Clasificación de tumores del cáncer de mama aplicando técnicas de aprendizaje automático

Adrián Barrios Trujillo

Grado de Ingeniería Informática

Inteligencia Artificial

Nombre Consultor/a: Elena Álvarez de la Campa

Nombre Profesor/a responsable de la asignatura: Susana Acedo Nadal

Enero de 2024

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Clasificación de tumores del cáncer de mama aplicando técnicas de aprendizaje automático*
Nombre del autor:	*Adrián Barrios Trujillo*
Nombre del consultor/a:	*Elena Álvarez de la Campa*
Nombre del PRA:	*Susana Acedo Nadal*
Fecha de entrega (mm/aaaa):	01/2024
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Inteligencia Artificial*
Idioma del trabajo:	Castellano
Número de créditos:	12
Palabras clave	*Breast cancer, tumor, machine learning*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El cáncer de mama, la forma más común de cáncer entre las mujeres de todo el mundo, tiene un impacto significativo en la salud global. La detección temprana es esencial para mejorar la tasa de supervivencia de los pacientes y mejorar su calidad de vida.

Por tanto, en este trabajo se han desarrollado modelos con capacidad de detección y clasificación de tumores mamarios mediante la aplicación de técnicas de aprendizaje automático, con el propósito de mostrar cómo esta tecnología puede ayudar a mejorar la detección de la enfermedad.

Para abordar este trabajo, se comienza con una presentación de la enfermedad del cáncer de mama, ofreciendo una visión global de qué es y cómo afecta a las personas. A continuación, se realiza un análisis de diversas técnicas de aprendizaje automático aplicadas en este estudio. Se explica en qué consisten, qué información aportan y cómo pueden ayudar a la clasificación de los tumores.

Los resultados obtenidos incluyen una serie de modelos entrenados para la clasificación de tumores malignos con un rendimiento superior al 90%, y cuyo rendimiento varía según el tratamiento previo de los datos.

Abstract (in English, 250 words or less):

Breast cancer, the most common form of cancer among women worldwide, has a significant impact on global health. Early detection is essential to increase the survival rate of patients and enhance their quality of life.

Therefore, in this study models with the capability to detect and classify breast tumors through the application of machine learning techniques have been developed, with the purpose of showing how this technology can help to improve the detection of this disease.

To address this work, we start with a presentation of the breast cancer disease, providing a global perspective on what it is and how it affects people. Next, an analysis of various machine learning techniques applied in this study is carried out. It is explained what they consist of, what information they provide, and how they can help in the classification of tumors.

The results obtained include a series of models trained for the classification of malignant tumors, achieving a performance exceeding 90% and whose performance varies according on the previous treatment of the data.

ÍNDICE

Lista de figuras

Lista de tablas

1. Introducción

1.1. Contexto y justificación del trabajo

El cáncer de mama es la forma más común de cáncer entre las mujeres y el tumor más diagnosticado en el mundo en 2021, superando incluso al cáncer de pulmón. Posee un impacto significativo en la salud de las mujeres de todo el mundo, por lo que su detección y diagnóstico temprano es esencial para ofrecer un tratamiento eficaz que aumente la tasa de supervivencia y mejore la calidad de vida de los pacientes.

Según la Organización Mundial de la Salud (OMS o WHO por sus siglas en inglés), en 2020 se registraron aproximadamente 2,3 millones de mujeres con cáncer de mama, y 685.000 fallecimientos debidos a esta enfermedad. A finales de año, 7,8 millones de mujeres que habían recibido el diagnóstico en los cinco años anteriores permanecían con vida, lo que lo posiciona como el tipo de cáncer más prevalente en el mundo.

Esta enfermedad también afecta a los hombres, aunque su nivel de incidencia representa entre el 0,5% y el 1% de los nuevos diagnósticos, y un 15-20% de los casos poseen antecedentes familiares, permitiendo un diagnóstico de cáncer hereditario que facilita la identificación de mutaciones genéticas que favorezcan esta enfermedad.

A partir de 1990, las tasas de supervivencia empezaron a mejorar gracias a campañas de detección temprana del cáncer de mama, complementadas con programas integrales de tratamiento con terapias farmacológicas eficaces.

Es por esta razón que este trabajo se enfoca en desarrollar un sistema de detección y clasificación de tumores mamarios mediante técnicas de aprendizaje automático. Su finalidad es distinguir entre tumores benignos y malignos, lo cual resulta crucial para determinar el tratamiento adecuado en la lucha contra esta enfermedad.

1.2. Objetivos del trabajo

1.2.1. Objetivo general

El objetivo principal de este trabajo es crear un modelo de *machine learning* capaz de analizar los datos de los tumores de cáncer de mama de varios pacientes, determine si son de carácter benigno o maligno, y facilite el diagnóstico precoz de esta enfermedad.

1.2.2. Objetivos específicos

Respecto a los objetivos específicos, se definen los siguientes:

- Realizar un estudio de la literatura médica para profundizar en la comprensión de la enfermedad del cáncer de mama, así como su detección y tratamiento.
- Presentar la base de datos seleccionada para la creación del modelo, identificando la información que aporta y explicando cómo se puede emplear para desarrollar nuestro sistema de detección y clasificación.
- Analizar en detalle las técnicas de aprendizaje automático que se emplearán en este trabajo, describiendo su funcionamiento y los resultados que se esperan obtener.
- Implementar los algoritmos de *machine learning* y evaluar el rendimiento del modelo en base a métricas clave como la exactitud, la precisión, la sensibilidad o el F1-score.
- Realizar un estudio minucioso de los resultados obtenidos del modelo, destacando tanto sus ventajas como las limitaciones encontradas respecto al presente contexto.

1.3. Enfoque y método seguido

Para la realización de este trabajo, optamos por emplear el conjunto de datos "*Breast Cancer Wisconsin (Diagnostic) Data Set*", disponible en Kaggle. Este *dataset* es ampliamente reconocido como uno de los más populares para proyectos de clasificación en *machine learning*, y nos proporciona una base ideal para nuestra investigación.

Para desarrollar nuestro modelo, comenzamos abordando los pasos convencionales del desarrollo de un proyecto de aprendizaje automático, que incluyen la observación y la limpieza de datos, seguido del análisis exploratorio de datos (EDA, por sus siglas en inglés), el tratamiento del conjunto de datos, y la creación final del modelo en base a distintos algoritmos.

La valoración del rendimiento de nuestro modelo ha sido elaborada con diversas métricas; sin embargo, no deseamos que nuestro modelo clasifique erróneamente los tumores, por lo que hemos priorizado la sensibilidad como la métrica a considerar, sin descartar las demás.

1.4. Planificación del trabajo

1.4.1. Tareas

Se ha desglosado una serie de tareas a realizar durante el desarrollo del trabajo, fundamentado en el análisis de los contenidos del proyecto, el calendario y las fechas de entrega.

En la tabla siguiente se especificarán las tareas asignadas para cada entrega, denominadas Pruebas de Evaluación Continua (PEC), junto con una estimación de la duración, la fecha de inicio y la fecha de conclusión de cada tarea, así como del conjunto de estas actividades dentro de la PEC.

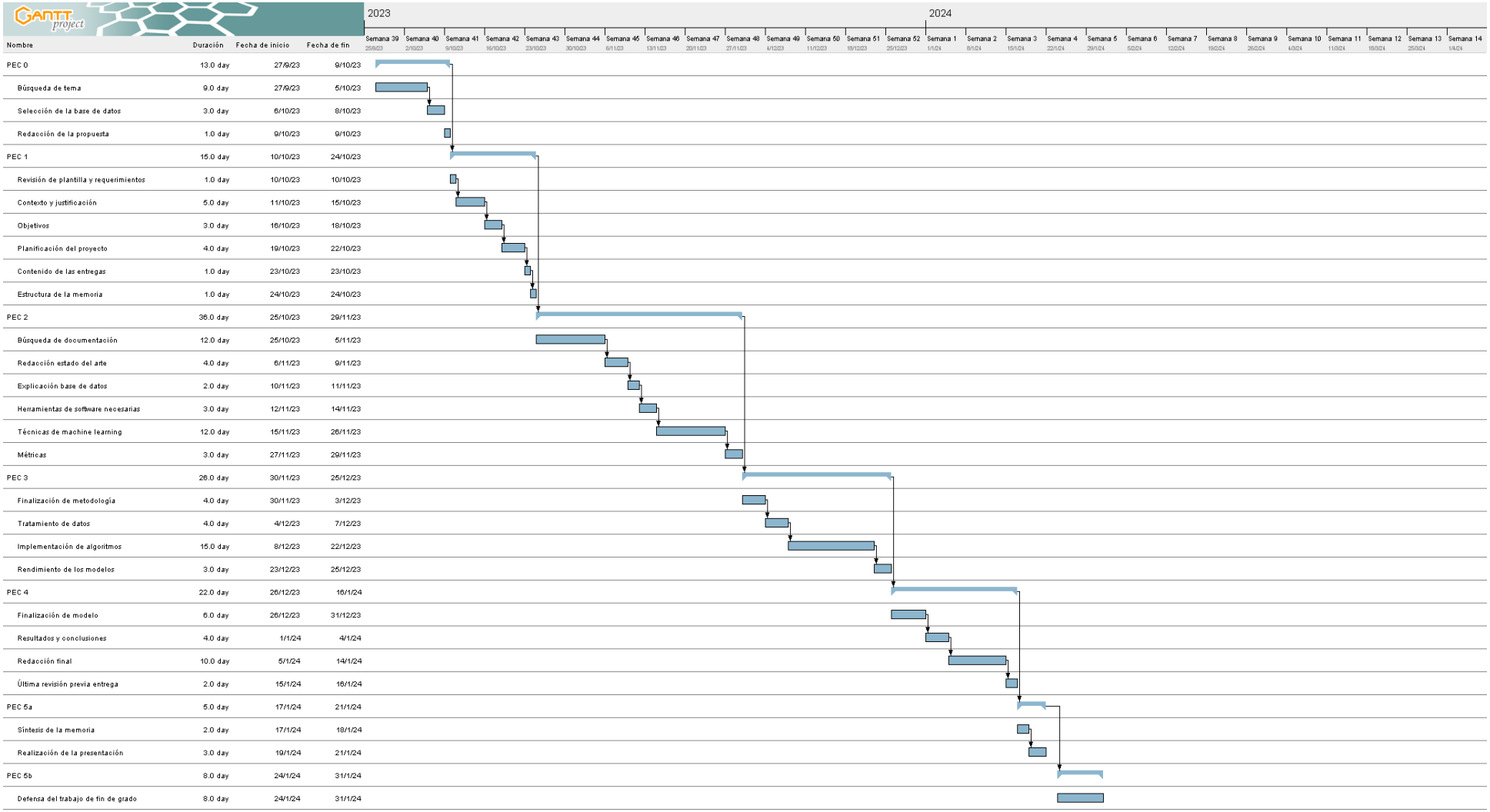
Tabla 1. Tareas y plazos

Nombre	Duración	Fecha de inicio	Fecha de fin
PEC 0	**13 días**	**27/09/2023**	**09/10/2023**
Búsqueda de tema	9 días	27/09/2023	05/10/2023
Selección de la base de datos	3 días	06/10/2023	08/10/2023
Redacción de la propuesta	1 días	09/10/2023	09/10/2023
PEC 1	**15 días**	**10/10/2023**	**24/10/2023**
Revisión de plantilla y requerimientos	1 días	10/10/2023	10/10/2023
Contexto y justificación	5 días	11/10/2023	15/10/2023
Objetivos	3 días	16/10/2023	18/10/2023
Planificación del proyecto	4 días	19/10/2023	22/10/2023
Contenido de las entregas	1 días	23/10/2023	23/10/2023
Estructura de la memoria	1 días	24/10/2023	24/10/2023
PEC 2	**36 días**	**25/10/2023**	**29/11/2023**
Búsqueda de documentación	12 días	25/10/2023	05/11/2023
Redacción estado del arte	4 días	06/11/2023	09/11/2023
Explicación base de datos	2 días	10/11/2023	11/11/2023
Herramientas de software necesarias	3 días	12/11/2023	14/11/2023
Técnicas de machine learning	12 días	15/11/2023	26/11/2023
Métricas	3 días	27/11/2023	29/11/2023
PEC 3	**26 días**	**30/11/2023**	**25/12/2023**
Finalización de metodología	4 días	30/11/2023	03/12/2023
Tratamiento de datos	4 días	04/12/2023	07/12/2023
Implementación de algoritmos	15 días	08/12/2023	22/12/2023
Rendimiento de los modelos	3 días	23/12/2023	25/12/2023
PEC 4	**22 días**	**26/12/2023**	**16/01/2024**
Finalización de modelo	6 días	26/12/2023	31/12/2023
Resultados y conclusiones	4 días	01/01/2024	04/01/2024
Redacción final	10 días	05/01/2024	14/01/2024
Última revisión previa entrega	2 días	15/01/2024	16/01/2024
PEC 5a	**5 días**	**17/01/2024**	**21/01/2024**
Síntesis de la memoria	2 días	17/01/2024	18/01/2024
Realización de la presentación	3 días	19/01/2024	21/01/2024
PEC 5b	**8 días**	**24/01/2024**	**31/01/2024**
Defensa del trabajo de fin de grado	8 días	24/01/2024	31/01/2024

1.4.2. Calendario

En la figura 1 definimos el calendario de ejecución de las tareas, realizado con GattProject. Se puede apreciar el cronograma en la siguiente página:

Figura 1. Cronograma



Nombre	Duración	Fecha de inicio	Fecha de fin
PEC 0	13.0 day	27/9/23	9/10/23
Búsqueda de tema	9.0 day	27/9/23	5/10/23
Selección de la base de datos	3.0 day	6/10/23	8/10/23
Redacción de la propuesta	1.0 day	9/10/23	9/10/23
PEC 1	15.0 day	10/10/23	24/10/23
Revisión de plantilla y requerimientos	1.0 day	10/10/23	10/10/23
Contexto y justificación	5.0 day	11/10/23	15/10/23
Objetivos	3.0 day	16/10/23	18/10/23
Planificación del proyecto	4.0 day	19/10/23	22/10/23
Contenido de las entregas	1.0 day	23/10/23	23/10/23
Estructura de la memoria	1.0 day	24/10/23	24/10/23
PEC 2	36.0 day	25/10/23	29/11/23
Búsqueda de documentación	12.0 day	25/10/23	5/11/23
Redacción estado del arte	4.0 day	6/11/23	9/11/23
Explicación base de datos	2.0 day	10/11/23	11/11/23
Herramientas de software necesarias	3.0 day	12/11/23	14/11/23
Técnicas de machine learning	12.0 day	15/11/23	26/11/23
Métricas	3.0 day	27/11/23	29/11/23
PEC 3	26.0 day	30/11/23	25/12/23
Finalización de metodología	4.0 day	30/11/23	3/12/23
Tratamiento de datos	4.0 day	4/12/23	7/12/23
Implementación de algoritmos	15.0 day	8/12/23	22/12/23
Rendimiento de los modelos	3.0 day	23/12/23	25/12/23
PEC 4	22.0 day	26/12/23	16/1/24
Finalización de modelo	6.0 day	26/12/23	31/12/23
Resultados y conclusiones	4.0 day	1/1/24	4/1/24
Redacción final	10.0 day	5/1/24	14/1/24
Última revisión previa entrega	2.0 day	15/1/24	16/1/24
PEC 5a	5.0 day	17/1/24	21/1/24
Síntesis de la memoria	2.0 day	17/1/24	18/1/24
Realización de la presentación	3.0 day	19/1/24	21/1/24
PEC 5b	8.0 day	24/1/24	31/1/24
Defensa del trabajo de fin de grado	8.0 day	24/1/24	31/1/24

1.4.3. Hitos

Los hitos corresponden a las fechas de entrega de cada PEC, y a lo largo de este trabajo, han servido como referencia para valorar el avance de las tareas realizadas. Resumimos en la siguiente tabla las mencionadas entregas:

Tabla 2. Hitos

Descripción del hito	PEC	Fecha de entrega
Redacción de la propuesta	PEC 0	09/10/2023
Definición del proyecto y estructura	PEC 1	24/10/2023
Primera fase del desarrollo de trabajo	PEC 2	29/11/2023
Segunda fase del desarrollo de trabajo	PEC 3	25/12/2023
Cierre de la memoria	PEC 4	16/01/2024
Presentación	PEC 5a	21/01/2024
Defensa del trabajo final de grado	PEC 5b	31/01/2024

1.5. Breve sumario de productos obtenidos

Tras la finalización de este proyecto, hemos logrado crear una serie de modelos de clasificación de tumores malignos mediante la aplicación de diversos algoritmos de aprendizaje automático y técnicas específicas de tratamiento de datos, los cuales serán detallados en las secciones subsiguientes.

Asimismo, en el transcurso de la realización de este trabajo, hemos contemplado posibles vías de investigación futuras que, aunque no fueron implementadas en este proyecto en particular, resultarían valiosas para su exploración debido al interés científico que puedan generar.

1.6. Breve descripción de los capítulos de la memoria

Los capítulos subsiguientes de la memoria comenzarán con una revisión exhaustiva del estado del arte, explorando en profundidad la enfermedad del cáncer de mama para proporcionar un contexto completo al análisis. Luego, nos adentraremos en la metodología empleada, destacando la elección del conjunto de datos, el *software* utilizado, los algoritmos de *machine learning* aplicados y las métricas de rendimiento seleccionadas.

Siguiendo esta introducción teórica, nos sumergiremos en la creación del modelo, realizando un análisis minucioso y un tratamiento cuidadoso de los datos del conjunto para asegurar la correcta aplicación de los algoritmos de aprendizaje automático.

Finalmente, procederemos a la fase de revisión de resultados y conclusiones. Este último capítulo ofrecerá un análisis de los logros alcanzados y las percepciones extraídas a lo largo del proyecto.

2. Estado del arte

2.1. Introducción sobre el término cáncer

El cáncer, según la definición de la Real Academia Española (RAE), es una "enfermedad neoplásica con transformación de las células, que proliferan de manera anormal e incontrolada". En términos más simples, esta enfermedad implica la presencia de una masa anormal de tejido que surge cuando las células se multiplican de manera descontrolada o no mueren cuando deberían.

El Instituto Nacional del Cáncer de Estados Unidos, de los Institutos Nacionales de la Salud (NIH o The National Institutes of Health), amplía esta descripción al señalar que el cáncer engloba diversas enfermedades donde las células anormales proliferan sin control e invaden tejidos cercanos. Este crecimiento incontrolado puede llevar a que las células cancerosas se diseminen a otras partes del cuerpo mediante el sistema sanguíneo y linfático.

La Organización Mundial de la Salud (OMS o WHO por sus siglas en inglés, World Health Organization) proporciona una perspectiva más amplia, definiendo el cáncer como un término genérico para un grupo diverso de enfermedades. En términos de la OMS, "«Cáncer» es un término genérico utilizado para designar un amplio grupo de enfermedades que pueden afectar a cualquier parte del organismo". En esta definición, la OMS destaca la multiplicación rápida de células anormales y su capacidad para invadir partes adyacentes del cuerpo o propagarse a otros órganos, en un proceso conocido como "metástasis". La extensión de estas metástasis es la principal causa de muerte asociada al cáncer, según datos de la organización.

En el año 2020, el cáncer se erigió como la principal causa de muerte a nivel mundial, con casi 10 millones de defunciones atribuidas a esta enfermedad. Entre los tipos de cáncer más prevalentes, se destacaron el de mama, pulmón, colorrectal, próstata, piel y gástrico.

Estas estadísticas reflejan la magnitud del impacto del cáncer en la salud pública, delineando la urgencia de comprender y abordar sus distintas manifestaciones. Por esta razón, y dentro de este amplio panorama de enfermedades, el presente trabajo se enfocará en una de las formas más relevantes de cáncer y el tipo más diagnosticado en 2021: el cáncer de mama.

2.2. Descripción de la enfermedad del cáncer de mama

La mama está formada por entre 10 y 20 secciones llamadas lóbulos, desglosados en segmentos más diminutos conocidos como lobulillos. Estos elementos, a su vez, albergan glándulas encargadas de una función vital durante la lactancia: la producción de leche. El líquido fluye desde los lobulillos hacia el pezón a través de conductos especializados, contribuyendo así al bienestar del recién nacido. Entre los lobulillos y los ductos, el espacio está repleto de tejido graso y fibroso, creando una estructura intricada y funcional.

Los vasos linfáticos presentes en las mamas son conducciones esenciales hacia los ganglios linfáticos, pequeños órganos redondos que cumplen funciones de protección al atrapar bacterias, células tumorales y otras sustancias nocivas. Este sistema linfático, con sus conexiones fundamentales a los ganglios linfáticos axilares, desempeña un papel crucial en la respuesta inmunitaria y la detección temprana de anomalías.

El cáncer de mama se manifiesta como una enfermedad que impacta principalmente a las células mamarias, consolidándose como una de las afecciones más prevalentes a nivel global. Esta condición se caracteriza por la proliferación incontrolada de células en la mama que, de no recibir tratamiento, puede desencadenar la formación de tumores y, en fases avanzadas, propiciar la propagación a otras partes del cuerpo, presentando un riesgo significativo para la vida de los individuos afectados, con consecuencias potencialmente fatales.

El desarrollo inicial de este tipo de cáncer tiene lugar en los conductos lactíferos o en los lobulillos de la mama. La detección temprana emerge como un factor crítico para un tratamiento eficaz, dado que permite intervenir en las primeras etapas de la enfermedad, cuando es más tratable y las opciones terapéuticas son más efectivas.

2.3. Factores de riesgo

Si bien las causas específicas del cáncer de mama no están completamente esclarecidas, diversos factores de riesgo han sido identificados, siendo la edad un factor clave, ya que la máxima incidencia se observa típicamente después de los 50 años. Sin embargo, un dato notable es que alrededor del 10% de los casos se diagnostican en mujeres menores de 40 años, subrayando la importancia de la conciencia y detección temprana incluso en edades tempranas.

Otro factor relevante son los antecedentes reproductivos, ya que la exposición hormonal a lo largo de la vida se ve modulada por eventos como la edad de la primera regla, la menopausia tardía y la nuliparidad (no haber estado embarazada nunca).

Otros factores de riesgo incluyen historial personal de cáncer de mama invasivo (que aumenta el riesgo de padecer cáncer de mama contralateral), hiperplasia atípica, densidad mamaria elevada en mamografías, uso de terapia hormonal sustitutiva después de la menopausia, exposición a radiaciones ionizantes (particularmente durante la pubertad), consumo de alcohol y obesidad.

Una categoría destacada es el cáncer de mama hereditario, que representa aproximadamente entre el 5 y el 10% de los casos, donde la enfermedad tiene su origen en alguna mutación genética. Mutaciones en los genes BRCA1 y BRCA2 comportan el 20-25% de los casos hereditarios, y la probabilidad de cáncer de mama en portadoras de mutación en BRCA1 o BRCA2 se sitúa alrededor del 57% a los 70 años. Además de estos genes, otros como PALB2, p53, CDH1, ATM, CHEK2 o BRIP1 también contribuyen a aumentar la probabilidad de padecer la enfermedad a lo largo de la vida.

2.4. Síntomas

El cáncer de mama puede manifestarse con una variedad de síntomas, siendo importante destacar que, en las fases iniciales, la mayoría de las personas pueden no experimentar ninguna manifestación clínica. Sin embargo, en etapas más avanzadas, se pueden observar combinaciones de síntomas que alertan sobre la presencia de la enfermedad.

Entre los síntomas comunes del cáncer de mama se incluyen la presencia de un nódulo o engrosamiento en el seno, a menudo sin dolor aparente. Además, cambios en el tamaño, forma o apariencia del seno, la aparición de hoyuelos, enrojecimiento, grietas u otros cambios en la piel, y alteraciones en el aspecto del pezón o la piel circundante (aréola) pueden ser señales de alerta. La secreción de líquido anómalo o sanguinolento por el pezón también se cuenta entre los posibles síntomas.

Es crucial destacar que la presencia de un nódulo anómalo en el seno debe motivar la búsqueda inmediata de atención médica, incluso si no se experimenta dolor. Aunque la mayoría de los nódulos en los senos no son cancerosos, el tratamiento efectivo es más probable cuando los nódulos son pequeños y no se han propagado a los ganglios linfáticos cercanos.

En algunos casos, el cáncer de mama puede avanzar y propagarse a otras partes del cuerpo, dando lugar a síntomas adicionales. Por ejemplo, la propagación a los ganglios linfáticos de la axila es un lugar común para la detección de la diseminación del cáncer. Con el tiempo, las células cancerosas pueden llegar a otras áreas, como los pulmones, el hígado, el cerebro y los huesos, dando lugar a nuevos síntomas relacionados con el cáncer, como dolor óseo o cefaleas.

2.5. Diagnóstico

El diagnóstico del cáncer de mama involucra pruebas de imagen como mamografías, ecografías y resonancias magnéticas, las cuales orientan pero no confirman el diagnóstico. La biopsia, que implica la extracción de tejido para su análisis microscópico, es necesaria para confirmar si hay presencia de células malignas. La biopsia puede realizarse mediante diferentes métodos, incluyendo agujas finas o gruesas, guiadas por palpación o ecografía.

Tras la confirmación del diagnóstico con biopsia, se realizan pruebas adicionales para evaluar la extensión del cáncer a otros órganos. Esto puede incluir radiografías de tórax, ecografías abdominales, gammagrafías óseas, tomografías axiales computerizadas (TAC) y tomografías por emisión de positrones (PET). Estas pruebas ayudan a determinar el estadio y guiar el plan de tratamiento.

El diagnóstico patológico final lo realiza un especialista en anatomía patológica, quien examina la muestra de la biopsia bajo el microscopio. El patólogo clasifica el cáncer si es invasivo o no invasivo; este último tipo también se denomina como

in situ, y son cánceres de mama que no se han extendido fuera del conducto (ductales in situ) o del lobulillo (lobulillares in situ).

El especialista analiza aspectos como el tamaño tumoral, tipo histológico, grado histológico, afectación ganglionar, receptores hormonales, las proteínas HER-2 y Ki-67, y perfiles de expresión génica. Esta información es crucial para determinar el pronóstico y el tratamiento más adecuado.

2.6. Estadificación

La supervivencia ante el cáncer de mama varía considerablemente según el estadio en el que se realiza el diagnóstico. Mientras que el estadio I, que es la etapa inicial, muestra una supervivencia superior al 98%, esta cifra disminuye drásticamente al 24% en el estadio IV, donde la enfermedad está en su fase más avanzada y se ha extendido a otras partes del cuerpo.

Para catalogar estos estadios, se emplea el sistema de clasificación TNM, donde se observan aspectos como el tamaño del tumor (T), junto a la extensión a ganglios linfáticos regionales (N) o hacia otras partes del cuerpo (M). Esta categorización se realiza después de haberse extraído el tumor y analizado el estado de los ganglios axilares. A continuación, se presenta una tabla extraída de la Sociedad Española de Oncología Médica o SEOM:

Tabla 3. Estadios del cáncer de mama

ESTADIO	SUB-ESTADIOS	
Estadio 0 o carcinoma in situ	**Carcinoma lobulillar in situ:** lesión en la que hay células anómalas en el revestimiento del lobulillo. Raramente se convierte en cáncer invasor, pero aumenta el riesgo de padecer cáncer de mama tanto en la mama de la lesión como en la contraletaral.	
	Carcinoma ductal in situ o carcinoma intraductal: lesión en la que hay células anómalas en el revestimiento de un conducto. No es una lesión invasiva, pero si se deja evolucionar, puede convertirse en un carcinoma infiltrante o invasor.	
Estadio I	El tumor mide menos de 2 cm y no se ha diseminado fuera de la mama.	
Estadio II	El tumor mide menos de 2 cm, pero ha afectado a ganglios linfáticos de la axila.	
	El tumor mide de 2 a 5 cm (con o sin diseminación ganglionar axilar).	
	El tumor mide más de 5 cm, pero no ha afectado a los ganglios linfáticos axilares.	
Estadio III o localmente avanzado	Estadio IIIA	El tumor mide más de 5 cm y se ha diseminado a los ganglios linfáticos axilares o a los ganglios situados detrás del esternón.
		El tumor mide menos de 5 cm y se ha diseminado a los ganglios linfáticos axilares de forma palpable o a los ganglios situados detrás del esternón.
	Estadio IIIB	Es un tumor de cualquier tamaño que afecta a la pared del tórax o a la piel de mama.
	Estadio IIIC Tumor de cualquier tamaño con:	Afectación de más de 10 ganglios axilares.
		Afectación de ganglios axilares y de ganglios situados detrás del esternón.
		Afectación de ganglios situados por debajo o por encima de la clavícula.
Estadio IV	El tumor se ha diseminado a otras partes del cuerpo.	

2.7. Tratamiento

El tratamiento del cáncer de mama es un proceso complejo que implica diversas estrategias y opciones, y la elección del enfoque adecuado depende de varios factores. Entre estos factores se incluyen la ubicación y el tamaño del tumor, la extensión de la enfermedad, las preferencias del paciente y las características individuales del tumor, como los receptores hormonales y la expresión del receptor HER2. Estos elementos guían a los profesionales en la determinación del plan de tratamiento más efectivo y personalizado para cada caso.

Una de las opciones comunes es la cirugía conservadora, donde se extirpa el tumor junto con el tejido sano circundante, preservando la mama en la medida de lo posible. Sin embargo, la viabilidad de esta cirugía depende en gran medida de factores como la ubicación y el tamaño del tumor. En situaciones más extensas o en casos que así lo prefiera la paciente, la mastectomía total (la extirpación de toda la mama, con tejido sano incluido) es otra alternativa, seguida a veces de la reconstrucción mamaria.

La evaluación de los ganglios linfáticos de la axila es crucial en la determinación del pronóstico, y aquí se presentan opciones como la linfadenectomía, que, aunque puede ser necesaria, conlleva efectos secundarios como adormecimiento o hinchazón del brazo. Para mitigar estos efectos, en algunos casos se utiliza la técnica del ganglio centinela, que localiza el primer ganglio linfático drenado por el tumor para su examen. No obstante, esta técnica no es apta para todos los casos y requiere un equipo multidisciplinario.

En adición a la cirugía, se emplean tratamientos adicionales, como la radioterapia, que utiliza rayos X de alta energía para eliminar células tumorales residuales post cirugía, y la terapia sistémica, que actúa en todo el organismo. La terapia sistémica incluye la quimioterapia, que utiliza drogas para detener el crecimiento o matar células tumorales, y la hormonoterapia, que interrumpe la acción hormonal que impulsa el crecimiento del tumor. Además, las terapias dirigidas, como anticuerpos monoclonales o inhibidores de ciclinas, atacan específicamente las células tumorales sin afectar las normales.

En casos de cáncer de mama triple negativo avanzado, la inmunoterapia se presenta como una opción, aprovechando el sistema inmune del paciente para combatir el cáncer. Medicamentos como pembrolizumab y atezolizumab están aprobados para este propósito.

También se diferencia el tipo de tratamiento según el estadio de la enfermedad: En etapas iniciales, se emplea el tratamiento adyuvante para erradicar células no detectables por la cirugía, utilizando quimioterapia, hormonoterapia y terapias dirigidas según el riesgo de recurrencia. En situaciones de estadios avanzados, el objetivo es la cronificación y paliación, y el tratamiento sistémico se selecciona según las características tumorales y las preferencias del paciente.

2.8. Seguimiento

Después de completar el tratamiento para estadios precoces de cáncer de mama, es crucial llevar a cabo un seguimiento adecuado para garantizar una supervisión continua de la salud. Este seguimiento se realiza cada 4-6 meses durante los primeros 5 años y posteriormente de forma anual. Incluye una exhaustiva revisión de la historia clínica, exploración física y una mamografía anual de la mama afectada y la contralateral. Aunque no se ha demostrado que un seguimiento intensivo con numerosas exploraciones incremente la supervivencia, es esencial para la detección temprana de posibles recurrencias.

Para aquellas pacientes que han recibido tratamiento con tamoxifeno, se aconseja realizar una revisión ginecológica anual, ya que este medicamento se ha asociado con un aumento del riesgo de cáncer de útero. En el caso de tratamientos con inhibidores de la aromatasa, se deben realizar densitometrías periódicas para evaluar la descalcificación ósea, un posible efecto secundario de estos fármacos.

También es fundamental que las mujeres que han superado el cáncer de mama mantengan un peso saludable, ya que el sobrepeso puede aumentar el riesgo de recaída. Además, se recomienda la práctica regular de actividad física, dado que el ejercicio físico se ha vinculado a un mejor pronóstico. Estos hábitos de vida saludables contribuyen significativamente al bienestar general y a la prevención de posibles complicaciones a largo plazo.

2.9. Impacto global

El cáncer de mama ha emergido como la principal causa global de incidencia de cáncer en 2020, superando al cáncer de pulmón, con 2,3 millones de nuevos casos, representando el 11,7% de todos los diagnósticos. A nivel mundial, ocupa el quinto lugar en mortalidad por cáncer, con 685.000 fallecimientos.

Las tasas de incidencia son notablemente más altas en países desarrollados, alcanzando un 88% más que en países en desarrollo, siendo Australia/Nueva Zelanda, Europa Occidental y América del Norte las regiones con tasas más elevadas. Sin embargo, las tasas de mortalidad son un 17% más altas en países en desarrollo, con Melanesia, África Occidental y Micronesia/Polinesia mostrando las tasas más altas.

La evolución de las tasas de incidencia de cáncer de mama ha sido bastante marcada en las últimas décadas. Durante los años 80 y 90, hubo un rápido aumento en países de América del Norte, Oceanía y Europa, atribuido a cambios en la prevalencia de factores de riesgo y a la adopción generalizada de la detección mamográfica. Desde principios de la década de 2000, las tasas se estabilizaron o incluso disminuyeron en algunos lugares, vinculadas a la reducción en el uso de terapia hormonal y posiblemente a un estancamiento en la participación en detección. Sin embargo, desde 2007, las tasas de incidencia han experimentado un lento aumento en los Estados Unidos y en varios países europeos y oceánicos.

Las disparidades en las tasas de supervivencia son evidentes, siendo el África subsahariana una región donde las tasas de incidencia y mortalidad aumentaron significativamente entre mediados de la década de 1990 y mediados de la década de 2010. La supervivencia es especialmente baja debido a presentaciones en etapas avanzadas, con un 77% de casos en etapas III/IV en el momento del diagnóstico. Se estima que entre el 28% y el 37% de las muertes por cáncer de mama en países subsaharianos podrían prevenirse mediante un diagnóstico más temprano y tratamiento adecuado.

La promoción de la conciencia sobre el cáncer de mama, el examen clínico y un tratamiento oportuno son imperativos en estas regiones. A pesar de las limitaciones de la mamografía, los programas de detección mamográfica a nivel de población siguen siendo esenciales para reducir la mortalidad a nivel mundial.

Además de comprender la magnitud y la complejidad del impacto global del cáncer de mama, la investigación sobre nuevas técnicas de detección conforma una valiosa contribución en la lucha contra el cáncer de mama. Por esta razón, la clasificación de tumores de cáncer de mama mediante técnicas de aprendizaje automático se presenta como una valiosa contribución en este ámbito. Al mejorar la precisión en la identificación de tumores, esta tecnología tiene el potencial de facilitar diagnósticos más tempranos y eficientes.

3. Metodología

En el presente capítulo, expondremos la base de datos utilizada y detallaremos las herramientas de software que serán empleadas a lo largo de este proyecto. Además, se proporcionará una explicación detallada sobre los distintos algoritmos de aprendizaje automático que se utilizarán en la creación del modelo, así como las métricas de rendimiento para evaluar los resultados obtenidos.

3.1. Elección de la base de datos

La base de datos elegida para este estudio corresponde al conjunto de datos de diagnóstico del cáncer de mama proporcionado por la Universidad de Wisconsin (también conocido como *Breast Cancer Wisconsin Dataset*), y se conforma como un conjunto de datos multidimensional diseñado para la clasificación de tumores mamarios como benignos o malignos.

El acceso a este *dataset* se realizó a través de Kaggle, una plataforma que no solo actúa como un repositorio central para diversos conjuntos de datos, sino que también ofrece una amplia gama de recursos educativos, facilita la interacción a través de foros especializados, proporciona herramientas para el análisis de datos, y fomenta la participación en competiciones de ciencia de datos y aprendizaje automático.

Este conjunto de datos proviene, a su vez, del UCI Machine Learning Repository, al cual fue donado el 31 de octubre de 1995. El repositorio, establecido en 1987 por David Aha y otros estudiantes graduados de la Universidad de California en Irvine (UC Irvine), posee más de 650 conjuntos de datos disponibles a la fecha de este estudio (enero de 2024). Su estructura permite acceder a los *datasets* según categorías específicas y ofrece la posibilidad de contribuir con conjuntos de datos propios, consolidándose como un recurso esencial para la investigación en el campo del *machine learning*.

Respecto a los datos, éstos se derivan de imágenes digitalizadas de aspirados de aguja fina (*fine needle aspirate* o FNA) de masas mamarias, y sus características se centran en describir los núcleos celulares presentes en estas imágenes. El plano de separación utilizado en este conjunto de datos se obtuvo mediante el Método *Multisurface Tree* (MSM-T), una técnica de clasificación que emplea programación lineal para construir un árbol de decisiones. Las características relevantes se seleccionaron mediante una búsqueda exhaustiva con un espacio de 1 a 4 características y de 1 a 3 planos de separación.

El programa lineal utilizado para obtener el plano de separación en el espacio tridimensional se describe en el artículo de K. P. Bennett y O. L. Mangasarian, titulado “Robust Linear Programming Discrimination of Two Linearly Inseparable Sets”, publicado en “Optimization Methods and Software 1” en 1992 (páginas 23-34).

En cuanto a las dimensiones del conjunto de datos, consta de 569 instancias, cada una caracterizada por 32 atributos. Dos de estos atributos corresponden al número ID (‘*id’*), que es el número de identificación asociado a la instancia, y al diagnóstico (‘*diagnosis’*), que indica si la instancia es maligna (M) o benigna (B) y que emplearemos como categoría para la clasificación del tumor.

Los 30 atributos restantes representan medidas estadísticas de los núcleos celulares analizados y proporcionan información sobre la forma, tamaño y textura de las células en las imágenes. Estas características están divididas en tres conjuntos: media o promedio (*mean*), error estándar (*standard error*), y el "peor" valor (*worst*), donde “peor” se refiere al valor más alto obtenido para cada medida estadística específica de los núcleos celulares.

Por ejemplo, si consideramos el atributo ‘*radius’*, el conjunto de datos registra el radio de la siguiente forma:

- *radius_mean*: media de distancias desde el centro hasta el perímetro.
- *radius_se*: error estándar para la media de distancias desde el centro hasta el perímetro.
- *radius_worst*: mayor valor medio para la media de las distancias desde el centro hasta el perímetro.

De esta manera, solo tendremos que diferenciar estas 10 variables:

- *radius* (radio, media de distancias desde el centro hasta el perímetro).
- *texture* (textura, calculada como la desviación estándar de los valores en escala de grises de las imágenes).
- *perimeter* (perímetro del tumor).
- *area* (área que abarca el tumor).
- *smoothness* (variación local en las longitudes del radio).
- *compactness* (compacidad del tumor, calculada como la relación entre el perímetro al cuadrado y el área del contorno menos uno).
- *concavity* (concavidad, la gravedad de las porciones cóncavas del contorno).
- *concave points* (puntos cóncavos, número de porciones cóncavas del contorno).
- *symmetry* (simetría del tumor en relación con su eje central).
- *fractal dimension* (dimensión Fractal, mide la complejidad de la forma del tumor – a mayor valor, más compleja)

En el apartado de "Desarrollo del modelo", profundizaremos más en los datos: presencia de valores nulos, nivel de equilibrio, etc.

Sobre la elección de esta base de datos, el "*Breast Cancer Wisconsin Dataset"* ha sido reconocido a lo largo de los años como un estándar en la investigación de aprendizaje automático. Su empleo en numerosos estudios y proyectos proporciona una valiosa oportunidad para comparar y contrastar enfoques, contribuyendo de esta forma a la robustez y validez de cualquier modelo que desarrollemos.

Además de su destacada relevancia en el ámbito médico y en la temática específica seleccionada para este trabajo, la accesibilidad y riqueza de información de este conjunto de datos lo convierten en una elección idónea para desarrollar un modelo de aprendizaje automático. Esta combinación de factores nos permite abordar de manera efectiva el desafío de crear un modelo preciso desde el punto de vista técnico y proporcionar un ejemplo concreto de la utilidad y aplicabilidad de estas tecnologías en el ámbito médico.

3.2. Herramientas de software empleadas

El desarrollo de los algoritmos de aprendizaje automático en nuestro proyecto se ha basado en el lenguaje de programación Python. Este lenguaje, que tiene como características ser un lenguaje interpretado, orientado a objetos y de alto nivel, se ha consolidado como la elección principal para la codificación orientada al aprendizaje automático. Su sintaxis clara no solo facilita la creación y mantenimiento del código, agilizando el ciclo de edición, prueba y depuración, sino que su mayor atractivo radica en un extenso conjunto de bibliotecas y marcos diseñados específicamente para el ámbito del aprendizaje automático.

Dentro de este ecosistema especializado, destacan librerías como NumPy, que posibilita cálculos numéricos eficientes; Scikit-learn, que proporciona herramientas esenciales para el aprendizaje automático; y Pandas, una poderosa herramienta para la manipulación y análisis de datos. Estas herramientas se han vuelto fundamentales en la implementación de modelos de *machine learning*, y haremos uso de ellas en nuestro proyecto.

A pesar de que Python puede presentar una velocidad comparativamente inferior frente a lenguajes compilados como C++ o Java, este inconveniente ha sido abordado eficazmente mediante librerías como NumPy y técnicas como la compilación en tiempo real, por lo que su rendimiento no será un problema. Además, la compatibilidad multiplataforma de Python permite la ejecución del código en distintos entornos sin necesidad de reescribirlo, proporcionando flexibilidad en el despliegue de aplicaciones.

En cuanto a la infraestructura elegida para la ejecución del código, hemos optado por Google Colaboratory como plataforma principal. Popularmente conocido como Google Colab, este entorno de cuaderno interactivo basado en Jupyter es directamente accesible desde el navegador y ofrece varias ventajas.

Para empezar, Colab facilita la escritura y ejecución de código Python sin necesidad de configuración adicional, por lo que no requerimos la instalación de ningún programa en nuestro equipo. También proporciona acceso gratuito a recursos informáticos, como GPUs, esenciales para tareas intensivas en el ámbito del aprendizaje automático.

Adicionalmente, la integración con Google Drive simplifica el almacenamiento y compartición de cuadernos; ha sido esta herramienta la que hemos empleado inicialmente para almacenar el conjunto de datos y, tras haber montado Google Drive en nuestro *notebook* de Google Golab, acceder fácilmente a él.

También hemos utilizado GitHub, una plataforma de hospedaje de código que facilita el control de versiones y la colaboración en proyecto, y que permite a los usuarios trabajar conjuntamente desde cualquier ubicación, brindando herramientas esenciales como repositorios, ramas, etc.

Respecto a los repositorios, sirven para organizar un proyecto y pueden contener archivos, carpetas, elementos multimedia y conjuntos de datos. En nuestro caso, se ha creado un repositorio para alojar el *notebook* de Google Colab con el código de nuestro modelo y una copia de la base de datos "*Breast Cancer Wisconsin Dataset*" en formato .csv, así como alternativa para cargar el conjunto de datos en caso de que surgiesen inconvenientes con Google Drive.

De esta forma, proporcionamos acceso al código a través de ambas vías, tanto por medio de Google Drive como de GitHub. Dado que en el pasado nos hemos enfrentado a ciertas limitaciones con la plataforma de Google Drive, como problemas en la conexión con el cuaderno o acceso limitado, hemos decidido ofrecer ambas opciones para garantizar la disponibilidad del código.

3.3. Modelos de aprendizaje automático

Los modelos de aprendizaje automático se pueden clasificar en base a una serie de criterios, los cuales incluyen:

- Si están o no entrenados con supervisión humana (aprendizaje supervisado, no supervisado, semisupervisado y aprendizaje por refuerzo).
- Si pueden aprender de manera incremental sobre la marcha (aprendizaje en línea versus aprendizaje por lotes).
- Si trabajan comparando simplemente nuevas instancias con instancias conocidas o si detectan patrones en los datos de entrenamiento y construyen un modelo predictivo (aprendizaje basado en instancias versus aprendizaje basado en modelos).

Estos criterios no son excluyentes y se pueden combinar de diversas formas, permitiendo la formación de modelos multifacéticos. Sin embargo, en este trabajo, limitaremos nuestra explicación a los modelos según entrenamiento con supervisión humana:

Modelos de aprendizaje supervisado

En el aprendizaje supervisado, el conjunto de entrenamiento proporcionado al algoritmo incluye soluciones deseadas, conocidas como etiquetas. Ejemplos comunes de tareas en este contexto son la clasificación y la regresión. Entre los algoritmos utilizados se encuentran K-Nearest Neighbors, Regresión Lineal, Regresión Logística, Máquinas de Soporte Vectorial (*Support Vector Machine* o SVM), Árboles de Decisión y Redes Neuronales.

Modelos de aprendizaje no supervisado

En este tipo de aprendizaje, el conjunto de entrenamiento no contiene etiquetas; los algoritmos en esta categoría se utilizan para explorar y descubrir patrones en los datos. Algunas tareas comunes son el agrupamiento (como K-Means o DBSCAN), la detección de anomalías y la reducción de dimensionalidad mediante técnicas como el Análisis de Componentes Principales (*Principal Component Analysis* o PCA) o el mapeo estocástico de vecinos t-distribuidos (*t-distributed Stochastic Neighbor Embedding* o t-SNE).

Modelos de aprendizaje semisupervisado

El aprendizaje semisupervisado aborda situaciones donde la anotación de datos es costosa o laboriosa. Aquí, se trabaja con conjuntos de datos que contienen tanto ejemplos etiquetados como no etiquetados. Este método ofrece flexibilidad al modelo al permitirle aprender tanto de ejemplos con etiquetas como de aquellos sin etiquetas. Al evitar depender completamente de datos anotados, el aprendizaje semisupervisado se convierte en una estrategia eficaz para optimizar recursos y superar las limitaciones asociadas con la recopilación extensiva de etiquetas.

Modelos de aprendizaje por refuerzo

El aprendizaje por refuerzo es un enfoque en el que un agente interactúa con un entorno, toma decisiones ejecutando acciones y recibe recompensas o penalizaciones como respuesta. A lo largo del tiempo, el agente aprende a desarrollar la mejor estrategia, conocida como política, con el objetivo de maximizar las recompensas obtenidas. Este proceso implica la adaptación continua del agente a su entorno mediante la evaluación de las consecuencias de sus acciones y la refinación de su política para lograr resultados más favorables. Es comúnmente utilizado en situaciones donde el agente debe tomar decisiones secuenciales para alcanzar metas específicas.

En la construcción de nuestro modelo, nos hemos centrado exclusivamente en técnicas supervisadas, aprovechando que el conjunto de datos contenía la etiqueta '*diagnosis*' para diferenciar entre tumor benigno y maligno. No obstante, consideramos que se podrían aplicar también técnicas no supervisadas si omitimos dicha variable antes de ejecutarlas, abriendo así una nueva línea de investigación que, dadas nuestras limitaciones temporales, no hemos podido estudiar. En los subapartados siguientes se detallan los algoritmos aplicados.

3.3.1. Algoritmo de regresión logística

La regresión logística se utiliza para tareas de clasificación binaria, prediciendo la probabilidad de que ocurra un evento, empleando una función logística para asignar variables de entrada a una puntuación de probabilidad, facilitando la toma de decisiones efectiva en la clasificación. La fórmula se muestra a continuación, donde x es una combinación lineal de las variables de entrada (equivalente a $(b + wX)$, siendo w el coeficiente de regresión).

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

Esta fórmula garantiza que la salida predicha se encuentre en el rango de 0 a 1, representando la probabilidad de pertenecer a una clase específica.

Proporciona velocidad rápida de entrenamiento e interpretabilidad. Sin embargo, su naturaleza lineal la hace menos adecuada para relaciones complejas y no lineales.

3.3.2. Algoritmo K-NN (*K-Nearest Neighbors*)

K-Vecinos más cercanos o *K-Nearest Neighbors* (abreviado como K-NN) es un algoritmo versátil utilizado tanto para tareas de clasificación como de regresión, así como en sistemas de búsqueda y recomendación. Parte del supuesto de que elementos similares tienden a tener una proximidad cercana en el espacio de características.

Para determinar esta proximidad, se miden distancias entre diferentes instancias, utilizando métricas como la distancia euclidiana. En modelos de clasificación, K-NN identifica los K vecinos más cercanos; en regresión, implica promediar sus valores. Aunque es simple y no requiere entrenamiento, a

excepción de seleccionar el valor de K, el algoritmo puede enfrentar desafíos con grandes conjuntos de datos y espacios de alta dimensión, siendo sensible a datos ruidosos y faltantes.

3.3.3. Algoritmo SVM (*Support Vector Machine*)

Las Máquinas de Vectores de Soporte o *Support Vector Machine* (SVM) son modelos versátiles de aprendizaje automático capaces de realizar clasificación lineal o no lineal, regresión e incluso detección de valores atípicos, y son especialmente eficaces para conjuntos de datos pequeños o medianos.

Las SVM buscan encontrar límites de decisión que separen las clases mientras maximizan el margen entre ellas. El objetivo es lograr la clasificación con mayor margen, es decir, encontrar la "calle más ancha" posible entre las clases. Los vectores de soporte, instancias ubicadas en el borde de la "calle", desempeñan un papel crucial en determinar el límite de decisión. Las SVM son sensibles a la escala de las características y es fundamental realizar un escalado para un mejor rendimiento.

Para nuestro modelo, hemos optado por aplicar SVM con clasificación lineal, una variante que establece un límite de decisión en forma de una línea recta en el espacio de características, con el objetivo de encontrar esta línea de manera óptima para separar las clases del conjunto de datos. Las SVM lineales son susceptibles a valores atípicos y solo funcionan si los datos son linealmente separables.

La fórmula de las SVM lineales se puede representar como $f(x) = wx + b$, donde $f(x)$ es la función de decisión, x es el vector de características, w es el vector de pesos o coeficientes asociados a las características, y b es el sesgo. El objetivo al trabajar con este algoritmo es encontrar los valores óptimos para w y b que maximicen el margen de separación entre las clases.

3.3.4. Algoritmo *Random Forest*

Los algoritmos de bosque aleatorio o *Random Forest* son métodos de aprendizaje automático que se destacan por su capacidad para crear múltiples árboles de decisión, denominados *decision trees*. Cada árbol se construye utilizando un subconjunto aleatorio de características del conjunto de datos original y, posteriormente, las decisiones individuales de estos árboles se combinan mediante técnicas como votación mayoritaria o promedio, dependiendo de si el problema es de clasificación o regresión, respectivamente.

La fortaleza de este algoritmo reside en su capacidad para mejorar la precisión predictiva. Al construir múltiples árboles con características diferentes, se reduce el riesgo de sobreajuste, especialmente en conjuntos de datos grandes y complejos. Este enfoque de diversificación también facilita la gestión eficaz de conjuntos de características extensos, permitiendo que cada árbol se especialice en un subconjunto específico de variables.

A pesar de estas ventajas, es importante tener en cuenta que los bosques aleatorios presentan algunas limitaciones. En comparación con los árboles de decisión individuales, los bosques aleatorios son generalmente menos interpretables, ya que la combinación de múltiples árboles dificulta seguir el flujo de decisiones de manera clara. Además, la construcción de varios árboles puede resultar en tiempos de entrenamiento más largos en comparación con modelos más simples.

3.3.5. Algoritmo XGBoost

XGBoost es un algoritmo perteneciente a la categoría de algoritmos de aumento de gradiente o *Gradient Boosting*. Este algoritmo opera construyendo secuencialmente múltiples árboles de decisión, corrigiendo los errores residuales en cada etapa.

La técnica distintiva de XGBoost es "*Gradient Boosting with Regularization*", que impone restricciones a la complejidad del modelo, evitando el sobreajuste al penalizar la contribución de cada árbol. Durante el entrenamiento, se optimiza una función de pérdida que mide la discrepancia entre las predicciones y los valores reales.

Con la adición de cada nuevo árbol, se ajustan los pesos de los ejemplos de entrenamiento, aplicando términos de regularización para controlar la complejidad y mejorar la precisión del modelo. Este proceso iterativo mejora la capacidad del modelo para generalizar a nuevos datos.

XGBoost destaca por su eficiencia computacional, velocidad de entrenamiento y capacidad para manejar conjuntos de datos grandes y dispersos. Además, ofrece opciones para personalizar la función de pérdida y cuenta con herramientas integradas para la selección automática de características. Como contraparte, puede ser más lento y sensible a datos ruidosos, siendo menos interpretable debido a la complejidad del modelo.

3.4. Métricas para evaluar el modelo

En el contexto de nuestro trabajo, deseamos analizar los resultados obtenidos tras la aplicación de algoritmos a una clasificación binaria; en otras palabras, evaluar la categorización entre tumores benignos y malignos. Es fundamental comprender que, en el ámbito del aprendizaje automático, la evaluación de modelos de clasificación difiere de la evaluación de modelos de regresión: en un modelo de clasificación buscamos evaluar su capacidad para asignar instancias a categorías específicas; en contraste, en un modelo de regresión se analiza la precisión en la predicción de los valores numéricos continuos obtenidos.

Esta diferencia en la naturaleza del modelo implica que se utilicen distintas métricas para evaluar sus resultados. En un modelo de clasificación, prevalece el uso de la exactitud (*accuracy*), precisión (*precision*), sensibilidad (*recall*), el F1-score y el área bajo la curva ROC (AU-ROC); por el contrario, en un modelo de regresión, se suelen emplear el error absoluto medio (*Mean Absolute Error* o MAE), error cuadrático medio (*Mean Squared Error* o MSE), raíz del error

cuadrático medio (*Root Mean Squared Error* o RMSE) y el coeficiente de determinación (*Coefficient of determination* o R^2). Dado que los algoritmos que utilizaremos son de clasificación, nos enfocaremos en el primer conjunto de métricas mencionadas (exceptuando AU-ROC).

Pero antes de llevar a cabo la valoración de los algoritmos aplicados, es necesario explicar el concepto de la "matriz de confusión", la cual sirve como base para construir las métricas que necesitaremos. Esta matriz permite analizar detalladamente las predicciones del modelo y compararlas con las clases reales de nuestro conjunto de datos.

La matriz de confusión, en esencia, es una tabla que describe el rendimiento de un modelo al clasificar instancias en diferentes categorías. Su estructura básica es bidimensional, con dos filas y dos columnas que representan las clases reales y las clases predichas. Las celdas de la matriz contienen recuentos o proporciones de instancias que pertenecen a cada combinación de clases.

Desglosemos los elementos clave de la matriz de confusión:

- **Verdaderos Positivos (*True Positive*, TP):** Instancias clasificadas correctamente como positivas por el modelo.
- **Verdaderos Negativos (*True Negative*, TN):** Instancias clasificadas correctamente como negativas por el modelo.
- **Falsos Positivos (*False Positive*, FP):** Instancias clasificadas incorrectamente como positivas por el modelo.
- **Falsos Negativos (*False Negative*, FN):** Instancias clasificadas incorrectamente como negativas por el modelo.

Estos valores son esenciales para calcular las métricas de rendimiento del modelo de clasificación, las cuales proporcionan una visión detallada de su eficacia al predecir las distintas clases, considerando aspectos como la calidad de las predicciones positivas, la capacidad de identificar instancias positivas y el equilibrio entre precisión y sensibilidad, como podremos observar a continuación:

Exactitud (*Accuracy*): Mide la proporción de predicciones correctas en relación con el total de predicciones. La fórmula básica para la exactitud es:

$$Exactitud = \frac{TP + TN}{TP + TN + FN + FP}$$

Precisión (*Precision*): Mide la proporción de instancias clasificadas como positivas que son verdaderamente positivas, es decir, nos aporta información sobre el rendimiento del modelo para predecir instancias positivas. Puede ser engañosa si se valora sin considerar otras métricas, especialmente en casos de desequilibrio de clases. Su cálculo se realiza de la siguiente manera:

$$Precisión = \frac{TP + TN}{TP + FP}$$

Sensibilidad (Recall): También conocida como Tasa de Verdaderos Positivos (*True Positive Rate* o TPR), mide la proporción de instancias positivas que el modelo logra identificar correctamente frente al total de instancias positivas, incluyendo las que fueron clasificadas incorrectamente como negativas. Una alta sensibilidad indica que el modelo tiene la capacidad de identificar la mayoría de las instancias positivas presentes en el conjunto de datos, minimizando así la posibilidad de falsos negativos. Se calcula como:

$$Sensibilidad = \frac{TP}{TP + FN}$$

F1-Score: El F1-Score busca encontrar un equilibrio entre precisión y sensibilidad, proporcionando una medida única que tiene en cuenta ambas métricas. Alcanza su valor máximo de 1 cuando tanto la precisión como la sensibilidad son óptimas, y es particularmente útil en situaciones donde se busca equilibrar la importancia de falsos positivos y falsos negativos La fórmula de la métrica F1-Score es la siguiente:

$$F1 - Score = \frac{2 \times Precisión \times Sensibilidad}{Precisión + Sensibilidad}$$

4. Desarrollo del modelo

En este apartado, presentaremos y explicaremos el desarrollo del modelo. En el archivo 'Breast_Cancer_Wisconsin_Analysis_Thesis.ipynb' se puede comprobar el desarrollo de las diferentes etapas del proyecto, el código empleado y los resultados obtenidos. En esta memoria no profundizaremos tanto en el código y contenido del *notebook*; en su lugar, avanzaremos a través de las fases del proyecto de manera más general. Por último, para acceder al archivo mencionado anteriormente, se ofrece el enlace al repositorio de GitHub:

https://github.com/adribatr/Breast_Cancer_Thesis

4.1. Configuración inicial y carga de datos

El inicio del proceso de desarrollo implicó la importación de todas las librerías necesarias para utilizar las funciones deseadas. Además, se configuró el cuaderno para que los resultados presentados en este entorno sean accesibles al usuario, como deshabilitar el límite en el número máximo de columnas mostradas por pantalla para que no haya omisión de éstas.

El conjunto de datos "*Breast Cancer Wisconsin Dataset"* ha sido inicialmente cargado en una variable '*data'* – el proceso de carga se realizó tanto montando Google Drive en el *notebook* como conectando al repositorio de GitHub.

Para confirmar que los datos se hubiesen cargado correctamente, comprobamos las 5 primeras filas del conjunto. Gracias a esta acción, podemos observar que se ha importado una columna extra, '*Unnamed: 32*', que trataremos más adelante.

4.2. Limpieza de datos

Empezamos la limpieza de los datos comprobando el estado inicial de los datos; para ello, se ha empleado la función info(), que muestra información básica para cada columna del *dataset*: el índice, su nombre, la cantidad de valores no nulos y su tipo. Además, proporciona un resumen de todos los tipos de datos presentes y la memoria utilizada por el conjunto de datos.

Figura 2. Estado inicial de datos

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

De este resumen, se ha podido confirmar que el *dataset* consta de 569 instancias y 33 columnas, en su mayoría numéricas y libres de valores nulos, exceptuando el caso de la columna extra, mencionada anteriormente.

También se ha invocado a la función describe(), que nos ha proporcionado una serie de estadísticas descriptivas para cada columna numérica. Esto implica que la columna '*diagnosis*', al ser categórica, no apareció en este resultado.

Estas estadísticas incluyen el conteo de valores, la media, la desviación estándar, los valores mínimo y máximo, así como los percentiles 25%, 50% (mediana) y 75%. Este análisis nos permite tener una visión general de la distribución y variabilidad de los datos numéricos en el conjunto.

Figura 3. Extracto de medidas estadísticas del conjunto inicial

```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.000000	906024.000000	8.813129e+06	9.113205e+08
radius_mean	569.0	1.412729e+01	3.524049e+00	6.981000	11.700000	13.370000	1.578000e+01	2.811000e+01
texture_mean	569.0	1.928965e+01	4.301036e+00	9.710000	16.170000	18.840000	2.180000e+01	3.928000e+01
perimeter_mean	569.0	9.196903e+01	2.429898e+01	43.790000	75.170000	86.240000	1.041000e+02	1.885000e+02
area_mean	569.0	6.548891e+02	3.519141e+02	143.500000	420.300000	551.100000	7.827000e+02	2.501000e+03
smoothness_mean	569.0	9.636028e-02	1.406413e-02	0.052630	0.086370	0.095870	1.053000e-01	1.634000e-01
compactness_mean	569.0	1.043410e-01	5.281276e-02	0.019380	0.064920	0.092630	1.304000e-01	3.454000e-01
concavity_mean	569.0	8.879932e-02	7.971981e-02	0.000000	0.029560	0.061540	1.307000e-01	4.268000e-01
concave points_mean	569.0	4.891915e-02	3.880284e-02	0.000000	0.020310	0.033500	7.400000e-02	2.012000e-01
symmetry_mean	569.0	1.811619e-01	2.741428e-02	0.106000	0.161900	0.179200	1.957000e-01	3.040000e-01
fractal_dimension_mean	569.0	6.279761e-02	7.060363e-03	0.049960	0.057700	0.061540	6.612000e-02	9.744000e-02
radius_se	569.0	4.051721e-01	2.773127e-01	0.111500	0.232400	0.324200	4.789000e-01	2.873000e+00
texture_se	569.0	1.216853e+00	5.516484e-01	0.360200	0.833900	1.108000	1.474000e+00	4.885000e+00
perimeter_se	569.0	2.866059e+00	2.021855e+00	0.757000	1.606000	2.287000	3.357000e+00	2.198000e+01
area_se	569.0	4.033708e+01	4.549101e+01	6.802000	17.850000	24.530000	4.519000e+01	5.422000e+02
smoothness_se	569.0	7.040979e-03	3.002518e-03	0.001713	0.005169	0.006380	8.146000e-03	3.113000e-02

De esta información, lo primero que habíamos observado es la posible existencia de valores extremos. Por ejemplo, en la característica '*area_se'* se puede apreciar que la media es 40,337079, mientras que el valor máximo es 542,2. Ante este escenario, decidimos incorporar un estudio de valores atípicos en la fase de análisis.

Después de esto, confirmamos la presencia de valores nulos, información que se había reflejado inicialmente con la función info(). En la siguiente figura podremos apreciar que la columna '*Unnamed: 32*' es la única con valor nulos; de hecho, es una columna vacía, pero es comprensible dado que fue generada durante el proceso de importación.

Figura 4. Presencia de valores nulos

```
data.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

También hemos realizado una comprobación sobre la presencia de filas duplicadas en el conjunto de datos, considerando las siguientes situaciones:

1. Una paciente aparece dos o más veces en el conjunto, conservando el mismo número de identificación (id).
2. Dos pacientes diferentes comparten exactamente la misma información en todas las columnas, pero con números de identificación diferentes.

Estas verificaciones se realizan para identificar posibles duplicados en el conjunto de datos, ya sea que una paciente aparezca varias veces o que haya coincidencias exactas en los datos de dos pacientes diferentes (caso muy improbable considerando la cantidad de características).

Figura 5. Presencia de filas duplicadas

```
duplicated_rows = data[data.duplicated()]
num_duplicated_rows = duplicated_rows.shape[0]
print(f"El dataset tiene {num_duplicated_rows} filas duplicadas.")

duplicated_rows_no_id = data[data.duplicated(subset=data.columns.difference(['id']))]
num_duplicated_rows_no_id = duplicated_rows_no_id.shape[0]
print(f"El dataset tiene {num_duplicated_rows_no_id} filas duplicadas (excluyendo la columna 'id').")
```

```
El dataset tiene 0 filas duplicadas.
El dataset tiene 0 filas duplicadas (excluyendo la columna 'id').
```

El último paso en este apartado ha sido la eliminación de aquellas características que no aportaban información relevante a nuestro estudio: '*Unnamed: 32'*, columna que se agregó después de importar el conjunto de datos al *notebook*, solo contiene valores nulos; y la columna '*id*', identificador único asociado a cada instancia y que no contribuye significativamente al análisis.

Además, y como hemos visto antes, casi todas las columnas son de naturaleza numérica, excepto '*diagnosis'*, que es categórica. Según la descripción del conjunto de datos, los únicos valores que contiene esta columna son 'B' (*benign*) para indicar si un tumor es benigno, y 'M' (*malignant*) si un tumor es maligno. Realizamos una comprobación para confirmar que son valores únicos:

Figura 6. Comprobación de valores en columna '*diagnosis'*

```
print("Valores únicos de la columna 'diagnosis':", list(data.diagnosis.unique()))
```

```
Valores únicos de la columna 'diagnosis': ['M', 'B']
```

Dado que muchos algoritmos de aprendizaje automático requieren entradas numéricas, procedimos a asignar el valor 0 a las instancias benignas y el valor 1 a las instancias malignas.

Figura 7. Reasignación de valores en columna '*diagnosis'*

```
data.diagnosis.replace({"B":0,"M":1}, inplace=True)
print("Valores únicos de la columna 'diagnosis':", list(data.diagnosis.unique()))
```

```
Valores únicos de la columna 'diagnosis': [1, 0]
```

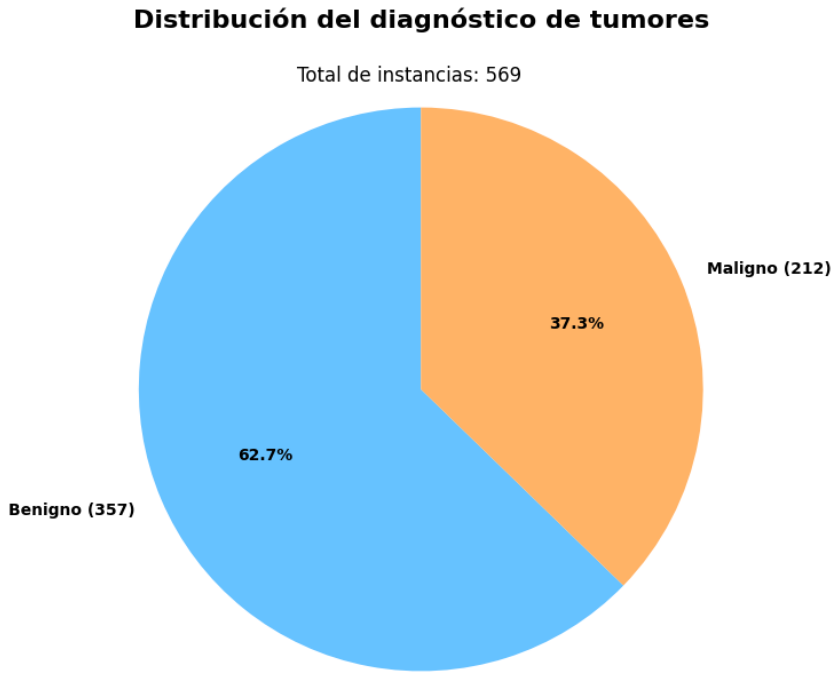
4.3. Análisis exploratorio de datos (EDA)

Después de la limpieza de datos, procedimos a realizar un estudio más profundo de los datos presentes en el conjunto de datos, lo que se suele denominar como análisis exploratorio de datos o EDA, por las siglas en inglés del término *Exploratory Data Analysis*.

4.3.1. Distribución de los datos

El análisis comenzó examinando la distribución de las dos clases en el conjunto de datos, con el objetivo de determinar si nos encontrábamos ante un *dataset* balanceado o desbalanceado.

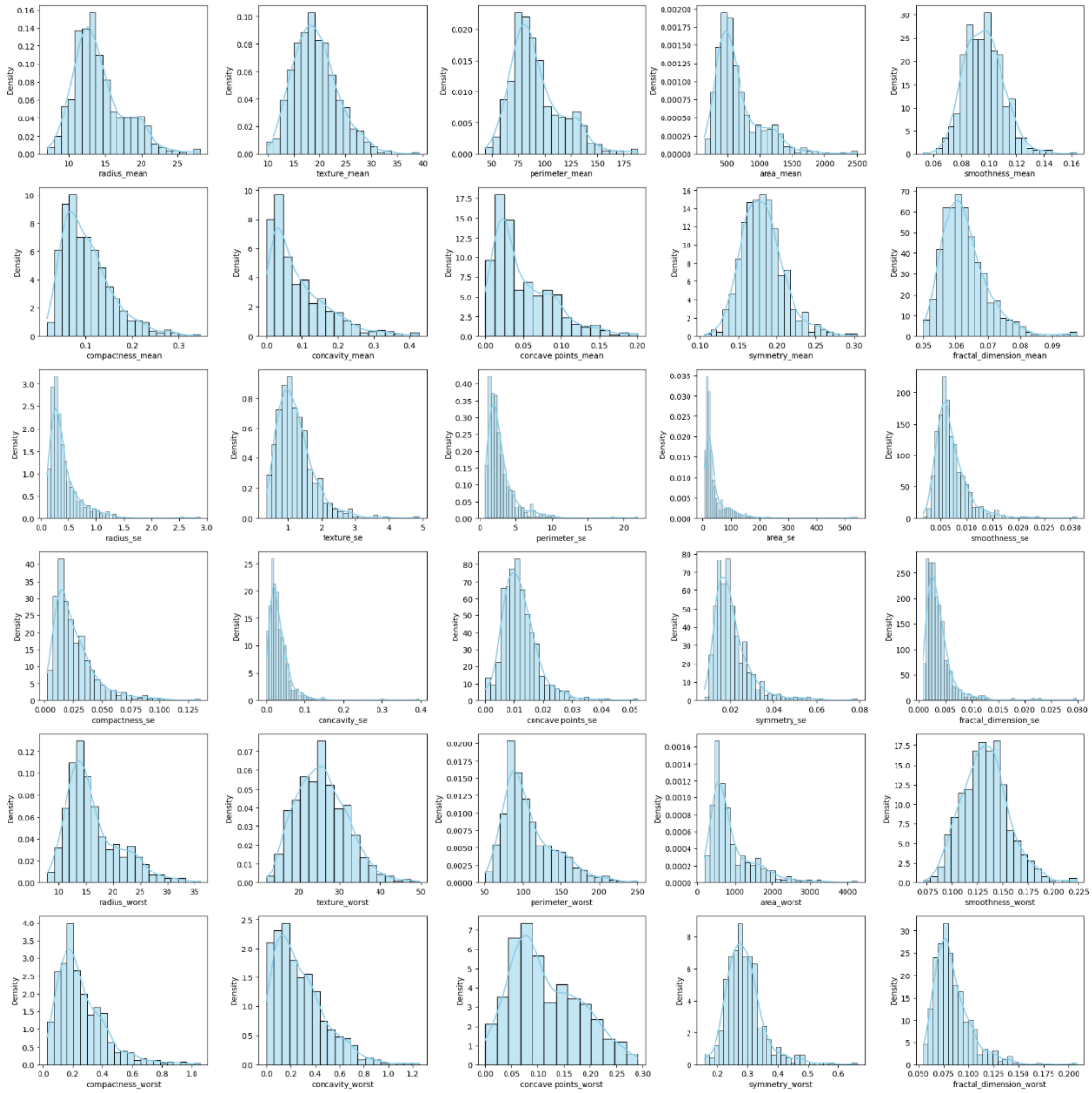
Figura 8. Distribución de clases



El conjunto de datos inicial está desbalanceado, con un 62,7% de las instancias benignas y el 37,3% restante como malignas. A pesar de este desequilibrio, no habíamos considerado que fuese un desbalance altamente significativo, así que optamos por no aplicar ninguna técnica de balanceo de clases. Sin embargo, somos conscientes de que aplicar técnicas como *undersampling, oversampling* o SMOTE (*Synthetic Minority Over-sampling Technique*) podrían afectar a los modelos, por lo que valoraríamos su uso en futuras iteraciones del análisis.

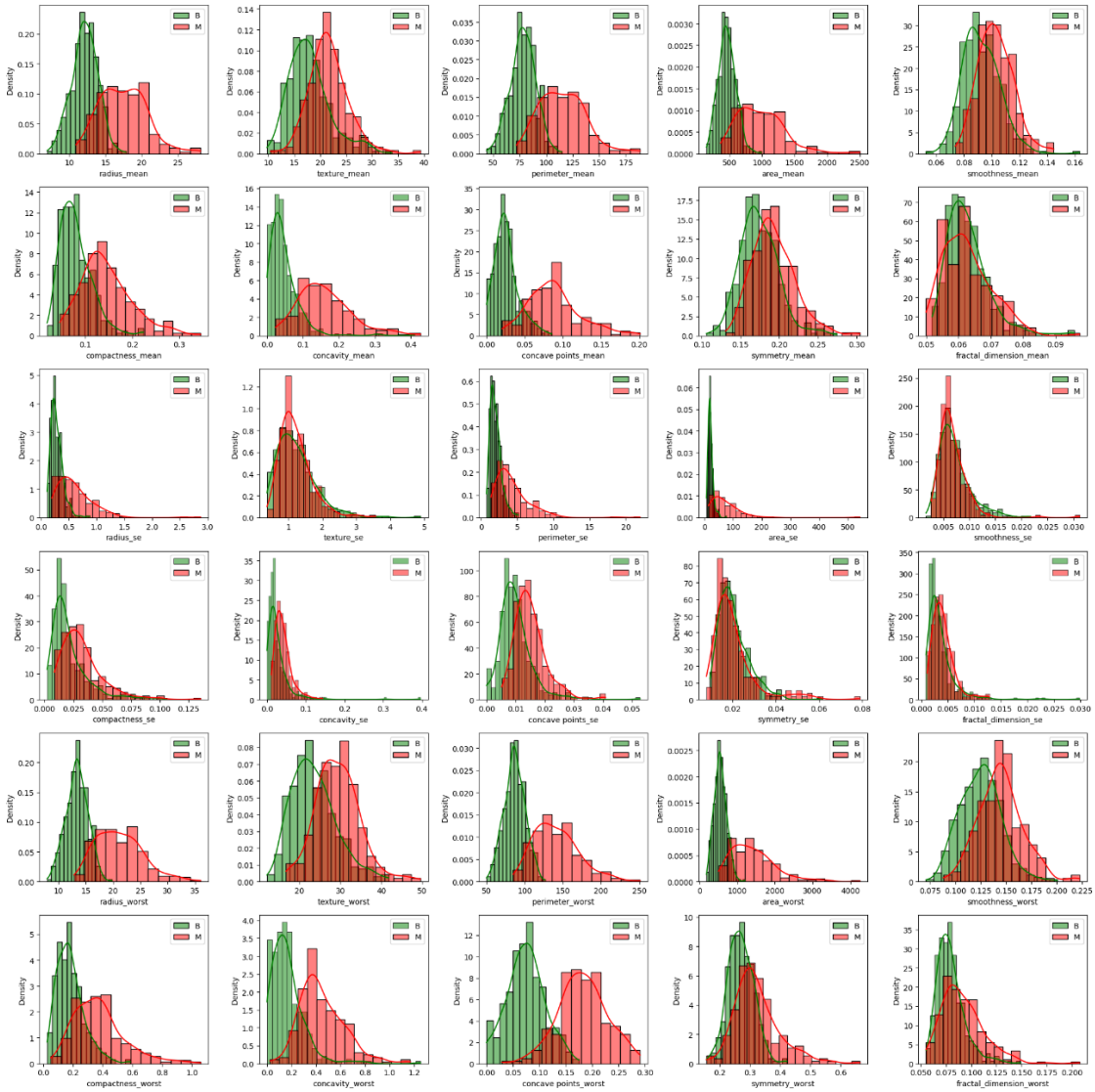
Faltaría comprobar la distribución del resto de datos, es decir, las características específicas de los tumores. Para ello, utilizamos histogramas que representen el estado conjunto de las instancias benignas y malignas, sin diferenciación entre clases.

Figura 9. Distribución conjunta de características



Se puede apreciar que diversas columnas exhiben una distribución con sesgo positivo o hacia la derecha, lo que indica que los datos se concentran a la izquierda y la "cola" se extiende hacia la derecha. Este sesgo podría ser resultado de la presencia de valores extremos u *outliers* que afectan la distribución en esa dirección. Sin embargo, ¿las instancias siguen una distribución similar independientemente de su clase?

Figura 10. Distribución por clases de características



En estos nuevos histogramas podemos ver que las características no se distribuyen de manera equitativa entre las instancias benignas y malignas. En la mayoría de los casos, las instancias malignas tienden a tener valores más altos que las instancias benignas, mostrando una tendencia a concentrarse hacia la derecha. En algunos casos, las distribuciones son prácticamente opuestas, como se evidencia en '*concave points_worst*'.

Este patrón podría sugerir la presencia de propiedades o patrones más prominentes en tumores malignos. Por lo tanto, hemos considerado importante calificar estas características como posibles indicadores clave al desarrollar modelos de clasificación.

En este caso, las columnas '*radius_mean*', '*perimeter_mean*', '*area_mean*', '*concavity_mean*', '*concave points_mean*', '*radius_worst*', '*perimeter_worst*', '*area_worst*', '*concavity_worst*' y '*concave points_worst*'. Es importante destacar que la mención de estas columnas no implica su elección definitiva para nuestro modelo, sino que indican características que podrían haber adquirido importancia tras una investigación más detallada.

4.3.2. Presencia de outliers

Hasta este punto en el análisis de los datos, habíamos observado indicios de que el conjunto de datos podría contener valores extremos, una posibilidad que se ha fortalecido durante la evaluación de la distribución de los datos. Ante este escenario, decidimos verificar la presencia de valores atípicos mediante el uso de *boxplots*.

Al crear los *boxplots*, hemos aplicado una diferenciación por clases y, con el fin de facilitar la visualización, los dividimos en subgrupos según correspondan a los subgrupos “mean”, “se” o “worst”.

Figura 11. *Outliers* del subgrupo ‘*mean’*

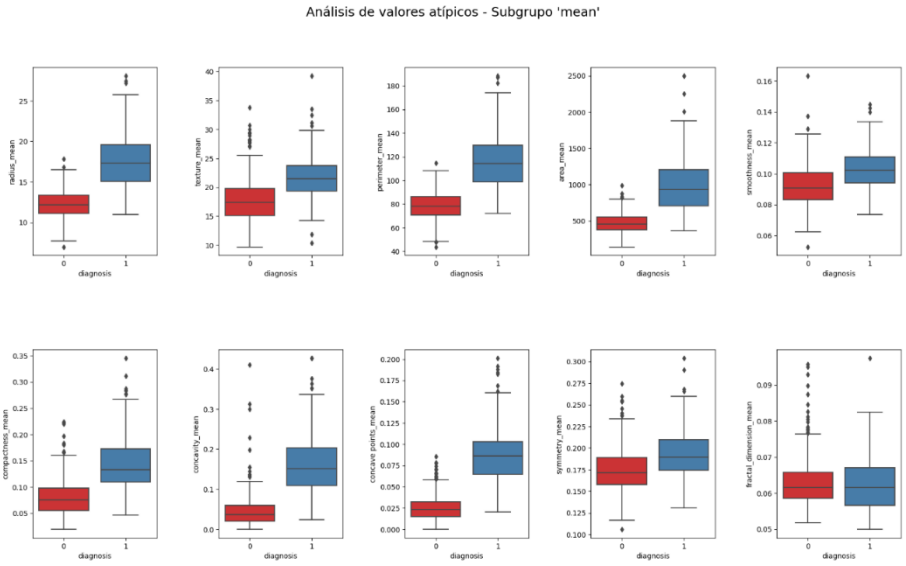


Figura 12. *Outliers* del subgrupo ‘*se’*

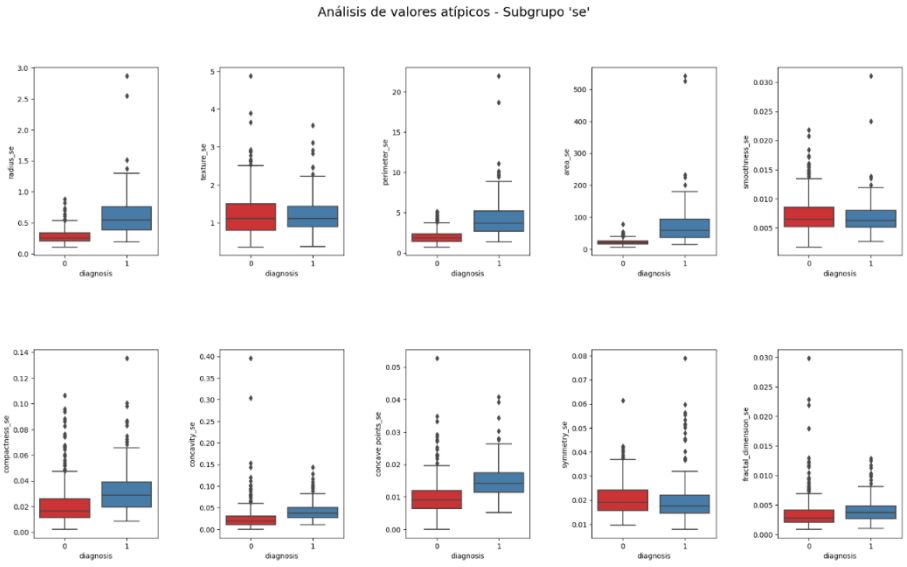
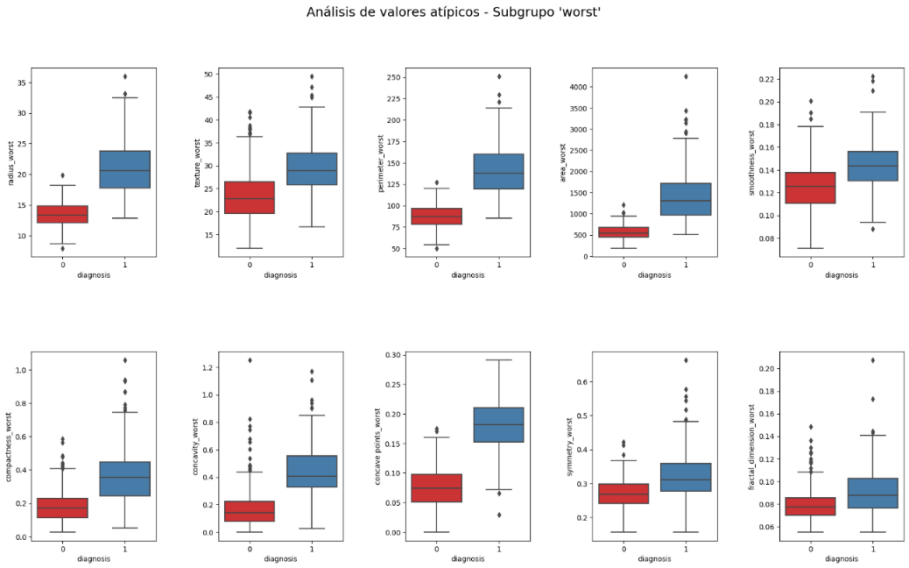


Figura 13. *Outliers* del subgrupo '*worst*'



Todas las características muestran valores atípicos; sin embargo, las características correspondientes al subgrupo '*se*' parecen exhibir un número mayor de *outliers*.

Para verificar el número de *outliers* en cada subgrupo, habíamos empleado el rango intercuartílico (IQR), una medida estadística que evalúa la dispersión de datos, calculada como la diferencia entre el tercer cuartil y el primer cuartil, siendo útil para identificar valores atípicos en un conjunto de datos.

De esta manera, podremos cuantificar cuántos valores se encuentran fuera del rango (gráficamente, aquellos valores que localizaban fuera de los "bigotes" de los *boxplots*).

Figura 14. Número y porcentajes de *outliers* por subgrupos

```
Número de outliers para subgrupo 'mean':
radius_mean               14
texture_mean               7
perimeter_mean            13
area_mean                 25
smoothness_mean            6
compactness_mean          16
concavity_mean            18
concave points_mean       10
symmetry_mean             15
fractal_dimension_mean    15
dtype: int64

Número de outliers para subgrupo 'se':
radius_se                 38
texture_se                20
perimeter_se              38
area_se                   65
smoothness_se             30
compactness_se            28
concavity_se              22
concave points_se         19
symmetry_se               27
fractal_dimension_se      28
dtype: int64

Número de outliers para subgrupo 'worst':
radius_worst              17
texture_worst              5
perimeter_worst           15
area_worst                35
smoothness_worst           7
compactness_worst         16
concavity_worst           12
concave points_worst       0
symmetry_worst            23
fractal_dimension_worst   24
dtype: int64
Número total de outliers para subgrupo 'mean': 139
Porcentaje de outliers para subgrupo 'mean': 24.43%

Número total de outliers para subgrupo 'se': 315
Porcentaje de outliers para subgrupo 'se': 55.36%

Número total de outliers para subgrupo 'worst': 154
Porcentaje de outliers para subgrupo 'worst': 27.07%
```

La presencia significativa de valores atípicos en el *dataset* se evidencia a través de porcentajes de valor considerable entre los distintos subgrupos. En particular, la proporción de *outliers* en el subgrupo '*se*', que representa el 55,36% de las instancias, indica una variabilidad considerable en las mediciones del error estándar.

Esta variación puede atribuirse a varias razones, como una marcada sensibilidad a condiciones variables o incluso a características biológicas específicas de las células mamarias, las cuales podrían estar contribuyendo a la discrepancia en las mediciones del grupo '*se*'.

Dado que estas características específicas podrían ser de interés, en este momento optamos por no abordar directamente los *outliers* identificados. No obstante, reconocemos su existencia y consideramos que, si volviésemos a crear el modelo, probaríamos a aplicar técnicas como Winsorizing, que consiste en sustituir los valores atípicos por los límites inferior y superior de un cierto rango; o la eliminación directa de los *outliers*.

4.3.3. Correlación entre características

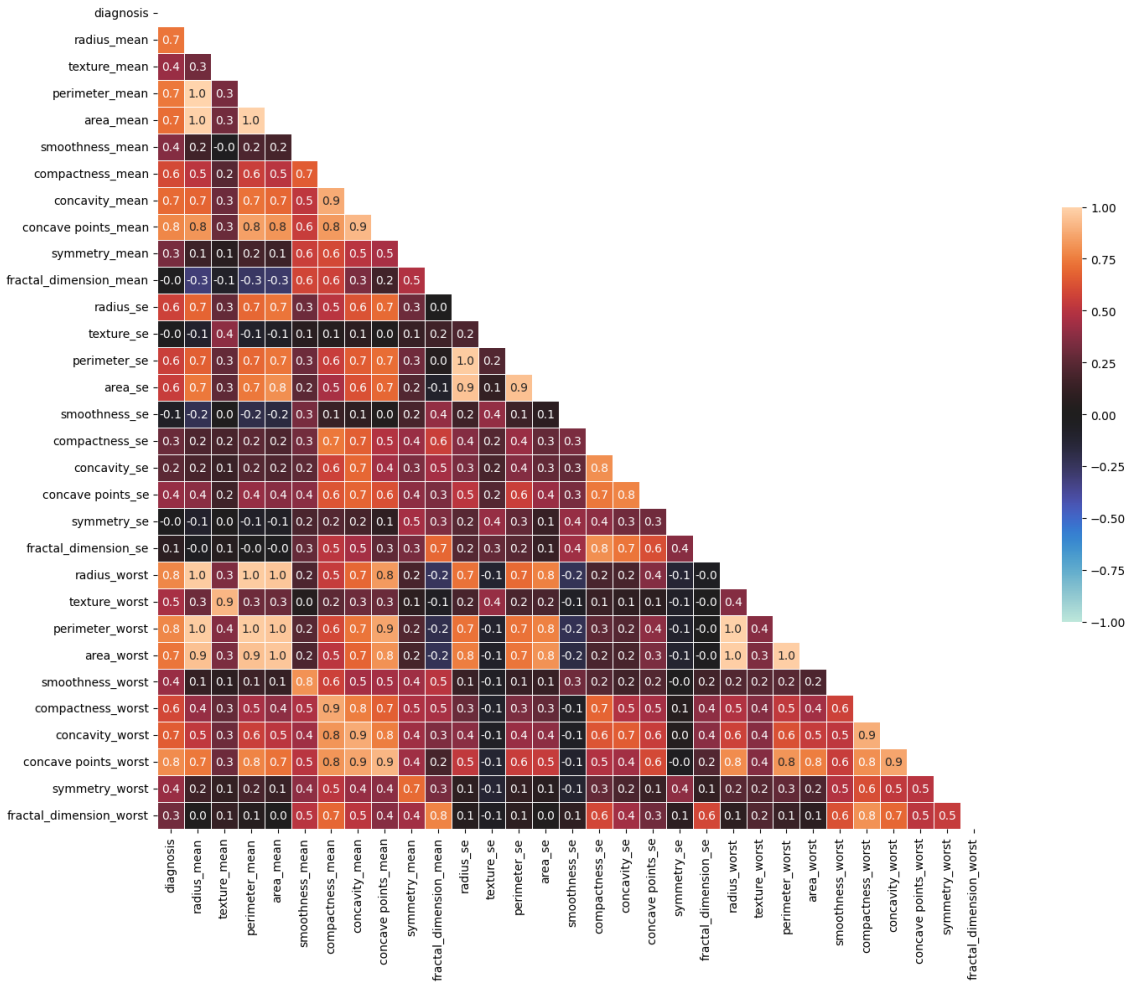
La presencia de correlaciones puede señalar posibles interdependencias entre las variables, y cuando variables independientes presentan una alta correlación, suele indicar que ambas están ofreciendo información similar. En este escenario, mantener ambas podría dar lugar a multicolinealidad en modelos subsiguientes.

La multicolinealidad puede tener un impacto negativo en los modelos de aprendizaje automático, especialmente en aquellos que se basan en supuestos de independencia entre las variables predictoras, como la regresión logística.

Por lo tanto, la eliminación de características altamente correlacionadas no solo mejora la interpretación del modelo, sino que también puede aumentar la precisión en algunos casos.

Para comprobar la correlación entre las características, se puede emplear la matriz de correlación, como mostramos en la siguiente figura:

Figura 15. Matriz de correlación



En la matriz de correlación, se evidencia la presencia de varias características altamente correlacionadas, con valores superiores a 0,9, incluso alcanzando 1. Esta observación sugiere que estas variables, aunque independientes, comparten una información considerablemente redundante o casi idéntica. En otras palabras, la alta correlación entre estas características indica que su variabilidad está fuertemente relacionada, y puede haber una redundancia en la información que aportan al análisis.

Para facilitar nuestro estudio, habíamos identificado los pares de columnas que tienen una correlación superior a 0,90. Dichos pares se guardaron en una tabla que mostraremos en la siguiente figura. Es importante mencionar que, en este caso, no consideraremos la correlación con '*diagnosis*', que representa las clases del *dataset*.

Figura 16. Pares de variables con correlación superior a 0,90

	Columna 1	Columna 2	Correlación
0	radius_mean	perimeter_mean	0.9979
1	radius_mean	area_mean	0.9874
2	radius_mean	radius_worst	0.9695
3	radius_mean	perimeter_worst	0.9651
4	radius_mean	area_worst	0.9411
5	texture_mean	texture_worst	0.9120
6	perimeter_mean	area_mean	0.9865
7	perimeter_mean	radius_worst	0.9695
8	perimeter_mean	perimeter_worst	0.9704
9	perimeter_mean	area_worst	0.9415
10	area_mean	radius_worst	0.9627
11	area_mean	perimeter_worst	0.9591
12	area_mean	area_worst	0.9592
13	concavity_mean	concave points_mean	0.9214
14	concave points_mean	concave points_worst	0.9102
15	radius_se	perimeter_se	0.9728
16	radius_se	area_se	0.9518
17	perimeter_se	area_se	0.9377
18	radius_worst	perimeter_worst	0.9937
19	radius_worst	area_worst	0.9840
20	perimeter_worst	area_worst	0.9776

Se han identificado 21 pares de columnas con una correlación significativa, y algunas de estas columnas aparecen en múltiples pares; esta repetición sugiere la presencia de multicolinealidad. En la figura que mostramos a continuación podremos observar los conjuntos de variables con alta correlacionalidad.

Figura 17. Conjuntos de variables altamente correlacionadas

	Columna Base	Columnas correlacionadas
0	radius_mean	perimeter_mean, area_mean, radius_worst, perimeter_worst, area_worst
1	texture_mean	texture_worst
2	perimeter_mean	area_mean, radius_worst, perimeter_worst, area_worst
3	area_mean	radius_worst, perimeter_worst, area_worst
4	concavity_mean	concave points_mean
5	concave points_mean	concave points_worst
6	radius_se	perimeter_se, area_se
7	perimeter_se	area_se
8	radius_worst	perimeter_worst, area_worst
9	perimeter_worst	area_worst

Las observaciones derivadas de esta información incluyen:

- Las características '*radius_mean*', '*perimeter_mean*' y '*area_mean*' están altamente correlacionadas, al igual que sus versiones '*se*' y '*worst*'. Esta correlación tiene sentido, ya que el perímetro y el área se derivan del radio, lo que implica que estas variables aportarían esencialmente la misma información sobre el tamaño físico de la observación. Por lo tanto, sería beneficioso elegir solo una de estas columnas para mejorar las predicciones del modelo.
- Las variables '*concavity_mean*' y '*concave points_mean*', que describen la forma del tumor, muestran una correlación significativa. Ambas características cuantifican aspectos diferentes de la concavidad en los núcleos celulares y, debido a su alta correlación, podrían proporcionar información redundante.
- Al explorar más a fondo esta relación, se puede observar que '*concavity_se*' y '*concave points_se*' presentan una correlación de 0,7718, mientras que '*concavity_worst*' y '*concave points_worst'* muestran una correlación de 0,8554. Aunque estas cifras no superan el umbral del 0,9, siguen siendo notables. Considerando también la alta correlación entre las medias de estas características, podría ser beneficioso seleccionar una característica de cada par, dado el vínculo existente entre '*concavity*' y '*concave points*'. (**Nota**: Los datos se pueden comprobar en la siguiente figura).
- Las columnas '*texture_mean*' y '*texture_worst*', así como '*concave points_mean*' y '*concave points_worst*', están fuertemente correlacionadas. En estos casos, podríamos optar por seleccionar solo una de las variables de cada par, ya que una representa la media y la otra el "peor" valor, es decir, el valor más alto obtenido en esa característica.

Figura 18. Correlación entre '*concavity*' y '*concave points*'

```
conc_par_mean = data[['concavity_mean', 'concave points_mean']].corr().iloc[0, 1]
conc_par_se = data[['concavity_se', 'concave points_se']].corr().iloc[0, 1]
conc_par_worst = data[['concavity_worst', 'concave points_worst']].corr().iloc[0, 1]

print("Correlación para el par 'concavity'-'concave points', grupo 'mean': {:.4f}".format(conc_par_mean))
print("Correlación para el par 'concavity'-'concave points', grupo 'se': {:.4f}".format(conc_par_se))
print("Correlación para el par 'concavity'-'concave points', grupo 'worst': {:.4f}".format(conc_par_worst))
```

```
Correlación para el par 'concavity'-'concave points', grupo 'mean': 0.9214
Correlación para el par 'concavity'-'concave points', grupo 'se': 0.7718
Correlación para el par 'concavity'-'concave points', grupo 'worst': 0.8554
```

4.4. Tratamiento de datos

Tras completar el análisis exploratorio de datos (EDA), avanzaremos hacia la selección de características para nuestro modelo. Dado el considerable número de columnas presente en el conjunto de datos, es aconsejable aplicar una reducción de dimensionalidad.

Para obtener un tamaño óptimo de características, se pueden emplear diversas técnicas, como la eliminación de características altamente correlacionadas, la aplicación de RFE, la combinación de RFE con validación cruzada, entre otras.

En este estudio, y en base al análisis previo, optamos por eliminar inicialmente aquellas variables fuertemente correlacionadas. Posteriormente, generamos varios *datasets* según la técnica que aplicada después de dicha eliminación.

Este enfoque nos permitiría evaluar qué método de selección de características es más efectivo, considerando como punto de partida nuestra premisa de que variables altamente correlacionadas proporcionarían información muy similar.

4.4.1. Eliminación de características altamente correlacionadas

Basándonos en los resultados obtenidos en el análisis de correlación entre las características, habíamos decidido excluir las columnas relacionadas con el perímetro (*perimeter*) y el área (*area*). Además, eliminaremos '*texture_worst*' debido a su fuerte correlación con '*texture_mean*'.

Para determinar qué columnas eliminar entre las relacionadas con '*concavity*' o '*concave points*', calculamos su correlación con la variable '*diagnosis*'. El objetivo es elegir la característica con una mayor correlación con las clases, y aquella con mayor correlación se consideraría más relevante para la clasificación de tumores malignos. En la siguiente figura mostramos los resultados de los cálculos de correlación, para cada característica y para la media de correlación de '*concavity*' y '*concave points*' respecto a la variable objetivo, '*diagnosis*'.

Figura 19. Correlación entre 'diagnosis' con '*concavity*' y '*concave points*'

```
Correlación de cada característica con 'diagnosis'
                    diagnosis
concavity_mean         0.6964
concave points_mean    0.7766
concavity_se           0.2537
concave points_se      0.4080
concavity_worst        0.6596
concave points_worst   0.7936

Media de correlaciones con 'diagnosis'
               diagnosis
concavity         0.5366
concave points    0.6594
```

Se puede observar que '*concave points*' presenta una correlación media superior con '*diagnosis*'.

Por tanto, las siguientes columnas han sido seleccionadas para ser eliminadas:

- *perimeter_mean*
- *perimeter_se*
- *perimeter_worst*
- *area_mean*
- *area_se*
- *area_worst*
- *texture_worst*
- *concavity_mean*
- *concavity_se*
- *concavity_worst*

Después de haber eliminado las columnas indicadas, el estado del nuevo *dataset* es el siguiente:

Figura 20. Estado del conjunto después de eliminación por correlación

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   diagnosis               569 non-null    int64
 1   radius_mean             569 non-null    float64
 2   texture_mean            569 non-null    float64
 3   smoothness_mean         569 non-null    float64
 4   compactness_mean        569 non-null    float64
 5   concave points_mean     569 non-null    float64
 6   symmetry_mean           569 non-null    float64
 7   fractal_dimension_mean  569 non-null    float64
 8   radius_se               569 non-null    float64
 9   texture_se              569 non-null    float64
 10  smoothness_se           569 non-null    float64
 11  compactness_se          569 non-null    float64
 12  concave points_se       569 non-null    float64
 13  symmetry_se             569 non-null    float64
 14  fractal_dimension_se    569 non-null    float64
 15  radius_worst            569 non-null    float64
 16  smoothness_worst        569 non-null    float64
 17  compactness_worst       569 non-null    float64
 18  concave points_worst    569 non-null    float64
 19  symmetry_worst          569 non-null    float64
 20  fractal_dimension_worst 569 non-null    float64
dtypes: float64(20), int64(1)
memory usage: 93.5 KB
```

El conjunto de datos revisado, después de la eliminación de las variables altamente correlacionadas, se compone ahora de 21 columnas. Una de estas columnas representa la clase de las instancias, mientras que las 20 restantes describen las características de los tumores.

A partir de este punto, vamos a emplear diversas técnicas de selección de características sobre este conjunto de datos actualizado. Sin embargo, antes de proceder, llevaremos a cabo la normalización de los datos. Este proceso busca asegurar que todas las características se encuentren en la misma escala, preparando así nuestros datos para un análisis más efectivo.

4.4.2. Estandarización de los datos

La estandarización de los datos asegura que todas las características contribuyan de manera equitativa durante el análisis, a la vez que evita que las variables con escalas más grandes dominen sobre aquellas con escalas más pequeñas.

Para normalizar los datos, existen diversas técnicas, como la normalización o la estandarización (también conocida como normalización Z-Score). En nuestro caso, optamos por la estandarización, método que consiste en ajustar las características para que tengan una media de cero y una desviación estándar de uno, y nos facilita la comparación entre diferentes características al ponerlas en la misma escala.

Aplicamos la estandarización a todas las variables, excepto '*diagnosis*', y verificamos el estado de los nuevos datos utilizando la función describe(), que nos proporcionará estadísticas resumidas, como la media y la desviación estándar. Este análisis nos permitirá asegurarnos de que los datos se han ajustado adecuadamente y están listos para la siguiente etapa del análisis.

Figura 21. Estado del conjunto tras estandarización de los datos

```
data_standard.describe().T
```

	count	mean	std	min	25%	50%	75%	max
diagnosis	569.0	3.725835e-01	0.483918	0.000000	0.000000	0.000000	1.000000	1.000000
radius_mean	569.0	-1.311195e-16	1.000000	-2.027864	-0.688779	-0.214893	0.468980	3.967796
texture_mean	569.0	6.243785e-17	1.000000	-2.227289	-0.725325	-0.104544	0.583662	4.647799
smoothness_mean	569.0	-8.366672e-16	1.000000	-3.109349	-0.710338	-0.034860	0.635640	4.766717
compactness_mean	569.0	1.998011e-16	1.000000	-1.608721	-0.746429	-0.221745	0.493423	4.564409
concave points_mean	569.0	-4.995028e-17	1.000000	-1.260710	-0.737295	-0.397372	0.646366	3.924477
symmetry_mean	569.0	1.748260e-16	1.000000	-2.741705	-0.702621	-0.071564	0.530313	4.480808
fractal_dimension_mean	569.0	4.838933e-16	1.000000	-1.818265	-0.722004	-0.178123	0.470569	4.906602
radius_se	569.0	2.497514e-16	1.000000	-1.058992	-0.623022	-0.291988	0.265866	8.899079
texture_se	569.0	-1.123881e-16	1.000000	-1.552898	-0.694198	-0.197324	0.466142	6.649429
smoothness_se	569.0	-1.545337e-16	1.000000	-1.774504	-0.623470	-0.220142	0.368031	8.022940
compactness_se	569.0	1.873136e-16	1.000000	-1.296957	-0.692317	-0.280773	0.389312	6.138081
concave points_se	569.0	2.497514e-17	1.000000	-1.911765	-0.673897	-0.140372	0.472241	6.643755
symmetry_se	569.0	9.365678e-17	1.000000	-1.531542	-0.651108	-0.219238	0.355380	7.065700
fractal_dimension_se	569.0	-6.243785e-18	1.000000	-1.096004	-0.584604	-0.229738	0.288388	9.842932
radius_worst	569.0	-8.241796e-16	1.000000	-1.725382	-0.674328	-0.268803	0.521557	4.090590
smoothness_worst	569.0	-2.122887e-16	1.000000	-2.680337	-0.690623	-0.046802	0.597020	3.951897
compactness_worst	569.0	-3.621395e-16	1.000000	-1.442609	-0.680485	-0.269264	0.539194	5.108382
concave points_worst	569.0	2.122887e-16	1.000000	-1.743529	-0.755735	-0.223272	0.711884	2.683516
symmetry_worst	569.0	2.622390e-16	1.000000	-2.159060	-0.641299	-0.127297	0.449742	6.040726
fractal_dimension_worst	569.0	-5.744282e-16	1.000000	-1.600431	-0.691303	-0.216254	0.450366	6.840837

La desviación estándar es 1 en todas las columnas que no sean '*diagnosis*', y la media es próxima a 0, por lo que confirmamos que hemos realizado bien el cálculo de estandarización, y estamos preparados para aplicar las técnicas de selección de características RFE y RFECV.

4.4.3. RFE

Recursive Feature Elimination (RFE) es una técnica de selección de características que se utiliza para identificar las variables más relevantes en un conjunto de datos. Se basa en la idea de eliminar de forma iterativa las características menos importantes, construir el modelo con las restantes y evaluar su rendimiento. Este proceso se repite hasta que se alcanza el número deseado de características; si no se sabe dicho número, se puede determinar utilizando técnicas como la validación cruzada para evaluar el rendimiento del modelo con diferentes cantidades de características.

RFE puede emplear diferentes algoritmos como base para evaluar la importancia de las características. Algunos de los algoritmos comúnmente utilizados incluyen máquinas de soporte vectorial (SVM), regresión logística, árboles de decisión, etc.

Nuestra elección ha sido emplear *Random Forest* como clasificador subyacente para RFE, debido a su capacidad para manejar conjuntos de datos complejos y su robustez frente al sobreajuste. Además, *Random Forest* puede proporcionar una medida de la importancia de las características directamente, lo que facilita el proceso de selección. En la siguiente figura mostraremos las variables escogidas por esta técnica y la importancia asociada.

Figura 22. Características seleccionadas con RFE

```
RFE - Características seleccionadas con importancias:
         Característica  Importancia
3   concave points_mean       0.2316
5          radius_worst       0.2175
8  concave points_worst       0.2013
0           radius_mean       0.1020
4             radius_se       0.0669
7     compactness_worst       0.0564
1          texture_mean       0.0411
2      compactness_mean       0.0356
9        symmetry_worst       0.0277
6      smoothness_worst       0.0200

Número de características seleccionadas por RFE: 10
```

La aplicación de RFE con *Random Forest* ha llevado a la selección de 10 columnas. Sin embargo, al entrenar un modelo con este algoritmo, se generan múltiples árboles de decisión mediante la utilización de diferentes subconjuntos aleatorios de características y datos de entrenamiento. La aleatoriedad inherente a este proceso puede dar lugar a distintos niveles de importancia para las

características en cada árbol, lo que incluye la posibilidad de variación del número de características, e incluso las características consideradas más importantes pueden fluctuar entre distintas ejecuciones del modelo.

4.4.4. RFECV

Durante nuestras evaluaciones, observamos que el número de características seleccionadas ha sido consistente mediante el uso de RFE, con 10 características. Sin embargo, para mejorar aún más esta elección, podemos emplear validación cruzada (*cross-validation*) mediante la técnica conocida como RFECV (*Recursive Feature Elimination with Cross-Validation*). En nuestro caso, especificaremos el uso de 10 (k) divisiones para la validación cruzada y evaluaremos la cantidad de características seleccionadas en este proceso.

Figura 23. Características seleccionadas con RFECV

```
RFECV - Características seleccionadas con importancias:
         Característica  Importancia
7  concave points_worst       0.2852
3   concave points_mean       0.2134
5          radius_worst       0.2030
0           radius_mean       0.1056
4             radius_se       0.0522
1          texture_mean       0.0446
6     compactness_worst       0.0417
2      compactness_mean       0.0313
8        symmetry_worst       0.0230

Número de características seleccionadas por RFECV: 9
```

La implementación de la validación cruzada mediante RFE nos señala que, después de 10 iteraciones, el número óptimo de características es de 9. Este número apenas ha reducido el número propuesto por RFE, con 10 columnas, por lo que ya estábamos cerca de la cantidad de variables necesarias para un buen rendimiento en nuestro modelo.

Guardaremos las características seleccionadas en una variable, tanto para RFE como RFECV, para su posterior uso en la prueba de los algoritmos de clasificación.

Figura 24. Estado del conjunto de datos tras aplicar RFE

```
data_RFE.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   diagnosis             569 non-null    int64
 1   radius_mean           569 non-null    float64
 2   texture_mean          569 non-null    float64
 3   compactness_mean      569 non-null    float64
 4   concave points_mean   569 non-null    float64
 5   radius_se             569 non-null    float64
 6   radius_worst          569 non-null    float64
 7   smoothness_worst      569 non-null    float64
 8   compactness_worst     569 non-null    float64
 9   concave points_worst  569 non-null    float64
 10  symmetry_worst        569 non-null    float64
dtypes: float64(10), int64(1)
memory usage: 49.0 KB
```

Figura 25. Estado del conjunto de datos tras aplicar RFECV

```
data_RFECV.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   diagnosis             569 non-null    int64
 1   radius_mean           569 non-null    float64
 2   texture_mean          569 non-null    float64
 3   compactness_mean      569 non-null    float64
 4   concave points_mean   569 non-null    float64
 5   radius_se             569 non-null    float64
 6   radius_worst          569 non-null    float64
 7   compactness_worst     569 non-null    float64
 8   concave points_worst  569 non-null    float64
 9   symmetry_worst        569 non-null    float64
dtypes: float64(9), int64(1)
memory usage: 44.6 KB
```

4.5. Creación del modelo

Los datos han sido preparados y están listos para ser utilizados por los diversos algoritmos de *machine learning* de clasificación supervisada que hemos escogido en este estudio.

Nuestro planteamiento ha sido identificar y definir primero las métricas de rendimiento esenciales para evaluar cada algoritmo, teniendo en cuenta la naturaleza del conjunto de datos: un problema de clasificación binaria.

Posteriormente, tras definir las métricas, decidimos dividir el conjunto de datos en un subconjunto de entrenamiento y otro de prueba. En el conjunto de entrenamiento, aplicamos la validación cruzada y ajustamos los hiperparámetros de cada algoritmo para maximizar la métrica '*recall'*. Esta elección se fundamenta en la premisa de que nuestro modelo debe ser altamente "sensible" para detectar tumores malignos, minimizando así los casos de falsos negativos.

Una vez entrenado y seleccionado el mejor modelo, es decir, aquel con los mejores hiperparámetros, lo evaluamos en el conjunto de prueba para obtener sus métricas.

Este proceso se repite para el conjunto de datos original ('*data'*), el conjunto de datos procesado solo mediante la comprobación de correlación y la estandarización de los datos ('*data_standard*'), el conjunto de datos cuyas características fueron seleccionadas mediante *Recursive Feature Elimination* o RFE con Random Forest ('*data_RFE'*), y el conjunto de datos tratado con RFE y validación cruzada ('*data_RFECV'*).

Finalmente, comparamos las métricas obtenidas para cada conjunto de datos y modelos. La combinación óptima se determina seleccionando aquella que presenta el mejor valor de '*recall'*, por la razón previamente explicada.

Esta decisión no sugiere que los demás modelos sean de baja calidad; de hecho, si valoramos el resto de las métricas, se podrá observar que sus valores son elevados y conforman modelos aptos para su uso en la clasificación de tumores.

4.5.1. Métricas de rendimiento

La evaluación del rendimiento de un modelo de clasificación es esencial para comprender su capacidad para realizar predicciones precisas y útiles en un conjunto de datos dado.

En el capítulo 3.4. Métricas para evaluar el modelo, explicamos que el cálculo de las métricas se basaría en las categorías asignadas en la matriz de confusión (TP, TN, FP, FN). También presentamos las métricas que emplearíamos en este proyecto, que por su nombre en inglés son: *accuracy, precision, recall* y *F1-score.* Para facilitar el cálculo de estas métricas, habíamos definido una función que recibe los conjuntos de entrenamiento y prueba.

Figura 26. Definición de la función calculate_metrics

```python
def calculate_metrics(y_true, y_pred):
    # Calculamos la matriz de confusión
    cm = confusion_matrix(y_true, y_pred)

    # Extraemos los valores de la matriz de confusión
    tn, fp, fn, tp = cm.ravel()

    # Se calculan las métricas
    accuracy = (tp + tn) / (tp + tn + fp + fn)
    precision = tp / (tp + fp) if (tp + fp) != 0 else 0
    recall = tp / (tp + fn) if (tp + fn) != 0 else 0
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) != 0 else 0

    # Métricas a porcentajes
    accuracy *= 100
    precision *= 100
    recall *= 100
    f1_score *= 100

    return accuracy, precision, recall, f1_score
```

4.5.2. Definición de modelos seleccionados

La función 'define_models' se generó para definir los algoritmos que aplicaremos en este proyecto. La encapsulación de un diccionario que contiene los distintos algoritmos de aprendizaje automático facilita su organización y nos permite referenciar a dichos modelos en las funciones siguientes.

Figura 27. Definición de la función define_models

```python
def define_models():
    models = {
        'Logistic Regression': LogisticRegression(max_iter=1000),
        'K-Nearest Neighbors': KNeighborsClassifier(),
        'Support Vector Machines': SVC(),
        'Random Forest': RandomForestClassifier(),
        'XGBoost': XGBClassifier()
    }
    return models
```

4.5.3. Validación cruzada y definición de hiperparámetros

La función 'define_hyperparameters' nos permite encapsular la selección de hiperparámetros con unos rangos definidos. De esta manera, la búsqueda de valores óptimos con GridSearchCV empleará estas definiciones, en base al algoritmo que corresponda. A continuación, se presenta un resumen de los algoritmos y sus respectivos hiperparámetros:

- **Logistic Regression:**
  - C: Inversa de la fuerza de regularización, controla la cantidad de regularización aplicada al modelo.
  - penalty: Especifica la norma utilizada en la regularización ('l2' para la norma L2, que penaliza los coeficientes del modelo al agregar la suma de sus cuadrados a la función de coste).

- **K-Nearest Neighbors:**
  - n_neighbors: Número de vecinos considerados durante la clasificación.
- **Support Vector Machines:**
  - C: Parámetro de penalización; controla el equilibrio entre tener un margen suave y clasificar correctamente los puntos de entrenamiento.
  - kernel: Tipo de kernel utilizado para transformar los datos ('linear' para clasificación lineal, sin aplicar transformaciones no lineales).
- **Random Forest:**
  - n_estimators: Número de árboles en el bosque.
  - max_depth: Profundidad máxima de los árboles, número máximo de niveles desde la raíz hasta las hojas.
- **XGBoost:**
  - n_estimators: Número de árboles a construir.
  - learning_rate: Tasa de aprendizaje que controla la contribución de cada árbol al modelo, regula la actualización de los pesos de los árboles durante el proceso de entrenamiento.

Estos hiperparámetros influyen en cómo se ajusta el modelo durante el proceso de entrenamiento y, por lo tanto, son cruciales para optimizar el rendimiento del modelo. En la figura que se muestra a continuación se puede apreciar su implementación y el rango definido para cada hiperpárametro:

Figura 28. Definición de la función define_hyperparameters

```
def define_hyperparameters(model_name):
    known_models = ['Logistic Regression', 'K-Nearest Neighbors', 'Support Vector Machines', 'Random Forest', 'XGBoost']

    if model_name not in known_models:
        print(f"Warning: Modelo desconocido '{model_name}'. Se usarán hiperparámetros por defecto.")
        return {}

    if model_name == 'Logistic Regression':
        return {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'penalty': ['l2']}
    elif model_name == 'K-Nearest Neighbors':
        return {'n_neighbors': [1, 3, 5, 7, 9]}
    elif model_name == 'Support Vector Machines':
        return {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'kernel': ['linear']}
    elif model_name == 'Random Forest':
        return {'n_estimators': [50, 100, 150], 'max_depth': [None, 10, 20, 30]}
    elif model_name == 'XGBoost':
        return {'n_estimators': [50, 100, 150], 'learning_rate': [0.01, 0.1, 0.2]}
    else:
        return {}  # Devuelve un diccionario vacío por defecto
```

4.5.4. Definición de evaluación del algoritmo individual

En esta sección, explicaremos la definición de una función que se encargará del entrenamiento y evaluación de nuestros modelos de aprendizaje automático de forma individual.

Pero primero, definiremos una función que extraiga las características y su importancia, si el algoritmo lo permite - KNN, por ejemplo, no tiene forma de calcular dicho valor.

Figura 29. Definición de la función get_feature_importance

```python
def get_feature_importance(model, X_train):
    if isinstance(model, LogisticRegression) or isinstance(model, SVC):
        # Regresión logística o SVM lineal
        if hasattr(model, 'coef_'):
            feature_importances = model.coef_[0]
            feature_names = X_train.columns
            return list(zip(feature_names, feature_importances))
    elif isinstance(model, RandomForestClassifier) or isinstance(model, XGBClassifier):
        # Random Forest o XGBoost
        if hasattr(model, 'feature_importances_'):
            feature_importances = model.feature_importances_
            feature_names = X_train.columns
            return list(zip(feature_names, feature_importances))
    elif isinstance(model, KNeighborsClassifier):
        # KNN no realiza selección de características, devuelve None
        return None
    else:
        # Otros modelos, se puede agregar lógica específica según sea necesario,
        print(f"Advertencia: No se ha implementado la lógica para obtener la importancia de características para el modelo {type(model)}.")
        return None
```

Tras definir esta función, procederemos a la función 'evaluate_model', que comienza aplicando la técnica de validación cruzada con el uso de GridSearchCV, una herramienta que explora diversas combinaciones de hiperparámetros para optimizar el rendimiento de cada algoritmo.

El conjunto de datos se divide en k-folds (k=10), es decir, 10 partes iguales, y el modelo se entrena y evalúa 10 veces, utilizando una parte diferente como conjunto de prueba en cada iteración.

En cada configuración de hiperparámetros definida en el espacio de búsqueda, GridSearchCV ajusta el modelo y calcula el promedio de la puntuación de validación cruzada a través de todos los pliegues. El objetivo es encontrar la combinación de hiperparámetros que maximice esta puntuación.

GridSearchCV se enfoca principalmente en maximizar la métrica de '*recall*', priorizando la capacidad del modelo para identificar tumores malignos y minimizar los falsos negativos.

Posteriormente, evaluamos las métricas obtenidas en el conjunto de prueba al aplicar nuestro mejor modelo, específico para el algoritmo en consideración. Estas métricas, junto al propio modelo y una lista con la importancia de las características en la obtención del modelo, serán devueltas por la función.

Figura 30. Definición de la función evaluate_model

```python
def evaluate_model(model_name, model, param_grid, X_train, y_train, X_test, y_test):
    # Configuramos la búsqueda de hiperparámetros con validación cruzada
    grid_search = GridSearchCV(model, param_grid, cv=10, scoring='recall')
    grid_search.fit(X_train, y_train)

    # Mejor modelo encontrado durante la búsqueda
    best_model = grid_search.best_estimator_

    # Se realizan las predicciones en el conjunto de prueba empleando el mejor modelo
    y_test_pred_best = best_model.predict(X_test)

    # Se obtienen las métricas de clasificación en el conjunto de prueba con el mejor modelo
    accuracy_test_best, precision_test_best, recall_test_best, f1_test_best = calculate_metrics(y_test, y_test_pred_best)

    # Si lo permite el modelo, se obtiene la importancia de las características
    ft_importance_list = get_feature_importance(best_model, X_train)

    # Almacenamos los resultados en un diccionario (excluyendo el propio modelo y la lista de importancias)
    results_dict = {
        'Modelo': model_name,
        'Mejores hiperparámetros': grid_search.best_params_,
        'Accuracy': accuracy_test_best,
        'Precision': precision_test_best,
        'Recall': recall_test_best,
        'F1-score': f1_test_best
    }

    # Devuelve resultados, mejor modelo y lista con la importancia de las características
    return results_dict, best_model, ft_importance_list
```

4.5.5. Definición de evaluación del conjunto de algoritmos

Por último, establecemos una función que aplique simultáneamente todos los algoritmos o una selección de ellos a un conjunto de datos. Aunque su ejecución puede ser más lenta si se consideran todos los algoritmos al mismo tiempo, nuestro objetivo no radica en optimizar el tiempo, sino en obtener y comparar métricas al aplicar los algoritmos deseados a un conjunto de datos específico.

El proceso comienza con la definición de una lista de resultados cuyo contenido corresponde a la información del mejor modelo para cada algoritmo: nombre del algoritmo, sus hiperparámetros y las métricas resultantes al aplicarse sobre el conjunto de prueba. También creamos listas para almacenar las instancias de los mejores modelos y la importancia de las características por modelo.

Tras definir los diccionarios correspondientes, se realiza una iteración sobre los modelos disponibles: primero, se comprueba que el modelo está entre los modelos seleccionados por parámetro; si se encuentra entre ellos, se obtienen los resultados y se preparan para su presentación. Este proceso se repite para todos los algoritmos seleccionados.

Finalmente, la función devuelve una tabla con el listado de resultados obtenidos, una tabla con las características y su importancia en el modelo (si lo permite), y las instancias de los mejores modelos obtenidos.

Figura 31. Definición de la función run_model_evaluation

```
def run_dataset_evaluation(dataset, selected_models=None):
    X = dataset.drop('diagnosis', axis=1)
    y = dataset['diagnosis']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Almacenamos todos los modelos definidos en una variable
    models_to_evaluate = define_models().items()
    results_list = []
    best_models = {}
    ft_importance_df = pd.DataFrame()

    # Iteramos sobre los modelos definidos
    for model_name, model in define_models().items():
        # Si se ha seleccionado un modelo (o varios) por parámetro, se omite el resto de modelos
        if selected_models is not None and model_name not in selected_models:
          continue

        # Definimos los hiperparámetros para el modelo actual
        param_grid = define_hyperparameters(model_name)
        # Evaluamos el modelo y obtenemos los resultados
        results_dict, best_model, feature_importance_list = evaluate_model(model_name, model, param_grid, X_train, y_train, X_test, y_test)
        # Agregamos los resultados a la lista
        results_list.append(results_dict)
        # Convertimos la lista de importancias de características en un DataFrame
        ft_importance_temp_df = pd.DataFrame(feature_importance_list, columns=['Característica', 'Importancia'])
        # Agregamos la columna 'Modelo' para poder acceder mejor a su contenido
        ft_importance_temp_df['Modelo'] = model_name
        # Reorganizamos el DataFrame para mantener solo las columnas definidas
        ft_importance_temp_df = ft_importance_temp_df[['Modelo', 'Característica', 'Importancia']]
        # Concatenamos el DataFrame de importancias de características
        ft_importance_df = pd.concat([ft_importance_df, ft_importance_temp_df], ignore_index=True)
        # Ordenamos por nombre del modelo y luego por importancia de características
        ft_importance_df.sort_values(by=['Modelo', 'Importancia'], ascending=[True, False], inplace=True)
        # Restablecemos los índices después de ordenar
        ft_importance_df.reset_index(drop=True, inplace=True)
        # Por último, almacenamos el mejor modelo en el diccionario de mejores modelos
        best_models[model_name] = best_model

    # Todos los resultados se guardan en un DataFrame
    results_df = pd.DataFrame(results_list)

    return results_df, ft_importance_df, best_models
```

4.5.6. Evaluación de los conjuntos de datos

Tras haber definido las funciones necesarias, procedimos a obtener los resultados para todos los conjuntos de datos definidos al inicio de esta sección.

Para cada conjunto de datos, generamos una tabla que incluye el nombre del modelo, los hiperparámetros del mejor modelo y las métricas obtenidas al aplicar dicho modelo al conjunto de prueba: accuracy, precision, recall y F1-score. También se guardan la lista de importancia de las características y las instancias de los mejores modelos.

En este apartado mostraremos solo las tablas obtenidas para cada modelo, y en el siguiente epígrafe estudiaremos los resultados.

Tabla 4. Resultados de los modelos en datos originales

Modelo	**Mejores hiperpárametros**	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Logistic Regression	{'C': 1000, 'penalty': 'l2'}	95.906433	93.750000	95.238095	94.488189
K-Nearest Neighbors	{'n_neighbors': 3}	94.152047	93.442623	90.476190	91.935484
Support Vector Machines	{'C': 100, 'kernel': 'linear'}	94.736842	92.187500	93.650794	92.913386
Random Forest	{'max_depth': 20, 'n_estimators': 50}	96.491228	96.721311	93.650794	95.161290
XGBoost	{'learning_rate': 0.1, 'n_estimators': 150}	95.906433	95.161290	93.650794	94.400000

Tabla 5. Resultados de los modelos en datos reducidos y escalados

Modelo	**Mejores hiperpárametros**	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Logistic Regression	{'C': 1000, 'penalty': 'l2'}	93.567251	87.142857	96.825397	91.729323
K-Nearest Neighbors	{'n_neighbors': 5}	95.321637	95.081967	92.063492	93.548387
Support Vector Machines	{'C': 1000, 'kernel': 'linear'}	91.812865	85.507246	93.650794	89.393939
Random Forest	{'max_depth': 10, 'n_estimators': 50}	95.906433	93.750000	95.238095	94.488189
XGBoost	{'learning_rate': 0.2, 'n_estimators': 100}	97.076023	95.312500	96.825397	96.062992

Tabla 6. Resultados de los modelos en datos tratados con RFE

Modelo	**Mejores hiperpárametros**	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Logistic Regression	{'C': 10, 'penalty': 'l2'}	98.245614	96.875000	98.412698	97.637795
K-Nearest Neighbors	{'n_neighbors': 3}	94.736842	92.187500	93.650794	92.913386
Support Vector Machines	{'C': 0.1, 'kernel': 'linear'}	97.660819	96.825397	96.825397	96.825397
Random Forest	{'max_depth': None, 'n_estimators': 100}	96.491228	95.238095	95.238095	95.238095
XGBoost	{'learning_rate': 0.2, 'n_estimators': 100}	96.491228	95.238095	95.238095	95.238095

Tabla 7. Resultados de los modelos en datos tratados con RFECV

Modelo	**Mejores hiperpárametros**	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Logistic Regression	{'C': 100, 'penalty': 'l2'}	96.491228	95.238095	95.238095	95.238095
K-Nearest Neighbors	{'n_neighbors': 3}	94.736842	93.548387	92.063492	92.800000
Support Vector Machines	{'C': 100, 'kernel': 'linear'}	96.491228	95.238095	95.238095	95.238095
Random Forest	{'max_depth': None, 'n_estimators': 50}	97.660819	98.360656	95.238095	96.774194
XGBoost	{'learning_rate': 0.1, 'n_estimators': 100}	96.491228	95.238095	95.238095	95.238095

5. Resultados

En esta sección comentaremos los resultados obtenidos para cada *dataset*, comenzando por el conjunto de datos original, contenido en la variable '*data*'.

Al aplicar el conjunto completo de algoritmos definidos en este trabajo, lo primero que podemos apreciar es el elevado coste computacional del proceso, provocando que el tiempo de obtención de los resultados que oscila entre 4 y 5 minutos. Este resultado era previsible, dado el volumen de características a procesar y la ausencia de escalado de los datos.

Además, se debe tener en cuenta la naturaleza inherente de algunos algoritmos (como *Random Forest*) y la manera en que se ha diseñado el entrenamiento de los modelos. Este último aspecto se refiere a la implementación de la validación cruzada, que, si bien mejora los resultados, requiere entrenar en 10 ocasiones (se recuerda que hemos empleado 10 *folds*) a cada modelo, por lo que el tiempo se esperaba fuese incrementado.

Después de haber mencionado esto, las métricas obtenidas indican que los algoritmos aplicados presentan un rendimiento elevado, pese a que los datos no hayan sido tratados. Esto puede deberse a cómo habían sido los datos recogidos para formar el *dataset*; por ejemplo, no había valores nulos ni duplicados, y los datos se obtuvieron mediante la técnica de MSM-T, lo que puede haber refinado aún más los datos obtenidos y almacenados en el conjunto. Para terminar, debemos mencionar que el algoritmo *Random Forest* destaca en precisión, mientras que la Regresión Logística lo supera en sensibilidad; como hemos explicado en esta memoria, es esta última métrica la que más nos importa, aunque si comparamos el valor F1-score de ambos algoritmos, que recordemos se construye a partir de la precisión y la sensibilidad, el mejor modelo para este conjunto de datos sería *Random Forest*.

Analicemos los datos de '*data_standard*', que habían sido tratados con la eliminación de las variables independientes altamente correlacionadas y el escalado de sus datos mediante estandarización. Respecto a su estado inicial, observamos que algunas métricas poseen un valor inferior al umbral del 90%, mientras que algunas mejoran sus resultados. Por ejemplo, la precisión de la regresión logística se reduce del 93,75% del conjunto original a 87,14%. No obstante, la sensibilidad se ha mantenido o incluso mejorado en todos los algoritmos, por lo que el rendimiento en este aspecto ha sido positivo. Con un 96,83%, tanto la regresión logística como XGBoost serían las mejores opciones para evitar la clasificación errónea de tumores malignos, aunque aún hay margen de mejora.

Los algoritmos aplicados a los datos tratados con RFE poseen un rendimiento superior que los dos conjuntos analizados anteriormente (exceptuando casos aislados). Queremos destacar el algoritmo de regresión logística, cuyo promedio de rendimiento para nuestras cuatro métricas supera el 97%, con especial énfasis en la exactitud y la sensibilidad, ambas con valores superiores al 98%.

Comprobemos el rendimiento para los datos transformados con RFECV, donde se observa que los algoritmos han demostrado una calidad superior en comparación con los dos primeros conjuntos. Sin embargo, cabe destacar que los resultados son menos favorables en comparación con los obtenidos mediante RFE. A pesar de que su objetivo es obtener el número óptimo de características mediante validación cruzada, esta discrepancia resalta la importancia de evaluar el rendimiento de diferentes enfoques de selección de características, ya que la inclusión de validación cruzada no siempre garantiza una mejora sustancial.

Para terminar con este apartado, queremos presentar la definición de una función que nos permite detectar qué conjunto de datos y qué algoritmos poseen el valor máximo para una métrica concreta. La función es la siguiente:

Figura 32. Definición de función de extracción de mejor sensibilidad

```python
def find_best_metric_row(*tables_and_names, metric):
    best_metric_row = None
    best_metric_value = float('-inf')

    # Lista para almacenar las tablas temporales
    temp_tables = []

    for table, dataset_name in tables_and_names:
        if isinstance(table, pd.DataFrame):
            # Creamos una copia temporal del DataFrame original
            temp_table = table.copy()

            # Se verificar si la tabla temporal tiene la columna 'Dataset'
            if 'Dataset' not in temp_table.columns:
                # Si no tiene la columna 'Dataset', añadirla al principio
                temp_table.insert(0, 'Dataset', dataset_name)
            else:
                # Si ya tiene la columna 'Dataset', insertar el valor dataset_name
                temp_table.loc[:, 'Dataset'] = dataset_name

            # Agregamos la tabla temporal a la lista
            temp_tables.append(temp_table)

            # Buscamos el máximo valor de la métrica en la tabla temporal
            max_metric_row = temp_table.loc[temp_table[metric].idxmax()]

            # Y verificamos si el máximo valor de la métrica en esta tabla supera al máximo global
            if max_metric_row[metric] > best_metric_value:
                best_metric_row = max_metric_row
                best_metric_value = max_metric_row[metric]

    # Se crea un nuevo DataFrame con la fila del mejor valor de la métrica
    best_metric_df = pd.DataFrame([best_metric_row])

    return best_metric_df
```

Si invocamos esta función indicando que deseamos obtener el mayor valor para la sensibilidad (*Recall*), obtenemos el siguiente resultado:

Tabla 8. Conjunto de datos y algoritmo con mejor sensibilidad

Dataset	Modelo	Mejores hiperpárametros	*Accuracy*	*Precision*	*Recall*	*F1-Score*
Data_RFE	Logistic Regression	{'C': 10, 'penalty': 'l2'}	98.245614	96.875	98.412698	97.637795

Por lo que se demuestra que el algoritmo que presenta el mayor valor de *'recall'* o sensibilidad es la regresión logística (*Logistic Regression*). Este resultado se obtuvo al aplicar el algoritmo al conjunto de datos que fue sometido a la técnica RFE, previo haber sido procesado con el análisis de la matriz de correlación y la normalización de sus datos.

También podemos extraer la importancia de las características si seleccionamos el conjunto de datos y el algoritmo de entre la lista de importancias obtenida con la función 'run_dataset_evaluation' (pero considerando que K-NN no tiene implementada este atributo). Emplearemos el resultado obtenido anteriormente para comprobar qué características son las más importantes en este modelo:

Figura 33. Importancia de las características para mejor modelo

```
model_name = 'Logistic Regression'
ft_importance_list = ft_importance_data_RFE
ft_importance_model = ft_importance_list[ft_importance_list['Modelo'] == model_name]
print(ft_importance_model)
```

	Modelo	Característica	Importancia
0	Logistic Regression	radius_worst	4.099874
1	Logistic Regression	concave points_mean	3.687922
2	Logistic Regression	radius_se	2.338202
3	Logistic Regression	concave points_worst	2.129113
4	Logistic Regression	texture_mean	1.764155
5	Logistic Regression	smoothness_worst	1.238813
6	Logistic Regression	symmetry_worst	1.071865
7	Logistic Regression	compactness_worst	0.937686
8	Logistic Regression	radius_mean	-0.246601
9	Logistic Regression	compactness_mean	-3.575471

6. Conclusiones

En el presente proyecto, hemos realizado un estudio exhaustivo sobre el cáncer de mama, enfermedad de gran importancia para la salud a nivel global. Para poder afrontar esta situación, desarrollamos un sistema capaz de determinar, a partir de datos tumorales, la naturaleza maligna o benigna de éstos.

La construcción de los modelos ha implicado afrontar los desafíos que concierne un proyecto de aprendizaje automático. Desde la selección de los datos correctos, hasta su análisis y tratamiento, finalizando así con la implementación de diversos algoritmos que pudiesen ajustarse al tema trabajado.

Los resultados obtenidos han sido en su mayoría favorables, con un rendimiento superior al 90%, por lo que podríamos considerar que hemos podido desarrollar modelos eficaces en la detección de malignidad de un tumor de cáncer de mama.

Sin embargo, somos conscientes de que este proyecto podría haberse enfocado con otra perspectiva, con decisiones completamente diferentes o la aplicación de otro tipo de técnicas. En esta memoria hemos comentado, en ciertos puntos, la posibilidad de afrontar algunas situaciones con una perspectiva distinta; de hecho, este conjunto de datos ha sido tratado de muchas maneras diferentes, con resultados dispares, aspecto que hemos aprovechado como una oportunidad para aprender a afrontar nuestro proyecto.

En otros aspectos, esta labor nos ha brindado la oportunidad de explorar un tema del cual carecíamos de la información necesaria para comprender completamente su alcance. Asimismo, nos ha permitido vislumbrar las aplicaciones de la inteligencia artificial en el ámbito de la medicina.

Por último, también queremos mencionar que la ejecución de un proyecto de esta envergadura nos ha permitido experimentar con aspectos inesperados que han servido como experiencias de las que esperamos hayamos aprendido.

7. Bibliografía

1. UC Irvine Machine Learning Repository. (n.d.). Breast Cancer Wisconsin (Diagnostic) Data Set. [en línea]. Recuperado de https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
2. UCI Machine Learning Repository. (n.d.). Breast Cancer Wisconsin (Diagnostic) Data Set. [en línea]. Recuperado de https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
3. Coursera. (n.d.). Kaggle. [en línea]. Recuperado de https://www.coursera.org/articles/kaggle
4. Universitat Oberta de Catalunya. (n.d.). Repositorio UOC-ML. [en línea]. Recuperado de http://datascience.recursos.uoc.edu/es/repositorio-uci-ml/
5. Real Academia Española. (2001). Cáncer. [en línea]. Recuperado de https://www.rae.es/drae2001/c%C3%A1ncer
6. Real Academia Española. (2001). Neoplasia. [en línea]. Recuperado de https://www.rae.es/drae2001/neoplasia
7. World Health Organization. (2022). Cancer. [en línea]. Recuperado de https://www.who.int/es/news-room/fact-sheets/detail/cancer
8. National Cancer Institute. (2022). Cáncer. [en línea]. Recuperado de https://www.cancer.gov/espanol
9. Sociedad Española de Oncología Médica. (2022). Cáncer de mama. [en línea]. Recuperado de https://seom.org/info-sobre-el-cancer/cancer-de-mama?showall=1&start=0
10. World Health Organization. (2022). Breast cancer. [en línea]. Recuperado de https://www.who.int/es/news-room/fact-sheets/detail/breast-cancer
11. Rodríguez-Salés, V., Ortiz-Barreda, G., & de Sanjosé, S. (2014). Revisión bibliográfica sobre la prevención del cáncer en personas inmigrantes residentes en España. Revista Española de Salud Pública, 88(6), 687-698. [en línea]. Recuperado de https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1135-57272014000600006
12. American Cancer Society. (2022). Cancer Facts & Figures 2022. [en línea]. Recuperado de https://acsjournals.onlinelibrary.wiley.com/doi/10.3322/caac.21660
13. Koss, L. G. (1992). Fine Needle Aspiration Biopsy of Nonpalpable Breast Lesions: A Review of 1885 Cases with Emphasis on Diagnostic Criteria and Cytologic-Histologic Correlations. Diagnostic Cytopathology, 8(3), 251-258. [en línea]. Recuperado de https://doi.org/10.1080/10556789208805504
14. O'Leary, D. P., & O'Neil, R. (2016). MSMT Multi-Sensor-Multi-Target Data Set. [en línea]. Recuperado de https://pages.cs.wisc.edu/~olvi/uwmp/msmt.html
15. Python Software Foundation. (n.d.). Python Brochure. [en línea]. Recuperado de https://www.python.org/doc/essays/blurb/
16. FreeCodeCamp. (2021). Lenguajes compilados vs interpretados: ¿cuál es la diferencia? [en línea]. Recuperado de https://www.freecodecamp.org/espanol/news/lenguajes-compilados-vs-interpretados/

17. Real Python. (n.d.). Object-Oriented Programming (OOP) in Python 3. [en línea]. Recuperado de https://realpython.com/python3-object-oriented-programming/#what-is-object-oriented-programming-in-python
18. DesarrolloWeb. (2017). ¿Qué es Python? [en línea]. Recuperado de https://desarrolloweb.com/articulos/2358.php
19. New Horizons. (n.d.). Why is Python Used for Machine Learning? [en línea]. Recuperado de https://www.newhorizons.com/resources/blog/why-is-python-used-for-machine-learning
20. NumPy. (n.d.). NumPy. [en línea]. Recuperado de https://numpy.org/
21. Scikit-learn. (n.d.). Scikit-learn. [en línea]. Recuperado de https://scikit-learn.org/stable/
22. Google. (n.d.). Google Colaboratory. [en línea]. Recuperado de https://colab.research.google.com/?hl=es
23. Google. (n.d.). Colaboratory FAQ. [en línea]. Recuperado de https://research.google.com/colaboratory/intl/es/faq.html
24. GitHub. (n.d.). GitHub. [en línea]. Recuperado de https://github.com/
25. GitHub. (n.d.). Hello World. [en línea]. Recuperado de https://docs.github.com/es/get-started/quickstart/hello-world
26. Müller, A. C., & Guido, S. (2017). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, Inc. [en línea]. Recuperado de https://www.oreilly.com/library/view/introduction-to-machine/9781449369880/
27. Fayrix. (2019). Machine Learning Metrics: A Comprehensive Guide. [en línea]. Recuperado de https://fayrix.com/machine-learning-metrics_es
28. Brownlee, J. (2021). Metrics to Evaluate Machine Learning Algorithms in Python. [en línea]. Recuperado de https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/
29. The Machine Learners. (n.d.). Métricas de clasificación. [en línea]. Recuperado de https://www.themachinelearners.com/metricas-de-clasificacion/
30. Neptune. (2021). Performance Metrics in Machine Learning: Complete Guide. [en línea]. Recuperado de https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide
31. AltexSoft. (2019). Machine Learning Metrics: How to Evaluate Your Model. [en línea]. Recuperado de https://www.altexsoft.com/blog/machine-learning-metrics/
32. scikit-learn. (n.d.). Model evaluation: quantifying the quality of predictions. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/model_evaluation.html
33. scikit-learn. (n.d.). sklearn.metrics. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
34. scikit-learn. (n.d.). sklearn.metrics.accuracy_score. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

35. scikit-learn. (n.d.). sklearn.metrics.precision_score. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html
36. scikit-learn. (n.d.). sklearn.metrics.recall_score. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html
37. scikit-learn. (n.d.). sklearn.metrics.f1_score. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
38. scikit-learn. (n.d.). sklearn.metrics.confusion_matrix. [en línea]. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
39. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics
40. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly. [en línea]. Recuperado de https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/
41. O'Reilly Media. (n.d.). The Machine Learning Algorithms for Clustering. [en línea]. Recuperado de https://learning.oreilly.com/library/view/the-machine-learning/9781805122500/
42. Saleh, H. (2018). *Machine Learning Fundamentals*. Packt Publishing.
43. Aamir S, Rahim A, Aamir Z, Abbasi SF, Khan MS, Alhaisoni M, Khan MA, Khan K, Ahmad J. Predicting Breast Cancer Leveraging Supervised Machine Learning Techniques. Comput Math Methods Med. 2022 Aug 16;2022:5869529. doi: 10.1155/2022/5869529. PMID: 36017156; PMCID: PMC9398810. [en línea]. Recuperado de https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9398810/