisis descriptivo

Este capítulo tiene como finalidad hacer una presentación de los bases de datos consideradas en el TFG, así como una descripción de las etapas de preprocesamiento seguidas.

4.1. Base de datos de MNIST

La base de datos de MNIST será el primer conjunto de datos usado en este proyecto. Es una base de datos disponible en el módulo *keras* de ***Tensorflow***, compuesta por 70.000 imágenes de números del 0 al 9 escrito a mano, distribuidos de manera **aleatoria** y **equivalente** por todo el set de datos. Un ejemplo de como es cada número se representa en la figura 4.1.

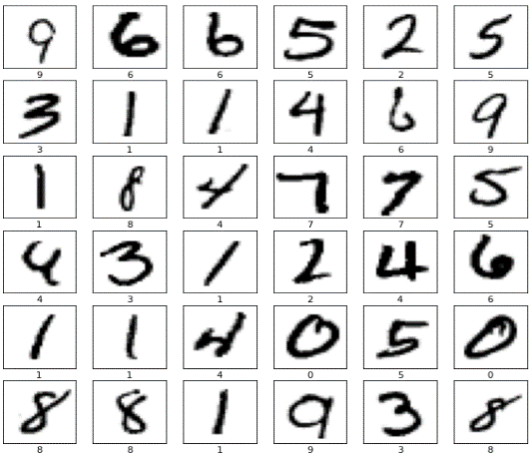

Figura 4.1: Muestra de las imágenes contenidas en la base de dados *mnist*.

Cada imagen tiene un formato de un array de 2 dimensiones de tamaño 28x28 píxeles, cuyos valores varían entre 0 y 1. El valor puede ser decimal. Valores cercanos a 0 son tonos blancos

y valores cercanos a 1 se representan como colores oscuros. La distribución de ejemplos en la base de datos es **balanceada**, es decir, que la proporción de casos para cada clase es muy parecida, evitando de esta forma que el algoritmo pueda tener inclinación a la predicción de un tipo de clase. Se puede ver la distribución en la figura 4.2. Cada clase tiene de media 6000 imágenes en el grupo de entrenamiento, siendo las de mayor número las de clase 1.

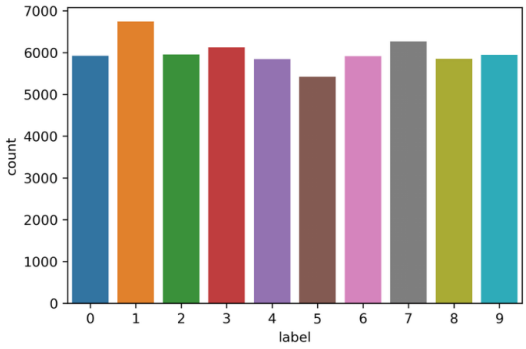

Figura 4.2: Distribución de clases *mnist* [7].

4.2. Base de datos de extracción de características

Es la segunda base de datos utilizada. Es una base de datos que contiene 1995 imágenes de cáncer de laringe, que han sido procesadas con un método de extracción de características. Las imágenes usadas son de tumores benignos y malignos de laringe, obtenidos por **Endoscopia de contacto**, una técnica mínimamente invasiva que da información de la estructura celular y vascular de la capa superficial de la laringe, usando un endoscopio que es insertado en el paciente y con la cámara adherida, se toma video de la zona de interés [57]. A estas imágenes se les añade un método de ***Narrow Band Imagin*** para mejorar la **visibilidad** de los tejidos de la imagen. La técnica de NBI funciona filtrando la luz blanca con unas longitudes de onda de luz específicas absorbidas por la hemoglobina, para mostrar con mayor **claridad** los tejidos [58]. Así, se obtienen las imagenes por Endoscopia de contacto con imagen de banda estrecha. Las imágenes obtenidas para crear la base de datos, fueron hechas con videos de *framerate* de 30 y aumento de x60, eligiendo los segmentos de manera **manual** y también los intervalos donde se observan las vesículas. Tras esto, se elegían 1 *frame* cada 3 *frames* y se obtenían imágenes de tamaño 1008 x 1280 píxeles que se almacenaban en el historial clínico de cada paciente. Los pacientes eran clasificados con el resultado de una biopsia para diagnosticar correctamente a cada paciente. Tras obtener las imágenes iniciales, se les aplicaban varios filtros para el procesado de la imagen y se obtenían imágenes CE + NBI de la laringe, clasificadas para cada paciente [9].

Para aplicar la técnica de extracción de características, se obtienen 5 indicadores para distinguir los diferentes patrones vasculares.

4.2.1. Indicadores basados en dirección

Indicadores basados en la dirección: se obtienen 2 indicadores de dirección: Dirección del gradiente del histograma (HGD) y la media de rotación de la imagen (RIA). El **gradiente** es el cambio direccional en **intensidad**, los cuales son más consistentes en objetos rectos que en curvos. Para obtener el HGD de cada imagen, primero se convierte la imagen en una imagen de 128x64 pixel. Ahora se calcula el gradiente de la imagen, haciéndolo para cada pixel siguiendo las fórmulas del gradiente

$$G_x(r,c) = I(r,c+1) - I(r,c-1) \;\; y \;\; G_y(r,c) = I(r-1,c) - I(r+1,c)$$

donde r es la fila, c es la columna y I (r, c) es el valor del pixel [r, c]. Este cálculo se realiza obteniendo el gradiente de la imagen. En la figura 4.3 se muestra un ejemplo de gradiente sobre una imagen de una B mayúscula con fondo negro. La imagen se divide en bloques de 8x8 y a

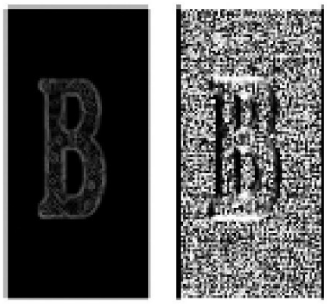

Figura 4.3: Gradiente de una imagen ejemplo [8].

cada bloque se le calcula un histograma de 9 puntos, con un rango de 0º a 180º. Esto se realiza para todos los bloques y se vuelven a crear bloques de 4 bloques de tamaño 2x2 a los que se les repite el paso anterior, pero con saltos de 8 píxeles y se crean vectores de 36 valores, los cuales se normalizan y se repite para toda la imagen, terminando con un HGD con 3780 características (7x15x36) [32]. La RIA consiste en el cálculo de la media de las filas para diferentes ángulos de rotación de la imagen. Esta media será pico cuando los vasos sean menos paralelos y tendrá comportamiento plano si son curvos. Para calcularlo la imagen se rotó de 0 a 360 grados en

pasos de 45 grados y calculamos la media como:

$$s^{\Theta}_{row}(x) = \frac{1}{N}\sum_{y=1}^{N} I_F(x,y)$$

siendo s la media del vector de filas para el ángulo θ, I el valor del píxel y N el número de filas. La RIA final será la concatenación de cada s.

4.2.2. Indicadores de curvatura

Indicadores de curvatura: se obtienen 3 indicadores, indicador de ángulo (ANG), de distancia (DIS) y curvatura. ANG y DIS se obtienen computando la **distancia** y el **ángulo** entre un punto de referencia definido (A en la figura 3) y cada pixel del tejido (C(x, y) en la figura 3), con el cálculo de la distancia euclídea

$$d(A,C) = \sqrt{(x_A - x_c)^2 + (y_A - y_C)^2}$$

y para el ángulo se usa un segundo punto de referencia (B) y se obtuvo el ángulo entre los vectores AC y AB con el arco tangente.

$$\Theta(A,C) = \arctan(\frac{\|\overrightarrow{AB} \times \overrightarrow{AC}\|}{\overrightarrow{AB}\overrightarrow{AC}})$$

Se obtiene DIS y ANG para cada segmento del vaso en cada imagen. El indicador CUR se obtiene calculando el nivel de curvatura del vaso, usando un método de aproximación de tangentes con un segmento digital recto de tamaño 2. Para cada punto pi, se calcula un DSS de longitud 2, centrado en pi. Este SDR es una aproximación de la tangente de pi, cuya longitud $li \approx ri$, que es el radio del círculo oscilante. Con esto calculamos un radio interno:

$$I_i = [(l_i - \frac{1}{2})^2 - \frac{1}{4}]$$

y un radio exterior:

$$O_i = [(l_i + \frac{1}{2})^2 - \frac{1}{4}]$$

y para calcular la curvatura de pi, usamos la estimación: [59]

$$E_i = \frac{2}{(I_i + O_i)}$$

El indicador final es la concatenación de las curvaturas de cada segmento del vaso. En la figura 4.4 se muestra gráficamente como calcular la curvatura. En la figura 4.5 está representado un

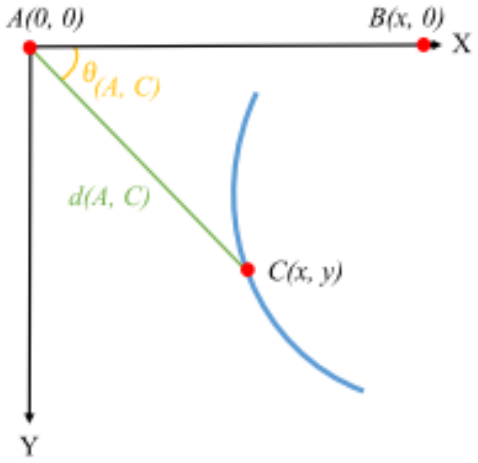

Figura 4.4: Construcción de los indicadores ANG y DIS.

ejemplo de cada indicador para una imagen ejemplo.

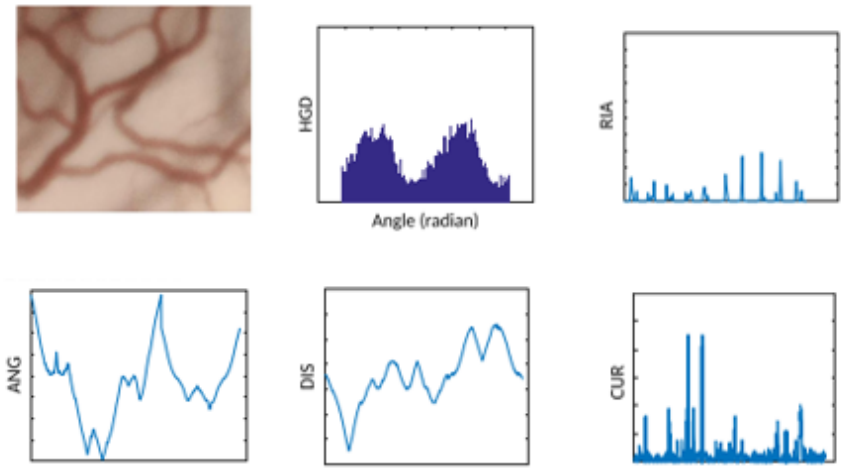

Figura 4.5: Indicadores HGD, RIA, ANG, DIS y CUR de la imagen [9].

4.2.3. Características

Tras obtener los indicadores de cada imagen, cuya forma está representada en la figura 5, se calculan 24 variables para representar a la imagen, a través de diferentes características obtenidas para cada indicador. Para HGD, se obtienen F1, F2 y F3 a partir de la energía total y

el valor mínimo de HGD y la diferencia entre el valor mínimo y máximo de HDG y F4 que es la suma de la energía concentrada en los picos significativos en HDG.

$$F_1 = \sum_{g=1}^{N} HGD^2(g)$$

$$F_2 = \min_{g}[s_m]$$

$$F_3 = \max_{g}[s_m]$$

$$F_4 = \frac{\sum_1^{n_p}[\sum_{on_i}^{off_i} HGDP_i^2]}{F_1}$$

Para RIA se obtienen F5 y F6 con la energía total y con el número de picos significativos, F7 como la suma de la energía concentrada en los picos significativos, y F8 como la suma de la ratio en amplitud y longitud de cada pico de RIA.

$$F_5 = \sum_{g=1}^{N} RIA^2(g)$$

$$F_6 = PEAKS[RIA]$$

$$F_7 = \frac{\frac{1}{n_p}\sum_1^{n_p}[\sum_{on_i}^{off_i} RIA^2(g)]}{F_5}$$

$$F_8 = \frac{1}{n_p}\sum_1^{np}[\frac{Amplitude(RIAP_i)}{Width(RIAP_i)}]$$

Para ANG se obtienen F9 a F12 usando el cambio de signo en ANG. Siendo M el número de vasos identificados en la imagen, para cada segmento m, se realiza la derivada de ANG y se obtiene en número de cambios de signos sm de cada segmento m y se obtienen las características con la media, el total de números de cambios de signo, el máximo y la mediana de sm. Además, se obtiene el error em de un polinomio de tercer grado para cada segmento del vaso y se obtiene la media y la mediana para tener F13 y F14.

$$F_9 = \frac{1}{M}\sum_{m=1}^{M} s_m$$

$$F_{10} = \sum_{m=1}^{M} s_m$$

$$F_{11} = \max_m s_m$$

$$F_{12} = median[s_m]$$

$$F_{13} = \frac{1}{M}\sum_{m=1}^{M} e_m$$

$$F_{14} = median[e_m]$$

Para DIS, se obtienen las características F15 a F20 con el mismo método que el usado en ANG respectivamente.

Para CUR se obtienen 4 características. F21, F22 y F23 son la energía total, la varianza y el número de picos significativos. F24 se calcula como el número de picos significativos por la amplitud de cada pico.

$$F_{21} = \sum_{g=1}^{n} CUR^2(g)$$

$$F_{22} = Variance[CUR]$$

$$F_{23} = Peaks[CUR]$$

$$F_{24} = n_p \times \sum_{1}^{n_p} Amplitude(CUR)$$

Las 2 últimas características se obtienen aplicando el método del esfuerzo del ciclista, que consiste en representar una imagen como una superficie **añosa**, con diferentes caminos entre el inicio y el final que cruza un ciclista como si fuese un Tour. De este trayecto se obtienen las características de **energía** y **potencia**. [60] Las imágenes obtenidas por CE-NBI son imágenes en 2D con valores para cada pixel y si representamos el valor de cada pixel como el valor en el eje z de un plano 3D, se crea un camino montañoso al que le creamos caminos para definir el tour del ciclista. El esfuerzo que tiene que hacer el ciclista para completar el trayecto se puede evaluar como la energía y la potencia del ciclista. Para calcular la energía y la potencia media de la imagen, se crean un número n de trayectorias. La energía y Potencia están definidas como:

$$P = F_T v \;\; y \;\; E = F_T vt$$

siendo v la velocidad, t el tiempo y FT la suma de las fuerzas que afectan a un ciclista: fuerza de gravedad (FG), fuerza de resistencia al aire (FA) y fuerza de resistencia al giro (FR). Con la potencia y la energía se obtienen las características de la imagen. Cada imagen es pre-procesada con un filtro de **Mediana** con un tamaño de 5x5 y se crean trayectorias rectas aleatoriamente de tamaño mínimo de 50 píxeles en la imagen, guardando la intensidad de cada píxel en la

trayectoria como un vector. Se crean vectores de trayectoria TPk(i) para N trayectorias. Ahora se calcula la energía y potencia de cada vector, dividiéndolo en secciones **no superpuestas** de tamaño 15 y con la fuerza y potencia de todos los vectores, se obtienen los parámetros de potencia y energía de la imagen. Así, obtenemos las características F25 (potencia) y F26 (energía). En la figura 4.6 vemos 2 ejemplos de transformación en etapas de 2 imágenes de CE-NBI y la figura 4.7 muestra las etapas seguidas para obtener de cada imagen las caracteríticas F25 y F26.

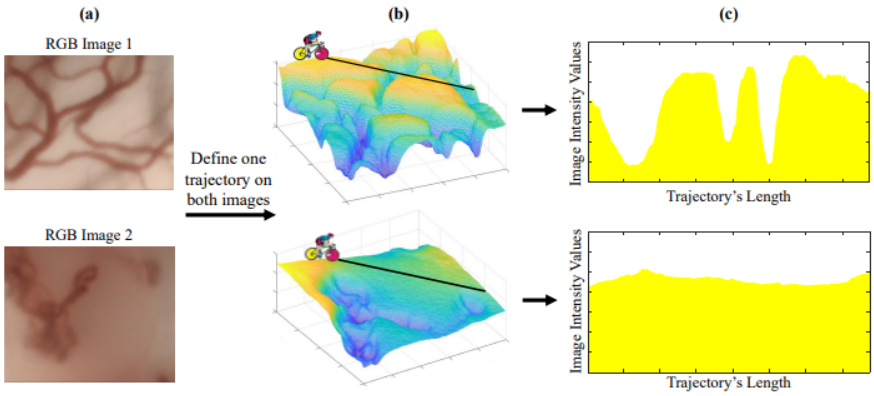

Figura 4.6: (a) imagen CE-NBI, (b) representación 3D de la imagen, (c) perfil de la etapa. [10].

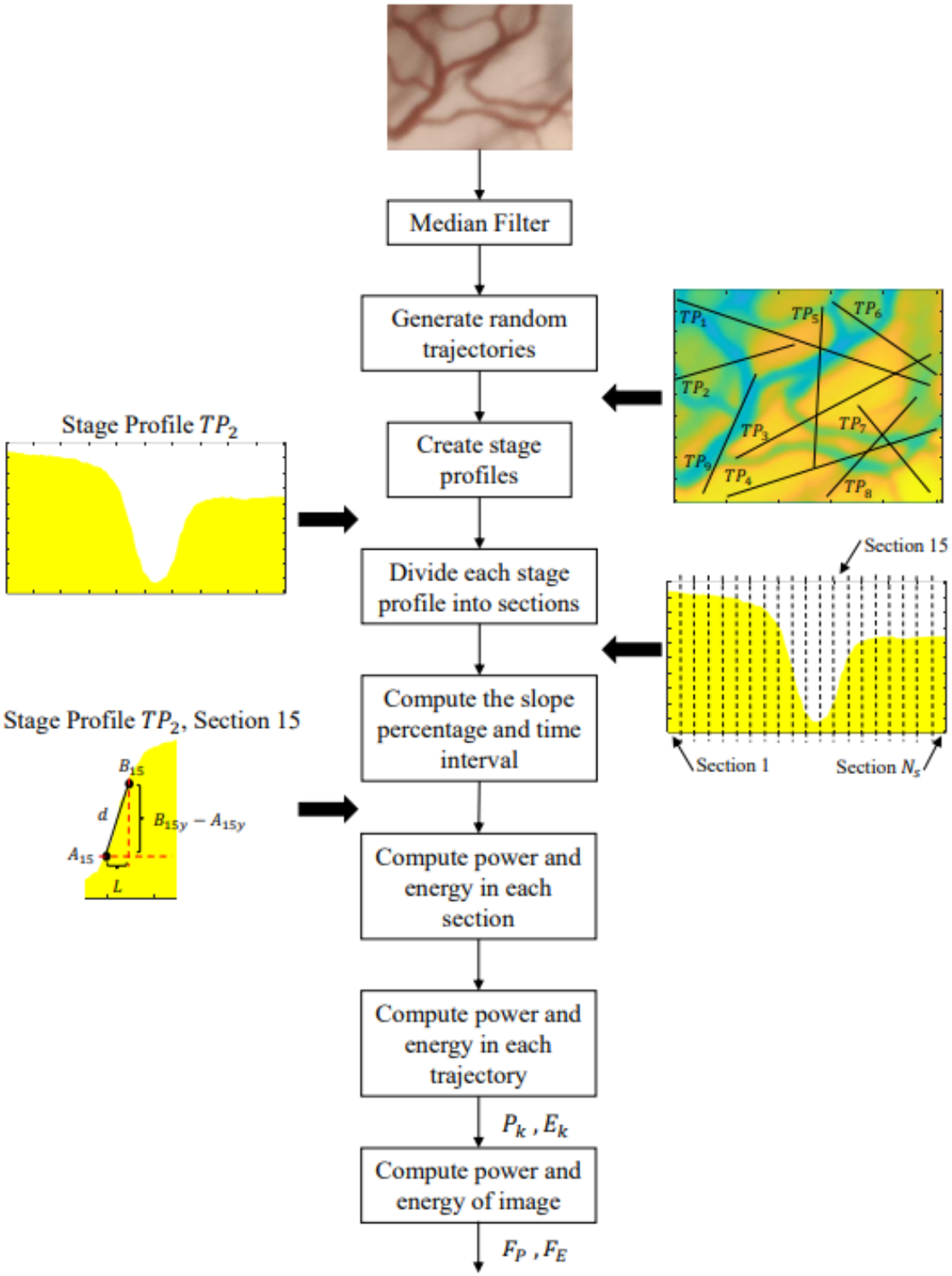

Figure 2. Feature extraction flowchart.

Figura 4.7: Esquema de extracción de Potencia y energía de la imagen.

En la figura 4.8 se puede observar la distribución de los valores contenidos en cada atributo de la base de datos. La variable F_27 es la salida del sistema, formada solo por valores 0 para representar tumores benignos y 1 para representar tumores malignos. El resto de variables son las obtenidas por los métodos anteriormente explicados. La distribución de cada variable es diferente, teniendo variables distribuidas como normales, como es el caso de F_2, pero lo más representativo de los datos es la diferencia de rangos entre las diferentes variables, observando que la variable F9 tiene un rango de valores [0.000 a 0.005], mientras que la variable F25 tiene un rango de [0, 25000] o la variable F26 con rango de [0, 7000], por lo que para el funcionamiento correcto de algunos algoritmos será recomendable hacer una normalización o estandarización de los datos. No presentan valores extremos que sean influyentes ninguna de las variables, pues están dentro de un rango aceptable para no afectar a la distribución media de los datos.

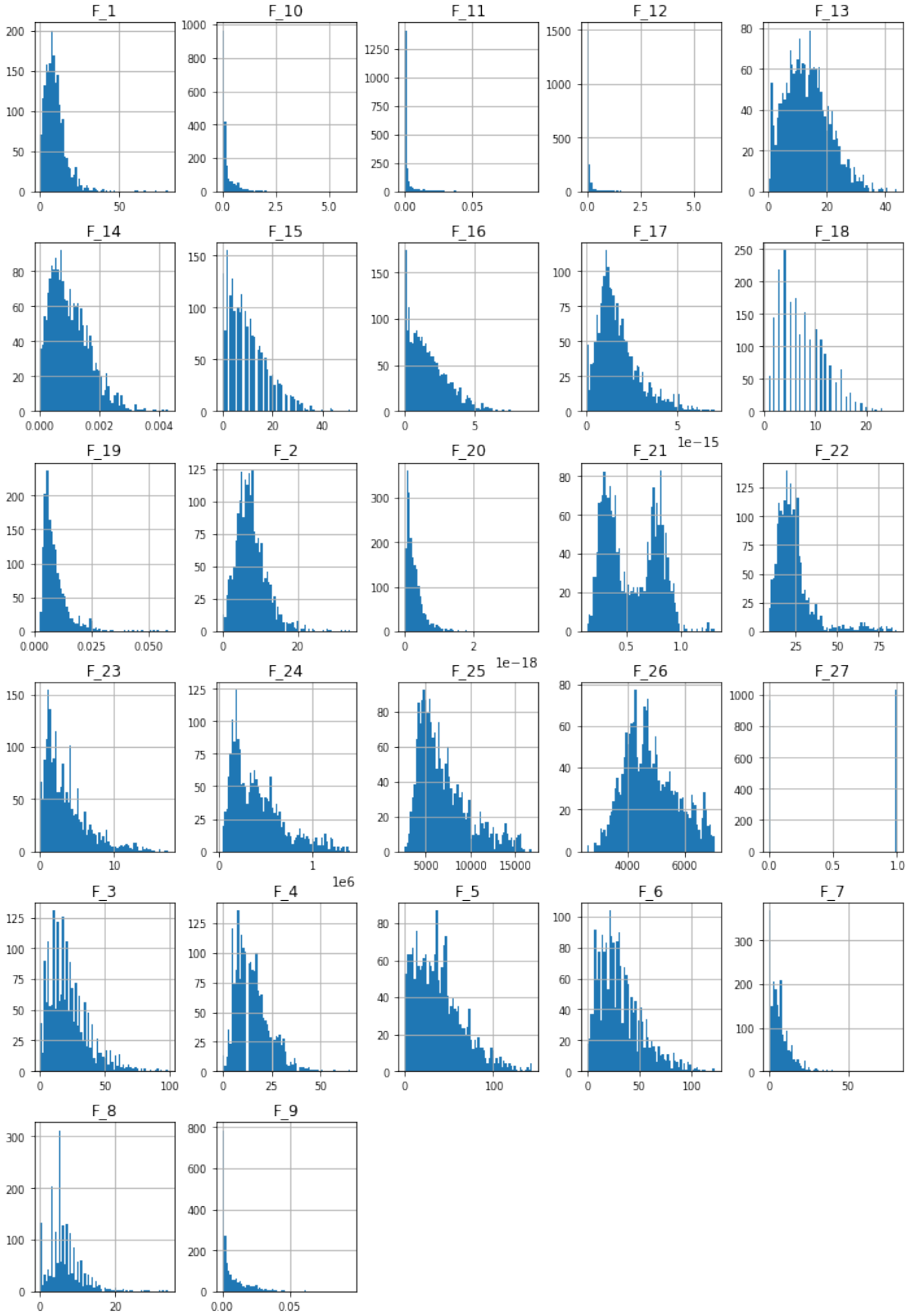

Figura 4.8: Gráficas de cada atributo de la base de datos.

Capítulo 5

Experimentos y resultados

En este capítulo se presentan los hiperparámetros usados para los algoritmos del proyecto, la herramienta utilizada para el desarrollo y se muestran los resultados de los clasificadores individuales y los sistemas de fusión de clasificadores con las 2 bases de datos del TFG.

5.1. Evaluación

La primera fase del trabajo es la creación de un modelo sencillo de fusión de clasificadores. Para ello, usaremos el set de datos disponible en *keras* llamado mnist. Este set contiene 60.000 imágenes de números escritos a mano del 0 al 9. Este set de datos lo dividimos en grupo de entrenamiento, con 50.000 imágenes; y grupo de test, con 10.000 imágenes. Para poder trabajar con los datos, se transforman en*arrays* de 1 dimensión usando la función *resize* de la librería cv2, transformando cada imagen en un array de 49 valores (49 variables). Ahora entrenamos los clasificadores y los 3 mejores se usarán: KNN, cuyos valores óptimos son k=50 y se usa método *accuracy* para puntuar; RF, con árboles de profundidad máxima 6 y criterio de gini; y SVM, usando un índice de tolerancia de 0.000001 y C= 0.8. Los parámetros se han decidido rápidamente y con pocas pruebas porque entrenar el modelo necesita mucho tiempo con Google Collab y no se tienen recursos superiores. Después de entrenar los clasificadores, la salida de cada clasificador se une en la entrada al método de votación, el cual ha sido creado a mano.

Después de trabajar con la primera base de datos y obtener los primeros resultados y entender cómo funcionan de manera básica los métodos de fusión de clasificadores, se procede a crear y entrenar los clasificadores y métodos más específicos de fusión con el segundo set de datos. Los datos son un archivo csv con 1995 imágenes transformadas en 26 variables y con

una variable de salida de valor 0 o 1. Todos los datos son números reales. Los datos se dividen en grupo de entrenamiento y grupo de test con la función train_test_split de sklearn, con una proporción del 80 % en el grupo de entrenamiento y 20 % en el grupo de test. Los datos son entrenados en los diferentes clasificadores 3 veces para cada modelo, pues se usan primero los datos *raw*, después los datos normalizados y finalmente los datos usando las 5 mejores variables marcadas por el método de chi cuadrado, que son las variables 23, 24, 25, 5 y 21. Para buscar los parámetros óptimos de cada clasificador se ha utilizado la función GridSearchCV de sklearn. Los parámetros obtenidos para cada clasificador son:

- PR: se busca el grado óptimo del clasificador entre grado del polinomio 1 y 6. El mejor grado encontrado es 2 para los 3 métodos de transformación de datos.
- LR: se busca nivel de tolerancia entre [1,0.1,0.01,0.001,0.0001,0.00001] y el coeficiente C en el rango [0,1] con saltos de 0.1. Se usa tol de 0.1 y C 0.5 para los datos *raw*, y tol 0.1 y C 1 para los datos normalizados y chi2.
- KNN: se busca el k óptimo entre 2 y 100 para el cual, el modelo no sobre ajusta ni generaliza. Se usará k de 80 para los datos *raw* y chi2 y k de 4 para los datos normalizados.
- DT: se busca la profundidad óptima para los árboles entre 2 y 15. Se usa profundidad de 14 para los datos *raw*, 12 para los datos con chi2 y 7 para los datos normalizados.
- RF: se busca la profundidad óptima para los árboles entre 2 y 15. Se usa profundidad de 14 para los datos *raw*, profundidad 11 para los datos con chi2 y 10 para los datos normalizados.
- SVM: se busca valor de tolerancia entre [1,0.1,0.01,0.001,0.0001,0.00001] y correlación en el rango [0,1] con saltos de 0.1. Se usa tol de 0.01 y C de 0.5 para los datos *raw* y tol de 0.1 y C de 1 para los datos normalizados y con chi2.

Tras evaluar y entrenar los modelos, se construyen los modelos de fusión. El primer modelo usa los métodos de votación creando una lista de los valores predichos y eligiendo la opción más popular, usando un α de 0.8, dando un poco más de importancia a los modelos con mejor rendimiento. Se hará con 3 grupos de diferentes clasificadores, el primero con KNN, RF y SVM, el segundo con LR, DT y RF y el tercero con todos los clasificadores menos GMM. Este proceso se hace para los datos *raw*, los datos normalizados y los datos con chi2. Tras esto, se hacen los modelos con el método de fusión MLP, usando keras. Se crean redes con 2 capas ocultas, con 10 neuronas en la primera capa y 5 neuronas en la segunda capa. Se usan los mismos 3 grupos

que en los métodos de votación, es decir, grupo con KNN, RF y SVM, grupo con LR, DT y RF y grupo con todos los clasificadores menos GMM, que obtuvo resultados subóptimos con las 2 bases de datos. El resultado es dado como un vector de 2 dimensiones con solo 2 resultados posibles: [1,0] si la salida es 0; y [0,1], si la salida es 1. Esto se ha hecho porque con una salida de 1 dimensión, el modelo solo obtenía resultado 0.

5.2. Google colab

Google Colab es la plataforma *online* sobre la que se desarrollará el proyecto. Es un servicio gratuito de Google que sigue un esquema de Producto como Servicio (PaaS). PaaS es un modelo en la nube que nos brinda un servidor de aplicaciones, además de almacenamiento, servidores y redes para el desarrollo del proyecto [61].

El proyecto fue desarrollado en *Google Colab*, creando varios *Jupyter Notebooks* que siguen el lenguaje de programación *Python*. Esta es la funcionalidad que tiene *Google Colab*, qué es permitir la creación y desarrollo de códigos *Python*, utilizando su servicio en la nube para la ejecución de código y almacenamiento de datos. Permite la creación de celdas de *Código, Markdown, Raw y Heading*. Además, el código desarrollado se puede exportar a archivos py o ipynb para trabajar fuera del entorno. Dispone de todas las librerías de Anaconda para una adecuada programación. La versión de *Python* utilizada en este proyecto es Python 3.8 notebook.

5.3. Resultados de la base de datos de MNIST

Se cargan los datos y las funciones de *sklearn* y *keras*. Después de transformar los datos en un vector unidimensional, los datos se separan en *train* y *test* y se entrenan los clasificadores.

953	1	0	1	4	6	9	1	1	4
0	1125	1	3	0	1	3	0	2	0
18	29	857	33	1	7	12	29	40	6
3	3	16	882	4	31	8	14	38	11
0	12	2	0	794	3	36	17	2	116
15	17	2	33	7	740	31	13	15	19
16	7	1	2	13	9	902	3	3	2
0	24	18	1	14	3	7	922	4	35
13	22	8	37	18	32	22	19	772	31
9	17	3	14	44	2	13	51	3	853

Cuadro 5.1: Matriz de confusión KNN.

936	2	2	4	6	2	14	2	12	0
0	1112	4	3	0	2	4	0	9	1
29	19	859	30	24	2	21	16	32	0
10	5	39	846	7	16	7	21	41	18
2	4	3	1	782	4	50	9	20	107
26	21	7	75	20	634	51	10	23	25
39	13	5	0	30	11	842	4	13	1
2	20	39	4	5	0	3	896	10	49
15	32	16	55	53	23	17	16	712	35
8	11	7	18	62	6	13	50	19	815

Cuadro 5.2: Matriz de confusión RF.

937	1	2	8	7	12	8	1	4	0
0	1110	3	2	0	4	4	0	12	0
13	6	862	26	10	12	23	25	54	1
7	2	26	865	4	42	7	13	37	7
1	2	10	1	790	19	40	20	4	95
20	22	17	65	15	683	23	16	21	10
21	5	14	1	25	23	857	4	7	1
4	16	30	2	12	7	7	901	11	38
8	26	27	41	16	42	15	14	755	30
9	9	9	15	73	14	13	37	20	810

Cuadro 5.3: Matriz de confusión SVM.

956	1	0	5	4	3	7	1	3	0
0	1120	3	1	0	2	2	0	7	0
22	19	890	19	5	5	12	21	39	0
8	4	24	887	3	26	6	15	32	5
1	9	4	0	826	6	36	15	3	82
17	23	15	51	12	722	25	8	8	11
25	8	8	0	19	8	883	3	4	0
2	21	30	2	14	3	5	912	8	31
14	29	17	47	19	27	13	15	769	24
13	19	6	18	47	2	11	40	7	846

Cuadro 5.4: Matriz de confusión método de votación.

5.3.1. Discusión de la base de datos de MNIST

Los clasificadores de KNN, RF y SVM obtienen un rendimiento de 0.88, 0.843 y 0.85, respectivamente, siendo estos clasificadores los mejores para el conjunto de datos. El resto solo supera el 0.8 de accuracy el modelo de regresión logística. Procedemos a realizar el método de fusión por votación y obtenemos el mejor rendimiento, con un accuracy de 0.881.

La naturaleza de esta base de datos, afecta al rendimiento del resto de clasificadores pues no se puede ver como un sencillo problema lineal, sino que es un problema en el que cada variable es relevante para determinar la salida. Por ello todos los clasificadores lineales tienen problemas a la hora de clasificar números similares, como son el 9 y el 4. La necesidad de evaluar todos los parámetros y el hecho de que todas las variables están distribuidas en un rango de 0 y 1, es la razón por la que los mejores clasificadores terminan siendo SVM, RF y KNN.

El mejor método de fusión por votación también se consigue solo usando esos 3 clasificadores y se observa que el método de votación usado, voto por mayoría, hace una media del rendimiento de los clasificadores, nunca siendo el mejor clasificador en rendimiento. Lo que se logra con el uso del método de votación es crear un modelo que se ajusta bien a todas las clases puesto que junta las características de todos los clasificadores que une.

5.4. Resultados de la base de datos sobre extracción de características.

Se realiza la normalización y filtrado chi cuadrado a la base de datos. Se entrenan los 7 clasificadores y se obtiene sus rendimientos con el grupo de *test*. Tras esto, se crean los modelos por votación y el MLP con los 3 grupos definidos. Los datos no necesitan pre procesado pues todas las variables son numéricas, no hay datos perdidos o datos externos. Procedemos a clasificar los modelos con los datos *raw*, los datos normalizados y los datos filtrados con chi2.

	MLR	PR	LR	KNN	DT	RF	SVM
precision 0	0.95	0.94	0.93	0.78	0.95	0.98	0.78
precision 1	0.95	0.88	0.91	0.80	0.95	0.96	0.80
recall	0.95	0.91	0.92	0.79	0.95	0.97	0.79
f1-score	0.95	0.91	0.92	0.79	0.95	0.97	0.79

Cuadro 5.5: Tabla de *precission*, *recall* y f1 score de los clasificadores con los datos raw.

Obtenemos *accuracy* superior a 0.75 en todos los modelos para los datos *raw*. El modelo de RF solo falla en predecir 11 resultados de 399 en el set de *test*, siendo el mejor modelo para los datos *raw*.

Entrenamos el modelo con los datos filtrados por chi2. En general la *accuracy* disminuye entre un 0.01 y 0.15 puntos para cada modelo excepto KNN y SVM que no pierden ni ganan rendimiento. La *accuracy* obtenida de llevar a cabo el método de votación con todos los clasificadores menos GMM es de 0.92.

	MLR	PR	LR	KNN	DT	RF	SVM
precision 0	0.93	0.95	0.78	0.78	0.89	0.93	0.77
precision 1	0.84	0.91	0.80	0.80	0.91	0.96	0.80
recall	0.88	0.93	0.79	0.79	0.90	0.95	0.79
f1-score	0.88	0.93	0.79	0.79	0.90	0.95	0.79

Cuadro 5.6: Tabla de *precission*, *recall* y f1 score de los clasificadores con los datos filtrados con chi2.

Normalizamos los datos y entrenamos los clasificadores. Tras normalizar los datos, los modelos de KNN y SVM mejoran en un 20 %, el resto mejora entre 1 y 4 % y solo la regresión polinómica empeora un 5 %.

	MLR	PR	LR	KNN	DT	RF	SVM
precision 0	0.95	0.89	0.95	0.97	0.95	0.98	0.97
precision 1	0.93	0.84	0.95	0.98	0.96	0.96	0.95
recall	0.94	0.86	0.95	0.98	0.96	0.97	0.96
f1-score	0.94	0.86	0.95	0.98	0.96	0.97	0.96

Cuadro 5.7: Tabla de *precission*, *recall* y f1 score de los clasificadores con los datos normalizados.

Ahora se proceden a crear los diferentes grupos de clasificadores para entrar en el sistema de fusión de clasificadores.

	Grupo 1	Grupo 2	Grupo 3
precision	0.79	0.96	0.95
recall	0.79	0.96	0.95
f1-score	0.79	0.96	0.95

Cuadro 5.8: Tabla de *precission*, *recall* y f1 score del método de votación para los datos *raw*.

	Grupo 1	Grupo 2	Grupo 3
precision 0	0.98	0.98	0.98
precision 1	0.96	0.96	0.96
recall	0.97	0.97	0.97
f1-score	0.97	0.97	0.97

Cuadro 5.9: Tabla de *precission*, *recall* y f1 score del método de votación para los datos normalizados.

Para los datos raw, la fusión de todos los clasificadores da un mal resultado puesto que los bajos rendimientos de algunos clasificadores afectan al rendimiento del método de votación. Para los datos agrupados y los datos normalizados se obtuvo un buen rendimiento de 0.97 de f-1 score como el mejor para todos los métodos con datos normalizados.

Ahora creamos el MLP para los datos *raw* y normalizados con los 3 diferentes grupos.

	Grupo 1	Grupo 2	Grupo 3
precision 0	0.98	1.00	0.99
precision 1	0.96	0.91	0.95
recall	0.97	0.94	0.97
f1-score	0.97	0.95	0.97

Cuadro 5.10: Tabla de *precission*, *recall* y f1 score del MLP para los datos *raw*.

	Grupo 1	Grupo 2	Grupo 3
precision 0	0.97	0.99	0.99
precision 1	0.96	0.95	0.95
recall	0.97	0.97	0.97
f1-score	0.97	0.97	0.97

Cuadro 5.11: Tabla de *precission*, *recall* y f1 score del MLP para los datos normalizados.

Para los datos *raw*, el primer grupo con un rendimiento de fallo de 10 predicciones entre los 399 y *accuracy* media de 0.97, para el segundo grupo se fallan 17 pero solo al predecir valor 0, se acertaron todos los 1's y para todos se obtuvo un rendimiento similar al primer grupo.

Los datos normalizados no dan una gran mejora del rendimiento, pues solo consiguen un 0.97 de f1-score, aunque consigue fallar solo 2 salidas de tipo 0, para el grupo 2 de clasificadores.

5.4.1. Discusión base de datos extracción de características

- **Datos sin modificar**: Los datos de esta base tienen una distribución ciertamente lineal y siguen distribuciones normalizadas, dando un buen rendimiento a todos los algoritmos. Pero la existencia de los atributos F-25 y F-26 hace que SVM y KNN tengan un muy mal rendimiento, pues son 2 variables que tienen rangos de valores de miles y cientos de miles, por lo que su dominio de rango hizo que ninguno de estos 2 clasificadores pudiera analizar correctamente el resto de atributos.
- **Datos chi cuadrado**: No se ha procedido a usar los clasificadores con chi cuadrado puesto que su rendimiento era muy bajo con respecto al resto de métodos, reduciendo los rendimientos de los algoritmos en un 5 %, puesto que las 26 variables son necesarias para un análisis correcto de la base de datos.

- **Datos normalizados**: Al equilibrar el rango de las variables, todos los clasificadores mejoran. Esta mejora general afecta a los métodos de fusión, puesto que mejoran su rendimiento, pero al estar todos los clasificadores muy parecidos en rendimiento y solo existir 2 tipos de salidas, no se consigue una gran mejora al unir las características, puesto que todos los clasificadores terminan dando rendimientos altos y equilibrados.

Capítulo 6

Conclusiones y líneas futuras

En este capítulo se presentan las conclusiones obtenidas tras la realización del presente TFG y además se incluyen varias líneas futuras de trabajo identificadas.

6.1. Conclusiones

Los métodos de fusión son un nuevo tipo de clasificación con interesantes aplicaciones y con un buen futuro en los métodos de *Machine Learning*. Este método puede unir las ventajas de diferentes métodos de clasificación para obtener una mejor predicción final.

El método de votación opera mal con grupos de clasificadores con rendimientos bastante amplios, no mejorando el rendimiento del mejor clasificador del sistema, puesto que no se les ha otorgado diferentes pesos a los clasificadores según los rendimientos obtenidos. Este método mejora en rendimiento al mejor clasificador individual del primer set de datos, puesto que se ha usado un grupo concreto de clasificadores con rendimientos con pequeña diferencia entre ellos, menos de un 5 % de precisión de diferencia individual entre el grupo de 3 clasificadores, pero con rendimientos no superiores al 90dándonos el sistema de votación una mejora ligera en la precisión. El rendimiento del método es malo en el segundo set de datos por la integración de clasificadores con altas diferencias de precisión, el grupo con todos los clasificadores tiene una diferencia de hasta el 20 % en precisión y en el grupo de un 10 % y tampoco mejora el rendimiento de los clasificadores con grupos de 3, pues la diferencia es de un 4 % pero en clasificadores con rendimiento superior al 90 %. Podemos decir que el método de votación es un buen método de fusión por la baja carga que requiere para funcionar, generando apenas un 5más en el gasto de recursos para funcionar como método de fusión, pero no obtiene mejora

de rendimiento con respecto a grupos de clasificadores con rendimientos superiores al 90 % o con grupos cuya diferencia de rendimiento es muy alta entre sí. Además, no es el mejor método en relación con la mejora de rendimiento obtenida, pues, aunque obtenga una mejora de rendimiento con respecto a usar un clasificador individual, la mejora apenas es del 1 %.

El método de MLP es un modelo bastante prometedor en los métodos de fusión de clasificadores, pues obtiene una gran mejora en rendimiento a la hora de predecir valor 0, precisión de 0,99 en el grupo 2; y la predicción del valor 1 es mejor que la mayoría de clasificadores, 0,96 en el grupo 1, obteniendo la mejor precisión para valor 0 entre todos los clasificadores y la segunda mejor precisión para el valor 1. Este suceso se debe a la unión de las cualidades diferentes de cada clasificador para predecir por sí mismo cada valor de salida, con el clasificador de RF siendo muy bueno para predecir 0 y el clasificador de KNN muy bueno para predecir 1. El modelo de MLP obtiene, en definitiva, una mejora para predecir los resultados al combinar los clasificadores y mejora el rendimiento de grupos de clasificadores muy amplios y con buen rendimiento, superior al 80 %. Esta mejora de rendimiento se podría aumentar al crear una red neuronal más compleja y con más capas, teniendo un campo de mejora bastante amplio en rendimiento obtenido. El problema que plantea el MLP, es que produce un aumento en el uso de recursos para llevar a cabo el método de fusión con MLP, en concreto un aumento en un 30 % de los recursos comparado a usar los clasificadores individualmente.

En definitiva, los métodos de fusión de clasificadores son un campo con muchas posibilidades y buen futuro a la hora de crear mejores modelos de clasificación, pero tienen el problema de que necesitan un aumento de los recursos usados para crear el modelo. Para valorar si se puede implementar un método de fusión de clasificadores se deberá estudiar si el aumento del coste que genera usar fusión de clasificadores es rentable con respecto a la mejora en rendimiento que obtenemos.

6.2. Líneas futuras

El proyecto se puede extrapolar para trabajar y mejorar diferentes proyectos que solo usan un clasificador tras el entrenamiento de los grupos de clasificadores y selección para utilizar todos los clasificadores y obtener mejoras en el rendimiento total del algoritmo de clasificación o regresión. Además, se puede trabajar con algoritmos de fusión más complejos, como los métodos bayesianos que trabajan sobre probabilidades a posteriori, o *fuzzy integrals* que es un método que trabaja sobre la salida de los clasificadores haciendo integrales definidas con los resultados obtenidos por los clasificadores entre los rangos de salidas posibles del sistema. Para

métodos más complejos se necesitarán mayores recursos por el aumento en la complejidad de los cálculos necesarios para crear el modelo.

Apéndice A

Código

Código desarrollado y publicado en *Google Colab*, accesible con el siguiente *link*:

https://colab.research.google.com/drive/1pCmDP4nIpY7HUHi3HHXBbVNKoeVQ2cb6?usp=sharing

Bibliografía

[1] Alexis Kevin De la Hoz Manotas, Ubaldo José Martínez-Palacio, and Fabio Enrique Mendoza-Palechor. Técnicas de ml en medicina cardiovascular. *Memorias*, 11(20):41–46, 2013.

[2] S Pértegas Díaz and S Pita Fernández. La distribución normal. *Cad Aten Primaria*, 8:268–274, 2001.

[3] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pages 986–996. Springer, 2003.

[4] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[5] Alexandre Kowalczyk. Support vector machines succinctly. *Syncfusion Inc*, 2017.

[6] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[7] Paula Ceccon, Guilherme Schardong, Simone Barbosa, Clarisse de Souza, and Hélio Lopes. Visual exploration of an ensemble of classifiers. *Computers Graphics*, 08 2019.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[9] Nazila Esmaeili, Alfredo Illanes, Axel Boese, Nikolaos Davaris, Christoph Arens, and Michael Friebe. Novel automated vessel pattern characterization of larynx contact endoscopic video images. *International journal of computer assisted radiology and surgery*, 14:1751–1761, 2019.

[10] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[11] Antonio Moreno, Eva Armengol, Javier Béjar Alonso, Luis Antonio Belanche Muñoz, Claudio Ulises Cortés García, Ricard Gavaldà Mestre, Juan Manuel Gimeno, Mario Martín Muñoz, and Miquel Sànchez-Marrè. Aprendizaje automático, 1994.

[12] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press, 2012.

[13] Piyush Jain, Sean CP Coogan, Sriram Ganapathi Subramanian, Mark Crowley, Steve Taylor, and Mike D Flannigan. A review of machine learning applications in wildfire science and management. *Environmental Reviews*, 28(4):478–505, 2020.

[14] Alexander L Fradkov. Early history of machine learning. *IFAC-PapersOnLine*, 53(2):1385–1390, 2020.

[15] F Rosenblatt. *The perceptron: a probabilistic model for information storage and organization in the brain*, volume 65. 1958.

[16] Chervonenkis A. Y. Vapnik V. N. On a perceptron class. *Automation and Remote Control*, 25:112–120, 1964.

[17] Yakubovich V.A. Machines that learn to recognize patterns. *Metodi Vichisleniy*, 2, 1963.

[18] Winston P. H. Learning structural descriptions from examples. 1970.

[19] Ito T. Fukushima K., Miyake S. eocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 5:826–834, 1983.

[20] FM Delpino, ÂK Costa, SR Farias, Alexandre Dias Porto Chiavegatto Filho, Ricardo Alexandre Arcêncio, and BP Nunes. Machine learning for predicting chronic diseases: a systematic review. *Public Health*, 205:14–25, 2022.

[21] Vishal Jain and Jyotir Moy Chatterjee. Machine learning with health care perspective. *Cham: Springer*, pages 1–415, 2020.

[22] Vega Marianella Álvarez, Mora Laura María Quirós, Mónica Valeria Cortés Badilla, et al. Inteligencia artificial y aprendizaje automático en medicina. *Revista médica sinergia*, 5(8):e557–e557, 2020.

[23] Adam Yala, Tal Schuster, Randy Miles, Regina Barzilay, and Constance Lehman. A deep learning model to triage screening mammograms: a simulation study. *Radiology*, 293(1):38–46, 2019.

[24] Alberto Prieto Sánchez. *Análisis y situación actual de las finanzas descentralizadas: un estudio empírico basado en técnicas de machine learning*. Universidad Pontificia, 2021.

[25] Muhammad Anees Khan, Kumail Abbas, Mazliham Mohd Su'ud, Anas A Salameh, Muhammad Mansoor Alam, Nida Aman, Mehreen Mehreen, Amin Jan, Nik Alif Amri Bin Nik Hashim, and Roslizawati Che Aziz. sn macro-economic data: Supervised learning techniques approach. *Sustainability*, 14(16):9964, 2022.

[26] Paul D Gader and Magdi A Mohamed. Multiple classifier fusion for handwritten word recognition. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 3, pages 2329–2334. IEEE, 1995.

[27] Fernando Huenupán Quinán. *Fusión de Múltiples Clasificadores en Verificación de Locutor*. Universidad de Chile, 2010.

[28] Dymitr Ruta and Bogdan Gabrys. An overview of classifier fusion methods. *Computing and Information systems*, 7(1):1–10, 2000.

[29] Michie Donald. Methodologies from machine learning in data analysis and software. *The Computer Journal*, 34(6):559–565, 1991.

[30] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

[31] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, 1(1):4–20, 2010.

[32] Arteaga Humberto Chaviano. Técnicas de aprendizaje supervisado y no supervisado para el aprendizaje automatizado de computadoras. In *Memorias del primer Congreso Interna-*

cional de Ciencias Pedagógicas: Por una educación integral, participativa e incluyente, pages 549–564. Instituto Superior Tecnológico Bolivariano, 2015.

[33] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons. b*, 4:51–62, 2017.

[34] Jesús Cáceres Tello and Servicios Informáticos. Reconocimiento de patrones y el aprendizaje no supervisado. *Universidad de Alcalá, Madrid*, 2007.

[35] Christian Vintimilla, Fabián Astudillo-Salinas, Erika Severeyn, Lorena Encalada, and Sara Wong. Agrupamiento de k-medias para estimación de insulino-resistencia en adultos mayores de cuenca. *Maskana*, 8:31–39, 2017.

[36] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. Selección dinámica de clasificadores para la estrategia uno-contra-uno: Evitando los clasificadores no competentes.

[37] Francisco Moreno-Seco, José M Inesta, Pedro J Ponce De León, and Luisa Micó. Comparison of classifier fusion methods for classification in pattern recognition tasks. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006. Proceedings*, pages 705–713. Springer, 2006.

[38] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE transactions on pattern analysis and machine intelligence*, 16(1):66–75, 1994.

[39] S Theodoridis et al. Pattern recognition, clustering: Basic concepts, 1999.

[40] James C Bezdek, James Keller, Raghu Krisnapuram, and Nikhil Pal. *Fuzzy models and algorithms for pattern recognition and image processing*, volume 4. Springer Science & Business Media, 1999.

[41] George J Klir and Tina A Folger. *Fuzzy sets, uncertainty, and information*. Prentice-Hall, Inc., 1987.

[42] Catherine M Ricardo. *Bases de datos*. McGraw Hill Educación, 2009.

[43] Rupert Miller and David Siegmund. Maximally selected chi square statistics. *Biometrics*, pages 1011–1016, 1982.

[44] Eva Ostertagová. Modelling using polynomial regression. *Procedia Engineering*, 48:500–506, 2012.

[45] Todd G Nick and Kathleen M Campbell. Logistic regression. *Topics in biostatistics*, pages 273–301, 2007.

[46] Michael P LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008.

[47] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[48] James Thorn. Random forest explained, 2020.

[49] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[50] Yunus Santur, Mehmet Karaköse, and Erhan Akin. Random forest based diagnosis approach for rail fault inspection in railways. In *2016 National Conference on Electrical, Electronics and Biomedical Engineering (ELECO)*, pages 745–750. IEEE, 2016.

[51] III ASPECTOS COMBINATORIOS DE LOS ARREGLOS and DE HIPERPLANOS. Tres lecciones en combinatoria algebraica.

[52] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

[53] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[54] Rodrigo Salas. Redes neuronales artificiales. *Universidad de Valparaıso. Departamento de Computación*, 1(1):1–7, 2004.

[55] Yawen Xiao, Jun Wu, Zongli Lin, and Xiaodong Zhao. A deep learning-based multi-model ensemble method for cancer prediction. *Computer methods and programs in biomedicine*, 153:1–9, 2018.

[56] José Manuel Sánchez Muñoz. Análisis de calidad cartográfica mediante el estudio de la matriz de confusión. *Pensamiento matemático*, 6(2):9–26, 2016.

[57] Awadhesh Kumar Mishra, Ajith Nilakantan, Kavita Sahai, Rakesh Datta, and Ajay Malik. Contact endoscopy of mucosal lesions of oral cavity–preliminary experience. *medical journal armed forces india*, 70(3):257–263, 2014.

[58] Louis Michel Wong Kee Song, Douglas G Adler, Jason D Conway, David L Diehl, Francis A Farraye, Sergey V Kantsevoy, Richard Kwon, Petar Mamula, Betsy Rodriguez, Raj J Shah, et al. Narrow band imaging and multiband imaging. *Gastrointestinal endoscopy*, 67(4):581–589, 2008.

[59] Simon Hermann and Reinhard Klette. Multigrid analysis of curvature estimators. Technical report, CITR, The University of Auckland, New Zealand, 2003.

[60] Nazila Esmaeili, Axel Boese, Nikolaos Davaris, Christoph Arens, Nassir Navab, Michael Friebe, and Alfredo Illanes. Cyclist effort features: A novel technique for image texture characterization applied to larynx cancer classification in contact endoscopy—narrow band imaging. *Diagnostics*, 11(3):432, 2021.

[61] William Y Chang, Hosame Abu-Amara, Jessica Feng Sanford, William Y Chang, Hosame Abu-Amara, and Jessica Feng Sanford. Building and configuring enterprise cloud services 3, 2. *Transforming Enterprise Cloud Services*, pages 273–308, 2010.