UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diagnóstico de la diabetes mediante el uso de técnicas de aprendizaje automático

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Antonio Martínez Leal

Tutor: Carlos David Martínez Hinarejos

Curso 2020-2021

Resum

La diabetis mellitus és una malaltia crònica que cada vegada està més estesa, afectant actualment a un de cada onze adults. Aquest trastorn metabòlic està relacionat amb un gran nombre de complicacions que representen una càrrega pesada per al pacient i el sistema sanitari.

El present treball pretén realitzar proves amb diferents tècniques d'aprenentatge automàtic amb l'objectiu de poder dir si una persona pateix o no diabetis, estudiant i comparant els diferents resultats obtinguts.

En primer lloc, s'analitzaran les tècniques actuals més rellevants i es detallarà l'estat de la qüestió. Seguidament es procedirà a la creació de diversos classificadors mitjançant l'ús de conjunts de dades públiques, identificant els paràmetres més importants i com influeixen en el funcionament dels diferents models. Després de realitzar aquest estudi es modificaran aquests models amb la finalitat de millorar els seus resultats i, posteriorment, replicar el desenvolupament per a una xarxa neuronal artificial.

Una vegada analitzats els valors obtinguts i determinades les millors implementacions es plantejarà la viabilitat de l'ús d'aquestes tècniques per al diagnòstic de la malaltia i, d'aquesta manera, intentar reduir el temps de detecció d'aquesta.

Paraules clau: Diabetis mellitus, Aprenentatge automàtic, Xarxes neuronals, Conjunt de dades, Classificació de mostres, Intel·ligència artificial

Resumen

La diabetes mellitus es una enfermedad crónica que cada vez está más extendida, afectando actualmente a uno de cada once adultos. Este trastorno metabólico está relacionado con un gran número de complicaciones que representan una carga pesada para el paciente y el sistema sanitario.

El presente trabajo pretende realizar pruebas con diferentes técnicas de aprendizaje automático con el objetivo de poder decir si una persona padece o no diabetes, estudiando y comparando los diferentes resultados obtenidos.

En primer lugar, se analizarán las técnicas actuales más relevantes y se detallará el estado de la cuestión. Seguidamente se procederá a la creación de varios clasificadores mediante el uso de conjuntos de datos públicos, identificando los parámetros más importantes y cómo influyen en el funcionamiento de los diferentes modelos. Tras realizar este estudio se modificarán dichos modelos con el fin de mejorar sus resultados y, posteriormente, replicar el desarrollo para una red neuronal artificial.

Una vez analizados los valores obtenidos y determinadas las mejores implementaciones se planteará la viabilidad del uso de estas técnicas para el diagnóstico de la enfermedad y, de esta forma, intentar reducir el tiempo de detección de la misma.

Palabras clave: Diabetes mellitus, Aprendizaje automático, Redes neuronales, Conjunto de datos, Clasificación de muestras, Inteligencia Artificial

Abstract

Diabetes mellitus is a chronic disease that is becoming more and more widespread, currently affecting one in eleven adults. This metabolic disorder is related to a large number of complications that represent a heavy burden for the patient and the healthcare system.

This work aims to carry out tests with different machine learning techniques in order to be able to tell if a person suffers from diabetes or not, studying and comparing the different results obtained.

First, the most relevant current techniques will be analyzed and the state of the matter will be detailed. Next, we will proceed to the creation of several classifiers through the use of public data sets, identifying the most important parameters and how they influence the performance of the different models. After conducting this study, these models will be modified in order to improve their results and, subsequently, the previous development will be replicated with an artificial neural network.

Once the obtained values have been analyzed and the best implementations determined, the feasibility of using these techniques for the diagnosis of the disease will be considered in order to try to reduce the detection time of the same.

Key words: Diabetes mellitus, Machine learning, Neural networks, Data set, Classification of samples, Artificial intelligence

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ADDM Red de seguimiento de autismo y de discapacidades del desarrollo, del inglés *Autism and Developmental Disabilities Monitoring network*.

ANN Red neuronal artificial, del inglés *Artificial Neural Network*.

ASD Trastorno del espectro autista, del inglés *Autism Spectrum Disorder*.

DM1 *Diabetes Mellitus* tipo 1.

DM2 *Diabetes Mellitus* tipo 2.

DMG *Diabetes Mellitus* Gestacional.

EDT Estructura de Desglose del Trabajo.

GB Potenciación del gradiente, del inglés *Gradient Boosting*.

GNB Clasificador bayesiano ingenuo, del inglés *Gaussian Naive Bayes*.

GPA Glucosa Plasmática en Ayunas.

IDE Entorno de desarrollo integrado, del inglés *Integrated Development Environment*.

IMC Índice de Masa Corporal.

JSON Notación de objetos JavaScript, del ingles JavaScript *Object Notation*.

KDE Estimación de la densidad de *kernel*, del inglés *Kernel Density Estimation*.

KNN k vecinos más próximos, del inglés *k-Nearest Neighbors*.

PTGO Prueba de Tolerancia a la Glucosa Oral.

PTSD Trastorno de estrés postraumático, del inglés *Posttraumatic Stress Disorder*.

ReLU Unidad lineal rectificada, del inglés *Rectified Linear Unit*.

RF Bosques aleatorios, del inglés *Random Forest*.

RFE Eliminación recursiva de características, del inglés *Recursive Feature Elimination*.

RGPD Reglamento General de Protección de Datos.

SVC Clasificación con vectores de soporte, del inglés *Support Vector Classification*.

Glosario

árbol de decisión Modelo de predicción cuya estructura se asemeja a un árbol. Está formado por una raíz, considerada nodo inicial, y nodos internos, los cuales dan paso a nuevas ramas o nodos terminales, conocidos como hojas.

array Colección de elementos identificados por al menos un índice o clave. Se trata de una de las estructuras de datos más antigua y utilizada en el ámbito de la programación.

correlación Medida estadística que expresa hasta que punto dos variables están relacionadas linealmente.

Creative Commons Organización que desarrolla licencias que permiten a los usuarios utilizar obras protegidas por derecho de autor sin necesidad de solicitar permiso.

data set Colección de números o valores que se relacionan con un tema en particular. Cada fila corresponde a un registro y sus valores se agrupan en diferentes variables representados en columnas.

desviación típica Medida estadística utilizada para cuantificar la variación de un conjunto de datos numérico.

diabetes mellitus Enfermedad metabólica que se caracteriza por un nivel elevado de glucosa en sangre producido por falta o funcionamiento incorrecto de la insulina.

entorno de desarrollo integrado Aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

glucosa Fuente primaria de síntesis de energía de las células. Comúnmente conocida como "el azúcar en la sangre".

Índice de Masa Corporal Razón matemática que asocia la masa y la talla de un individuo. Utilizado para clasificar a las personas según su peso.

insulina Hormona producida por el páncreas que permite que la glucosa se utilice como fuente de energía.

kernel Función simétrica que cuenta con una área igual a uno bajo la curva de la función y cuyo valor, llamado densidad, no puede ser negativo.

matriz de confusión Herramienta que permite valorar cómo de bueno es un modelo de clasificación. En concreto, permite distinguir cuándo una clase es confundida por otra, poniendo en relación las predicciones realizadas con los resultados correctos que se deberían haber alcanzado.

Open Source *Software* publicado bajo una licencia de código abierto que permite ser utilizado, modificado y redistribuido por cualquier persona. Generalmente desarrollado de forma colaborativa.

outlier Valor atípico que es numéricamente distinto al resto, generalmente pertenecientes a una población diferente al resto de muestras.

overfitting Sobreajuste que se produce al cuando un modelo depende excesivamente de los datos de entrenamiento y no es capaz de generalizar los resultados a nuevos datos de prueba.

perceptrón Forma más simple de una red neuronal utilizable en la clasificación de problemas linealmente separables..

presión sanguínea Tensión ejercida por la sangre sobre las paredes de los vasos sanguíneos, la cual se puede distinguir entre presión venosa y, la más conocida, presión arterial. Está última está compuesta por dos componentes: la presión arterial diastólica y la presión arterial sistólica.

presión arterial sistólica Medida durante el latido del corazón, momento de máxima presión. También conocida como presión arterial alta.

presión arterial diastólica Medida durante el descanso entre dos latidos, momento de presión mínima. También conocida como presión arterial baja y, usualmente, la que mayor atención recibe.

pueblo pima Grupo indígena localizado en Arizona (Estados Unidos) y en Sonora y Chihuahua (México).

red neuronal artificial Modelo computacional formado por un conjunto de unidades, neuronas, que, tras someterse a diversas operaciones al pasar por unos niveles llamados capas, producen una salida.

smartwatch Reloj de pulsera que cuenta con más funcionalidades que los relojes convecionales como pueden ser la capacidad de acceder a internet, recibir llamadas, realizar pagos o monitorizar la salud del usuario.

valor nulo Término utilizado para hacer referencia a la nada, usado para indicar que se no apunta a un dato u objeto válido.

CAPÍTULO 1
Introducción

A lo largo de nuestra evolución como especie hemos tenido que confrontar diferentes problemas con el objetivo de alcanzar una vida más cómoda, sencilla y duradera. Dichos problemas, que han cambiando con el paso del tiempo, inicialmente procedían de necesidades básicas como encontrar refugio, calor o comida, dificultades que en la actualidad se dan por resueltas en gran parte del mundo. En consecuencia, nuestras preocupaciones giran entorno a casos más específicos que todavía escapan de nuestro alcance, como es el caso de la cura de una enfermedad concreta o la reducción de los costes y residuos en los procesos industriales.

Este trabajo parte de una variación del primer ejemplo donde el objetivo no es la cura, sino la detección de la misma. De esta forma, a lo largo del documento se va a proceder a estudiar la detección de la diabetes, analizando en primer lugar los métodos existentes para su diagnóstico con el fin de que puedan ser complementados por una amplia variedad de técnicas de aprendizaje automático.

1.1 Motivación

Cerca de 9 % de la población actual del planeta padece diabetes, es decir, unas 400 millones de personas. Se trata de un grave problema global que no preocupa únicamente por el número de casos existentes, el cual ya es sorprendentemente elevado, sino por seguir una tendencia al alza que amenaza con incluso llegar a doblarse [1].

Esta tendencia está presente a nivel nacional, donde el porcentaje de la población con diabetes ha pasado de representar un 4.09 % en 1993 hasta un 7.80 % en los últimos datos registrados de 2017 según el Ministerio de Sanidad, registros observables en la Figura 1.1. En otras palabras, se trata de una de las enfermedades no contagiosas más frecuentes del mundo [2].

Gracias a los últimos avances realizados en el campo, ahora estos pacientes cuentan con herramientas como FreeStyle Libre[1], un sensor que actualiza los datos de glucosa cada minuto, que permiten controlar la enfermedad de una forma más precisa. No obstante, como se verá en el Capítulo 2, los tests utilizados para el diagnóstico de la diabetes no han recibido cambios que les permitan ir a la par con el resto de los avances.

Comúnmente, el aumento de personas con diabetes es asociado con el crecimiento de las tasas de obesidad, la falta de deporte y la ausencia de una dieta equilibrada. No obstante, no siempre es así: un 13 % de las personas diabéticas en España son de tipo 1 (DM1), las cuales, generalmente, llegan a esta condición por un trastorno autoinmune

[1] https://www.freestylelibre.es/libre/

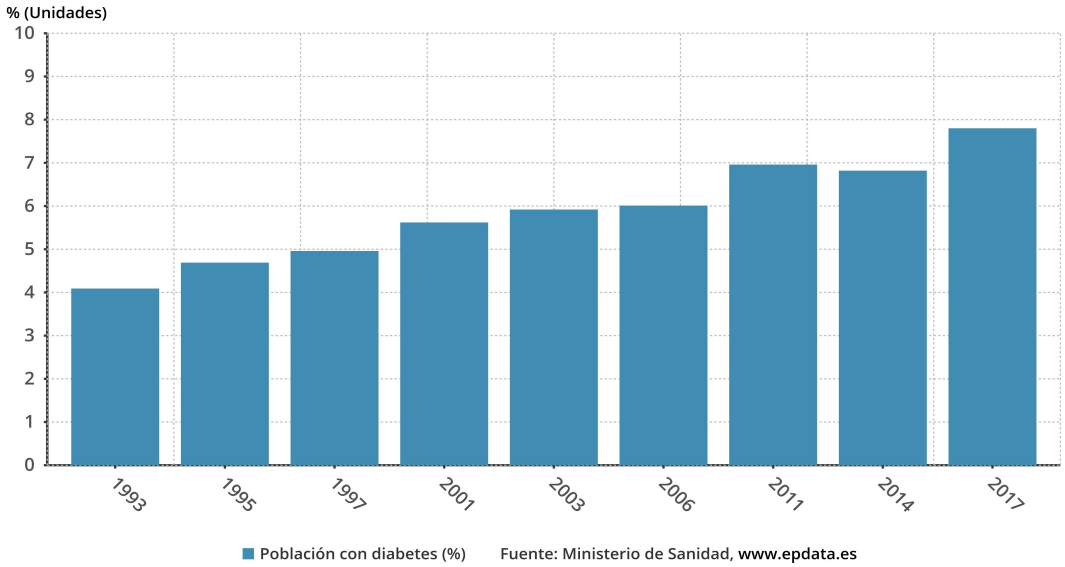

Figura 1.1: Evolución de la población con diabetes en España. Fuente: [3].

que no se puede identificar a simple vista y que contribuye a que muchos casos estén todavía sin diagnosticar [1].

Personalmente, este es el principal motivo por el que se está desarrollando este trabajo: soy una persona diabética cuyo caso ha sido diagnósticado hace menos de dos años y, pese a seguir un control bastante estricto de mi salud, había pasado desapercibido. Afortunadamente no llegué a presentar problemas serios, pero de no ser por mi insistencia al personal médico tal vez podría haberse dado una situación diferente.

Se estima que existen más de dos millones de personas diabéticas de diferentes tipos por diagnosticar en España, hecho que sin duda no recibe la atención suficiente en comparación a los costes que la diabetes supone anualmente al país: 17.630 millones de euros indirectos, 5.447 millones de euros directos y un gran número de consecuencias que recaen en la salud de los pacientes. Estos costes se ven incrementados por las detecciones tardías [1].

1.2 Objetivos

El objetivo principal de este trabajo es desarrollar una serie de modelos basados en apredizaje automático que permitan detectar si una persona padece diabetes o no, complementando los métodos existentes para la detección de la misma. Estas implementaciones partirán de modelos sencillos a los que se les aplicarán técnicas más complejas en función de los resultados obtenidos, con el fin de potenciar aquellos que cuenten con una mayor proyección.

De esta forma, al final del proyecto se dispondrá de diferentes alternativas, cada una con un nivel de complejidad diferente, lo que permitirá tener un amplio abanico de opciones. Tomando en cuenta estas últimas palabras, se propone como objetivo secundario determinar en qué situaciones puede ser mejor un modelo en comparación a los demás y si existen casos en los que, a pesar de que una implementación ofrezca peores resultados, determinar si destaca más en un ámbito en concreto.

1.3 Impacto esperado

Se espera que los resultados obtenidos al final del proyecto puedan servir para disipar dudas acerca de los siguientes temas:

- La importancia de utilizar métodos alternativos para la detección de enfermedades.
- El valor de los datos ya recolectados y cómo su reutilización puede aportar un gran número de avances que no se habían considerado.
- Los tipos de modelos que mejor se adaptan al caso para ser utilizados en proyectos de mayor envergadura.

En definitiva, este estudio pretende hacerse eco del potencial de diferentes técnicas de inteligencia artificial en el campo sanitario con la intención de que, tras haber podido considerar la viabilidad del proyecto en base a las conclusiones alcanzadas en este escrito, pueda ser retomado por equipos con una mayor experiencia en la utilización de las técnicas actuales y llegar, en el mejor de los casos, a crear un sistema capaz de incorporarse en el servicio de salud.

Se trata de una idea muy ambiciosa pero no lejana de la realidad: cada vez existen más instrumentos financieros dispuestos a destinar recursos a estas causas. El programa europeo Horizonte 2020[2] es uno de los proyectos de más renombre en la actualidad y uno de los retos sociales que pretende alcanzar, *Salud, cambio demográfico y bienestar*[3], está fuertemente entrelazado con las metas que se persiguen en este estudio. De la misma forma, *Salud y bienestar* también forma parte de los 17 Objetivos de Desarrollo Sostenible[4] de las Naciones Unidas, desafíos globales que pretenden alcanzar un mundo más equitativo, saludable y en sintonía con la naturaleza, muy presentes en los países líderes mundiales y que cada vez atraen más atención.

1.4 Estructura de la memoria

El contenido restante de este documento se estructura en ocho capítulos diferentes. En el Capítulo 2 se realiza una revisión y crítica del estado del arte. Seguidamente, en el Capítulo 3 se analiza de forma detallada el problema, estableciéndose un plan de trabajo adecuado. A continuación, en el Capítulo 4 se diseña una solución a partir del análisis anterior y en el Capítulo 5 se indica cómo se ha desarrollado. En el Capítulo 6 se detallan las pruebas y evaluaciones realizadas con el fin de comentar los resultados obtenidos. Posteriormente, en el Capítulo 7 se tendrán en cuenta todos los aspectos necesarios que harían falta para una posible implementación y de qué forma podría incluirse en el sistema actual. Por último, antes de cerrar el trabajo, en el Capítulo 8 y en el Capítulo 9 se extraen las conclusiones que servirán como base a los posibles trabajos futuros que deriven del escrito.

[2] https://ec.europa.eu/programmes/horizon2020/en/what-horizon-2020
[3] https://cordis.europa.eu/programme/id/H2020-EU.3.1./es
[4] https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/

CAPÍTULO 2
Estado de la cuestión

2.1 Tipos de diabetes

En el Capítulo 1 se han mencionado datos relevantes acerca de la diabetes, aportando estadísticas sobre su presencia y los motivos por los que no debe pasarse por alto. Sin embargo, todavía quedan dellates técnicos por ver, por lo que a continuación se realizará un breve resumen de en qué consiste la enfermedad exactamente.

En la actualidad se utiliza el término *diabetes mellitus* indiferentemente, pero es importante comprender que bajo este nombre se agrupan una cifra de enfermedades bastante notoria. Este hecho se debe al número de distintos transtornos metabólicos que se producen en presencia de concentraciones elevadas de glucosa en la sangre, cada uno con diferentes causas y consecuencias. Los tipos más comunes son la *diabetes mellitus* tipo 1 (DM1), la *diabetes mellitus* tipo 2 (DM2) y la *diabetes mellitus* gestacional (DMG) [4] [5].

La primera de ellas se produce por la destrucción autoinmune de las células β del páncreas, hecho que imposibilita la capacidad de generar insulina. Todavía se desconoce la causa exacta que provoca esta respuesta autoinmune; no obstante, existe cierto componente genético en la predisposición a desarrollarla. Suele detectarse en niños o en adultos jóvenes, los cuales, posteriormente, deberán seguir un tratamiento dependiente de insulina [6].

En cambio, las personas con DM2 cuentan con un organismo con demasiada resistencia a esta hormona. La aparición de esta variante viene motivada por la falta de actividad física y la acumulación de tejido adiposo, es decir, el sedentarismo y el exceso de peso son los principales culpables. Un cambio en el estilo de vida puede llegar a servir de único tratamiento pero, en los casos que no es así, se puede requerir de inyecciones de insulina [6].

Finalmente, la diabetes gestacional, que está presente en mujeres embarazadas, muestra similitudes con DM2 a pesar de no llegar a padecer la patología. Aparece debido al notable esfuerzo metabólico que supone un embarazo para la mujer y, en caso de no controlarse, puede llegar a afectar tanto a la madre como al feto, desarrollando ambos un mayor riesgo de padecer DM2 [6].

2.2 Detección de la diabetes

De igual forma que podemos encontrar distintos casos de diabetes, también existen diferentes tipos de tests específicos a utilizar según el caso. Estos precisan de un análisis de la sangre y se describen brevemente a continuación:

- **Prueba A1C**: Test que estudia los niveles de glucosa en sangre de los últimos tres meses.
- **Prueba de glucosa plasmática en ayunas (GPA)**: Mide el nivel de glucosa en sangre después de un ayuno de ocho horas.
- **Prueba de glucosa plasmática aleatoria**: Estudia el nivel de glucosa en sangre de un momento en concreto sin necesidad de ayunar.
- **Test de O'Sullivan**: Toma una muestra del valor de glucosa una hora después de haber consumido un líquido dulce.
- **Prueba de tolerancia a la glucosa oral (PTGO)**: Analiza numerosas medidas de glucosa en sangre a lo largo de un periodo de dos a tres horas. La primera de ellas se realiza en ayunas y las posteriores tras haber tomado glucosa líquida.

Los tres primeros tests predominan para la detección de la diabetes de tipos 1 y 2, mientras que los restantes se emplean para el diagnóstico de la diabetes gestacional. No obstante, cabe tener en cuenta que, debido a las similitudes que comparten estos trastornos, es bastante usual la necesidad de combinar las pruebas anteriores para obtener más detalles del caso bajo estudio [4].

2.3 Uso de aprendizaje automático en la detección de enfermedades

Uno de los principales problemas que la medicina afronta gira en torno a la identificación de los pacientes que puedan desarrollar mayor riesgo de contraer una enfermedad y, de esta forma, poder tomar medidas para dificultar, retrasar o incluso prevenir su aparición. Por lo tanto, las enfermedades que más pueden beneficiarse de una identificación temprana son aquellas que cuentan con una alta mortalidad y coste [7].

Atendiendo al contexto anterior, sobresalen estudios de distintas aproximaciones de aprendizaje automático que facilitan la detección de diferentes patologías. Por ejemplo, mediante la combinación de diversos modelos se puede llegar a determinar la presencia o ausencia de la enfermedad de las arterias coronarias, destacando el uso de selectores de características óptimas para determinar los datos más influyentes. Dichos modelos pueden ser utilizados para formar una aplicación móvil completamente funcional que no requiere de supervisión constante, sirviendo para complementar los informes médicos ya existentes y producir evaluaciones más detalladas [8].

Asimismo, también se pueden emplear algoritmos de aprendizaje automático para el diagnóstico de trastornos mentales como el estrés postraumático (PTSD), la esquizofrenia, la depresión, el autismo (ASD) o el trastorno bipolar. La elección del modelo depende de las características propias del caso, motivo por el que no se puede determinar una opción superior a las demás. No obstante, en este campo destaca la utilización de máquinas de vectores soporte (SVC), potenciación del gradiente (GB), clasificadores de bayesianos, k vecinos más cercanos (KNN) y *Random Forest* (RF) [9], siendo la implementación hecha por Maenner et al. [10] un ejemplo exitoso del último modelo y donde se consigue identificar casos de autismo con una precisión de 86.5 % a partir de palabras y frases obtenidas de actos de evaluación ejercidos por la red de seguimiento de autismo y de discapacidades del desarrollo (ADDM)[1].

[1] https://www.cdc.gov/ncbddd/autism/addm.html

Consecuentemente, los prometedores resultados obtenidos en el sector de la salud con la inclusión del aprendizaje automático han conseguido despertar también un gran interés económico. Llaman la atención empresas como Niramai[2], una *startup* que utiliza la inteligencia artificial para detectar el cáncer de mama de forma no invasiva y con menor coste que los métodos ya establecidos. Las dudas acerca de la rentabilidad de estos proyectos se disipan rápidamente en cuanto se conoce que Niramai ha recaudado en cuatro años siete millones de dólares en fondos y realizado más de 70 instalaciones en diferentes centros hospitalarios de India, abriendo las puertas a una tecnología que ya ha demostrado tener la capacidad de facilitar el seguimiento sanitario de las personas [11].

2.4 Crítica del estado de la cuestión

Tras analizar los métodos empleados para el diágnostico de la diabetes y cómo se utiliza el aprendizaje automático para la detección de enfermedades, corresponde realizar una crítica de la unión de ambos conceptos y principal tema del escrito: el diagnóstico de la diabetes mediante el uso de técnicas de aprendizaje automático.

Los enfermos de diabetes han podido contar con una gran variedad de innovaciones tecnológicas en los últimos años, como son los sistemas de seguimiento de glucosa o las bombas de insulina. No obstante, la tecnología empleada para la detección de la enfermedad no goza de la misma salud. Los tests PTGO, piedra angular de la detección de la diabetes, parten de un concepto que surgió hace 100 años y cuyas limitaciones de rendimiento ponen en duda su desempeño a la hora de identificar individuos de alto riesgo [12].

Este test, conjuntamente a los presentados anteriormente, requiere de un análisis de sangre que no se puede reemplazar de forma inmediata pero, sin duda alguna, se puede complementar con técnicas de aprendizaje automático con el fin de mejorar la precisión de los resultados obtenidos y facilitar la incorporación de nuevas tecnologías a la ciencia.

Afortunadamente, esta visión parece compartirse en el sector de la investigación, surgiendo, en consecuencia, un mayor número de estudios y proyectos relacionados con la materia, como es el caso de *Pima Indians Diabetes Database*[3], competición que tiene como objetivo determinar si un paciente tiene diabetes y que ha servido de inspiración para la elaboración de este trabajo.

Finalmente, cabe destacar que la identificación de la diabetes sin utilizar extracciones de sangre se trata de una idea ambiciosa pero no lejana: existen dispositivos *smartwatch* capaces de detectar múltiples condiciones médicas. Uno de los estudios más sorprendentes, llevado a cabo por la empresa de salud Cardiogram[4], es capaz de detectar diabetes, colesterol alto, presión sanguínea alta y apnea del sueño con unas precisiones del 84.51 %, 74.41 %, 80.86 % y 82.98 % respectivamente [13], datos realmente prometedores teniendo en cuenta que es un método no invasivo que utiliza un producto ya extendido y de fácil acceso económico en comparación a los costes que puede ocasionar una detección tardía.

[2] https://www.niramai.com/
[3] https://www.kaggle.com/uciml/pima-indians-diabetes-database/
[4] https://cardiogram.com/research/

CAPÍTULO 3
Análisis del problema

En este capítulo se procederá a establecer los requisitos del proyecto y se comprobará que no entran en conflicto con ninguna cuestión legal o ética. De este modo, se obtendrá una descripción completa del comportamiento del sistema antes de su propio desarrollo, hecho que nos permitirá diseñar una solución posteriormente.

Es importante remarcar que no se pretende crear una aplicación disponible para usuarios. El objetivo principal sigue siendo comparar cómo se adaptan los diferentes modelos de aprendizaje automático elegidos a los datos proporcionados y de qué manera se pueden obtener mejores resultados. De todas formas, aunque el número de casos de uso se verá drásticamente reducido, siempre conviene realizar un análisis previo que evite contratiempos inesperados y, más tarde, se mencionarán en el Capítulo 7 aquellas partes en las que no se ha podido entrar en detalle.

3.1 Especificación de requisitos

Los requisitos del proyecto se pueden clasificar como funcionales y no funcionales. Los primeros, como bien indica su nombre, definen funciones que el sistema debe cumplir, mientras que los no funcionales hacen referencia a características del sistema relacionadas con aspectos como el rendimiento, la disponibilidad y otros factores impuestos por el cliente.

En las siguientes tablas (Tablas 3.1 a 3.8) se indican los requisitos que el proyecto debe tener en cuenta.

3.1.1. Requisitos funcionales

Estos requisitos se presentan en las Tablas 3.1 a 3.4.

Identificador	RF-01
Nombre	Detección de categorías determinantes
Descripción	Una vez proporcionado el conjunto de datos se determinará las características que influyen en el entrenamiento y la evaluación de los modelos

Tabla 3.1: Requisito funcional RF-01.

Identificador	RF-02
Nombre	Resultados del modelo
Descripción	Tras entrenar y evaluar un modelo, el sistema debe proporcionar automáticamente los datos que permitan valorar su éxito

Tabla 3.2: Requisito funcional RF-02.

Identificador	RF-03
Nombre	Registro de pruebas realizadas
Descripción	El sistema almacenará descripción, resultados y fecha de las pruebas realizadas

Tabla 3.3: Requisito funcional RF-03.

Identificador	RF-04
Nombre	Generación de informes
Descripción	El sistema generará automáticamente un informe en formato pdf donde se puedan comparar los diferentes resultados obtenidos

Tabla 3.4: Requisito funcional RF-04.

3.1.2. Requisitos no funcionales

Estos requisitos se muestran en las Tablas 3.5 a 3.8.

Identificador	RNF-01
Nombre	Limpieza de conjuntos de datos
Descripción	El sistema debe eliminar los *outliers* presentes en los conjuntos de datos usados

Tabla 3.5: Requisito no funcional RNF-01.

Identificador	RNF-02
Nombre	Conjuntos de datos en csv
Descripción	Los conjuntos de datos que se traten utilizarán el formato csv

Tabla 3.6: Requisito no funcional RNF-02.

Identificador	RNF-03
Nombre	Implementación descentralizada
Descripción	La implementación se realizará por módulos de forma que se pueda modificar cada parte sin afectar a todo el conjunto

Tabla 3.7: Requisito no funcional RNF-03.

Identificador	RNF-04
Nombre	Código desarrollado en inglés
Descripción	Tanto el código como los comentarios del mismo se realizarán en lengua inglesa con el objetivo de llegar a un mayor número de personas

Tabla 3.8: Requisito no funcional RNF-04.

3.2 Análisis del marco legal y ético

Para poder realizar las pruebas pertinentes con los diferentes modelos de aprendizaje automático se debe tener en cuenta la normativa vigente del Reglamento General de Protección de Datos (RGPD), la cual afectará a la elección del *data set* que posibilitará los experimentos.

En primer lugar, deberá contar con datos anonimizados que imposibiliten identificar al paciente, hecho que adquiere importancia cuando se tiene en cuenta que estamos hablando de información relativa a la salud de una persona física [14]. De no respetarse, el paciente podría ser víctima de intentos de estafa en los que buscan beneficiarse de su condición de diabético, o de cualquier otro aspecto privado deducible de sus datos, con la idea de obtener su información financiera [15].

Además, debemos contar con la posibilidad de compartir y modificar la colección de datos que utilicemos en el proyecto para así poder aplicar técnicas que mejoren los resultados. Para ello, conviene elegir un conjunto de datos asociado a una licencia *Creative Commons* que permita realizar variaciones en la obra [16], preferiblemente una dedicada a dominio público como CC0[1].

Finalmente, a la hora de modificar los modelos bajo estudio para obtener mejores resultados se debe tener en mente la importancia de los falsos negativos: personas diabéticas que no son diagnosticadas como tales. En nuestro caso, no solo se debe buscar aquellos modelos que consigan un mayor número de aciertos, sino que también disminuyan el número de falsos negativos debido a las consecuencias que se pueden producir.

[1] https://creativecommons.org/share-your-work/public-domain/cc0

CAPÍTULO 4
Diseño de la solución

4.1 Solución propuesta

Python[1] es el lenguaje que mejor se adapta a los requisitos previos, ofreciendo además facilidades para una mayor legilibilidad del código. Por ello, el proyecto se realizará en torno a este lenguaje, el cual cuenta con una serie de librerías muy potentes dentro del campo bajo estudio y que nos evitarán tener que empezar de cero, pudiendo destinar más tiempo a probar diferentas alternativas como principal ventaja.

En cuanto a la estructura a seguir, se dividirá el proyecto en carpetas y cada implementación en un archivo diferente para que cualquier persona interesada por el estudio pueda comprender los diferentes pasos que se han seguido. La utilización de Jupyter Notebook[2] permitirá también distribuir el código en bloques de ejecución, los cuales pueden ser realmente útiles para acompañar el código desarrollado con gráficas y explicaciones del mismo, siendo este uno de los motivos por lo que es muy usado por los científicos de datos.

Todos estos diferentes paquetes de *software* pueden entrar en conflicto con la instalación local que tengamos en nuestro ordenador personal, por lo se optará por hacer uso de Anaconda[3]. De esta forma, dispondremos de un entorno virtual que no se verá afectado por cambios producidos en otros trabajos.

Finalmente, se deberá utilizar un entorno de desarrollo integrado que permita trabajar con todos estos servicios a la vez, siendo Pycharm[4] la opción elegida. Este entorno no solo cuenta con soporte para todos los componentes anteriormente elegidos, sino que además forma parte de paquetes de herramientas como GitHub Student Developer Pack[5] que ofrecen acceso gratuito a estudiantes y nos evita cualquier sobrecoste.

4.2 Plan de trabajo

El trabajo a ser ejecutado será dividido y organizado en tareas, las cuales serán agrupadas en una estructura de desglose de trabajo (EDT). Como se puede observar en la Figura 4.1, se han planificado cinco fases distintas, las cuales se detallarán brevemente a continuación.

[1] https://www.python.org/
[2] https://jupyter.org/
[3] https://www.anaconda.com/
[4] https://www.jetbrains.com/es-es/pycharm/
[5] https://education.github.com/pack

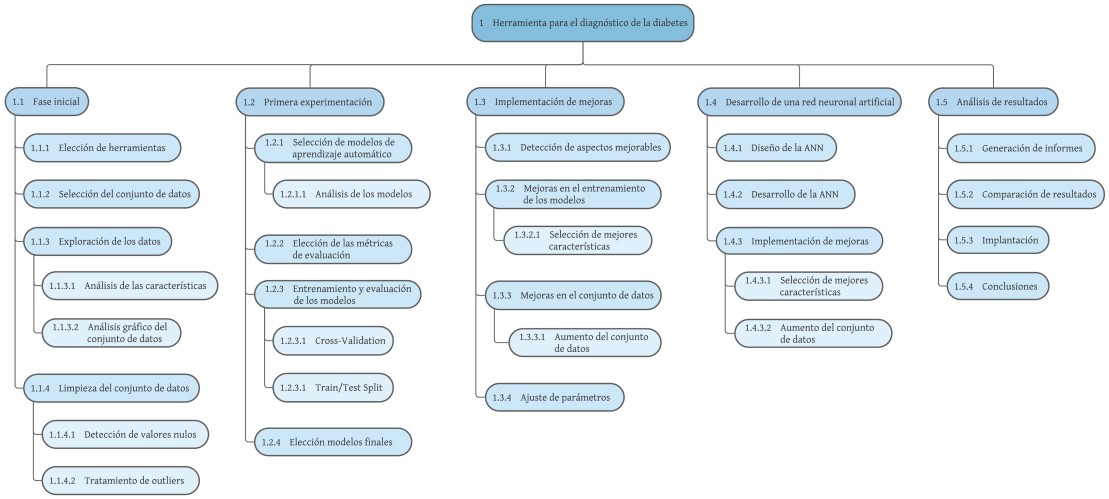

Figura 4.1: Vista de la EDT del proyecto. A continuación se incluirán imágenes ampliadas de las ramas que la componen para permitir la lectura de las mismas.

La **Fase inicial**, ilustrada en la Figura 4.2, gira entorno a la elección de las herramientas a utilizar, tarea que se abordará en la Sección 4.3, y la selección de un data set, materia que dará paso al inicio del desarrollo de la solución propuesta (Capítulo 5). Es de vital importancia atender debidamente estas tareas, puesto que su resultado marcará todo el estudio.

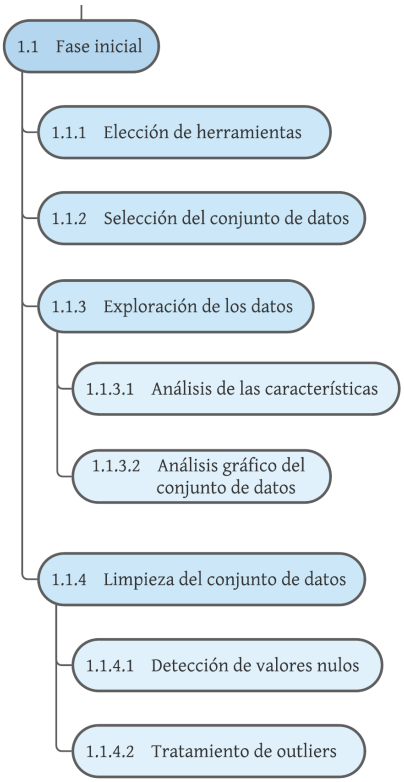

Figura 4.2: Rama de la EDT asociada a la **Fase inicial**.

Seguidamente nos encontramos con la Figura 4.3, la cual detalla la **Primera experimentación**, fase en la que se realizará un conjunto de pruebas con el objetivo de determinar aquellos modelos que mejor pueden adaptarse al caso bajo estudio. Para ello se

utilizarán dos métodos de entrenamiento distintos, así como diversos modelos de aprendizaje automático.

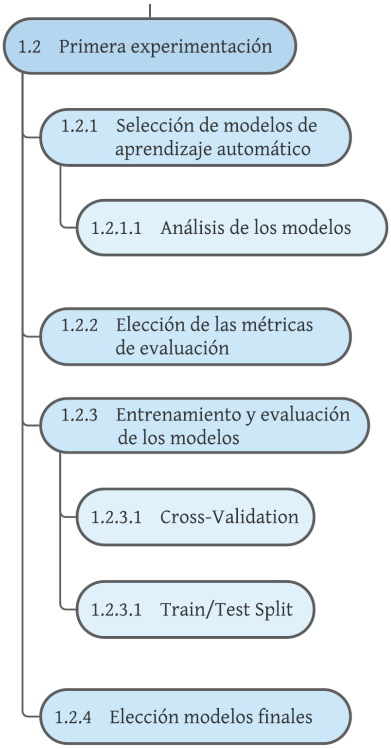

Figura 4.3: Rama de la EDT asociada a la **Primera experimentación**.

Posteriormente, siguiendo el esquema de la Figura 4.4, se detectarán los aspectos mejorables de los modelos seleccionados, con el objetivo de realizar una **Implementación de mejoras**. Dichas modificaciones se aplicarán tanto al conjunto de datos como a los modelos de entramiento y, una vez obtenidos los nuevos valores, se podrán comparar con los anteriores.

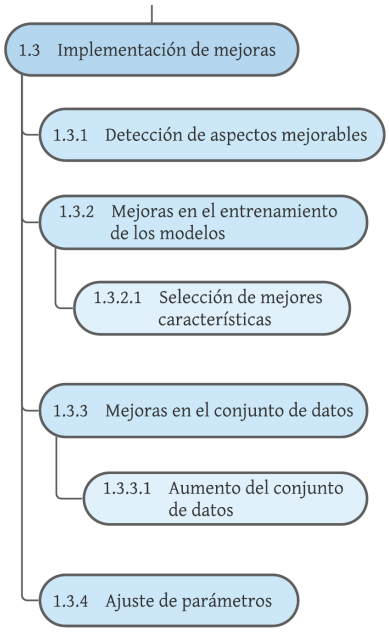

Figura 4.4: Rama de la EDT asociada a la **Implementación de mejoras**.

Una vez terminados de optimizar estos modelos se probará con una configuración más compleja: una ANN. **El desarrollo de una red neuronal artificial**, detallado en la Figura 4.5, requerirá de una implementación completamente nueva en la que se utilizarán otras librerías distintas a las que se les intentarán aplicar las mejoras anteriormente vistas.

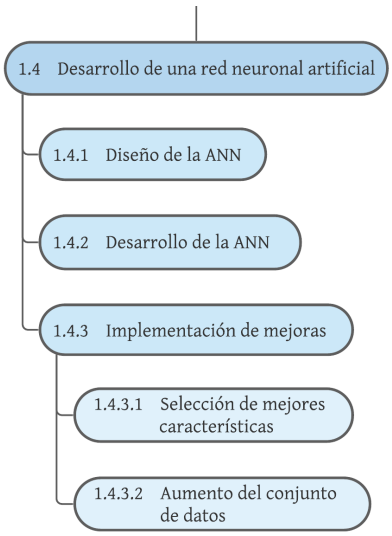

Figura 4.5: Rama de la EDT asociada al **Desarrollo de una red neuronal**.

Por último, en la Figura 4.6 se observan los pasos finales que cerrarán el proyecto, destacando la generación de los informes y la comparación de los resultados obtenidos con las diferentes configuraciones. En este **Análisis de resultados** se tendrá en cuenta también una hipotética implantación, la cual será continuada por una explicación de las conclusiones alcanzadas tras la finalización del desarrollo.

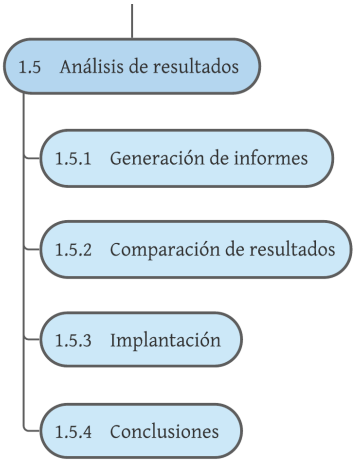

Figura 4.6: Rama de la EDT asociada al **Análisis de resultados**.

4.3 Herramientas utilizadas

La primera de las tareas vistas en el Plan de trabajo y que da inicio a la **Fase inicial** corresponde a la elección de las herramientas a utilizar. A continuación se exponen y describen aquellas que se prevé que serán utilizadas en el desarrollo del proyecto:

- **Python 3.9**: Lenguaje de programación interpretado, dinámico y multiparadigma que hace hincapié en la legibilidad del código.
- **Anaconda**: Herramienta que permite crear entornos virtuales que simplifican el despliegue y la administración de los paquetes de *software*.
- **Pycharm**: Entorno de desarrollo integrado (IDE) específico para Python con soporte a Anaconda integrado.
- **Archivos CSV**: Tipo de formato sencillo para representar datos en forma de tabla donde las columnas se separan por comas y las filas por saltos de línea.
- **Jupyter Notebook**: Entorno interactivo basado en la web para crear documentos divididos en bloques de ejecución. La parte de código desarrollado en Python se distribuirá en estos documentos.
- **NumPy**[6]: Biblioteca de Python que permite crear vectores y matrices grandes multidimensionales con los que realizar operaciones de alto nivel.
- **Pandas**[7]: Extensión de NumPy escrita para facilitar la manipulación y el análisis de datos.
- **Matplotlib**[8]: Biblioteca de Python para generar gráficos a partir de listas o *arrays*.
- **Seaborn**[9]: Biblioteca de Python basada en Matplotlib que proporciona una mayor variedad de patrones de visualización.
- **Scikit Learn**[10]: Biblioteca de Python para el uso de aprendizaje automático que cuenta con diversos algoritmos de clasificación, regresión y análisis de grupos.
- **PyTorch**[11]: Biblioteca de Python de aprendizaje automático que destaca en el desarrollo de redes neuronales. Da soporte para su ejecución en tarjetas gráficas, hecho que agiliza los procesos de cálculo.
- **GitHub**[12]: Sistema de control de versiones que permite alojar proyectos. Se trata de una plataforma muy utilizada para compartir proyectos *Open Source*.
- **Trello**[13]: *Software* de administración de proyectos con interfaz web que simula un tablón donde se pueden distribuir tareas.
- **Lucidchart**[14]: Herramienta de diagramación basada en la web que permite crear diagramas de flujo y esquemas como una EDT.
- **gnuplot**[15]: Programa de línea de comandos que puede generar gráficos de funciones bidimensionales y tridimensionales.

[6] https://numpy.org/
[7] https://pandas.pydata.org/
[8] https://matplotlib.org/
[9] https://seaborn.pydata.org/
[10] https://scikit-learn.org/
[11] https://pytorch.org/
[12] https://github.com/
[13] https://trello.com/
[14] https://www.lucidchart.com/
[15] http://www.gnuplot.info/

CAPÍTULO 5
Desarrollo de la solución propuesta

A continuación se detallará el desarrollo de la solución propuesta, parte presente a lo largo de toda la EDT. No obstante, aunque existen tareas como la elección de herramientas, la descripción de la implantación y la extracción de conclusiones que corresponden a capítulos diferentes, se procederá a seguir la misma estructura para mantener cierta coherencia organizativa.

Todo el código desarrollado y las soluciones obtenidas están disponibles en el siguiente repositorio público https://github.com/AntonioM15/TFG-Diabetes-Detection .

5.1 Fase inicial

Una vez se han determinado las herramientas a utilizar corresponde dar paso a la elección del data set sobre el que se realizarán todas las pruebas, motivo por el que debe estudiarse y tratarse concienzudamente antes de considerarlo un conjunto válido.

5.1.1. Selección del conjunto de datos

Tras hacer una búsqueda exhaustiva encontramos los siguientes conjuntos prometedores que cumplen con los requisitos establecidos:

- **Diabetes 130-US hospitals for years 1999-2008 Data Set**[1]: Conjunto que proporciona información sobre los pacientes diabéticos hospitalizados a lo largo de diez años, indicando los que han necesitado ser ingresados de nuevo.
- **Diabetes Data Set**[2]: Contiene los valores de glucosa registrados en pacientes diabéticos tomados de un dispositivo eléctronico y de mediciones convencionales.
- **Pima Indians Diabetes Database**[3]: *Data set* centrado en mujeres del pueblo pima de más de 21 años que, tras proporcionar una serie de datos médicos relacionados con la diabetes, indica aquellas que padecen la enfermedad y las que no.

Observamos que la tercera propuesta es la que mejor se adapta a nuestro caso y, aunque los dos primeros conjuntos aportan información interesante sobre características relacionadas a la diabetes, son registros de personas diabéticas exclusivamente, por lo que no se pueden clasificar las muestras de la forma que se desea. Por lo tanto, Pima Indians Diabetes Database será el *data set* que se utilice de ahora en adelante.

[1] https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008
[2] https://archive.ics.uci.edu/ml/datasets/diabetes
[3] https://www.kaggle.com/uciml/pima-indians-diabetes-database

5.1.2. Exploración de los datos

Antes de nada se procederá a determinar las dimensiones del *data set*. Como se puede ver en la Figura 5.1, se trata de un conjunto bastante reducido que cuenta con tan solo 768 muestras, de las cuales 268 son personas diabéticas.

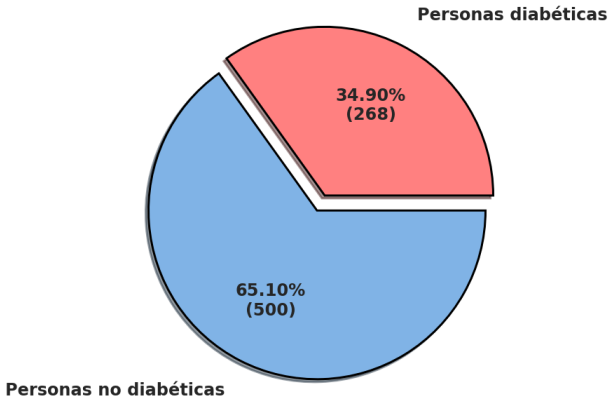

Figura 5.1: Distribución de las muestras en Pima Indians Diabetes Database.

Además de indicar si una persona del conjunto de datos es diabética o no, Pima Indians Diabetes Database cuenta con ocho columnas diferentes cuyos análisis gráficos están disponibles desde la Figura 5.2 a la 5.9, representando las siguientes variables:

- **Pregnancies**: Número de embarazos tenidos.
- **Glucose**: Concentración de glucosa en plasma dos horas después de realizar una PTGO.
- **BloodPressure**: Presión sanguínea, en concreto, la presión arterial diastólica (mm Hg).
- **SkinThickness**: Espesor del pliegue de la piel del tríceps (mm). Valor usado para estimar el nivel de grasa corporal.
- **Insulin**: Concentración de insulina dos horas después de realizar una PTGO.
- **BMI**: Índice de masa corporal (IMC).
- **DiabetesPedigreeFunction**: Riesgo de desarrollar la diabetes atendiendo al historial familiar.
- **Age**: Edad en años.

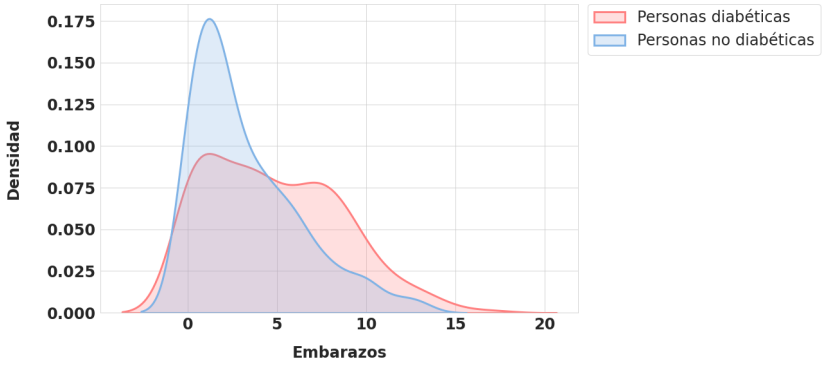

Figura 5.2: Distribución del número de embarazos en Pima Indians Diabetes Database.

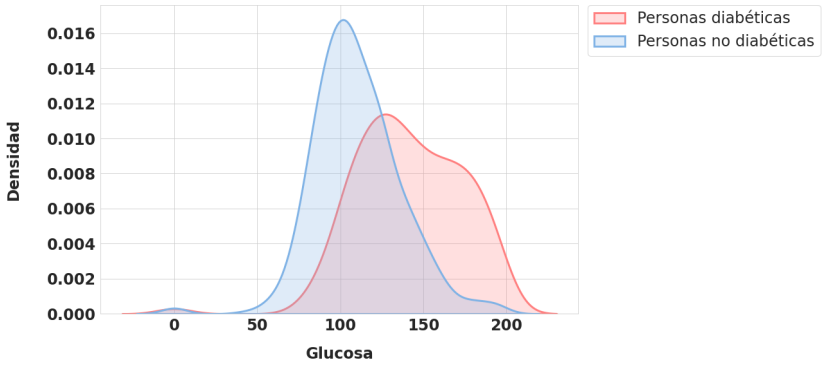

Figura 5.3: Distribución del valor de glucosa en Pima Indians Diabetes Database.

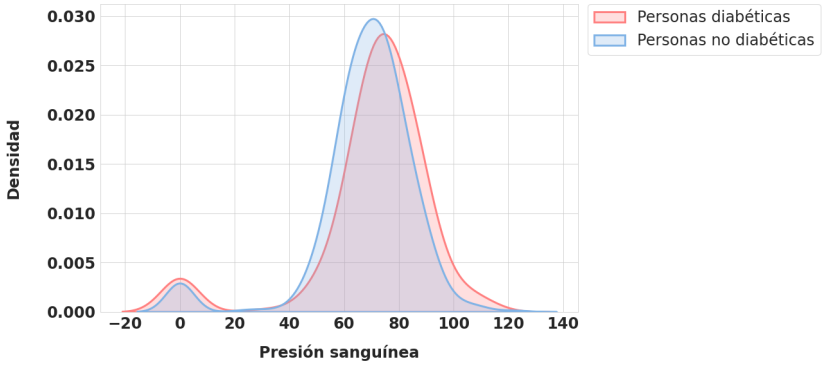

Figura 5.4: Distribución del valor de la presión sanguínea en Pima Indians Diabetes Database.

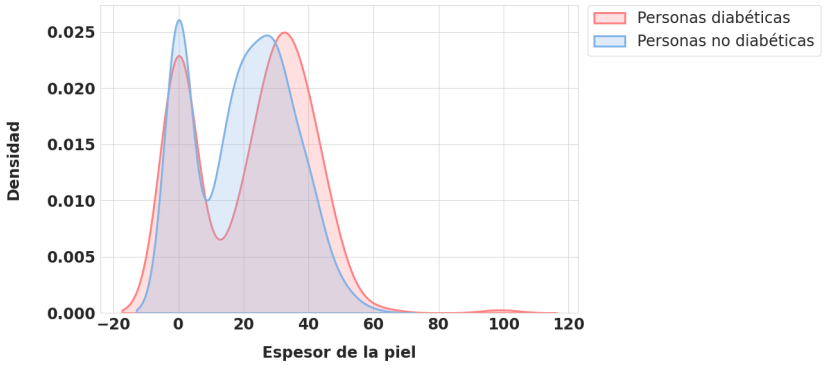

Figura 5.5: Distribución del espesor del pliegue de la piel en Pima Indians Diabetes Database.

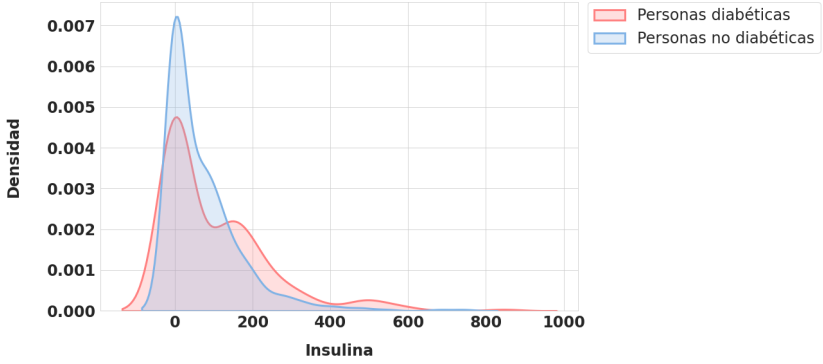

Figura 5.6: Distribución de la concentración de insulina en Pima Indians Diabetes Database.

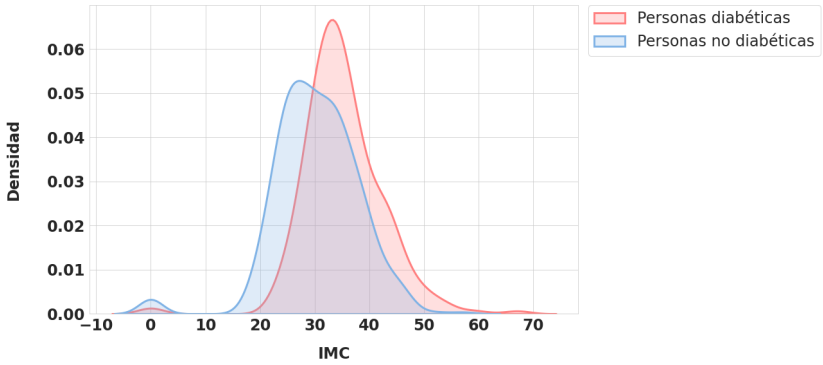

Figura 5.7: Distribución del valor de IMC en Pima Indians Diabetes Database.

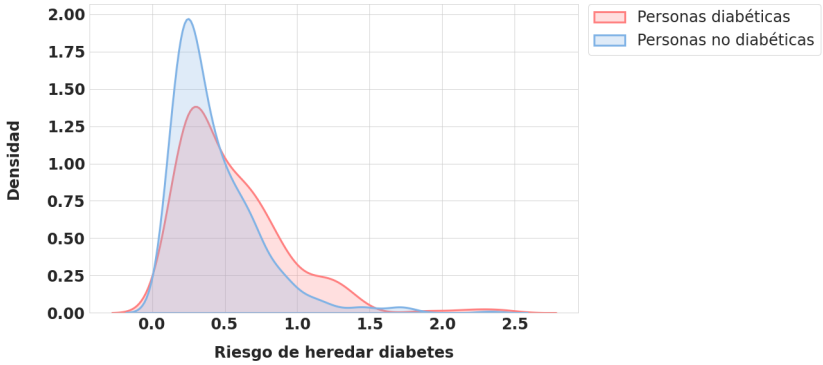

Figura 5.8: Distribución del riesgo de heredar diabetes en Pima Indians Diabetes Database.

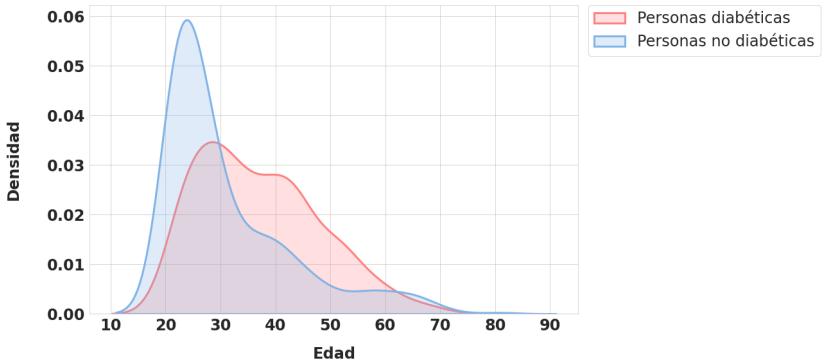

Figura 5.9: Distribución de los registros de edad en Pima Indians Diabetes Database.

Se ha decidido representar las gráficas anteriores mediante una serie de estimaciones de la densidad de *kernel* (KDE) para comparar los dos grupos de muestras. Básicamente, como se puede deducir de la Figura 5.10, una estimación de la densidad de *kernel* es el resultado de sumar todos los kernels normalizados que se obtienen a partir de los valores existentes en el *data set*. De esta forma, a partir de la densidad obtenida, se puede saber los valores que más predominan en cada característica y poder analizar a simple vista las diferencias más destacadas [17]. Debe mencionarse que la normalización se ha aplicado a cada grupo por separado y no a la distribución completa, con el fin de poder confrontar dos cantidades dispares de muestras.

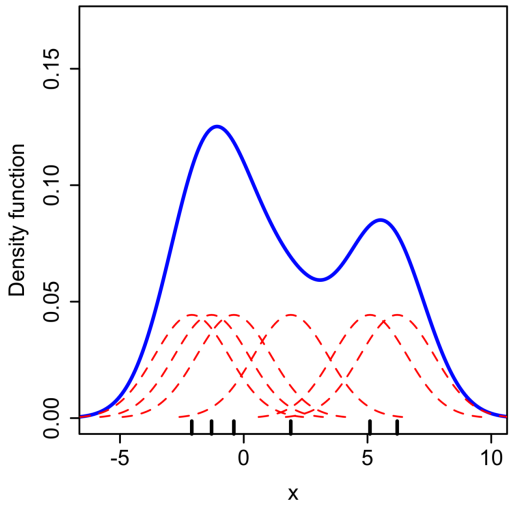

Figura 5.10: Formación de un KDE, línea continua azul, a partir de la suma *kernels* normalizados. Se observa que se obtiene un valor de densidad mayor en la zona izquierda, la cual presenta una concentración superior de *kernels*. Fuente: Adaptada de [18].

Primeramente, se obseva que el conjunto de datos sigue las tendencias vistas en el Capítulo 2: las personas diabéticas tienden a haber pasado por un mayor número de embarazos y también parecen contar con una mayor edad. Consecuentemente, en la Figura 5.3 se puede apreciar como las personas diabéticas presentan una concentración más elevada de glucosa, gráfica que destaca sobre todas las demás. De igual forma, se percibe un mayor número de personas diabéticas entre aquellas que presentan más riesgo de desarrollar la enfermedad según su historial familiar. Sin duda alguna, se deberá prestar especial atención a estas características a la hora de proceder con la experimentación.

En otro orden de cosas, se advierten una gran cantidad de valores indeseados dentro del conjunto. Aunque es perfectamente normal que una mujer tenga cero embarazos, no lo es que no se tenga glucosa en plasma o presión sanguínea. Se entiende que estos valores habrán sido almacenados por error, por lo que deberán tratarse las muestras afectadas para que no influya negativamente en el resultado.

Por último, pero no menos importante, se encuentra un hecho que parece ser contradictorio: en la Figura 5.6 se puede ver como la concentración de insulina no decrece en las personas diabéticas. Este hecho requiere de un análisis en detalle del caso, por lo que se procede a determinar las correlaciones de las variables, resultados visibles en el mapa de calor de la Figura 5.11.

Se puede ver como, al igual que el número de embarazos, la glucosa y el riesgo de desarrollar diabetes, los cuales ya se habían destacado anteriormente, el valor de la insulina presenta una correlación positiva, valor que todavía se pronuncia más, hasta llegar a 0.3, si se consideran solo los valores distintos a cero. En otras palabras, conforme más insulina tiene una persona más posibilidades tiene de ser diabética, suceso muy extraño que parece corresponder a un error. No obstante, tiene su explicación: los datos pueden haber sido recogidos de personas que padezcan únicamente DM2. Esta hipótesis está respaldada por L. J. Baier y R. L. Hanson en el artículo "Genetic Studies of the Etiology of Type 2 Diabetes in Pima Indians", donde se comenta que la población tiene una mezcla europea mínima, no llegando a desarrollar la autoinmunidad presente en la DM1[19]. Así pues, es completamente normal que, tratándose en exclusiva de una diabetes de tipo 2 caracterizada por demostrar más resistencia a la insulina, las personas afectadas por la enfermedad requieran producir más cantidad de esta hormona.

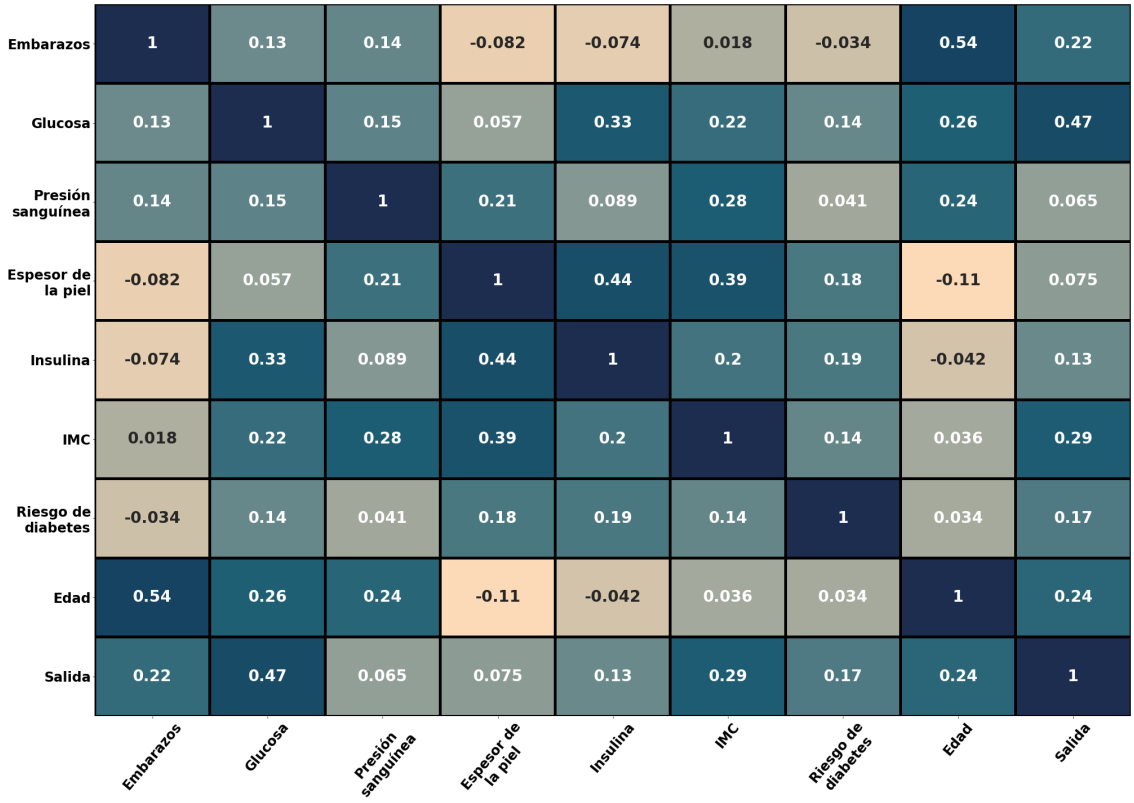

Figura 5.11: Correlaciones de las variables pertenecientes al conjunto de datos.

5.1.3. Limpieza del conjunto de datos

Inmediatamente después de explorar el conjunto de datos se debe preparar para poder ser utilizado. Para ello primero se procede a comprobar si existen valores nulos, suceso que no ocurre para este banco de datos. Este hecho puede significar que el excesivo número de valores iguales a cero encontrados durante la exploración de los datos representase las variables de las que no se tenía información, por lo que se analizarán estos casos en compensación.

	Muestras con valor 0	Valor mínimo[1]	Valor máximo	Media[1 2]	Desviación típica[1 2]
Embarazos	111	0	17	4.947	3.162
Glucosa	5	44	199	121.687	30.529
Presión sanguínea	35	24	122	72.405	12.369
Espesor de la piel	227	7	99	29.153	10.440
Insulina	374	14	846	155.548	118.773
IMC	11	18.2	67.1	32.457	6.925
Riesgo de diabetes	0	0.078	2.420	0.472	0.331
Edad	0	21	81	33.241	11.747

[1] Cálculos hechos a partir de los valores distintos a 0, exceptuando el número de embarazos.
[2] Números redondeados a la diezmilésima.

Tabla 5.1: Resumen estadístico de Pima Indians Diabetes Database.

Tras indagar en la información recabada en la Tabla 5.1, y teniendo en cuenta que es totalmente comprensible que existan mujeres sin haber pasado por un embarazo, se extraen las siguientes conclusiones:

- Una cantidad importante de las muestras registradas cuenta con un valor igual a cero en las variables que hacen referencia a la espesor de la piel o a la concentración de insulina. Considerando que pasa en ambos grupos por igual, se supone que no representan valores reales.
- El resto de categorías no cuentan con demasiados registros iguales a cero, por lo que se puede conservar gran parte del conjunto dándoles más importancia a éstas.
- En general, ignorando por un momento la cantidad de valores iguales a cero, no existen muchos más outliers destacables. Los casos que más sobresalen serían el máximo valor del espesor de la piel o el máximo de la concentración de la insulina, pero, tras comprobrar que la desviación típica no alcanza valores demasiado altos en comparación a la media, no se puede afirmar que sean valores no deseados. En cuanto a las demás características, número de embarazos incluidos, tampoco parecen salirse de la norma.

Por lo tanto, tras haber analizado los datos almacenados en cada categoría solo queda tratar aquellos iguales a cero. Sin embargo, puesto que existen demasiados casos relacionados con el espesor de la piel y la insulina, solo trataremos aquellos pertenecientes a la glucosa, la presión sanguínea y el IMC. Las muestras afectadas, un total de 44 ya que existen muestras con más de un valor indeseado, proceden a eliminarse del conjunto. En otros conjuntos es común sustituir estos registros por un valor medio pero, puesto que existen un mayor número de personas no diabéticas, de hacerlo estaríamos facilitando la aparición de falsos positivos.

Así pues, después de estos cambios, el conjunto resultante se puede ver en la Tabla 5.2.

	Número de personas diabéticas	Número de personas no diabéticas
Conjunto original	268	500
Conjunto tratado	249	475

Tabla 5.2: Dimensiones de los conjuntos de datos.

5.2 Primera experimentación

Esta fase tiene como objetivo realizar una serie de pruebas sobre distintos modelos de aprendizaje automático para así detectar los que producen los mejores resultados.

Para poder conseguir un modelo de aprendizaje automático debe utilizarse un conjunto de datos con el que entrenar el modelo. El concepto de entrenamiento se refiere a ajustar los parámetros del modelo en desarrollo para que se acerque a los datos de salida deseados y, así, se le puede dar uso en situaciones reales como, en nuestro caso, determinar si una persona es diabética o no. No obstante, es necesario conocer cómo de buena es la implementación realizada antes de ponerla en práctica, por lo que se destina una parte de los datos para estimar el error producido. Atendiendo a estas declaraciones, a continuación se determinarán los modelos, métricas y tipos de entrenamiento que harán posible el desarrollo.

5.2.1. Selección de modelos de aprendizaje automático

Para este primer barrido se utilizarán los siguientes modelos incluidos en Scikit Learn:

- **KNeighborsClassifier (KNN)**: Conocido como k vecinos más próximos. Pretende explotar el concepto de que las muestras que están cerca con respecto a una métrica definida son similares, por lo que pueden llegar a compartir características [20].
- **GaussianNB (GNB)**: Un clasificador bayesiano ingenuo es un clasificador probabilístico que se basa en una condición ingenua: la independencia condicional de las causas, es decir, la probabilidad de un suceso no está influida porque otro ocurra o no [20].
- **SVC**: Conjunto de algoritmos que hace referencia a la clasificación con vectores soporte (SVC). Permiten encontrar la forma óptima de clasificar entre clases maximizando la separación entre ellas. Esta separación está definida mediante vectores [21].
- **GradientBoostingClassifier (GB)**: La potenciación del gradiente es una técnica que permite construir un conjunto de árboles de decisión paso a paso con el objetivo de minimizar una función de pérdida [20].
- **RandomForestClassifier (RF)**: Clase basada en un conjunto de árboles de decisión cuya política de división presenta un nivel medio de aleatoriedad [20].

Dichos modelos se han elegido porque se considera que pueden adaptarse correctamente al conjunto de datos y, además, como se ha visto en la Sección 2.3, ya han sido utilizados anteriormente para el desarrollo de algoritmos de aprendizaje automático en la detección de otras enfermedades.

5.2.2. Elección de las métricas de evaluación

En este apartado se detallarán las métricas elegidas para la evaluación de los modelos anteriores. Para este trabajo se medirán los números obtenidos mediante los conceptos de *accuracy*, *precision*, *recall* y *F1 Score* [20], métricas que se explican mejor a partir de la noción de matriz de confusión representada en la Figura 5.12:

- ***Accuracy***: Se conoce comúnmente como exactitud y se trata de uno de las valores más utilizados para determinar el funcionamiento de una implementación de aprendizaje automático. Básicamente, se calcula dividiendo el número de predicciones correctas entre el número total de predicciones.

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

- ***Precision***: Traducido como precisión, hace referencia a cómo de exacta es la implementación desarrollada en cuanto al número de positivos predichos. Es un valor a tener en cuenta cuando el coste de falsos positivos es elevado.

$$precision = \frac{VP}{VP + FP}$$

- ***Recall***: Valor que hace referencia a la exhaustividad de un modelo y que es realmente importante calcular cuando el coste de los falsos negativos es muy elevado.

El estudio llevado a cabo en este trabajo se encuentra en esta situación: una persona diabética que no se clasifica como tal presenta una mayor cantidad de riesgos que el caso contrario.

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

- ***F1 Score***: Referenciado como valor-F en castellano, es una métrica a la que se recurre cuando se busca un cierto equilibrio entre precisión y exhaustividad, siendo especialmente recomendada cuando hay una distribución de clases desigual con, generalmente, un gran número de negativos.

$$f1 = 2 * \frac{precision * recall}{precision + recall}$$

Teniendo estas definiciones en cuenta, aunque todas las métricas son importantes y no debe ignorarse ninguna, en este proyecto se otorgará mayor importancia a los valores de *recall* y *f1*, evitando en la medida de lo posible la utilización de sus traducciones ya que no están ampliamente extendidas en comparación a sus versiones inglesas y pueden inducir cierta confusión.

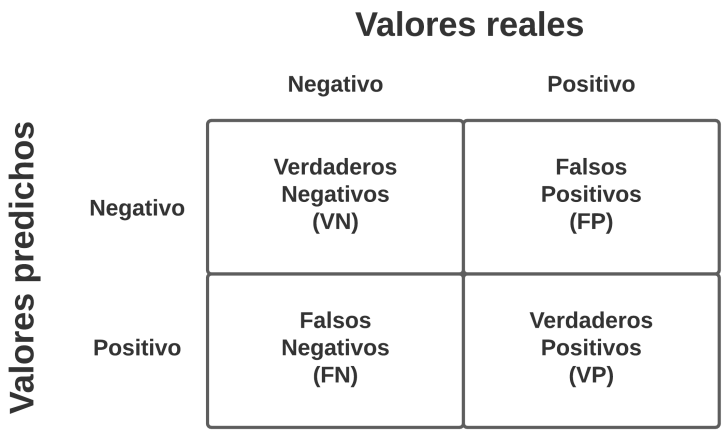

Figura 5.12: Matriz de confusión genérica.

5.2.3. Entrenamiento y evaluación de los modelos

Una vez comprendido el funcionamiento del desarrollo que se va a realizar y con qué métricas se va a evaluar, se puede empezar con el entrenamiento de los modelos. Para ello, se debe elegir un modelo de evaluación y, dada la naturaleza de esta primera prueba, aprovecharemos para experimentar con dos distintos:

- ***Cross-Validation***: Conocido en castellano como validación cruzada. Es un procedimiento de remuestreo que se utiliza para evaluar modelos de aprendizaje automático que cuentan con un *data set* bastante limitado y, usualmente, no produce resultados demasiado sesgados. El método que generalmente se sigue, representado en la Figura 5.13, consiste en dividir las muestras en k grupos, realizándose k iteraciones en las que, en cada una de ellas, se utiliza una combinación diferente de k - 1 grupos para entrenamiento y se destina el conjunto sobrante para evaluación. Una vez terminado el proceso se determina su desempeño realizando la media ponderada de las actuaciones anteriores [22].

- ***Train/Test Split***: Es un procedimiento en el que el conjunto de datos se divide en dos subconjuntos: el primero de ellos se utiliza para entrenar un modelo de aprendizaje automático mientras que el segundo es reservado para la evaluación del mismo. Puede ser complicado de utilizar en un *data set* que cuente con un número de muestras reducido, hecho que no debe ignorarse ya que puede ser problemático con las dimensiones de Pima Indians Diabetes Database [23].

  Por otra parte, tener solo dos subconjuntos puede provocar un sobreajuste (*overfitting*) producido al modificar el modelo con la intención de mejorar los resultados obtenidos sobre la misma porción de datos, motivo por el que es interesante destinar una parte del *data set* a un grupo de validación. Tras realizar el entrenamiento, se puede evaluar el modelo desarrollado con este nuevo subconjunto para ajustar los parámetros y alcanzar unas cifras mejores. Posteriormente, una vez finalizados los cambios, se procede a realizar la evaluación final con el conjunto de evaluación sin tener que preocuparse de haber adaptado el modelo a los datos con los que se valora, incidente que aportaría un error demasiado optimista. En la Figura 5.14 se representa gráficamente estas dos formas de dividir el conjunto datos en *Train/Test Split*.

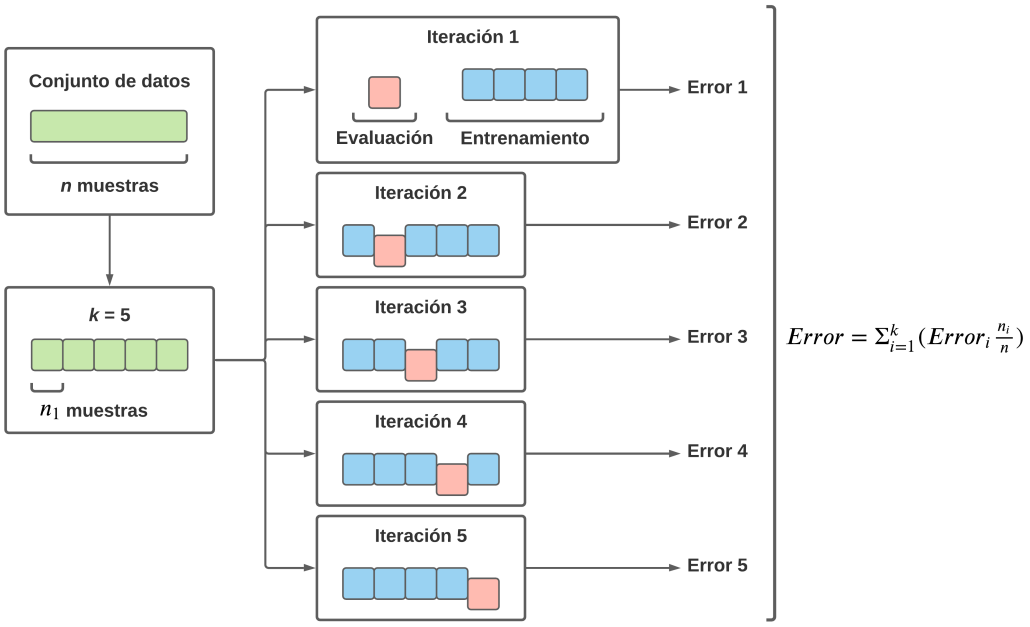

Figura 5.13: Procedimiento del modelo *Cross-Validation*.

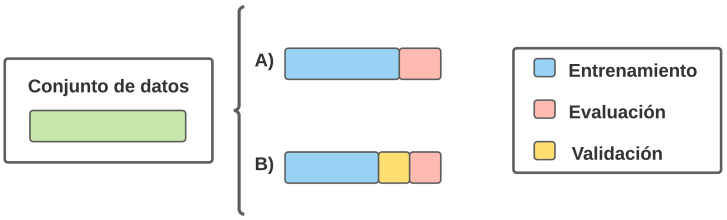

Figura 5.14: Formas de dividir el conjunto de datos para aplicar *Train/Test Split*. En A) se observa un planteamiento sencillo formado por los subconjuntos de entrenamiento y evaluación. En cambio, B) está formado por tres grupos, destinando una porción de los datos para validación.

A la hora de llevar a cabo la obtención de los resultados se distinguirán dos tipos de casos: los resultados alcanzados tras realizar una prueba individual, formada por entrenamiento y evaluación de los resultados, y los valores medios obtenidos tras repetir un número elevado de veces una prueba individual, experimento que proporciona una mayor fiabilidad.

Las pruebas individuales se realizarán utilizando una semilla aleatoria determinada, es decir, se ejecutará el experimento cada vez bajo las mismas condiciones para poder replicar los resultados y disponer de una forma más justa de comparar las implementaciones, ya que de no hacerse así, hechos como distribuir el conjunto de datos de una forma diferente podrían alterar los valores obtenidos.

Por otra parte, cuando la implementación lo permita, se repetirá el experimento una gran cantidad de ocasiones para obtener unos resultados más fiables con los que realizar las comparaciones. Al contar con una mayor cantidad de datos no es necesario reutilizar las mismas semillas porque, aunque alguna ejecución diste mucho de las demás, sus números pasarán relativamente desapercibidos tras calcular los valores medios.

5.2.4. Elección de los modelos finales

Hechas todas las explicaciones se procede a entrenar los cinco modelos explicados en el punto 5.2.1 con el conjunto de datos tratado con anterioridad. Dichos modelos, caracterizados por ser unas implementaciones básicas de las clases facilitadas por la librería de Scikit Learn, destinarán un 75 % de las muestras para el entrenamiento y el 25 % restante para su evaluación, alcanzando los resultados presentados en las Tablas 5.3 y 5.4.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
KNN	72.0	59.7	51.3	54.6
GNB	75.7	65.9	62.2	63.3
SVC	75.9	72.3	46.2	56.0
GB	77.5	69.6	60.6	64.4
RF	77.4	70.8	59.1	63.7

Tabla 5.3: Media de los resultados obtenidos tras repetir 100 veces el entrenamiento básico con *Cross-Validation*.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
KNN	71.7	60.0	52.5	55.8
GNB	75.8	66.0	61.0	63.2
SVC	75.7	73.2	46.6	56.8
GB	75.8	66.9	58.6	62.3
RF	76.6	69.2	57.7	62.7

Tabla 5.4: Media de los resultados obtenidos tras repetir 100 veces el entrenamiento básico con *Train/Test Split*.

Sin ninguna duda GaussianNB (GNB), GradientBoostingClassifier (GB) y RandomForestClassifier (RF) son las alternativas más prometedoras. Estos tres modelos presentan un valor de *recall* y *f1* mucho más atractivo que KNeighborsClassifier (KNN) y SVC, llegando incluso a mostrar una abultada diferencia de un 16 %. Además, los tres modelos elegidos son también los que mayor *accuracy* tienen, característica que refuerza la decisión tomada. Cabe mencionar que la implementación basada en las máquinas de vectores

soporte (SVC) no parece desenvolverse mal del todo, pero presenta un valor bastante pobre de *recall* que, dada la importancia del mismo en este proyecto, la convierten en una opción poco esperanzadora que conviene descartar para poder distribuir más los recursos disponibles.

En cuanto a la elección del modelo de evaluación, se puede observar que ambos alcanzan resultados parejos. No obstante, dada la necesidad de disponer de mayor flexibilidad para manipular y distribuir las muestras del *data set*, se optará por sacrificar parte de la fiabilidad extra proporcionada por *Cross-Validation* por una mayor sencillez, continuando la experimentación con *Train/Test Split*.

Por último, puesto que habrá casos en los que no sea factible repetir una prueba numerosas veces, se adjunta en la Tabla 5.5 los resultados de una única prueba de *Train/Test Split* con los modelos elegidos para facilitar la comparación de los resultados, estando todos los valores iniciales agrupados bajo un mismo apartado. En estos nuevos valores se alcanzan unas cifras mejores, hecho complatemente normal teniendo en cuenta que cada ejecución genera unos resultados distintos, los cuales pueden estar por debajo o por encima de la media, siendo en esta ocasión la segunda posibilidad. Por lo tanto, no se le debe dar mucha más importancia que la de tener que utilizar la misma semilla en las demás pruebas individuales para partir de las mismas condiciones.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
GNB	76.2	64.6	67.7	66.1
GB	79.0	70.0	67.7	68.9
RF	77.3	68.4	62.9	65.5

Tabla 5.5: Resultados del entrenamiento básico con *Train/Test Split*.

5.3 Implementación de mejoras

Aunque ciertamente los resultados anteriores no son un mal inicio, todavía pueden mejorarse aplicando distintas técnicas. Para ello, primero se analizarán los puntos débiles del conjunto para, posteriormente, poderlos subsanar. Con este fin, de igual forma que se ha hecho en la exploración y limpieza del conjunto de datos, se distribuirán todas las implementaciones realizadas en diferentes documentos de Jupyter Notebook para que, de esta forma, cualquier usuario pueda replicar o analizar un caso concreto aisladamente, sin necesidad de comprender el proyecto entero. Las librerías de Pandas y Scikit Learn también tendrán un papel fundamental en esta sección, por lo que se estudiará de qué forma pueden ser más beneficiosas.

5.3.1. Detección de aspectos mejorables

Tras analizar el *data set* se observan dos hechos dignos de mención:

- **El conjunto de datos es demasiado reducido**: En la Figura 5.1 ya se había podido comprobar que tan solo se contaba con 768 muestras, que es un número demasiado pequeño. Si se fuerza demasiado un algoritmo a adaptarse al mismo para reducir el error obtenido se puede acabar alcanzando un sobreajuste del modelo que limite la capacidad de generalización [20], por lo que deberá considerarse aumentar la cantidad de muestras.
- **El conjunto de datos cuenta con características poco influyentes**: El análisis de correlaciones realizado sobre la Figura 5.11 ya dejaba ver que había variables con

muy poca correlación y su participación en los entrenamientos puede entorpecer la creación de un modelo óptimo. Por lo tanto, deberán determinarse aquellas características que dificultan el desempeño y eliminarlas para alcanzar mejores resultados y agilizar el proceso.

5.3.2. Mejoras en el conjunto de datos

Aumentar los datos, más conocido como *Data Augmentation*, es una técnica más propia de la clasificación de imágenes [20]; no obstante, dadas las reducidas dimensiones del conjunto de datos, se debe apostar por ella. En primer lugar, se guardará la parte del *data set* correspondiente al subconjunto de evaluación en un fichero distinto para no manipular esa parte. Seguidamente, se procederá a modificar las muestras restantes introduciendo cierta aleatoriedad a las mismas. Es importante considerar que de aplicar cambios muy bruscos se puede hacer que la muestra pierda cualquier sentido (por ejemplo, una persona que no sea diabética pero que tenga el valor de glucosa más alto de todos), y con cambios muy leves el modelo se entrenará continuamente sobre los mismos valores. En este caso se ha optado aplicar dos componentes aleatorios que se restrinjan entre ellos, llamados random_value y random_variation, visibles en las siguientes líneas de pseudocódigo:

```
random_variation = 0.01 + random.random()/10
value = variable[position, feature]
random_value = random.uniform(-value, +value)
variable[position, feature] = (value + random_value * random_variation)
```

Si se analizan esas instrucciones en detalle se observa que random_variation solo puede tomar valores entre 0.01 y 0.10, limitando a que una cifra solo pueda variar un 10 % de su valor original como máximo y, de forma general, bastante menos que eso. Se trata de una postura bastante conservadora, pero siempre se puede modificar si se observa que los resultados no son suficientemente significativos.

Por otro lado, también se han realizado experimentos variando el porcentaje de muestras destinada a entrenamiento y evaluación, probando no solo con la ya usada distribución 75 % - 25 % sino también con 80 % - 20 %. Todas estás pruebas están documentadas en el repositorio del proyecto y, puesto que se obtienen resultados similares a los alcanzados con la configuración 75 % - 25 %, no se mencionarán a la hora de realizar las comparativas, ya que no es recomendable confrontar dos pruebas que no han efectuado el mismo procedimiento.

Tras los cambios mencionados se obtienen los resultados plasmados en la Tabla 5.6, los cuales proceden de una prueba individual ya que, al ser el contenido del subconjunto de evaluación siempre el mismo, no se producirán variaciones derivadas de distintas distribuciones de los datos. Por lo tanto, con el objetivo de ser lo más justos posibles, se compararán estos resultados con la Tabla 5.5.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
GNB	76.2	68.3	65.2	66.7
GB	80.1	78.8	62.1	69.5
RF	73.5	65.0	59.1	61.9

Tabla 5.6: Resultados del entrenamiento tras aplicar *Data Augmentation* con *Train/Test Split* 75-25.

Se puede ver como *Data Augmentation* parece ser beneficiosa para los modelos de GNB y GB, produciendo una ligera mejoría del *f1*. Sin embargo, también destaca que RF

empeora notablemente en todas las métricas, siendo un posible indicativo de que puede no ser la mejor técnica para esta situación.

5.3.3. Mejoras en el entrenamiento de los modelos

La selección de características se realizará aplicando *Recursive Feature Elimination* (RFE), un método disponible en Scikit Learn que se ajusta a un modelo y elimina las características más débiles hasta que se alcanza un número especificado. Para encontrar el número óptimo de características a conservar es normal utilizar la validación cruzada; no obstante, en este proyecto se hará uso de una clase equivalente disponible para *Train/Test Split* ya que será el método de evaluación que se utilizará.

Tras aplicar RFE en *Gradient Boosting* y *Random Forest* observamos que las mejores variables son la glucosa, el IMC, el riesgo de desarrollar diabetes atendiendo al historial familiar y la edad. No obstante, la clase utilizada no está disponible para el clasificador bayesiano ingenuo pero, atendiendo a que los dos otros modelos han coincidido con las características a conservar y que son de las que mayor correlación presentan con la salida, se optará por utilizarlas en los tres modelos.

Al aplicar esta técnica se consiguen los valores representados en la Tabla 5.7.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
GNB	77.0	70.4	57.3	63.0
GB	76.0	67.0	60.0	63.0
RF	75.9	66.5	59.8	62.9

Tabla 5.7: Media de los resultados obtenidos tras repetir 100 veces el entrenamiento con selección de características en *Train/Test Split* 75-25.

En comparación con los obtenidos en la Tabla 5.4 se puede observar como GB sigue mejorando de media el modelo básico, llegando a incrementar ligeramente los valores de todas las métricas. Por otro lado, GBN sacrifica parte del valor de *recall* a cambio de mejorar los valores de *accuracy* y *precision*, y RF hace lo opuesto, llegando a mejorar el valor de *f1* por primera vez.

Una vez vistas ambas técnicas, *Data Augmentation* y la selección de características, queda combinarlas para ver cómo se acoplan entre sí y poder determinar si pueden llegar a formar parte en el modelo final. Los resultados de este ensayo se exponen en la Tabla 5.8.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
GNB	80.0	76.4	63.6	69.4
GB	80.7	76.3	68.2	72.0
RF	71.3	61.7	56.1	58.7

Tabla 5.8: Resultados tras aplicar *Data Augmentation* y selección de características en *Train/Test Split* 75-25.

Esta combinación parece ser todo un éxito en el caso de GBN y GB, alcanzando los valores más altos de *accuracy*, *precision* y *f1* para el primero y de *accuracy*, *recall* y *f1* en el segundo. En definitiva, mejora todos los datos de la Tabla 5.5 menos el *recall* de GBN, que ve su valor reducido. No obstante, este éxito no es compartido por RF: este modelo falla estrepitosamente y no mejora en absoluto con ninguna de las técnicas aplicadas.

5.3.4. Ajuste de parámetros

En el punto 5.2.3 se ha hablado de la necesidad de utilizar un subconjunto de validación para cuando se ajustan los hiperparámetros, valores de las configuraciones utilizadas durante el entrenamiento de un modelo, pero no hemos llegado a ponerlo en práctica. Esto se debe a que hemos utilizado los valores por defecto, pero siempre se pueden modificar para obtener mejores resultados.

En concreto, se van a intentar mejorar los hiperparámetros de *learning rate*, *maximum depth* y *number of boosting stages* de *Gradient Boosting*, modelo que mejor resultados ha presentado hasta el momento y cuyos parámetros se han seleccionado a partir de la documentación de GradientBoostingClassifier:

- ***Learning rate***: La tasa de aprendizaje determina cuánto cambia el modelo cada vez que se actualizan los pesos al detectar un error. Una tasa demasiado elevada puede provocar que no se lleguen a encontrar los valores mínimos, mientras que una demasiado baja puede hacer que se retrase la convergencia [20]. Se probará con los valores 0.01, 0.05, 0.1, 0.3, 0.5 y 1.

- ***Maximum depth***: Máximo número de nodos por árbol. Dado que por defecto utiliza 3 nodos, se realizará el experimento con valores pequeños como 1, 2, 4, 8, 16 y 32.

- ***Number of boosting stages***: Determina el número de etapas a realizar en su construcción escalonada. En este caso se probará con un mayor número de posibilidades, siendo los números elegidos 1, 2, 4, 8, 16, 32, 64, 100, 200, 300 y 500.

Para encontrar los mejores valores no se va a probar con todas las combinaciones posibles, puesto que tomaría una cantidad de tiempo completamente desorbitada. En su lugar, se hará uso de la clase GridSearchCV disponible en Scikit Learn, que permite hacer una búsqueda exhaustiva de los mejores hiperparámetros especificados para un estimador. Como indica su nombre, esta implementación está pensada para realizarse en *Cross Validation* por lo que para adaptarla también usaremos ShuffleSplit, una estrategia de *Cross Validation*, que permite controlar el número de repeticiones a realizar, permitiendo así replicar el comportamiento de *Train/Test Split*.

Tras probar con las diferentes cifras anunciadas anteriormente se obtiene que la mejor combinación posible está formada por 0.5 de *learning rate*, 4 de *maximum depth* y 16 etapas de potenciación, hiperparámetros que, tras reentrenar el 80 % previo utilizado, producen los resultados visibles en la Tabla 5.9 con el 20 % que se había reservado para evaluación.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
GB	79.3	73.9	65.4	69.4

Tabla 5.9: Resultados tras aplicar *Data Augmentation*, selección de características y ajuste de hiperparámetros en *Train/Test Split* 80-20.

Se puede ver como las cifras obtenidas son muy similares a las ya conseguidas en la Tabla 5.8, significando que los valores por defecto no eran una mala elección. Posiblemente la pequeña variación de los resultados haya sido provocada por variar la distribución utilizando un conjunto de validación, decisión recomendable para evitar overfitting aunque sea a cambio de un ligero deterioro de los resultados.

5.4 Desarrollo de una red neuronal artificial

Una vez terminadas las implementaciones anteriores se va a replicar el desarrollo anterior con una red neuronal artificial (ANN), proporcionando así un amplio repertorio de alternativas que justifiquen la utilización de modelos de aprendizaje automático para la detección de la diabetes.

Como sugiere su nombre, las redes neuronales artificiales son sistemas inspirados en las redes neuronales biológicas que constituyen los cerebros de los animales y, de forma generalizada, presentan los siguientes componentes [24]:

- **Neuronas**: Conjunto de unidades interconectadas que forman la red. Tras recibir una entrada, la neurona produce una salida que es propagada por la red de una forma determinada. Una neurona puede contar con múltiples conexiones.
- **Enlaces**: Conexiones entre las neuronas; cada enlace cuenta con un neurona de entrada y otra de salida, especificando cómo se propaga la información a lo largo de la red. A estos enlaces se les puede asignar un peso para modificar la importancia de los mismos.
- **Capas**: Las neuronas están agrupadas por capas, existiendo siempre como mínimo dos de ellas: la capa de entrada y la de salida. Estos conjuntos pueden conectarse con capas ocultas intermedias que permiten elevar la complejidad de la red y, conjuntamente a los enlaces, las capas dan forma a la estructura de la red. Todas las neuronas de una capa tienen las mismas funciones de propagación y de activación.
- **Funciones de propagación**: Función que calcula la entrada de una neurona a partir de las salidas recibidas.
- **Funciones de activación**: Función que, dada la entrada, determina la salida producida por la neurona.

Para poder desarrollar la ANN deberemos definir cada uno de estos elementos, proceso que se describirá durante su diseño y que permitirá una mejor comprensión de los conceptos.

5.4.1. Diseño de la ANN

Continuando con la misma filosofía que hemos seguido a lo largo del proyecto, se empezará desarrollando un modelo sencillo que permita ser mejorado posteriormente con técnicas más complejas. De tal manera, se utilizará una red neuronal *feed-forward*, es decir, un conjunto de perceptrones organizados por capas donde la información solo avanza hacia delante sin proporcionar retroalimentación a las neuronas anteriores.

Por sí mismo un perceptron cuenta con bastantes limitaciones; no obstante, con la inclusión de más capas se pueden atajar problemas de una mayor complejidad. Según el teorema de aproximación universal, una red *feed-forward* con una capa oculta puede representar cualquier función [24], motivo por el que nuestro modelo más básico partirá de dos capas ocultas, cada una de ellas formada por 20 neuronas inicialmente.

Las capas ocultas estarán conectadas linealmente y utilizarán una unidad lineal rectificada (ReLU) como función de activación, funciones ilustradas en la Figura 5.15.

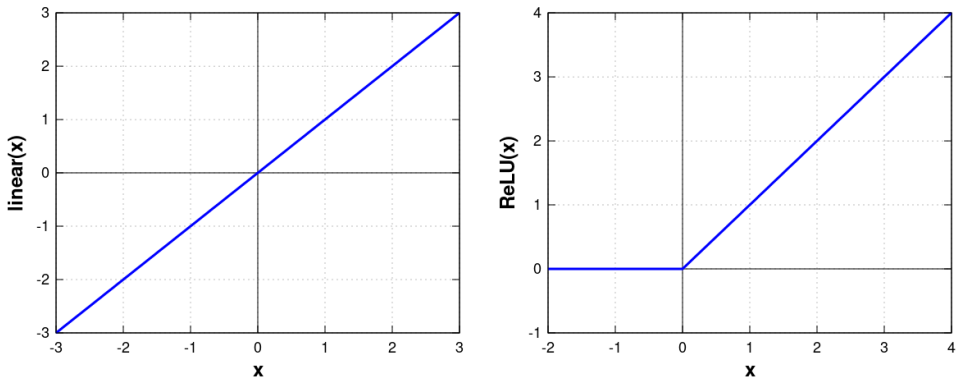

Figura 5.15: Funciones utilizadas en la red neuronal. A la izquierda se observa una función lineal definida como $f(x) = x$ mientras que a la derecha está representada una función ReLU definida como $f(x) = max(0, x)$.

Finalmente, poniendo en común todas las características mencionadas se genera el esquema visible en la Figura 5.16.

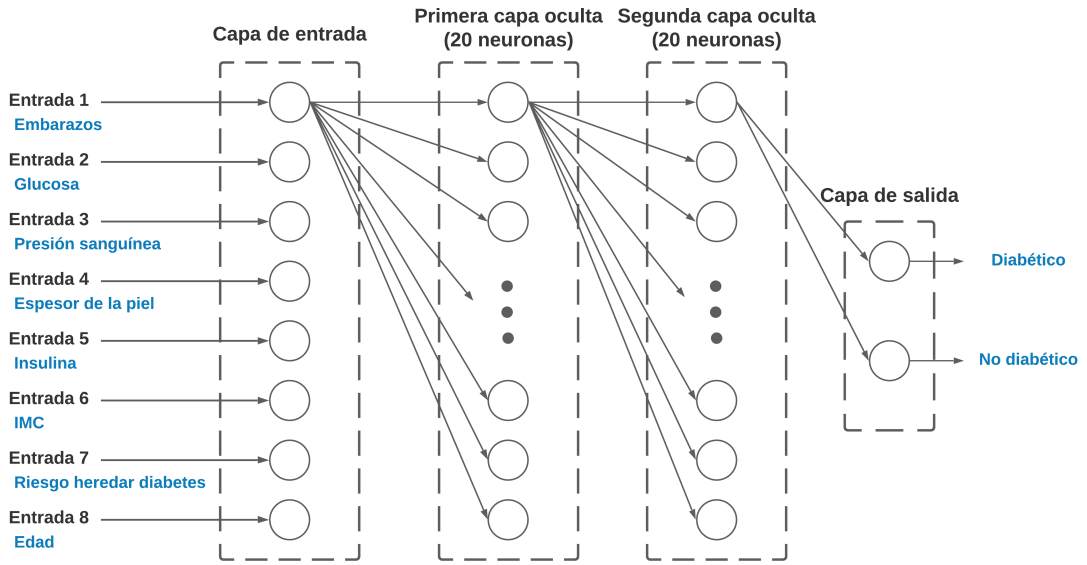

Figura 5.16: Esquema de la primera ANN implementada. Cada característica del conjunto de datos corresponde a una neurona de la capa de entrada. Cada neurona de la capa de entrada está conectada con todas las neuronas de la siguiente capa (en la imagen solo se muestran las conexiones de la primera para no afectar a la claridad de la ilustración, pero es extendible a todas las demás), comportamiento que se repite con todas las capas ocultas hasta llegar a la capa de salida que, al ser una clasificación binaria, estará formada por dos unidades.

5.4.2. Desarrollo de la ANN

Para el desarrollo de la red neuronal artificial se utilizará la misma distribución empleada en el punto 5.3.4, reservando un 60 % para entrenamiento, 20 % para validación y 20 % para evaluación, no pudiéndose usar inicialmente el reparto 75-25 puesto que se ajustará un hiperparámetro desde el principio (sin embargo, luego se replicará el experimento para esa partición utilizando ya el mejor valor del hiperparámetro con el fin contar con un mayor repertorio de soluciones). También se continuará utilizando las mis-

mas métricas de evaluación, motivo por el que se dependerá de la librería de Scikit Learn de nuevo, la cual se utilizará con PyTorch por primera vez en este trabajo.

Para poder implementar la configuración presentada en la sección anterior se utilizará nn.Module, una clase base pensada para que todas las redes neuronales hereden de ella. De tal forma, se alcanza la siguiente clase personalizada:

```
class Custom_ANN_Model(nn.Module):
    """ Artificial Neural Network model used
    """

    def __init__(self, in_features=8, neurons_first_layer=20,
        neurons_second_layer=20, out_features=2):
        """ Inits Custom_ANN_Model hidden layers.

        :param in_features: int Number of input features
        :params neurons_X_layer: int Number of neurons in the X hidden layer
        :param out_features: int Number of output features
        """

        super(Custom_ANN_Model, self).__init__()
        # Definition of the two hidden layers
        self.f_connected1 = nn.Linear(in_features=in_features, out_features=
            neurons_first_layer)
        self.f_connected2 = nn.Linear(in_features=neurons_first_layer,
            out_features=neurons_second_layer)
        self.out = nn.Linear(in_features=neurons_second_layer, out_features=
            out_features)

    def forward(self, h):
        """ Defines the computation performed at every call.
        """

        # Layers activation
        h = F.relu(self.f_connected1(h))
        h = F.relu(self.f_connected2(h))
        h = self.out(h)
        return h
```

En ella se puede ver inicializada el esquema de la red neuronal definido, contando con ocho neuronas en la capa de entrada, 20 en las capas ocultas y dos en la capa de salida. También se observa la función de activación en el método forward() y la función de propagación en la inicialización de la clase. Acto seguido se procederá a entrenar el modelo de la misma forma que se ha hecho con los de aprendizaje automático, destacando las siguientes diferencias:

- **Hiperparámetro *epoch***: Valor que hace referencia a un ciclo a través del conjunto de datos de entrenamiento completo [25]. Para estimar este valor se han probado distintos valores de *epochs* hasta alcanzar el menor error en el conjunto de validación, cuyo valor ha sido 900.

- **Estandarización de las columnas**: Se ha aprovechado este experimento para introducir esta nueva técnica. De esta forma, mediante el uso de StandardScaler(), una clase de Scikit Learn, se han ajustado los valores de las diferentes columnas para que hagan uso de una escala común, ya que sus valores llegaban a tener rangos completamente distintos (por ejemplo, edad, formado por un rango que va desde 21 a 81, y glucosa, de 44 a 199).

- **Función de pérdida**: Función que calcula la distancia entre la salida actual del algoritmo y la salida esperada [25]. Se utilizará la función de pérdida de entropía cruzada (*Cross Entropy Loss*), típica de las tareas de clasificación.

Tras estos cambios se registran los resultados obtenidos para la red neuronal artificial en las tablas 5.10 y 5.11, reentrenando el 80 % previo utilizado (entrenamiento y validación) para compararlo con el 20 % que se había apartado para evaluación.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
ANN	70.1	57.1	57.2	56.8

Tabla 5.10: Media de los resultados obtenidos tras repetir 100 veces *Train/Test Split* 80-20 en la ANN.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
ANN	64.8	51.7	56.6	54.1

Tabla 5.11: Resultados tras realizar una prueba de *Train/Test Split* 80-20 en la ANN.

Como se puede apreciar, esta primera implementación produce unos resultados inferiores a los vistos en los modelos de aprendizaje automático, motivo por el que se prestará especial importancia a la inclusión de las mejoras.

5.4.3. Implementación de mejoras

Se seguirá el mismo orden que con los otros modelos de aprendizaje automático. En primer lugar, se aplicará *Data Augmentation*, proceso bastante sencillo ya que se reutilizará el conjunto de datos obtenido con anterioridad, siendo sustituir el *data set* de entrada el único cambio necesario. Los resultados de este experimento están reflejados en la Tabla 5.12.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
ANN	70.3	59.6	53.8	56.6

Tabla 5.12: Resultados tras aplicar *Data Augmentation* en el entrenamiento *Train/Test Split* 80-20 de la ANN.

Estos resultados mejoran bastante los primeros valores obtenidos en la Tabla 5.11, llegando a superar *accuracy*, *precision* y *f1*, y teniendo un pequeño aumento del error en el *recall* como inconveniente.

Seguidamente, se procederá a aplicar la selección de características, caso en el que sí que se podrá comparar la media de los resultados obtenidos, a diferencia de la situación anterior en la que se ha hecho uso de una prueba individual con la misma semilla. En este ensayo se utilizarán las mismas características empleadas a lo largo de todo el proyecto, las cuales eran la glucosa, el IMC, el riesgo de desarrollar diabetes atendiendo al historial familiar y la edad. Los frutos de este experimento están disponibles en la Tabla 5.13.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
ANN	70.8	57.8	57.3	57.2

Tabla 5.13: Media de los resultados obtenidos tras repetir 100 veces el entrenamiento con selección de características con *Train/Test Split* 80-20 de la ANN.

De igual manera que la técnica anterior, la selección de características parece ser bien recibida por la red neuronal artificial: todas las métricas analizadas mejoran en comparación a los números de la Tabla 5.10. Hecho esto, se comprobará cómo combinan ambas implementaciones, resultados disponibles en la Tabla 5.14.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
ANN	75.2	69.0	55.8	61.7

Tabla 5.14: Resultados tras aplicar *Data Augmentation* y selección de características en el entrenamiento *Train/Test Split* 80-20 de la ANN.

Una vez más se ha podido comprobar que ambas técnicas se acoplan perfectamente, alcanzando unos valores similares a los vistos en los modelos de aprendizaje automático, mejorando todas las métricas menos el *recall*, que vuelve a tener un ligero aumento del error posiblemente provocado por la aplicación de *Data Augmentation* como ha sucedido anteriormente.

Una vez terminadas todas las pruebas se procederá a probar con distintas configuraciones variando el número de capas ocultas y de sus neuronas. Estas configuraciones continuarán utilizando los mismos valores en los parametros vistos, descubriendo que, como se observa en la Tabla 5.15, se llegan a mejorar todavía más las cifras conseguidas utilizando cuatro capas de 10 neuronas, llegando a superar el *recall* obtenido para GNB y a muy cerca de GB.

Modelo	***Accuracy* (%)**	***Precision* (%)**	***Recall* (%)**	***F1 Score* (%)**
ANN con cuatro capas y diez neuronas	78.6	72.3	65.4	68.7
ANN con ocho capas y diez neuronas	76.6	68.9	63.5	66.0

Tabla 5.15: Resultados tras aplicar *Data Augmentation* y selección de características en el entrenamiento *Train/Test Split* 80-20 con distintas configuraciones de ANN.

CAPÍTULO 6
Pruebas y evaluaciones

A lo largo del escrito se han analizado los diferentes resultados obtenidos en cada implementación ejecutada. No obstante, esos valores por sí solos no tienen ningún significado. Por lo tanto, en este capítulo se procederá a recopilar todos los resultados mencionados anteriormente para, así, poder evaluar correctamente el desempeño de los diferentes modelos. Es digno de mención que este procedimiento será realmente útil a la hora de comprobar las conclusiones alcanzadas en el trabajo, por lo que se invertirá todo el tiempo que sea necesario para presentar las soluciones obtenidas de una forma clara y concisa.

6.1 Generación de informes

Para la generación de informes se procederá a crear dos clases de Python que permitan automatizar el proceso. De esta forma, se podrá realizar un breve experimento sin tener que preocuparse de registrar manualmente los resultados una vez finalizado.

La primera de las clases, con nombre TestRecorder, permite guardar los resultados obtenidos como un archivo JSON, facilitando su posterior lectura. Los datos almacenados están compuestos por el momento en el que se ha realizado el experimento, una breve descripción del ensayo y las cuatro métricas analizadas hasta el momento (*accuracy*, *precision*, *recall* y *F1 Score*). A continuación se muestra un ejemplo de cómo se puede utilizar una vez terminado un ensayo.

```
1 from python_project.test_recorder import TestRecorder
2
3 test_description = "Media de Train/Test Split 75-25"
4 results = TestRecorder(test_description, accuracy, precision, recall, f1)
5 results.data_to_json()
```

Por otro lado se dispone de CustomReport, una herramienta que a partir de los JSON creados es capaz de elaborar un informe pdf, visible en la Figura 6.1. Para su creación se han utilizado las librerías datetime, json, json2html y pdfkit, las cuales permiten ahorrar un gran número de pasos complejos.

test	description	accuracy	precision	recall	f1	date
	ANN with Feature Selection and Data Augmentation. 8 Layers of 10 neurons	76.6	68.8	63.5	66.0	09/06/2021, 13:29:52
	ANN with Feature Selection and Data Augmentation. 4 Layers of 10 neurons	78.6	72.3	65.4	68.7	09/06/2021, 13:30:34

Figura 6.1: Imagen del PDF generado a partir de los archivos JSON guardados.

6.2 Comparación de resultados

Finalmente, en las Tablas 6.1, 6.2, 6.3 y 6.4 se recopilan todos los resultados obtenidos para los modelos elegidos, coloreados de verde aquellos que mejoran los valores iniciales y, de rojo, los que los empeoran. También se destacan las mejores cifras en negrita y se han abreviado los nombres de las columnas para poder disponer de más espacio para la descripción de los ensayos, siendo las métricas utilizadas las mismas que han servido de referencia a lo largo de todo el proyecto.

GBN				
Ensayo	***ACC* (%)**	***PRC* (%)**	***RCL* (%)**	***F1* (%)**
Media entrenamiento básico con *Train/Test Split* 75-25	75.8	66.0	**61.0**	**63.2**
Media entrenamiento con selección de características en *Train/Test Split* 75-25	**77.0**	**70.4**	57.3	63.0
Prueba individual entrenamiento básico con *Train/Test Split* 75-25	76.2	64.6	**67.7**	66.1
Prueba individual con *Data Augmentation* en *Train/Test Split* 75-25	76.2	68.3	65.2	66.7
Prueba individual con S.C. y *D.A.* en *Train/Test Split* 75-25	**80.0**	**76.4**	63.6	**69.4**

Tabla 6.1: Recopilación de los resultados obtenidos en *Gaussian Naive Bayes* (GBN).

GB				
Ensayo	***ACC* (%)**	***PRC* (%)**	***RCL* (%)**	***F1* (%)**
Media entrenamiento básico con *Train/Test Split* 75-25	75.8	66.9	58.6	62.3
Media entrenamiento con selección de características en *Train/Test Split* 75-25	**76.0**	**67.0**	**60.0**	**63.0**
Prueba individual entrenamiento básico con *Train/Test Split* 75-25	79.0	70.0	67.7	68.9
Prueba individual con *Data Augmentation* en *Train/Test Split* 75-25	80.1	**78.8**	62.1	69.5
Prueba individual con S.C. y *D.A.* en *Train/Test Split* 75-25	**80.7**	76.3	**68.2**	**72.0**
Prueba individual con S.C., *D.A.* y ajuste de parámetros en *Train/Test Split* 80-20	79.3	73.9	65.4	69.4

Tabla 6.2: Recopilación de los resultados obtenidos en *Gradient Boosting* (GB).

RF				
Ensayo	***ACC* (%)**	***PRC* (%)**	***RCL* (%)**	***F1* (%)**
Media entrenamiento básico con *Train/Test Split* 75-25	**76.6**	**69.2**	57.7	62.7
Media entrenamiento con selección de características en *Train/Test Split* 75-25	75.9	66.5	**59.8**	**62.9**
Prueba individual entrenamiento básico con *Train/Test Split* 75-25	**77.3**	**68.4**	**62.9**	**65.5**
Prueba individual con *Data Augmentation* en *Train/Test Split* 75-25	73.5	65.0	59.1	61.9
Prueba individual con S.C. y *D.A.* en *Train/Test Split* 75-25	71.3	61.7	56.1	58.7

Tabla 6.3: Recopilación de los resultados obtenidos en *Random Forest* (RF).

ANN				
Ensayo	***ACC* (%)**	***PRC* (%)**	***RCL* (%)**	***F1* (%)**
Media ANN básica con *Train/Test Split* 80-20	70.1	57.1	57.2	56.8
Media ANN con selección de características en *Train/Test Split* 80-20	**70.8**	**57.8**	**57.3**	**57.2**
Prueba individual ANN básica con *Train/Test Split* 80-20	64.8	51.7	56.6	54.1
Prueba individual con *Data Augmentation* en *Train/Test Split* 80-20	70.3	59.6	53.8	56.6
Prueba individual con S.C. y *D.A.* en *Train/Test Split* 80-20	75.2	69.0	55.8	61.7
Prueba individual en una ANN de cuatro capas y diez neuronas con S.C. y *D.A.* en *Train/Test Split* 80-20	**78.6**	**72.3**	**65.4**	**68.7**
Prueba individual en una ANN de ocho capas y diez neuronas con S.C. y *D.A.* en *Train/Test Split* 80-20	76.6	68.9	63.5	66.0

Tabla 6.4: Recopilación de los resultados obtenidos en la red neuronal artificial (ANN).

Una vez coloreados los valores alcanzados resulta mucho más fácil analizar cómo han evolucionado los distintos modelos. En primer lugar *Gaussian Naive Bayes* alcanza unos valores bastante positivos, tanto *accuracy* como *f1* mejoran cerca de un 3 % respecto a la primera implementación, y *precision* llega reducir casi un 12 % su error, hecho realmente meritoso. No obstante, se puede comprobar como el *recall*, métrica que tiene asociada una gran importancia debido a su relación con los falsos negativos, empeora considerablemente, siendo un punto negativo que no debe pasar inadvertido.

Por otro lado, *Gradient Boosting* llega alcanzar los valores más altos de todo el proyecto: ningún otro modelo consigue superarlo en cualquier métrica. La prueba realizada tras aplicar selección de características y *Data Augmentation* es el mayor éxito del escrito, consiguiendo superar los primeros valores alcanzados con la implementación básica.

En cambio, con el modelo de *Random Forest* no se han podido superar los valores iniciales, entendiendo que este modelo no se adapta tan bien al problema como se pensaba.

Sigue habiendo obtenido unas cifras aceptables pero, sin duda alguna, no se trata de la mejor opción a profundizar en caso de querer ampliar el trabajo desarrollado.

Por último pero no menos importante, la red neuronal artificial ha obtenido, de la misma forma que GB, unos resultados muy favorables. Las configuraciones de cuatro y ocho capas ejecutadas al final han conseguido superar con creces los primeros resultados alcanzados en la Tabla 5.11 y, aunque no llega a superar los de *Gradient Boosting*, es la implementación que más ha conseguido superar sus propios resultados iniciales, llegando a mejorar en casi un 10 % y 14 % el *recall* y *f1* respectivamente.

En definitiva, se concluye que de los cuatro modelos potenciados solo uno ha acabado siendo descartado. En cuanto a los tres restantes, alcanzan unos resultados similares en todas las métricas, es decir, no hay uno que se adapte mejor a un tipo de caso concreto (podría resultar que uno de ellos fuera mejor tratando falsos positivos y otro tratando falsos negativos), por lo que no se distingue una situación que sea más beneficiosa para una implementación en concreto. En todo caso, los modelos de aprendizaje automático clásico pueden preferirse cuando se desee realizar un despliegue rápido, mientras que la red neuronal puede ser más útil si se aumenta la complejidad del problema.

CAPÍTULO 7
Implantación

Aunque este proyecto cuenta con una gran cantidad de horas detrás, todavía cuenta con ciertas carencias que le impiden ser implantado en un entorno real. Para poder realizar una implementación deberán considerarse los siguientes aspectos:

- **Aumentar el conjunto de datos**: En general, los modelos implementados han conseguido adaptarse al *data set* utilizado; no obstante, como se ha mencionado a lo largo del documento, se cree que el reducido número de muestras ha sido uno de los factores más limitantes. Para compensar este hecho deberían recolectarse un gran número de nuevas muestras para formar un nuevo conjunto mayor que no restrinja tanto la experimentación y permita unos mejores resultados.
- **Preparar una interfaz gráfica**: Las personas encargadas de detectar la diabetes en los pacientes no tienen por qué saber cómo funcionan los modelos desarrollados. Si de verdad se desea que esta propuesta sea utilizada se debe facilitar su funcionamiento a los usuarios, proporcionando una interfaz que, con tan solo indicar las variables bajo estudio, proporcione el resultado de la clasificación.
- **Descentralización del software desarrollado**: Aunque hoy en día es común utilizar ordenadores en las consultas de los hospitales, se desconoce la potencia o disponibilidad de los mismos. Atendiendo a este hecho y a que posiblemente en países con mayores dificultades económicas exista una situación distinta a la observada en España, es interesante dividir el software en partes de forma que la clasificación de nuevas muestras se lleve a cabo en servidores especializados que puedan aprovechar mejor los recursos. De esta manera, podría obtenerse la información deseada con cualquier dispositivo capaz de realizar la petición, ahorrando también tener que adaptar la herramienta a sistemas de diferentes especificaciones.
- **Creación de un sistema de alarma**: El proyecto desarrollado tiene como objetivo facilitar la detección de la diabetes, pero si el paciente no acude a realizar las pruebas pertinentes no se podrá proporcionar una gran ayuda. Por lo tanto, debería considerarse la creación de un sistema de alarma para detectar con anterioridad los pacientes que pueden ser diabéticos sin saberlo por no haber sufrido un percance grave todavía.

Una vez abarcados estos aspectos se podría llevar a cabo la instalación y configuración del sistema, el cual podría seguir nutriéndose a partir de los nuevos datos proporcionados por los pacientes con el objetivo de mejorar más su funcionamiento.

CAPÍTULO 8
Conclusiones

En este proyecto se ha elaborado un sistema con el objetivo principal de desarrollar una serie de modelos basados en apredizaje automático que permitan detectar si una persona padece diabetes o no. Dichos modelos debían partir de una implementación sencilla a la que se aplicasen técnicas más complejas en función de los resultados obtenidos, potenciando los que contasen con mayor proyección. Como se ha podido seguir a lo largo del Capítulo 5, este objetivo se ha cumplido exitosamente, llegando a elaborar diferentes versiones y configuraciones de cada modelo realizado y aumentando la complejidad de las mismas en cada cambio.

Además, como objetivo secundario se proponía determinar las mejores situaciones para cada modelo resultante. No obstante, como se ha podido ver en el Capítulo 6, los tres modelos fructíferos restantes producían unos valores muy similares, siendo imposible determinar situaciones en las que cada uno de ellos fuera mejor que los demás. Sin embargo, sí que se han analizado las tres implementaciones para poder decir cuándo era mejor utilizar un modelo de aprendizaje automático o la red neuronal.

Estos objetivos anteriores han estado presentes en todo el desarrollo del sistema, el cual se ha llevado a cabo siguiendo la estructura planteada en el Capítulo 4. El hecho de haber dividido el desarrollo en distintas fases ha facilitado la creación de distintos objetivos derivados que permitían un mejor seguimiento. De esta forma, en cada punto del proyecto se podía tener una idea clara de lo que se debía hacer, puesto que tras la consecución de estos subobjetivos se acabaría alcanzando las metas generales propuestas.

Entre los subobjetivos más destacados se encuentran la elección de herramientas, la exploración y tratamiento del conjunto de datos y el desarrollo de las diferentes implementaciones agrupado en secciones más reducidas.

En resumen, el plan ideado ha permitido completar el proyecto satisfactoriamente, alcanzando unos resultados muy positivos que fomentan la utilización de técnicas de inteligencia artificial para la detección de enfemedades como la diabetes, aportando así una mayor variedad de alternativas disponibles.

8.1 Relación con los estudios cursados

Este proyecto hace un gran uso de los conocimientos adquiridos a lo largo del grado de ingeniería informática, motivo por el que se van a describir aquellos que han tenido mayor peso.

En primer lugar, como es esperable, asignaturas como "Introducción a la informática y a la programación" y "Programación" han tenido un papel realmente importante pa-

ra poder haber creado un código entendible y funcional que posibilitara la ejecución de los experimentos. Este código ha estado desarrollado con el lenguaje de programación Python, herramienta que he sabido utilizar gracias a su utilización en las asignaturas de "Sistemas de almacenamiento y recuperación de información" y "Algorítmica". Conjuntamente a las prácticas externas en empresa, todas estas asignaturas me han posibilitado generar la confianza suficiente para poder utilizarlo por mi cuenta, descubriendo gracias a eso instrumentos realmente útiles como Jupyter Notebook.

Por otro lado, "Sistemas inteligentes", "Percepción" y "Aprendizaje automático" son materias indispensables para este trabajo, puesto que es donde se fundamenta todo el conocimiento relacionado con el aprendizaje automático, concepto sobre el que gira el escrito. Sin ellas no se podría haber iniciado el proyecto en sí.

Por último, también es importante reconocer dos asignaturas que pueden no ser tan evidentes pero que han tenido un papel fundamental: "Gestión de proyectos" y "Estadística". La primera de ellas ha posibilitado seguir un plan de trabajo marcado por unos requisitos funcionales y no funcionales, sin lo que el desarrollo se habría retrasado notablemente, mientras que gracias a los conceptos vistos en la asignatura de estadística se ha podido analizar el conjunto de datos en detalle, facilitando la eliminación de los outliers y ayudando a predecir que partes debían recibir una mayor atención.

Además de haberse puesto en práctica los conocimientos adquiridos también se han desarrollado las siguientes competencias transversales:

- **Comprensión e integración**: Competencia empleada a la hora de entender el estado de la cuestión actual así como las diferentes técnicas existentes y cómo replicarlas.
- **Aplicación y pensamiento práctico**: En el proyecto se ha tenido que poner en práctica los conocimiento teóricos estudiados, siendo necesario adaptarse a la situación y no aplicar la misma fórmula una y otra vez.
- **Análisis y resolución de problemas**: La exploración de los datos y elaboración del plan del trabajo no habría sido posible sin esta capacidad.
- **Innovación, creatividad y emprendimiento**: Se ha intentado afrontar un problema cotidiano, la detección de la diabetes, de una forma diferente, aplicando una gran cantidad de técnicas y probando diferentes modelos.
- **Diseño y proyecto**: Se ha diseñado cuidadosamente el plan de trabajo a seguir, detallando las tareas y el orden de las mismas.
- **Responsabilidad ética, medioambiental y profesional**: Competencia puesta en uso realizando un análisis del marco legal y ético.
- **Comunicación efectiva**: Esta memoria ha sido redactada intentando expresar las ideas de una forma clara, rigurosa y convincente para que los lectores no tengan problema en entenderla.
- **Pensamiento crítico**: En números puntos de la memoria se han cuestionado las ideas establecida o los datos alcanzados, siendo el más curioso el descubrimiento de que todas las personas partícipes en el conjunto de datos padecían DM2.
- **Conocimiento de problemas contemporáneos**: Las tareas de clasificación mediante modelos de aprendizaje automático llenan los titulares hoy en día, sin duda alguna una cuestión de las más actuales.

- **Aprendizaje permanente**: En este proyecto se han utilizado un gran número de herramientas que eran totalmente desconocidas. Aprender su correcto funcionamiento y cómo buscar información acerca de ellas ha sido indispensable.
- **Planificación y gestión del tiempo**: La planificación y gestión del tiempo, aunque no ha sido perfecta, ha tenido una gran importancia, siendo las reuniones establecidas con el tutor del trabajo de gran ayuda para mantener un desarrollo correcto.
- **Instrumental específica**: Instrumentos como Jupyter Notebook y las diferentes librerías de Python empleadas (Scikit Learn, PyTorch, Pandas, etc.) son sin duda herramientas específicas asociadas a la rama de la titulación.

En definitiva, este proyecto cubre una extensa parte de las competencias vistas en el grado y, en concreto, de la rama de computación estudiada, hecho que ha servido para poder poner en práctica una gran cantidad de conocimientos y habilidades que se espera poder seguir explotando en el futuro laboral.

CAPÍTULO 9
Trabajos futuros

En el Capítulo 7 se han mencionado cuatro aspectos por abordar antes de realizar una hipotética implantación. Dichos aspectos pueden ser tratados como distintos trabajos futuros a realizar, ya que cuentan con una complejidad equiparable al trabajo realizado en este proyecto. Además, a lo largo del desarrollo se han observado una serie de puntos que podrían mejorarse.

El primero de ellos tiene que ver con la complejidad de los modelos diseñados. En todo momento se ha optado por la creación de unos modelos sencillos que permitieran la realización de un gran número de experimentos. No obstante, tal vez hubiera sido más beneficioso reducir este número a cambio de aplicar una técnicas más complejas que facilitasen alcanzar mejores resultados.

Por otro lado, también se han intentado abarcar demasiados modelos. Al final da la sensación de que no se ha tocado techo con ellos, por lo que sería sensato elegir alguno de ellos e intentar desarrollarlo al máximo.

Finalmente, en el Capítulo 8 se ha podido observar que el objetivo secundario no ha podido alcanzarse con todo el éxito que se hubiera querido, por lo que un estudio más profundo sobre los modelos finales podría ilustrar las mejores situaciones para cada uno.

En resumen, existen una gran cantidad de puntos por detallar que pueden dar mucho de sí, por lo que animo encarecidamente a todas las personas interesadas a intentar afrontarlos. Sus descubrimientos pueden llegar a facilitar la vida de muchas, y cada vez más, personas.

Bibliografía

[1] La diabetes a través de las estadísticas, 17 de julio de 2018. [En línea]. Disponible en https://www.freestylelibre.es/libre/diabetes-blog/la-diabetes-a-traves-de-las-estadisticas.html. [Último acceso: 24 de junio de 2021].

[2] Europa Press. La diabetes en España, en datos y gráficos, 25 de septiembre de 2020. [En línea]. Disponible en https://www.epdata.es/datos/diabetes-espana-datos-graficos/472. [Último acceso: 24 de junio de 2021].

[3] Europa Press. Evolución de la población con diabetes en España, 25 de septiembre de 2020. [En línea]. Imagen disponible en https://www.epdata.es/porcentaje-poblacion-mayor-quince-anos-diabetes-espana/4a31e195-50a7-4470-94ef-a85e6adc481b. [Último acceso: 24 de junio de 2021].

[4] National Institute of Diabetes and Digestive and Kidney Diseases. Diabetes Tests & Diagnosis, Diciembre 2016. [En línea]. Disponible en https://www.niddk.nih.gov/health-information/diabetes/overview/tests-diagnosis. [Último acceso: 18 de abril de 2021].

[5] S. Ghosh y A. Collier. *Diabetes*. Edinburgh : Churchill Livingston/Elsevier, segunda edición, 2012, pp. 1-28.

[6] Federación Española de Diabetes. Tipos de diabetes. [En línea]. Disponible en https://fedesp.es/diabetes/tipos/. [Último acceso: 24 de abril de 2021].

[7] V. Kumar. *Healthcare analytics made simple: techniques in healthcare computing using machine learning and Python*. Birmingham, England : Packt, primera edición, 2018, pp. 12-17.

[8] P. Sharma, K. Choudhary, K. Gupta, R. Chawla, D. Gupta y A. Sharma. "Artificial plant optimization algorithm to detect heart rate & presence of heart disease using machine learning", *Artificial Intelligence in Medicine*, vol. 102, enero 2020.

[9] G. Cho, J. Yim, Y. Choi, J. Ko y S.-H. Lee. "Review of Machine Learning Algorithms for Diagnosing Mental Illness", *Psychiatry investigation*, vol. 16, no. 4, pp. 262–269, 2019.

[10] M. J. Maenner, M. Yeargin-Allsopp, K. Van Naarden Braun, D. L. Christensen y L. A. Schieve. "Development of a Machine Learning Algorithm for the Surveillance of Autism Spectrum Disorder", *PLoS ONE*, vol. 11, no. 12, diciembre 2016.

[11] A. T. Joseph. This startup uses A.I. to detect breast cancer early, 30 de abril de 2021. [En línea]. Disponible en https://www.fortuneindia.com/enterprise/this-startup-uses-ai-to-detect-breast-cancer-early/105450. [Último acceso: 1 de mayo de 2021].

[12] R. Jagannathan, J. S. Neves, B. Dorcely, S. T. Chung, K. Tamura, M. Rhee y M. Bergman. "The Oral Glucose Tolerance Test: 100 Years Later", *Dove Medical Press*, vol. 2020:13, pp. 3787-3805, octubre 2020.

[13] B. Ballinger et al. "DeepHeart: Semi-Supervised Sequence Learning for Cardiovascular Risk Prediction", *AAAI*, vol. 32, no. 1, febrero 2018.

[14] Agencia Española de Protección de Datos. "Guía para pacientes y usuarios de la Sanidad", noviembre 2019.

[15] Diabetes Self-Management. Beware of This Diabetes Scam, 16 de agosto de 2019. [En línea]. Disponible en https://www.diabetesselfmanagement.com/blog/beware-of-this-diabetes-scam/. [Último acceso: 15 de julio de 2021].

[16] Creative Commons. Sobre las licencias. [En línea]. Disponible en https://creativecommons.org/licenses/?lang=es. [Último acceso: 18 de julio de 2021].

[17] N. Pramanik. Kernel Density Estimation, 24 de septiembre de 2019. [En línea]. Disponible en https://medium.com/analytics-vidhya/kernel-density-estimation-kernel-construction-and-bandwidth-optimization-using-maximum-b1dfce127073. [Último acceso: 25 de julio de 2021].

[18] Drleft, Wikipedia. Comparison of a histogram and a kernel density estimate, 28 de septiembre de 2010. [En línea]. Imagen disponible en https://en.wikipedia.org/wiki/Kernel_density_estimation#/media/File:Comparison_of_1D_histogram_and_KDE.png. [Último acceso: 25 de julio de 2021].

[19] L. J. Baier y R. L. Hanson. "Genetic Studies of the Etiology of Type 2 Diabetes in Pima Indians", *Diabetes*, vol. 53, no. 5, pp. 1181–1186, mayo 2004.

[20] G. Bonaccorso. *Machine learning algorithms: popular algorithms for data science and machine learning*. Birmingham, Mumbai : Packt Publishing, segunda edición, 2018, pp. 164-175, 185-290 y 475-476.

[21] J. Martínez Heras. Máquinas de Vectores de Soporte, 28 de mayo de 2019. [En línea]. Disponible en https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/. [Último acceso: 29 de julio de 2021].

[22] J. Brownlee. A Gentle Introduction to k-fold Cross-Validation, 3 de agosto de 2020. [En línea]. Disponible en https://machinelearningmastery.com/k-fold-cross-validation/. [Último acceso: 29 de julio de 2021].

[23] J. Brownlee. Train-Test Split for Evaluating Machine Learning Algorithms, 26 de agosto de 2020. [En línea]. Disponible en https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/. [Último acceso: 29 de julio de 2021].

[24] I. Vasilev, D. Slater, G. Spacagna, P. Roelants y V. Zocca. *Python deep learning: exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*. Birmingham: Packt Publishing Ltd., segunda edición, 2019.

[25] F. Chollet. *Deep learning with Python*. Shelter Island, NY: Manning, 2018.