UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Comparación de técnicas de machine learning y estadística multivariante para la predicción del cáncer de mama utilizando biomarcadores obtenidos a partir de imágenes de resonancia magnética

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Alcalá Belmonte, Miguel

Tutor/a: Ferrer Riquelme, Alberto José

Director/a Experimental: AGUADO SARRIO, ERIC

CURSO ACADÉMICO: 2021/2022

Resumen

La motivación del presente trabajo es abordar un estudio acerca de la implementación de técnicas de *machine learning* y estadística multivariante a un problema real y muy grave, como es el diagnóstico del cáncer de mama. Esto se llevará a cabo mediante el empleo de dichas técnicas a unos biomarcadores obtenidos mediante resonancia magnética.

En primer lugar, se pondrá en contexto el problema en cuestión, el cáncer de mama, la resonancia magnética y la obtención de sus respectivos biomarcadores.

Más adelante, se hablará del estado del arte de la Inteligencia Artificial, especialmente en el ámbito de la salud y la medicina.

Se plantearán distintos modelos, quedarán expuestos sus fundamentos matemáticos/estadísticos, y los resultados esperados de cada uno.

Para finalizar, se aplicarán a los datos, y se verán los resultados esperados de cada uno.

Palabras clave: cáncer, mama, *machine learning*, inteligencia artificial

Abstract

The motivation of the present work is to approach a study about the implementation of machine learning and multivariate statistics techniques to a real and very serious problem, as is the diagnosis of breast cancer. This will be done through the use of these techniques to some biomarkers obtained by magnetic resonance imaging.

First, the problem in question, breast cancer, magnetic resonance imaging and the obtaining of their respective biomarkers, will be put into context.

Later, the state of the art of Artificial Intelligence, especially in the field of health and medicine, will be discussed.

Different models will be introduced, their mathematical/statistical foundations will be explained, and the expected results of each one.

Finally, they will be applied to data, and the expected results of each one will be shown.

Keywords : cancer, breast, machine learning, artificial intelligence

Índice de figuras

Índice de tablas

Índice de contenidos

1. Introducción

El aprendizaje automático (*machine learning*) es una rama de la inteligencia artificial que permite que las máquinas aprendan sin ser expresamente programadas para ello. Una habilidad indispensable para hacer sistemas capaces de identificar patrones entre los datos para hacer predicciones. Esta tecnología está presente en un sinfín de aplicaciones como las recomendaciones de HBO, las respuestas inteligentes del correo electrónico, o los automóviles que se desplazan sin necesidad de conductor.

En pocas palabras, el *machine learning* se basa en el reconocimiento de patrones. Mediante su uso, se consigue convertir una muestra de datos en un programa informático capaz de extraer inferencias de nuevos conjuntos de datos para los que no ha sido entrenado previamente.

Mediante el empleo de técnicas de *machine learning* se puede reducir potencialmente el número de fallos en algunas situaciones en las que la toma de decisiones es fundamental.

Dichos errores pueden provocar desde un fallo a la hora de empaquetar un pedido *online*, a accidentes aéreos o malos diagnósticos sanitarios. Aquí es donde las técnicas de *machine learning* tienen la oportunidad de detectar estos errores y mitigarlos, ya que la mayoría de estos errores se deben a fallos humanos.

La interpretación de los datos que aparecen en forma de imagen o vídeo puede ser una tarea difícil. Los expertos en la materia tienen que formarse durante muchos años para alcanzar la capacidad de discernir los fenómenos médicos y, además, tienen que aprender activamente nuevos contenidos. Sin embargo, la demanda es cada vez mayor y hay una importante escasez de expertos en este campo. Por consiguiente, es necesario un nuevo abordaje, y la Inteligencia Artificial promete ser la herramienta que se utilizará para llenar este vacío en la demanda.

La esperanza de vida de la población mundial ha aumentado llamativamente en los últimos años y sigue incrementándose. Se estima, que para 2050, el número de personas de 65 años o más alcanzará los 1500 millones, lo que equivale aproximadamente al 16% de la población mundial. Las principales amenazas de salud que afectan a las personas están pasando a ser enfermedades que reflejan el envejecimiento y el cambio del estilo de vida, especialmente en países desarrollados, donde el aumento de enfermedades crónicas no transmisibles, como las enfermedades cardíacas o el cáncer, constituyen el principal problema de salud. Se estima que para 2050, el 70% de las enfermedades sean de este tipo.

Con el aumento de las historias clínicas electrónicas, los datos de estilo de vida y de salud y la capacidad de realizar un mejor y más rápido análisis de los datos, las tendencias digitales están cambiando profundamente el sistema sanitario.

Esto también está conduciendo a avances en los tratamientos basados en las correlaciones que se generan por la recogida y la integración de los datos de la asistencia sanitaria. El uso de estos datos en grandes volúmenes es relativamente nuevo para el sector sanitario, en comparación con otros sectores bien establecidos, como los servicios financieros. Recientemente, muchas empresas digitales están tratando de irrumpir en el sistema sanitario mediante la adopción de un abordaje tecnológico de los datos de la asistencia sanitaria.

Por ejemplo, gracias a la aparición de las HCE (historias clínicas electrónicas), se pueden recopilar, por métodos digitales y de forma sistematizada, muchos aspectos de la salud del paciente, que puede ser útil en diversos aspectos asistenciales (información recopilada por los médicos en las consultas, historial clínico, diagnóstico, planes de tratamiento…).

Como la HCE, hay muchísimas herramientas, que permiten recoger enormes volúmenes de datos sobre la asistencia sanitaria, lo que hace que se necesiten métodos analíticos potentes para procesar e interpretar esos datos a fin de que tengan sentido y creen valor.

Figura 1. Ejemplos de obtención de datos en la asistencia sanitaria

Mientras que los datos masivos como entidad son inútiles, el procesamiento de datos masivos para hacer predicciones o tomar decisiones utilizando la inteligencia artificial tiene el potencial de transformar la práctica actual de la asistencia clínica. [1]

2. Cáncer de mama

De entre todos los cánceres con mayor prevalencia, el cáncer de mama es el tumor maligno más frecuente en mujeres y la primera causa de muerte por cáncer en mujeres en los países desarrollados (Tabla 1). Su incidencia está en aumento, pero la mortalidad está en descenso gracias a los avances en el tratamiento y al diagnóstico precoz. [2]

Los factores de riesgo son:

- Factores genéticos y familiares
- Edad tardía de primer embarazo
- Menarquia precoz
- Menopausia tardía
- Nuliparidad
- Terapia hormonal sustitutiva
- Irradiación
- Antecedentes personales de otros cánceres
- Alto nivel socioeconómico, dieta rica en grasas y obesidad
- Sindrome de Klinefelter

INCIDENCIA Y MORTALIDAD POR CÁNCER EN ESPAÑA				
INCIDENCIA	1º	2º	3º	4º
Global	Colorrectal	Próstata	Pulmón	**Mama**
Hombres	Próstata	Pulmón	Colorrectal	
Mujeres	**Mama**	Colorrectal	Útero	
MORTALIDAD	1º	2º	3º	
Global	Pulmón	Colorrectal	Páncreas	
Hombres	Pulmón	Colorrectal	Próstata	
Mujeres	**Mama**	Colorrectal	Pulmón	

Tabla 1. Informe de la sociedad española de oncología médica

Existen varios tipos de cáncer de mama diferenciados por su tipología y localización, aunque los que aparecen con más frecuencia son dos:

- **El carcinoma ductal in situ (CDIS)** es un cáncer de mama no infiltrante en el que las células anormales se encuentran en la pared interior del conducto galactóforo (canal que sirve para llevar la leche al pezón).

- **El carcinoma lobular in situ (CLIS)** es un tipo de cáncer que se desarrolla en las células de la glándula mamaria y permanece delimitado en el interior del lóbulo.

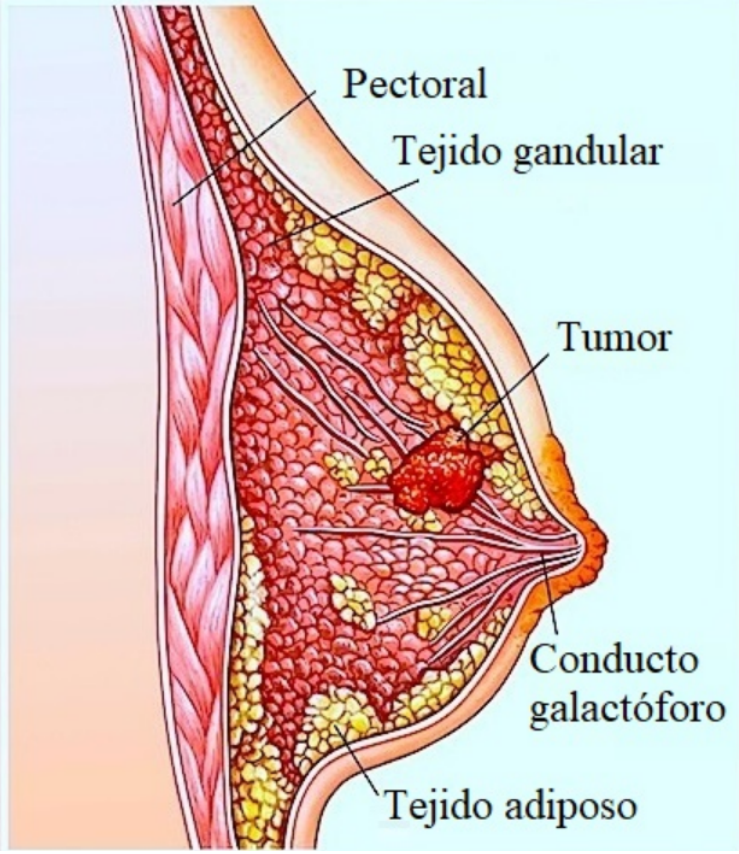


Figura 2. Estructura interna de la mama con tumor

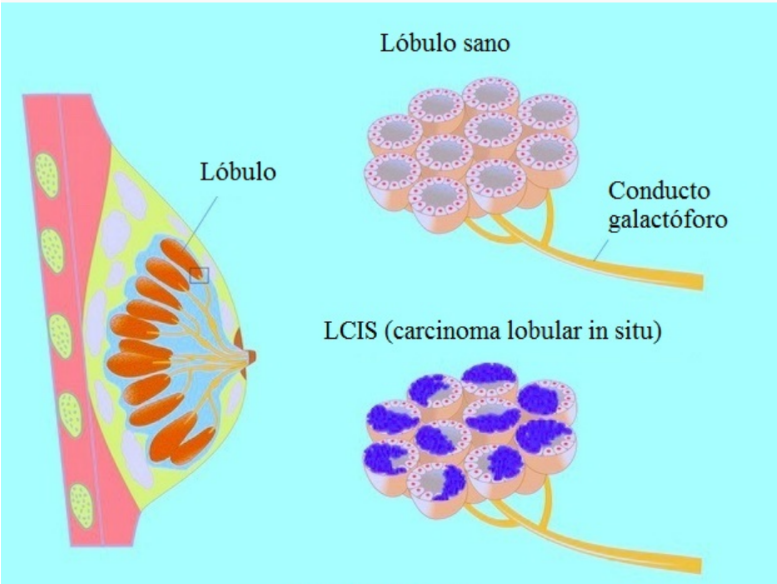


Figura 3. Lóbulos según el tipo de cáncer

Aparte de la localización, el cáncer está definido también por la extensión que ha alcanzado, así como el estadio en el que se encuentra.

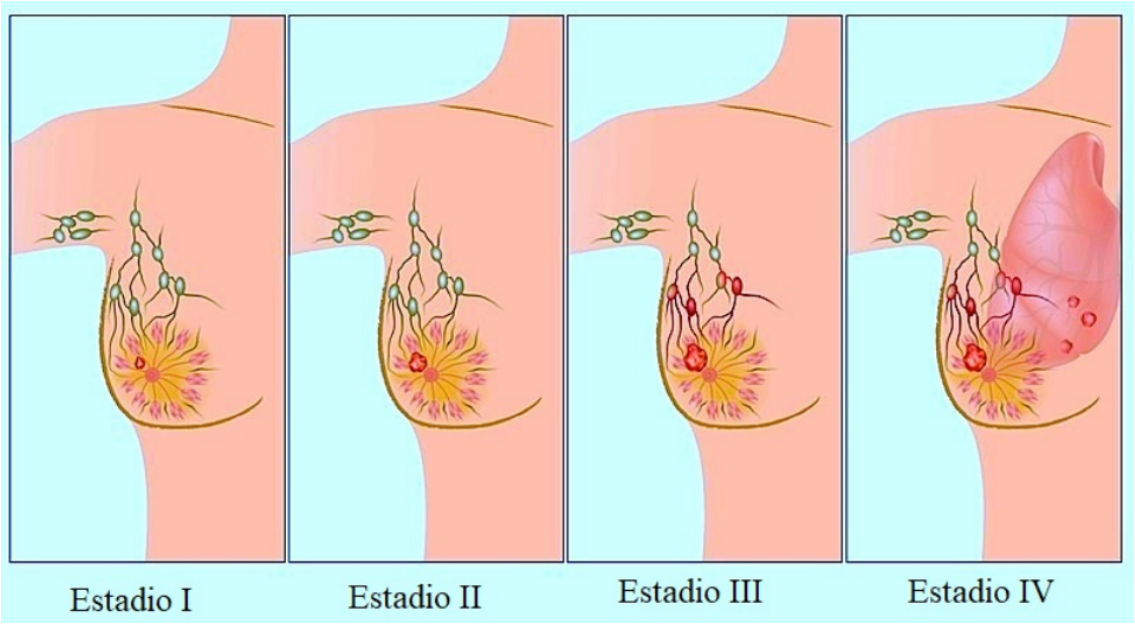


Figura 4. Estadios del cáncer de mama

La importancia de la detección temprana del cáncer de mama proviene de dos factores. El primero es la alta tasa de mortalidad que tiene el cáncer cuando progresa a estadios más avanzados. La segunda surge a partir de la elevada tasa de supervivencia que tienen los pacientes a los que se les ha detectado y tratado en el primer estadio.

Esta elevada tasa de supervivencia proviene de que en los primeros estadíos es posible delimitar con mucha precisión el alcance del cáncer ya que la metástasis se produce a través del sistema linfático. A cada región de la mama le corresponde un único ganglio linfático de la axila, por lo que se emplea un contraste para detectar dicho ganglio una vez diagnosticado el cáncer. En el caso de que no se detecten células cancerígenas en el ganglio es muy probable que el cáncer sea únicamente local.

Para la detección temprana del cáncer, la técnica más extendida es el *screening* de la población mediante mamografía. Esta técnica es adecuada para la detección de neoplasias en las mamas pero una vez detectadas no resulta adecuada para la caracterización de la lesión encontrada.

Para una detección más precisa y una mejor caracterización del cáncer es necesario recurrir a otras técnicas, entre las que destaca la resonancia magnética debido a su gran flexibilidad por las múltiples configuraciones que es posible emplear y la capacidad de obtener información útil para el desarrollo de biomarcadores que ayuden en el diagnóstico y tratamiento.

Debido a la falta de indicadores más allá de los genéticos, que permitan enfocar los esfuerzos en la prevención, y al elevado número de muertes que provoca, el cáncer de mama ha sido el foco de atención de la comunidad sanitaria durante mucho tiempo, dirigiendo gran parte de los esfuerzos al diagnóstico precoz de la enfermedad. La tasa de supervivencia de este tipo de

cánceres es casi del 100% si se detectan en estadios muy tempranos, descendiendo drásticamente a menos del 15% cuando se detectan en estadios avanzados.

Entre la gran variedad de métodos de diagnóstico existentes, el más extendido en la actualidad es el cribado de la población mediante el uso de la mamografía, que se basa en el uso de rayos X con el fin de identificar alteraciones que puedan indicar la presencia de cáncer.

Aunque la mamografía se utiliza ampliamente y muchos países tienen programas de detección precoz del cáncer basados en ella, esta técnica presenta varios problemas. Por un lado, el uso de radiaciones ionizantes siempre supone un riesgo. También hay que considerar las molestias asociadas a la aplicación de la técnica. Por último, debido a las características de la imagen radiográfica, en la que sólo se pueden apreciar proyecciones bidimensionales de las densidades de un volumen tridimensional, la detección precoz de todos los tumores (incluidos los incipientes), o la medición correcta de su tamaño o forma, presentan dificultades.

Por este motivo, se han desarrollado otras técnicas de imagen diferentes, entre ellas la resonancia magnética.

3. Resonancia magnética

La resonancia magnética nuclear (RMN) de mama es hoy en día una técnica muy consolidada en la valoración de la patología mamaria y cada vez tiene más aplicaciones clínicas como método diagnóstico complementario a la mamografía y la ecografía.

Es la técnica con mayor sensibilidad para detectar cáncer de mama (90-100%), aunque presenta una baja/moderada especificidad (60-80%), por lo que es muy importante correlacionar los hallazgos con antecedentes personales, exploración física, mamografía y ecografía para evitar realizar biopsias innecesarias, ansiedad en la paciente y gasto innecesario.

Hay estudios que afirman que cuando se diagnostica un cáncer de mama, existe un 3-5% de probabilidad de que la RMN detecte un cáncer mamográficamente oculto en la mama contralateral. [3]

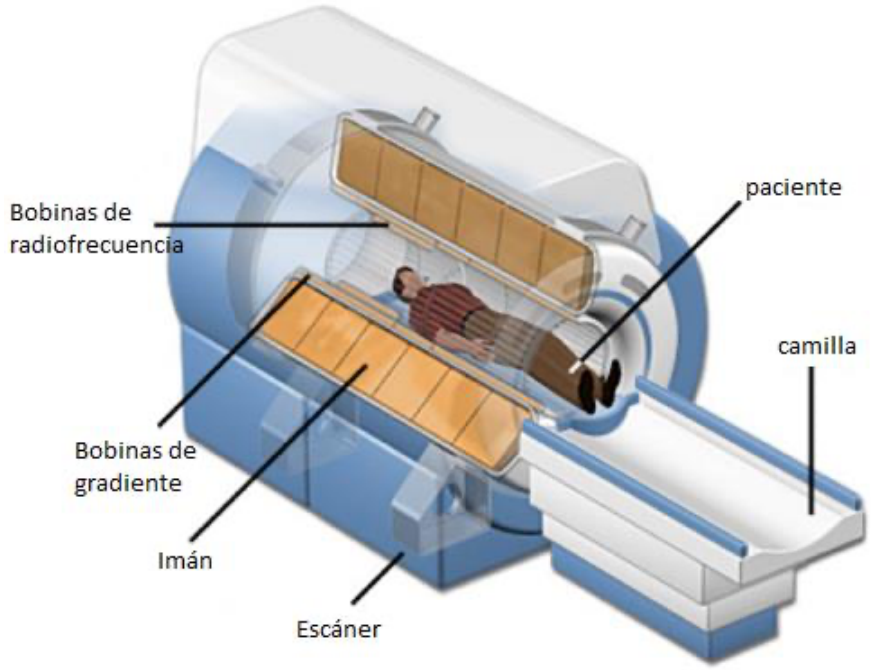


Figura 5. Estructura de máquina de resonancia magnética

Esta técnica es muy recomendable por su capacidad para detectar precozmente la angiogénesis y la neovascularización, así como la proliferación celular, principales indicadores de los procesos tumorales. Normalmente, la información anatómica se obtiene a través de secuencias morfológicas de RMN. Típicamente, la RMN muestra el tumor como una lesión focal de diferentes intensidades de señal con respecto al tejido circundante. Dentro de estas secuencias de RMN se encuentran las secuencias potenciadas en T2, que incluyen imágenes morfológicas en escala de grises relacionadas con los tiempos de disminución de la magnetización transversal de las moléculas de agua. Estas imágenes son muy útiles para determinar la extensión del tumor en algunos casos, ya que permite valorar la infiltración del tumor a otras estructuras.

Sin embargo, para estudiar los procesos fisiológicos indicadores de desarrollo tumoral (como la vascularización y la proliferación celular), se han desarrollado técnicas de adquisición de imágenes funcionales en RMN. Las respuestas fisiopatológicas preceden a las morfológicas y

pueden cuantificarse mediante técnicas funcionales. Por tanto, las imágenes funcionales obtenidas mediante RMN tienen un papel muy importante en el diagnóstico precoz y, además, en la evaluación de la respuesta tumoral, facilitando las aplicaciones personales de medicina de precisión. Este tipo de imágenes, complementarias a la RMN convencional (insuficiente para discriminar la extensión de la parte invasiva del tumor), proporcionan un método de diagnóstico con alta sensibilidad y especificidad, basado en la adquisición, entre otros, de la información sobre la vascularización y la celularización de los tejidos.

Las dos técnicas funcionales más importantes para evaluar dichos fenómenos son la difusión (celularización) y la perfusión (vascularización). Siendo la primera de ellas no invasiva, lo que supone una ventaja en lo que se refiere a la salud del paciente. La perfusión es ligeramente invasiva (introduce un contraste vía intravenosa) pero afecta mucho menos a la seguridad del paciente que otras técnicas de imagen como los rayos X.

La difusión se refiere al movimiento aleatorio de moléculas en un fluido (líquido o gas). Dependiendo de la libertad con la que dichas partículas pueden moverse se habla de difusión isotrópica cuando el movimiento es idéntico en todas las direcciones y anisotrópica cuando presenta una restricción en alguna dirección. La RMN de difusión se emplea en el diagnóstico del cáncer debido a que la presencia de tejido tumoral altera la estructura normal del tejido, afectando a su vez la difusión dentro del mismo. La relación entre las intensidades medidas y la magnitud de la difusión fue descrita como un modelo exponencial decreciente en 1965 por Stejskal y Tannner. [4]

La secuencia del tensor de difusión (DTI) se emplea en tejidos anisotrópico como la mama (ductos mamarios) cuando los modelos isotrópicos exponenciales no son suficientemente buenos para caracterizar la movilidad de las moléculas de agua, ya que depende de la dirección en la que la difusión se calcula. Para esto, con el objetivo de caracterizar la magnitud de la difusión según la orientación de los gradientes aplicados, es necesario emplear otro tipo de secuencias de difusión modificando las direcciones en las que el gradiente del campo magnético se aplica. Por lo tanto, la caída de la señal va a ser mayor en las direcciones en las que la magnitud de la difusión sea mayor. La detección del tumor se estudia calculando en cada píxel lo que se conoce como el tensor de difusión (*Diffusion Tensor Imaging*, DTI), a partir del cual se obtienen unos biomarcadores que nos proporcionan información adicional acerca de los tejidos a analizar. [5], [6]

En el estudio de la perfusión (también es conocido por su acrónimo DCE-MRI, en inglés *Dynamic contrast-enhanced magnetic resonance imaging*), la neoangiogénesis rara vez se producen en sujetos sanos, pero está muy presente en condiciones patológicas como los tumores. Así, la formación de estos nuevos y tortuosos vasos produce un aumento de la perfusión sanguínea. En la RMN, estos procesos pueden estudiarse con la DCE-MRI. En esta técnica, se administra por vía intravenosa un medio de contraste exógeno a base de gadolinio. Se difunde desde la red capilar al espacio extracelular extravascular, vuelve al sistema vascular y es filtrado progresivamente por los riñones. El tamaño molecular relativamente pequeño de estos agentes les permite atravesar el endotelio vascular por difusión pasiva hacia el tejido, estableciendo una relación dinámica entre los cambios de intensidad de la señal de la imagen y la cantidad de medio de contraste que pasa y difunde hacia un determinado tejido. Esta secuencia requiere una alta resolución temporal, pero manteniendo la resolución espacial para poder detectar pequeñas variaciones en el interior del tejido. Así, la adquisición se realiza normalmente durante unos 5-6 minutos en diferentes instantes distribuidos de forma no-equidistante debido a que al principio se toman imágenes rápidamente, cada pocos segundos, para caracterizar mejor la captación inicial del contraste. Mientras que, en los instantes finales, se toman imágenes de forma más espaciada, ya que es la parte de la curva menos relevante. La capacidad de analizar la angiogénesis tumoral de forma cuantitativa y reproducible a partir de imágenes de DCE-MR tiene importantes aplicaciones para representar y gradar los tumores, así como para evaluar la respuesta terapéutica al principio del tratamiento.

4. Biomarcadores de imagen

En este trabajo se van a utilizar un conjunto de biomarcadores de imagen obtenidos a partir de las secuencias DTI y DCE-MRI, que conforman la base de datos con la que se han llevado a cabo los análisis.

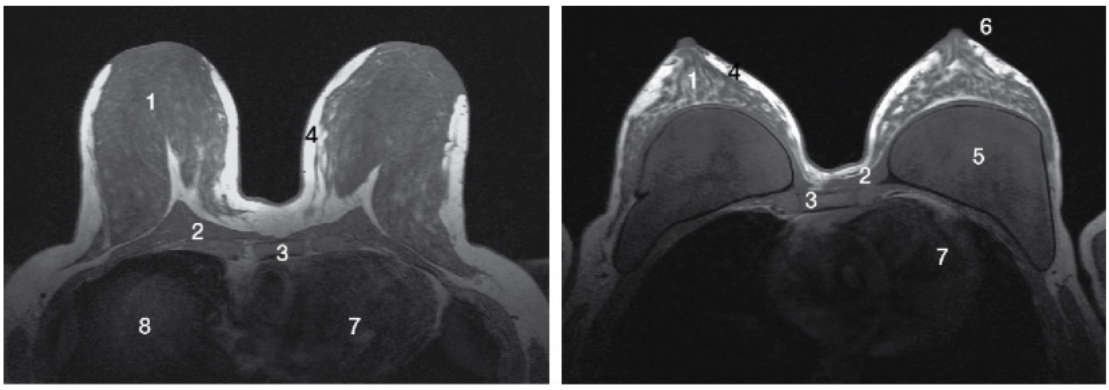


Figura 6. Imagen de resonancia magnética

En la Figura 6 vemos cómo es una imagen de resonancia magnética, a partir de la cual se obtienen los biomarcadores.

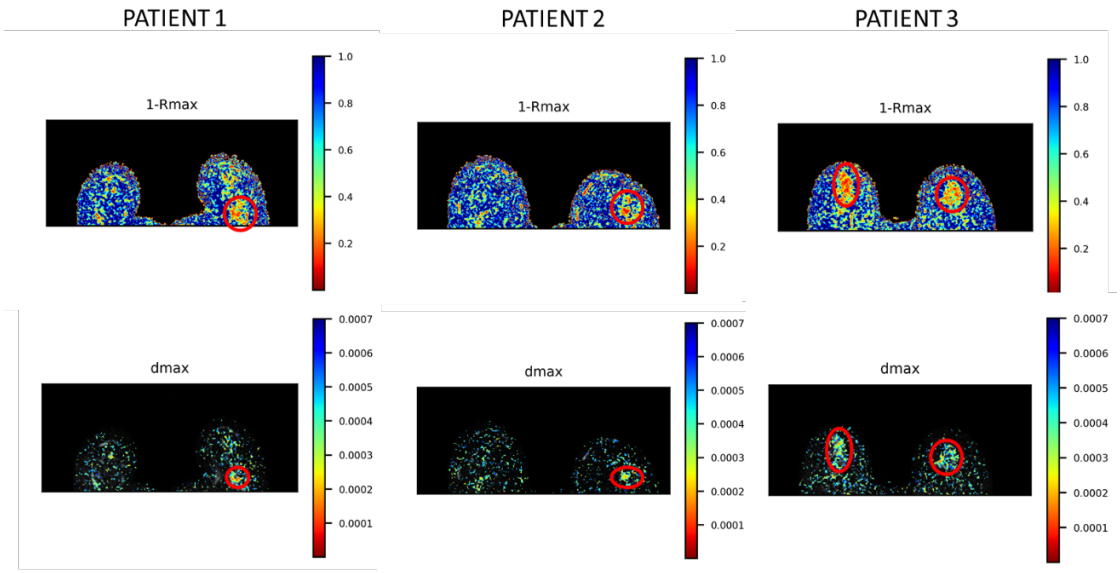


Figura 7. Ejemplos de biomarcadores DTI obtenidos a partir de la imagen.

En el ejemplo de la Figura 7, vemos dos biomarcadores reflejados para 3 pacientes distintos. Estos biomarcadores son, concretamente, 1-Rmax y dmax (cuyo significado aparece reflejado en la Tabla 3).

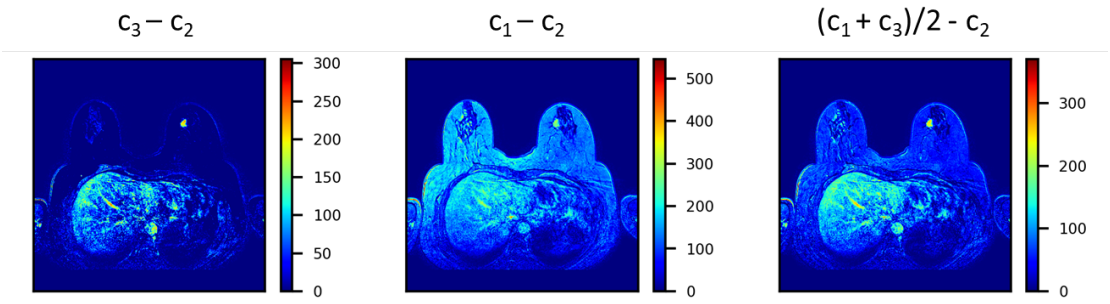


Figura 8. Ejemplos de biomarcadores de Perfusión obtenidos a partir de la imagen

En el ejemplo de la Figura 8, vemos tres biomarcadores reflejados para un paciente. Son biomarcadores de la secuencia DCE-MRI, concretamente las combinaciones de C_1, C_2 y C_3 que aparecen en la Tabla 4.

4.1 Base de datos de difusión

El modelo a partir del cual se obtienen estos biomarcadores es DTI, por sus siglas en inglés *Diffusion Tensor Imaging.*

El DTI es una matriz de 3x3 en un entorno tridimensional, conocida como tensor de difusión, que permite modelar cómo se produce la difusión en cada dirección del espacio. Así, el comportamiento de la difusión en un píxel concreto puede modelarse como un elipsoide que se representa mediante el tensor de difusión **D**, que indica la magnitud de la difusión en los 3 ejes espaciales (*"x"*, *"y"* y *"z"*) del marco de referencia [7]:

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D_{xx}} & \boldsymbol{D_{xy}} & \boldsymbol{D_{xz}} \\ \boldsymbol{D_{xy}} & \boldsymbol{D_{yy}} & \boldsymbol{D_{yz}} \\ \boldsymbol{D_{xz}} & \boldsymbol{D_{yz}} & \boldsymbol{D_{zz}} \end{pmatrix} = \boldsymbol{V} \begin{pmatrix} \boldsymbol{\lambda_1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\lambda_2} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\lambda_3} \end{pmatrix} \boldsymbol{V^T}$$

Donde la matriz **V** contiene en columnas los v_a vectores propios (*eigenvectors*) del tensor de difusión **D**, y λ_a (a=1,2,3) son sus correspondientes valores propios (*eigenvalues*), obtenidos tras diagonalizar **D**.

La figura 9 muestra el tensor de difusión **D** representado por un elipsoide en un espacio tridimensional. Nótese que este elipsoide tiene tres ejes principales, definidos por los vectores propios v_a, y sus valores propios asociados λ_a son proporcionales a los cuadrados de las longitudes de los semiejes del elipsoide.

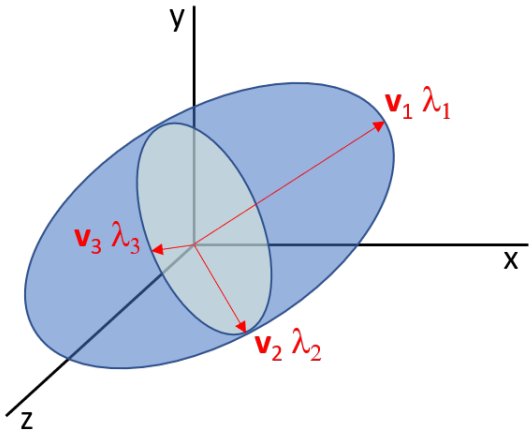


Figura 9. Tensor de difusión con ejes principales (vectores propios $\mathbf{v}_a$) y valores propios λ_a (a=1,2,3).

En esta base de datos hay tres parámetros principales, los autovalores del tensor de difusión (λ_a (a=1,2,3), a partir de los cuales se calculan el resto de los biomarcadores:

Parámetro	Expresión
Representa la dirección de mayor difusión, la orientación del tejido donde las moléculas de agua se mueven con mayor libertad. En la mama esta suele ser la dirección en el eje de los ductos mamarios	L1: λ1
Representan los autovalores de los vectores ortogonales a la dirección de máxima difusión por lo que son similares entre sí y más pequeñas que L1.	L2: λ2
	L3: λ3

Tabla 2. Lambdas de la base de datos de difusión

Combinando dichos parámetros, se obtienen unos nuevos biomarcadores de mucha utilidad:

Parámetro	Sinónimo	Fórmula
Bio1: I_1	Traza	$\lambda_1 + \lambda_2 + \lambda_3$
Bio2: I_2		$\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1$
Bio3: I_3	Determinante	$\lambda_1\lambda_2\lambda_3$
Bio4: I_4	$I_1^2 + 2I_2$	$\lambda_1^2 + \lambda_2^2 + \lambda_3^2$
Bio5: D_{av}	$\frac{I_1}{3}$	$\frac{\lambda_1 + \lambda_2 + \lambda_3}{3}$
Bio6: D_{surf}	$\left(\frac{I_2}{3}\right)^{1/2}$	$\left(\frac{\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1}{3}\right)^{1/2}$
Bio7: D_{vol}	${I_3}^{1/3}$	$(\lambda_1\lambda_2\lambda_3)^{1/2}$
Bio8: D_{mag}	$\left(\frac{I_4}{3}\right)^{1/2}$	$\left(\frac{{\lambda_1}^2 + {\lambda_2}^2 + {\lambda_3}^2}{3}\right)^{1/2}$
Bio9: D_{an}:D_{an}	$6{D_{av}}^2 - 2I_2$	$(\lambda_1 - D_{av})^2 + (\lambda_2 - D_{av})^2 + (\lambda_3 - D_{av})^2$
Bio10: "K"	$\frac{I_2}{I_1}$	$\frac{\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1}{D_{av}}$
Bio11: "H"	$3\frac{I_3}{I_2}$	$\frac{3\lambda_1\lambda_2\lambda_3}{\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1}$
Bio12: Anisotropía fraccional (FA)	$\frac{(\frac{3D_{an}:D_{an}}{2D:D})^{1/2}}{D_{av}}$	$\sqrt{\frac{3[(\lambda_1 - D_{av})^2 + (\lambda_2 - D_{av})^2 + (\lambda_3 - D_{av})^2]}{2({\lambda_1}^2 + {\lambda_2}^2 + {\lambda_3}^2)}}$
Bio13: Anisotropía relativa escalada (sRA)	$\frac{(\frac{D_{an}:D_{an}}{6})^{1/2}}{D_{av}}$	$\sqrt{\frac{(\lambda_1 - D_{av})^2 + (\lambda_2 - D_{av})^2 + (\lambda_3 - D_{av})^2}{\sqrt{6}\,D_{av}}}$
Bio14: Volume ratio (VR)	$\frac{1 - I_3}{D_{av}^3}$	$1 - \frac{\lambda_1\lambda_2\lambda_3}{D_{av}^3}$
Bio15: UA_{surf}	$1 - \frac{D_{surf}}{D_{av}}$	$1 - \frac{[\frac{(\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1)}{3}]^{1/2}}{D_{av}}$
Bio16: UA_{vol}	$1 - \frac{D_{vol}}{D_{av}}$	$1 - \frac{(\lambda_1\lambda_2\lambda_3)^{1/3}}{D_{av}}$
Bio17: $UA_{vol,\ surf}$	$1 - \frac{D_{vol}}{D_{surf}}$	$1 - \frac{(\lambda_1\lambda_2\lambda_3)^{1/3}}{[\frac{(\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1)}{3}]^{1/2}}$
Bio18: Índice de latencia (LI_N)	$\frac{(FA + FA^2)}{2}$	$\frac{(\frac{\left(\frac{3D_{an}:D_{an}}{2D:D}\right)^{\frac{1}{2}}}{D_{av}} + (\frac{(\frac{3D_{an}:D_{an}}{2D:D})^{1/2}}{D_{av}}}{2}$
Bio19: Difusión transversal		$\frac{\lambda_2 + \lambda_3}{2}$
Bio20: LT ratio		$\frac{\lambda_1}{(\lambda_2 + \lambda_3)/2}$

Tabla 3. Parámetros de difusión calculados a partir de las lambdas

A la base de datos se añade la variable correspondiente al paciente en cuestión, y al tipo de tejido, es decir 1 si es tejido sano o 0 si es tejido tumoral.

	L1-L3	1-L3/L1	L1	L2	L3	Bio1	Bio2	Bio3	Bio4	Bio5	Bio6	Bio7	Bio8
0	0.000623	0.928957	0.000671	0.000398	0.000048	0.001117	3.180362e-07	1.272676e-11	6.106928e-07	0.000372	0.000326	0.000233	0.000451
1	0.001425	0.828675	0.001720	0.000488	0.000295	0.002503	1.490122e-06	2.473707e-10	3.282844e-06	0.000834	0.000705	0.000628	0.001046
2	0.001211	0.695278	0.001742	0.001304	0.000531	0.003578	3.890025e-06	1.206502e-09	5.018692e-06	0.001193	0.001139	0.001065	0.001293
3	0.002079	0.914340	0.002274	0.000897	0.000195	0.003366	2.656756e-06	3.972336e-10	6.014137e-06	0.001122	0.000941	0.000735	0.001416
4	0.001055	0.779737	0.001353	0.000634	0.000298	0.002286	1.451184e-06	2.559587e-10	2.323171e-06	0.000762	0.000696	0.000635	0.000880
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8217	0.000431	0.260884	0.001653	0.001599	0.001222	0.004473	6.615468e-06	3.228725e-09	6.780894e-06	0.001491	0.001485	0.001478	0.001503
8218	0.000679	0.402885	0.001685	0.001498	0.001006	0.004190	5.728128e-06	2.540643e-09	6.097031e-06	0.001397	0.001382	0.001365	0.001426
8219	0.001011	0.645881	0.001565	0.001275	0.000554	0.003394	3.569032e-06	1.105633e-09	4.381631e-06	0.001131	0.001091	0.001034	0.001209
8220	0.001127	0.802068	0.001406	0.000879	0.000278	0.002563	1.871204e-06	3.437531e-10	2.825804e-06	0.000854	0.000790	0.000701	0.000971
8221	5368.000000	5308.000000	5308.000000	5199.000000	4911.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figura 10. Primera parte base de datos de difusión

	Bio9	Bio10	Bio11	Bio12	Bio13	Bio14	Bio15	Bio16	Bio17	Bio18	Bio19	Bio20	Paciente	Sano
0	1.951044e-07	0.000854	0.000120	0.692258	0.484492	0.753166	0.125204	0.372710	0.282930	0.585739	0.000223	3.008175	1.0	1
1	1.195148e-06	0.001786	0.000498	0.738978	0.535010	0.573883	0.155155	0.247494	0.109298	0.642533	0.000391	4.393897	1.0	1
2	7.524441e-07	0.003262	0.000930	0.474228	0.296961	0.288556	0.045110	0.107283	0.065110	0.349560	0.000918	1.898567	1.0	1
3	2.238254e-06	0.002368	0.000449	0.747160	0.544415	0.718680	0.161184	0.344761	0.218852	0.652704	0.000546	4.167747	1.0	1
4	5.813248e-07	0.001904	0.000529	0.612653	0.408498	0.421455	0.087241	0.166743	0.087101	0.493998	0.000466	2.903005	1.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8217	1.102844e-07	0.004436	0.001464	0.156192	0.090920	0.026212	0.004142	0.008815	0.004692	0.090294	0.001410	1.172273	24.0	0
8218	2.459350e-07	0.004102	0.001331	0.245978	0.144969	0.067240	0.010564	0.022936	0.012504	0.153242	0.001252	1.346040	24.0	0
8219	5.417331e-07	0.003155	0.000929	0.430646	0.265594	0.236492	0.035915	0.086018	0.051969	0.308051	0.000915	1.710538	24.0	0
8220	6.363995e-07	0.002190	0.000551	0.581219	0.381230	0.448634	0.075520	0.180001	0.113016	0.459517	0.000579	2.429106	24.0	0
8221	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0

Figura 11. Segunda parte base de datos de difusión

Por lo que finalmente queda una base de datos formada por 8222 individuos y con 27 atributos.

Dependiendo del modelo de *machine learning* que se utilice, es necesario centrar y escalar los datos, ya que tenemos información medida en distintas unidades.

4.2 Base de datos de perfusión

Para obtener los biomarcadores de perfusión se emplea un modelo estadístico multivariante basado en variables latentes llamado Resolución de curvas multivariantes por mínimos cuadrados alternos (MCR-ALS, por su definición en inglés *multivariate curve resolution-alternating least squares*) [8], [9].

MCR es una alternativa a PCA, que no impone ortogonalidad en las direcciones de proyección. La ortogonalidad de las componentes principales es una limitación para modelar diferentes comportamientos de perfusión que no son necesariamente ortogonales. MCR puede proporcionar comportamientos fisiológicamente interpretables imponiendo un conocimiento a priori en el modelo. MCR-ALS es un método iterativo que realiza una descomposición bilineal de una matriz **S** mediante un algoritmo de optimización de mínimos cuadrados alternos.

$$\boldsymbol{S = C(D)^T + E}$$

Donde **S** contiene la intensidad de señal registrada para cada vóxel en filas; $\mathbf{D^T}$ es una matriz que contiene en sus filas cada uno de los comportamientos dinámicos de perfusión modelados (Figura 12). **C** recoge en sus filas la contribución relativa de cada comportamiento dinámico modelado en cada vóxel de la imagen (Figura 13) y **E** es una matriz residual [10], [11]. En la figura 12 se muestra un esquema sobre las dimensiones de las matrices.

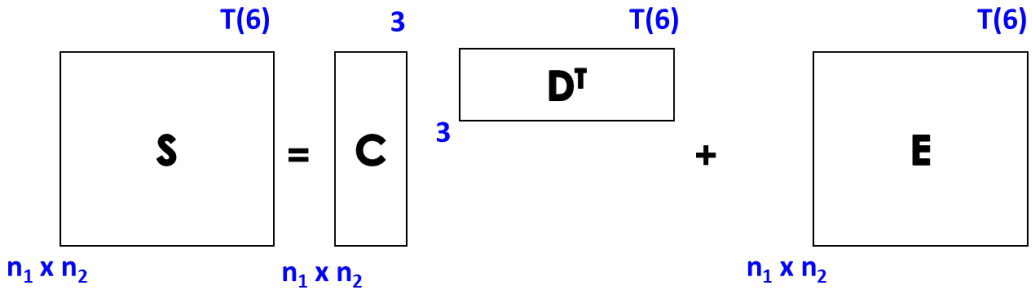


Figura 12. Esquema del modelo MCR que contiene las dimensiones de las matrices para el análisis de perfusión DCE-MRI. Los 3 componentes considerados en este modelo son los comportamientos dinámicos relacionados con el NT (tejido normal), el VT (tejido vascularizado) y el CMA (llegada del medio de contraste).

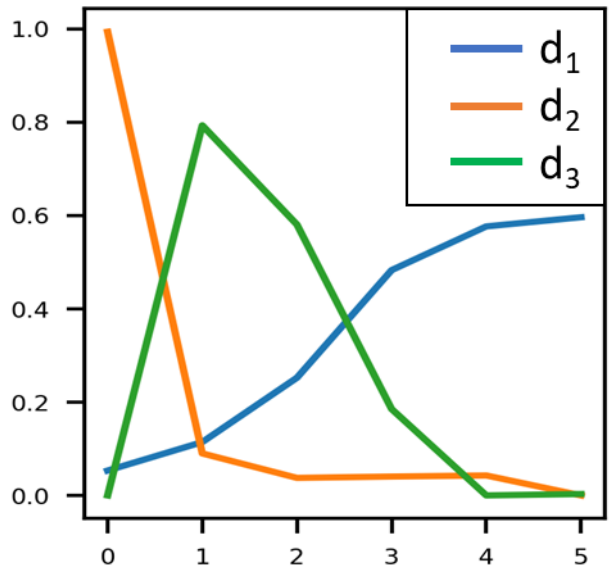


Figura 13. Comportamientos dinámicos obtenidos de MCR-ALS. d_1(NT), d_2(CMA) y d_3(VT).

El análisis también se llevará a cabo con los biomarcadores obtenidos mediante MCR aplicado a la técnica de perfusión. En este caso el número de variables es mucho menor.

Parámetro	**Expresión**
Scores de tipo NT, que se refiere a la curva de captación de la perfusión que tendría un tejido sano	C1
Scores de tipo *contrast media arrival* (CMA) que es un artefacto, un ruido estructural provocado por la llegada del contraste al tejido	C2
Scores de tipo VT, que es la curva de captación de perfusión que tendría un tejido vascularizado (signos de inicio de procesos tumorales, neovascularización y angiogénesis)	C3
Suma de cuadrados residual del modelo MCR (básicamente los residuos del modelo, aquella variabilidad que el modelo MCR no es capaz de explicar y que utilizamos como nuevo biomarcador)	RSS
Operaciones con los biomarcadores anteriores, para quitar la contribución del comportamiento artefacto a los mapas de distribución	C3-C2
	C1-C2
	(C1+C3)/2-C2

Tabla 4. Parámetros de la base de datos de perfusión

De nuevo, se añade la variable correspondiente al paciente en cuestión, y al tipo de tejido, es decir 1 si es tejido sano o 0 si es tejido tumoral.

	C1	C2	C3	RSS	C3_C2	C1_C2	C1C3_C2	Paciente	Sano
0	359.134206	202.266992	154.802201	349.082202	0.000000	156.867215	54.701212	1	1
1	362.748346	184.425396	132.843883	290.497417	0.000000	178.322950	63.370719	1	1
2	338.771623	158.453906	132.980731	21.224631	0.000000	180.317717	77.422271	1	1
3	312.529493	164.226614	157.393083	60.262141	0.000000	148.302880	70.734675	1	1
4	312.282653	177.782607	156.181376	169.098622	0.000000	134.500046	56.449407	1	1
...	...	...	...	...	...	...	...	...	...
3493	412.120807	55.294206	268.692148	455.142658	213.397942	356.826601	285.112271	24	0
3494	305.532491	103.760141	143.154142	523.246687	39.394001	201.772350	120.583175	24	0
3495	335.098320	109.224360	145.229861	1661.911072	36.005501	225.873961	130.939731	24	0
3496	374.654512	126.033480	158.113020	1483.942050	32.079541	248.621032	140.350287	24	0
3497	282.977570	119.260323	84.299174	1429.872148	0.000000	163.717246	64.378049	24	0

7576 rows × 9 columns

Figura 14. Base de datos de perfusión

5. IA en el ámbito de la salud

El aprendizaje automático (*machine learning*) es una subdisciplina de la inteligencia artificial y constituye un conjunto de técnicas utilizadas para resolver el análisis de datos mediante la búsqueda de patrones entre ellos. [1]

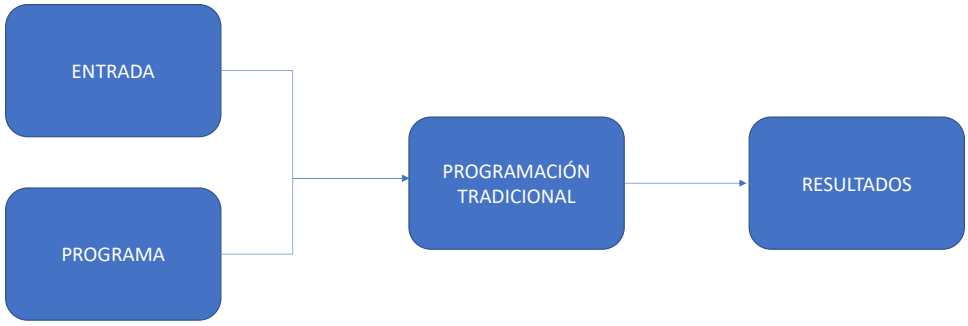


Figura 15. Funcionamiento programación tradicional

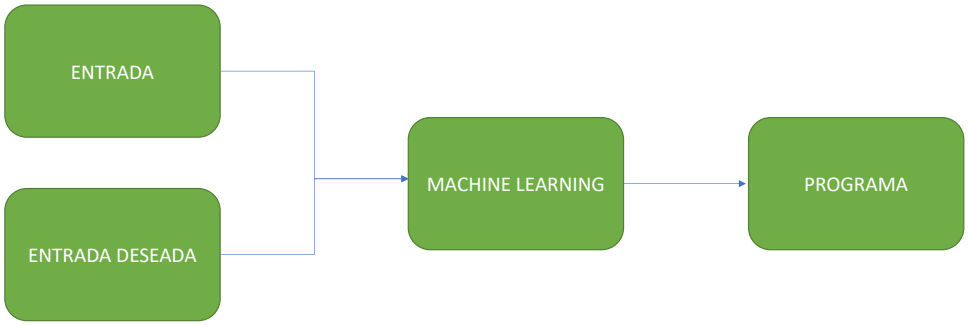


Figura 16. Funcionamiento machine learning

El objetivo de la inteligencia artificial es la elaboración modelos informáticos que muestren "comportamientos inteligentes" como los humanos, según Boris Katz. Esto significa que las máquinas pueden llevar a cabo acciones propias de los seres humanos, como comprender un texto escrito manualmente, reconocer qué hay en una imagen, o realizar una acción en el mundo real, como llevar comida a domicilio.

El *machine learning* es una forma de utilizar la inteligencia artificial. El pionero de ella, Arthur Samuel, la definió en los años 50 como "el campo de estudio que da a los ordenadores la capacidad de aprender sin ser programados explícitamente".

El profesor Mikey Shulman, especialista en inteligencia artificial para las comunidades financiera y de inteligencia de Estados Unidos, comparó la forma tradicional de programar ordenadores con

la repostería, en la que una receta exige cantidades precisas de ingredientes y le dice al panadero que mezcle durante un tiempo exacto. La programación tradicional también requiere crear instrucciones detalladas para que el ordenador las siga.

El *machine learning* consiste en dejar que los ordenadores aprendan a programarse a sí mismos a través de la experiencia. Sin datos entrenamiento y sin tiempo para ello, es imposible que un ordenador lleve a cabo este tipo de misiones, a diferencia de los humanos.

El proceso siempre comienza con datos, ya sean números, imágenes o texto. Se recopilan y se pre-procesan para ser utilizados como datos de entrenamiento, constituyendo la información con la que se entrenará el modelo de *machine learning*. Cuantos más datos, mejor será el programa.

El siguiente paso es la elección del modelo de *machine learning* y el suministro de los datos al mismo. Es entonces cuando el modelo informático comienza a entrenarse con el fin de encontrar patrones, para hacer descripciones o predicciones. Con el tiempo, el programador humano también puede ajustar el modelo, incluso cambiando sus parámetros, para ayudar a obtener resultados más precisos.

La función de un sistema de *machine learning* puede ser descriptiva, lo que significa que el sistema utiliza los datos para explicar lo que ha sucedido o predictiva, lo que significa que el sistema utiliza los datos para predecir lo que sucederá. [12]

El *machine learning* puede clasificarse en tres tipos de aprendizaje: 1) aprendizaje supervisado,2) aprendizaje no supervisado y 3) aprendizaje semisupervisado.

5.1 Aprendizaje supervisado

El aprendizaje supervisado utiliza conjuntos de datos con resultados preetiquetados para entrenar a los algoritmos en la resolución de problemas de clasificación y regresión. Emplea variables de entrada para predecir un resultado definido y es adecuado para los problemas de regresión, pero lleva tiempo porque es necesario el etiquetado manual de los datos (supervisión) y requiere una gran cantidad de datos.

Los tipos más comunes de algoritmos de aprendizaje supervisado incluyen los árboles de decisión, los clasificadores bayesianos ingenuos (Nave Bayes) y las máquinas vectoriales de soporte.
Los árboles de decisión agrupan los atributos mediante una clasificación basada en valores y se utilizan para la clasificación de los datos.

Los clasificadores bayesianos ingenuos se utilizan principalmente para clasificar datos de texto y emplean métodos basados en la probabilidad de sucesos.

Las máquinas de vectores soporte también se utilizan para la clasificación de datos y emplean el principio de cálculo de márgenes, una forma de evaluar la distancia entre clases.

El aprendizaje supervisado es adecuado para la generación de modelos predictivos mediante el establecimiento de relaciones entre los rasgos del paciente (entrada) y el resultado de interés (salida). Los clasificadores son los algoritmos de *machine learning* que se utilizan predominantemente para las aplicaciones sanitarias y se emplean, por ejemplo, para sugerir posibles diagnósticos, identificar un grupo de pacientes, diferenciar entre clases de documentos y definir los valores anormales de diversos resultados de laboratorio.

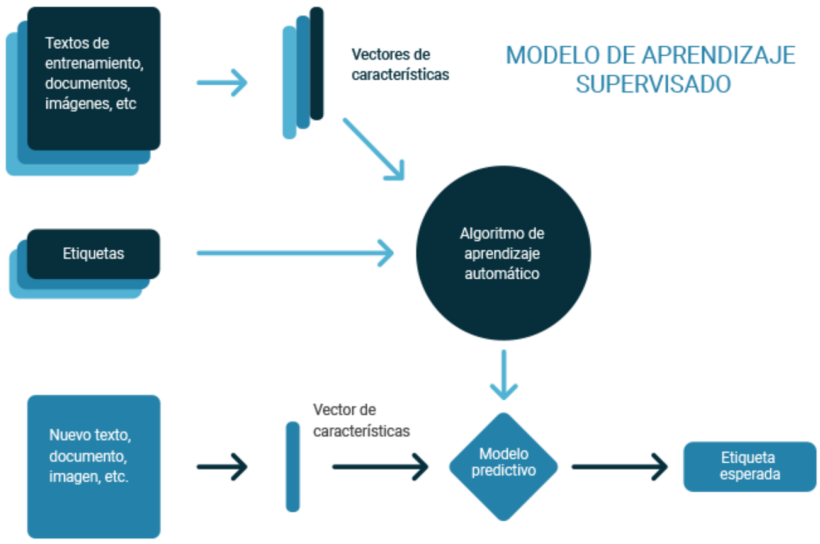


Figura 17. Modelo de aprendizaje supervisado

5.1.1 Regresión lineal

La regresión lineal es un modelo estadístico predictivo que relaciona la variable objetivo con las demás variables regresoras de forma lineal. Si se tiene una única variable de entrada, se define regresión lineal simple, si hay más, se define como regresión lineal múltiple. [13]

En la regresión lineal las relaciones de cada variable de entrada con la variable de salida se modelan de forma lineal, como su propio nombre indica. Es el modelo predictivo más simple.

Se pretende minimizar la distancia entre los datos y la función lineal modelada, como se observa en la Figura 19.

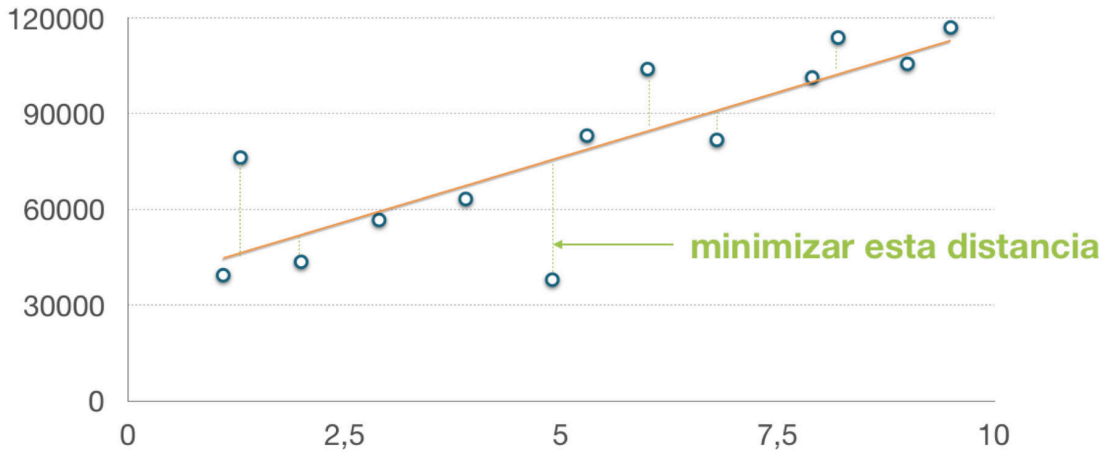


Figura 19. Ejemplo de regresión lineal

Se suele aplicar el método de los mínimos cuadrados para el modelado de esta regresión:

• Y es la variable objetivo.

• $X_1,X_2,...,X_m$ son las variables regresoras o explicativas.

• $\beta_0,\beta_1,\beta_2,...,\beta_p$ son los parámetros del modelo e indican el peso o importancia que tiene cada variable X_j en nuestra función. Particularizando para el individuo *i*-ésimo:

$$y_i = \beta 0 + \beta 1 x_i 1 + ... + \beta_p x_{im} + \varepsilon_i$$

siendo εi la perturbación o error asociada al individuo i-ésimo.

Agrupando las X_j´s y β´s en un vector, nos queda la siguiente expresión:

$$y_i = \boldsymbol{x}^T \boldsymbol{\beta} + \varepsilon_i$$

El modelo puede expresarse para todos los individuos como sigue:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$$

Donde podemos definir:

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$$

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1{}^T \\ \boldsymbol{x}_2{}^T \\ \dots \\ \boldsymbol{x}_n{}^T \end{pmatrix}$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \dots \\ \beta_p \end{pmatrix}$$

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_n \end{pmatrix}$$

Dado un conjunto de datos, este modelo asume una relación lineal entre la variable dependiente *y* y las variables explicativas X_j. Es necesario tener en cuenta el error ε, necesario para el modelado de la relación.

El modelo de regresión lineal no solo modela relaciones lineales en las variables de entrada; con transformaciones en las variables de salida (e.g. logaritmos, inversas, etc.) o de entrada (términos cuadráticos, cúbicos, inversas, logaritmos, etc) este modelo es capaz de modelar relaciones no lineales entre la variable de salida y las de entrada. Sin embargo, no todas las relaciones no-lineales pueden ser modeladas por este tipo de modelos; en esos casos es necesario recurrir a otras técnicas predictivas.

5.1.2 Regresión logística

La regresión logística es un modelo estadístico para estudiar las relaciones entre un conjunto de variables explicativas X*j* y una variable cualitativa dicotómica Y.

Se trata de un modelo lineal generalizado que utiliza una función logística como función de enlace y permite predecir la probabilidad de que ocurra un evento (valor de 1) o no (valor de 0) a partir de la optimización de los coeficientes de regresión. Este resultado siempre varía entre 0 y 1.

Cuando el valor predicho supera un umbral, se considera que es probable que ocurra el evento, mientras que cuando ese valor está por debajo del mismo umbral, no es así. [14]

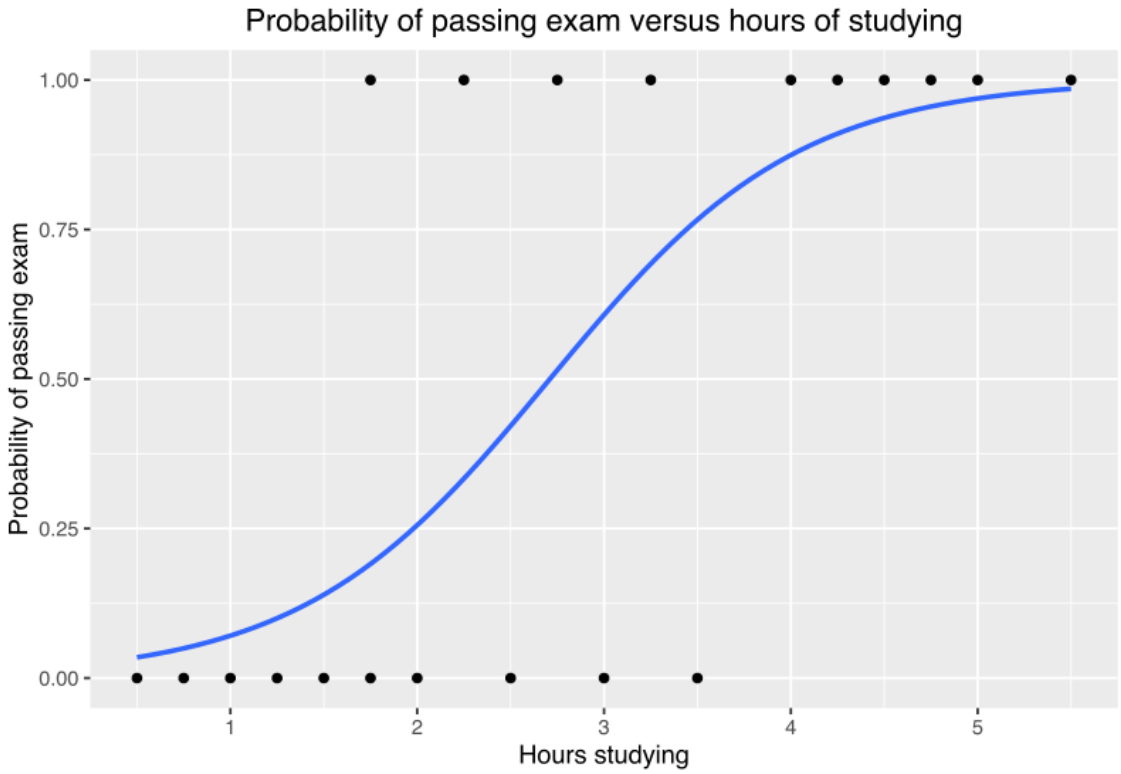


Figura 20. Ejemplo regresión logística. Horas estudiando (eje X) vs probabilidad de aprobar el examen (eje Y).

En la Figura 20 se puede observar un ejemplo de regresión logística, en este caso se quiere modelar la probabilidad de aprobar un examen teniendo en cuenta las horas de estudio empleadas. Se puede apreciar una función sigmoidea, la función logística estándar se modela de la siguiente forma, siendo la entrada el tiempo empleado en horas (h).

$$S(h) = \frac{e^h}{e^h + 1} = \frac{1}{1 + e^{-h}}$$

Asumiendo a continuación que *h* es una función lineal definida por la variable x,

$$h = \beta_0 + \beta_1 x$$

Por lo que se obtiene que:

$$p(x) = S(h) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Siendo *p*(x) la probabilidad de que ocurra el evento. Considerando un umbral, se podría contestar a la pregunta "¿Aprobará el examen habiendo estudiado h horas?", obteniendo una respuesta de Sí o No.

5.1.3 Regresión PLS

La regresión en mínimos cuadrados parciales, PLS (por su definición en inglés *Partial Least Squares*) permite encontrar las distintas relaciones entre la variable objetivo, y el resto de las variables regresoras. Se define como **X** a la matriz regresora, e **Y** a la matriz de respuestas.

En este caso se buscan una serie de coeficientes de regresión que maximicen la correlación entre las variables regresoras y la objetivo.

Se asemeja al método de aprendizaje no supervisado PCA, que se explicará posteriormente, ya que se descompone la matriz de variables original en variables latentes que explican las relaciones internas.

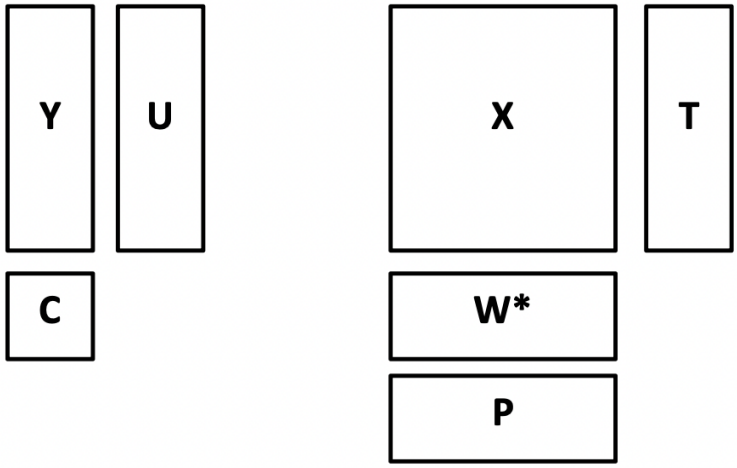


Figura 21. Esquema modelo PLS

Como se observa en la figura 21, la matriz **X** se descompone en 3 matrices. La matriz **T**, q contiene los vectores de scores, y se calculan de forma que se maximice la covarianza con los scores **u** de la matriz **Y**. Los pesos **w*** son las combinaciones lineales de las variables de la matriz **X** que dan como resultado los scores **t** y dan información sobre la relación de las variables de **X** con los scores U. Los loadings **p** se calculan de forma que son las direcciones que mejor modelan la variabilidad de la matriz X.

La matriz **Y** también se descompone en dos matrices, una de scores llamada matriz **U**, que está relacionada con los scores **t** de la matriz **X**, y una matriz de pesos **C**, que da información sobre la relación de las variables **Y** con los scores **t**.

5.1.4 Máquinas de vectores soporte

El algoritmo SVM (por su nombre en inglés *support vector machine*) es uno de los métodos de predicción más robustos.

El algoritmo se encarga de construir un hiperplano o un conjunto de hiperplanos en un espacio con una dimensión elevada y puede usarse tanto para clasificación o regresión.

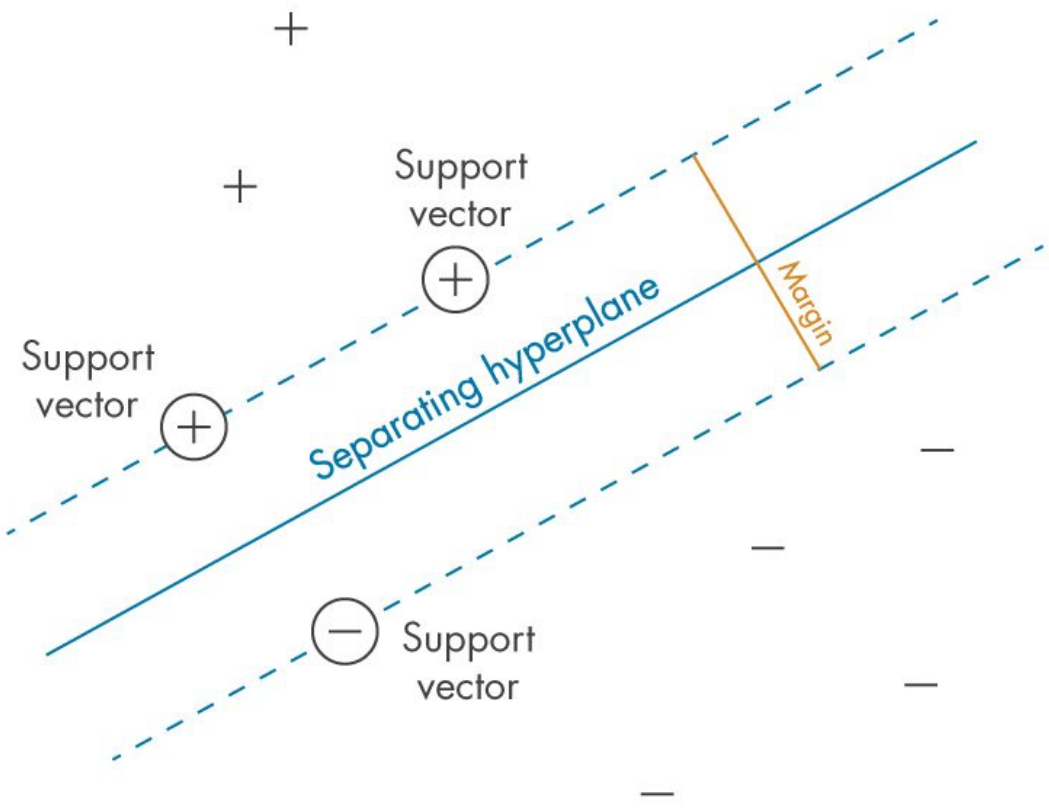


Figura 22. Idea SVM

El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. Esto implica el hiperplano con el margen más amplio entre las dos clases, representado por los signos más y menos en la Figura 21. El margen se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores. En la mayoría de los problemas prácticos, el algoritmo maximiza el margen flexible permitiendo un pequeño número de clasificaciones erróneas.

Los vectores de soporte hacen referencia a un subconjunto de las observaciones de entrenamiento que identifican la ubicación del hiperplano de separación. El algoritmo SVM estándar está formulado para problemas de clasificación binaria; los problemas multiclase normalmente se reducen a una serie de problemas binarios.

En la imagen de la Figura 22, los puntos están representados en p=2 dimensiones, entonces se busca un plano que esté en p−1 dimensión lo que sería una recta (ya que los individuos están representados en dos dimensiones), que separe los conjuntos de datos. [15]

Si la separación se pueda establecer de forma lineal, cualquier punto x que esté en el hiperplano separador, satisface la siguiente ecuación:

$$\boldsymbol{x^T w} + b = 0$$

Siendo:

- w un vector perpendicular al hiperplano separador.
- b se define como sesgo (*bias*) para posicionarlo respecto al origen
- x cualquier punto del hiperplano separador.

Definiendo los puntos + como $y_i = 1$ y los puntos de color - como $y_i = -1$, en la figura 22, se tiene que:

$$\boldsymbol{x_i^T w + b \geq +a, y_i = +1}$$

$$\boldsymbol{x_i^T w + b \leq +a, y_i = -1}$$

$$\boldsymbol{y_i\left(x_i^T w + b\right) \leq a}$$

Por lo que ahora se puede determinar el margen separador (en amarillo en la figura), de la siguiente forma:

$$\boldsymbol{M = d_+ + d_- = \frac{2a}{||w||}}$$

Se fija a = 1 para maximizar dicho el hiperplano separador, y que sea máximo es el objetivo.

5.1.5 Métodos basados en árboles

Los árboles de decisión son modelos predictivos formados por reglas binarias (si/no) con las que se consigue repartir las observaciones en función de sus atributos y predecir así el valor de la variable respuesta.

La idea del funcionamiento de un árbol de decisión se puede ver en la Figura 23, donde podemos buscar la respuesta a "¿Podremos jugar al fútbol esta tarde según la meteorología?"

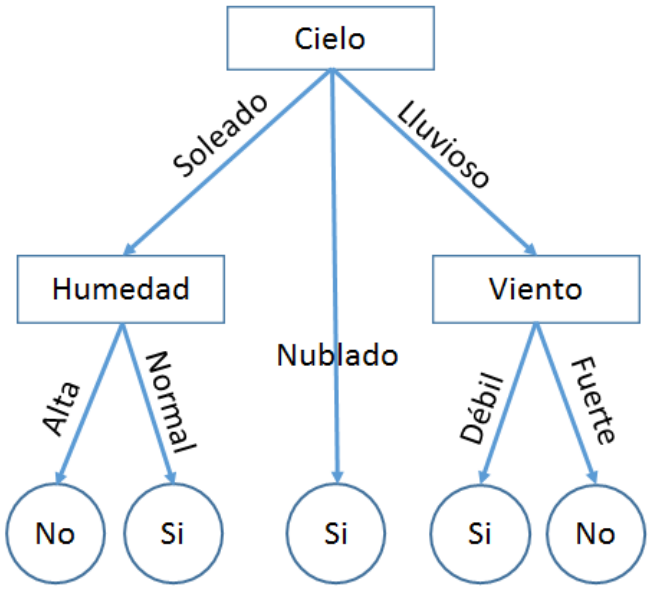


Figura 23. Ejemplo árbol de decisión

Cuando las variables predictoras interaccionan entre ellas de forma compleja, de tal manera que una función lineal no es capaz de interpretar correctamente el problema, es complicado encontrar un modelo que sea capaz de reflejar la relación de dichas variables.

Los métodos estadísticos y de *machine learning* basados en árboles son muy potentes cuando esta situación se presenta, ya que estos métodos engloban a un conjunto de técnicas supervisadas no paramétricas que consiguen segmentar el espacio de los predictores en regiones simples, dentro de las cuales es más sencillo manejar las interacciones.

Un modelo *Random Forest* está formado por un conjunto (*ensemble*) de árboles de decisión individuales, cada uno entrenado con una muestra aleatoria extraída de los datos de entrenamiento originales. Esto implica que cada árbol se entrena con unos datos ligeramente distintos. En cada árbol individual, las observaciones se van distribuyendo por bifurcaciones (nodos) generando la estructura del árbol hasta alcanzar un nodo terminal. La predicción de una nueva observación se obtiene teniendo en cuenta las predicciones de todos los árboles individuales que conforman el *Random Forest*.

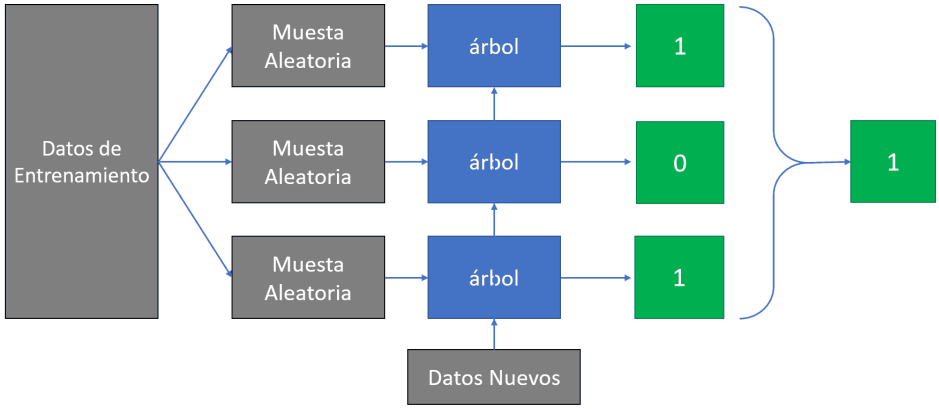


Figura 24. Funcionamiento Random Forest

Boosting es otra estrategia de *ensemble* que se puede emplear en estos casos. La idea del *boosting* es ajustar, de forma secuencial, múltiples modelos sencillos que predicen solo ligeramente mejor que lo esperado por azar. Cada nuevo modelo emplea información del modelo anterior para aprender de sus errores, mejorando iteración a iteración. A diferencia del método de *random forest*, el *boosting* no hace uso de muestreo repetido, los árboles que forman el modelo se distinguen por la importancia (peso) de las observaciones, que es distinta en cada iteración.

Entre los muchos algoritmos de boosting, los más empleados son *AdaBoost, Gradient Boosting* y *Stochastic Gradient Boosting*. Todos ellos se caracterizan por tener una cantidad considerable de hiperparámetros. [16]

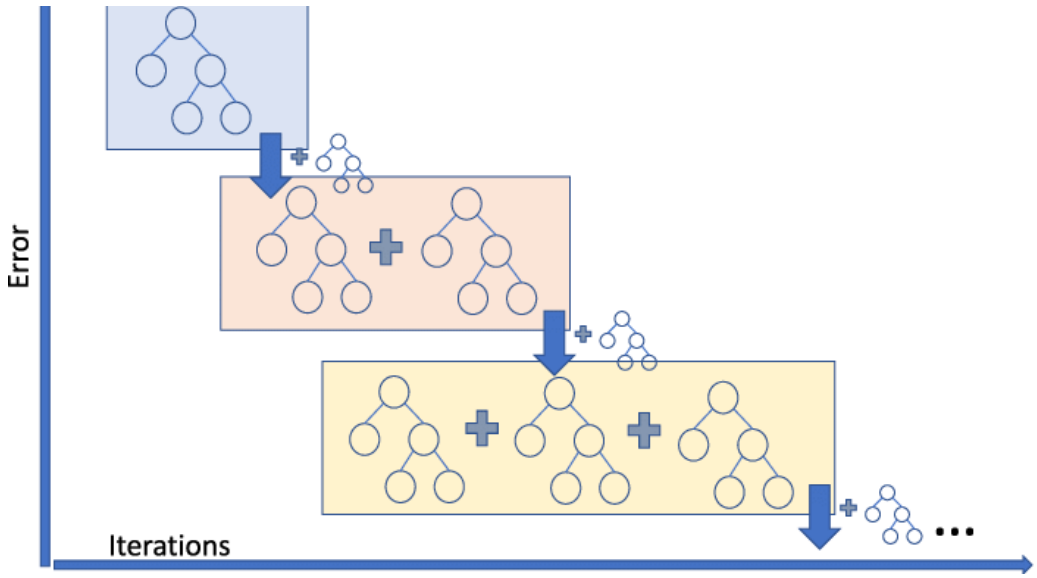


Figura 25. Funcionamiento de los algoritmos Boosting

5.1.6 KNN

El algoritmo de los K-Vecinos KNN (por su definición en inglés K-*Nearest-Neighbor*), puede utilizarse tanto para clasificación como para regresión.

"*K*" significa el número de individuos "más cercanos" que el algoritmo tiene en cuenta para clasificar un nuevo individuo, y este método se basa en dichos individuos para clasificar a uno nuevo.

Es un algoritmo basado en instancia, es decir, memoriza las instancias de entrenamiento, por ello requiere mucha memoria, y por tanto funciona mejor con bases de datos no muy grandes.

Funciona de la siguiente forma:

1. Se calcula la distancia entre el nuevo individuo y los demás (pertenecientes a los datos de entrenamiento)

2. Mediante el uso de la distancia euclídea se puede determinar qué individuos se encuentran más cercanos al nuevo individuo. Una vez lo tenemos determinado, es el momento de escoger el valor de *K*.

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

3. Se clasifica al nuevo individuo dependiendo de la clase a la que pertenezcan la mayoría de los K vecinos.

Es un crucial definir apropiadamente el valor de *K*. Tomar los valores impares para que no haya empate es algo usual. Hay que buscar un balance ya que cuantos más puntos, el coste computacional será mayor

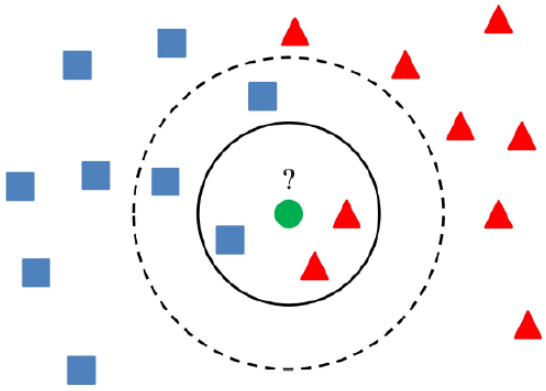


Figura 26. Ejemplo KNN

En el caso de la Figura 26, si tomamos *K* = 3, clasificaríamos el punto verde como rojo. En cambio, si tomamos *K* = 5, lo clasificaríamos como azul, al haber una mayoría de individuos de esa clase.

5.2 Aprendizaje no supervisado

El aprendizaje no supervisado utiliza conjuntos de datos con entradas no etiquetadas y permite que los algoritmos extraigan patrones, características y estructuras de los datos de forma autónoma. Utiliza las características aprendidas previamente para reconocer los datos que se le presentan.

Este método se utiliza para detectar patrones ocultos en los datos, y su objetivo es encontrar esos patrones sin la retroalimentación humana. El aprendizaje no supervisado es útil para la extracción y la agrupación de características.

Los algoritmos comunes que se utilizan incluyen la agrupación de K-medias (K-*Means Clustering*) y el análisis de componentes principales (PCA, *principal component analysis*).

La agrupación de K-medias implica la agrupación automática de datos en K conglomerados o grupos (*clusters*) distintos, basados en características compartidas. El algoritmo suele estar provisto de un conjunto determinado de atributos para cada elemento y para el número de conglomerados que se van a utilizar. A continuación, divide los elementos en combinaciones de atributos que se ajustan con mayor precisión al número de grupos.

El PCA es una técnica muy útil para la detección de observaciones anómalas (*outliers*) y para la extracción de características (*features*) de conjuntos de datos, posibilitando una reducción en el número de variables a usar y agilizando los cálculos. Utiliza proyecciones ortogonales para comprimir las variables originales que están correlacionadas en nuevas variables latentes (*features*) incorrelacionadas.

Algunas de las aplicaciones más comunes de los métodos de agrupación en la asistencia sanitaria incluyen la gestión de riesgos para los resultados clínicos y los reembolsos de los seguros de salud, la identificación de pacientes similares con enfermedades complejas o raras, o la gestión de la salud de la población mediante la agrupación de pacientes. [1]

Figura 27. Tipos de aprendizaje no supervisado

5.2.1 Clustering

Se entiende por *clustering* la tarea de dividir la población o los puntos de datos en una serie de grupos, de manera que los puntos de datos de los mismos grupos sean más similares a otros puntos de datos del mismo grupo y menos similares a los puntos de datos de otros grupos. Básicamente, se trata de una agrupación de objetos sobre la base de la similitud y la disimilitud entre ellos.

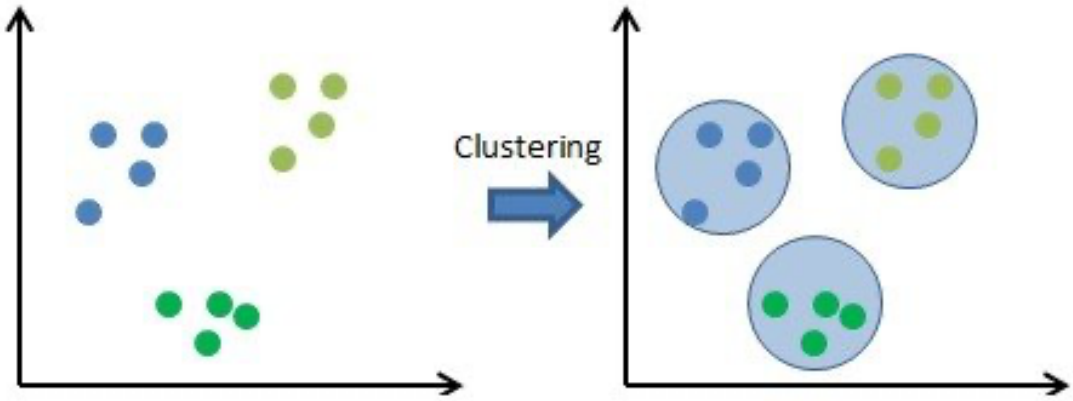


Figura 28. Idea del clustering

El *clustering* es muy útil ya que determina la agrupación intrínseca entre los datos no etiquetados presentes.

No hay un criterio óptimo único para una buena agrupación; esto depende del objetivo del estudio y de la naturaleza de los datos. Por ejemplo, podríamos estar interesados en encontrar representantes de grupos homogéneos (reducción de datos), en encontrar *clusters* "naturales" y describir sus propiedades desconocidas (tipos de datos "naturales"), en encontrar agrupaciones útiles y adecuadas (clases de datos "útiles") o en encontrar objetos de datos inusuales (detección de valores atípicos). [17]

Este algoritmo puede usar distintas medidas de distancia o disimilaridad, dando lugar en cada caso a *clusters* diferentes.

Hay varios métodos para llevar a cabo una técnica de *clustering*

5.2.1.1 Métodos basados en la densidad:

Estos métodos consideran los *clusters* como la región densa que tiene algunas similitudes y diferencias con la región densa inferior del espacio. Estos métodos tienen una buena precisión y la capacidad de fusionar dos *clusters*. Por ejemplo, DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), OPTICS (*Ordering Points to Identify Clustering Structure*), etc.

5.2.1.2 Métodos basados en la jerarquía:

Los *clusters* formados en este método forman una estructura de tipo árbol basada en la jerarquía. Los nuevos *clusters* se forman a partir de los previamente formados. Se divide en dos categorías

- Aglomerativo (enfoque ascendente)
- Divisivo (enfoque descendente)

Algunos ejemplos serían CURE (*Clustering Using Representatives*), BIRCH (*Balanced Iterative Reducing Clustering and using Hierarchies*), etc.

5.2.1.3 Métodos de partición:

Estos métodos dividen los objetos en *K clusters* y cada partición forma un *cluster*. Este método se utiliza para optimizar una función de similitud de criterio objetivo como cuando la distancia es un parámetro importante. Por ejemplo K-means, CLARANS (*Clustering Large Applications based upon Randomized Search),* etc.

5.2.1.4 Métodos basados en cuadrículas:

En este método, el espacio de datos se formula en un número finito de celdas que forman una estructura tipo rejilla. Todas las operaciones de *clustering* realizadas en estas cuadrículas son rápidas e independientes del número de objetos de datos. Por ejemplo, STING (*Statistical Information Grid*), *Wave Cluster*, CLIQUE (*Clustering In Quest*), etc.

5.2.2 PCA. Reducción de dimensionalidad

El análisis de componentes principales (PCA por sus siglas en inglés *Principal Component Analysis*) es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información más relevante.

El método consiste en descomponer una base de datos original, de *N* observaciones y *M* variables en una serie de variables latentes denominadas componentes principales, las cuales se obtienen mediante combinación lineal de las originales.

Estas componentes tienen dos vectores característicos, el de scores **t** y el de loadings **p**. Los scores corresponden a las proyecciones de cada observación. Sobre las direcciones representadas por los loadings. Los loadings representan las direcciones de máxima variabilidad y son perpendiculares entre sí, pudiendo extraer como máximo tantas componentes como el rango de la matriz **X**. Esta estructura se visualiza en la Figura 29. [18]

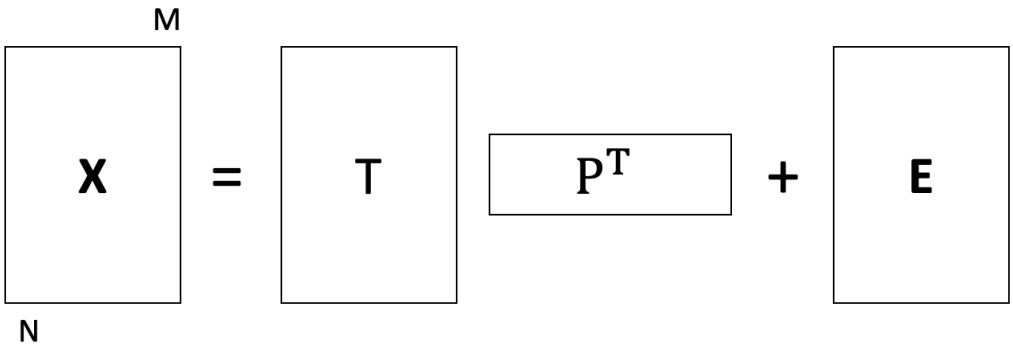


Figura 29. Esquema PCA

Cada componente explica una proporción de la variabilidad total de los datos, siendo la primera componente principal la que más explica, seguido de la segunda, y así sucesivamente

El método de análisis de componentes principales permite “condensar” la información aportada por múltiples variables en solo unas pocas componentes. Gracias a esa capacidad, se considera útil de aplicar tanto en el propio análisis como en el análisis exploratorio previo, ya que aporta mucha información sobre la posible presencia de datos anómalos así como sobre la estructura de relación entre individuos y entre variables. [19]

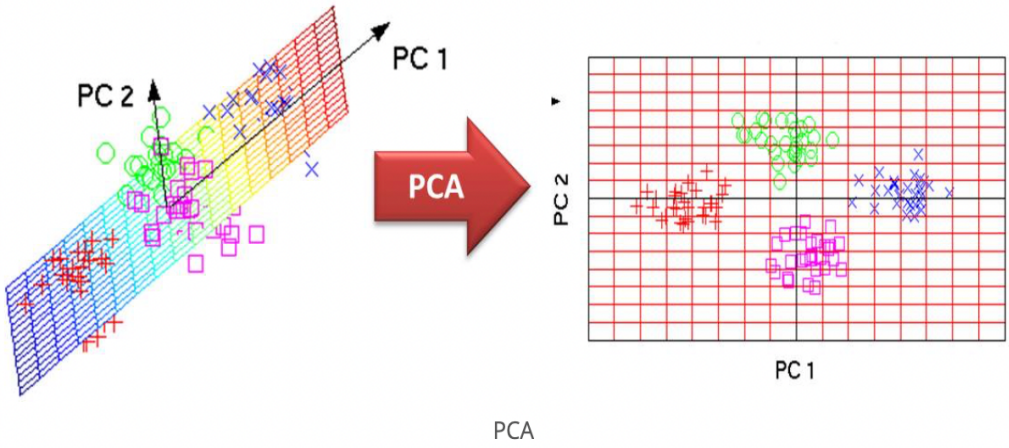


Figura 30. Idea PCA

5.3 Aprendizaje semisupervisado

El aprendizaje semisupervisado se puede considerar como una mezcla entre el aprendizaje supervisado y no supervisado, ya que solo se etiqueta un subconjunto de la salida de datos, ya que requiere una investigación intensiva para obtener los datos totalmente etiquetados. Puede ser útil cuando ya se dispone de datos sin etiquetar y es tedioso etiquetarlos.

Los métodos de aprendizaje semisupervisado incluyen los modelos generativos, el autoentrenamiento y la máquina vectorial de soporte transductivo.

Los modelos generativos utilizan distribuciones mixtas, como los modelos de mezcla Gaussiana, como identificadores para la predicción.

El autoentrenamiento es un modelo en el que se entrena primero a los clasificadores con datos etiquetados y posteriormente se introducen datos no etiquetados; las predicciones se añaden al conjunto de entrenamiento.[1]

5.4 Redes neuronales artificiales

Las RNA (Redes Neuronales Artificiales) son modelos basados en el funcionamiento del cerebro y de las neuronas biológicas, y se utilizan para resolver problemas computacionales.

Las redes están formadas por 3 tipos de capas diferenciadas, una capa de entrada, una o más capas ocultas y una capa de salida. En las capas encontramos nodos, y estos se conectan a otros con un peso y un umbral asociado. [21]

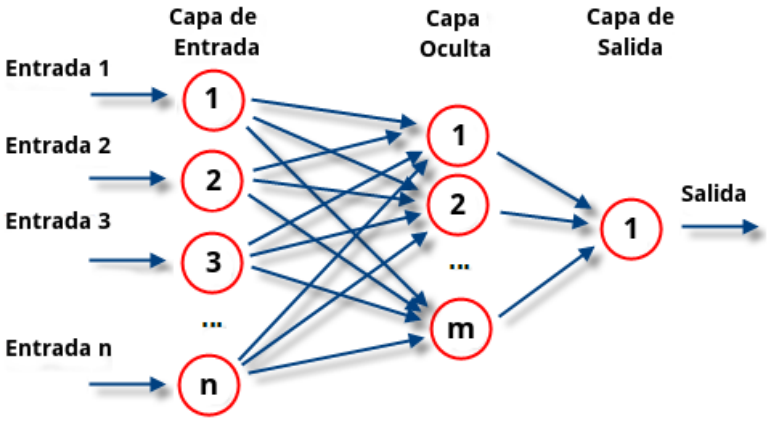


Figura 31. Partes de una red neuronal artificial

Se puede imaginar una red neuronal como una extensión de la regresión lineal para detectar relaciones complejas no lineales entre las variables de entrada y la variable de salida. Las RNA se pueden entrenar utilizando tanto el aprendizaje supervisado como el no supervisado en el proceso de optimizar la exactitud de sus predicciones.

El objetivo principal de este modelo es aprender modificándose automáticamente a sí mismo de forma que puede llegar a realizar tareas complejas que no podrían ser realizadas mediante la clásica programación basada en reglas. De esta forma se pueden automatizar funciones que en un principio solo podrían ser realizadas por personas.

Para comprender su funcionamiento hay que imaginarse cada nodo individual como un modelo de regresión lineal, compuesto por datos de entrada, pesos, un sesgo y una salida. La fórmula de un nodo con m datos de entrada sería:

$$\sum_{i=1}^{m} w_i x_i + bias$$

$$output = f(x) = \begin{cases} 1 \; if \; \sum_{i}^{m} w_i x_i + bias \geq 0 \\ 0 \;\; if \; \sum_{i}^{m} w_i x_i + bias < 0 \end{cases}$$

Una vez que se determina una capa de entrada, se asignan ponderaciones que ayudan a determinar la "importancia" de cualquier variable, de tal manera que aquellas con unas ponderaciones mayores contribuirán más significativamente a la salida en comparación con otras entradas. Todas las entradas se multiplican por sus respectivos pesos y se suman. Después, la salida pasa por una función de activación, que determina la salida. Según la función de activación definida, si esa salida supera un umbral determinado, se activa el nodo, pasando los datos a la siguiente capa de la red. Este proceso de paso de datos de una capa a la siguiente define a esta red neuronal como una red *feedforward* (hacia delante).

Al entrenar un modelo, es necesario evaluar su precisión utilizando una función de coste. Esta función puede ser el error medio cuadrático medio MSE (por su definición en inglés *mean squared error*), cuya fórmula es la siguiente:

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (y' - y)^2$$

Siendo *m* el número de individuos, *y* el valor real e *y'* el valor predicho.

Por lo que el objetivo ahora es minimizar nuestra función de coste para garantizar la corrección del ajuste para cualquier observación dada. A medida que el modelo ajusta sus pesos y su sesgo, utiliza la función de coste y el aprendizaje por refuerzo para alcanzar el punto de convergencia, o el mínimo local. El proceso por el que el algoritmo ajusta sus pesos es a través del descenso de gradiente, lo que permite al modelo determinar la dirección que debe tomar para reducir los errores. Con cada ejemplo de entrenamiento, los parámetros del modelo se ajustan para converger gradualmente en el mínimo.

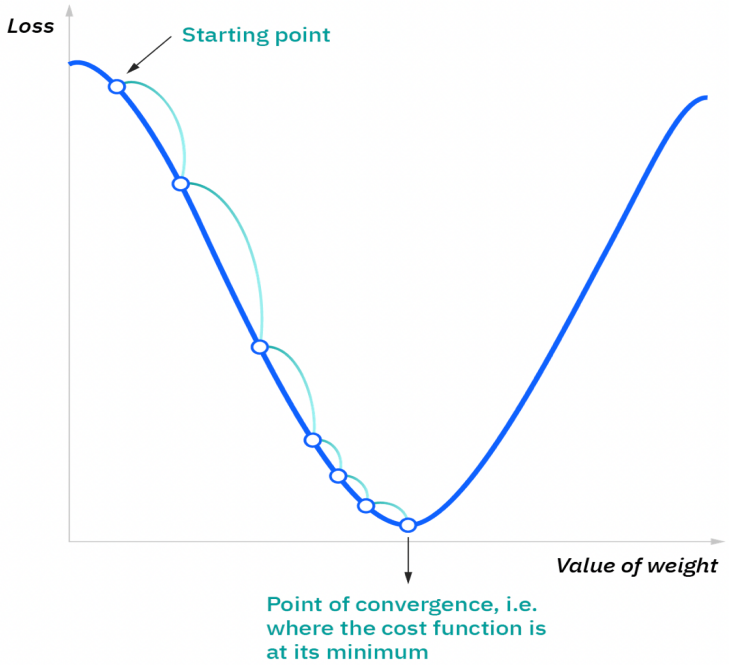


Figura 32. Convergencia de la función de coste. Valor del peso en el eje X y el error en el eje Y. El punto de convergencia es cuando la función de coste está en su mínimo.

Se puede entrenar el modelo a través de la *backpropagation* (retropropragación), es decir, moverse de la salida a la entrada, lo que nos permite calcular y atribuir el error asociado a cada neurona, para ajustar y adaptar los parámetros del modelo o modelos de forma adecuada. Con este método se consigue que la red "aprenda", consiguiendo un modelo capaz de obtener resultados muy acertados.

Las redes neuronales se pueden clasificar en diferentes tipos, que se utilizan para distintos propósitos.

El algoritmo perceptrón fue creado por Frank Rosenblatt en 1958. Tiene una sola neurona y es la forma más simple de una red neuronal:

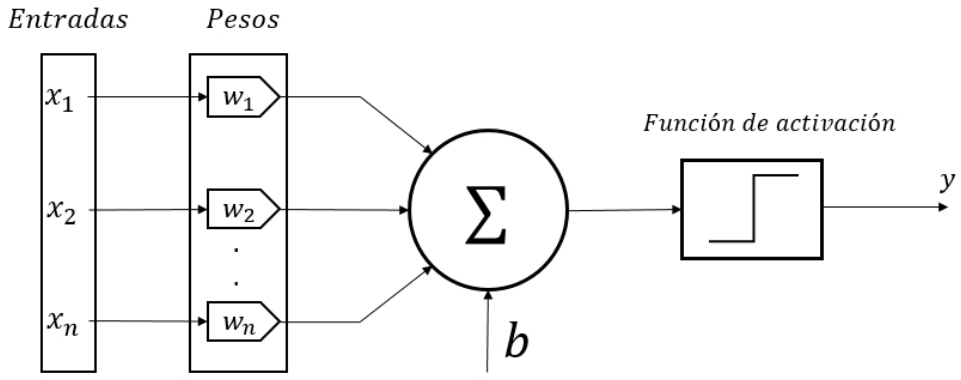


Figura 33. Estructura perceptrón

Los perceptrones multicapa (MLP por sus siglas en inglés, *Multi Layer Perceptron*) se componen de una capa de entrada, una o varias capas ocultas y una capa de salida. Se podría decir que es el acoplamiento de muchos perceptrones para modelar problemas no lineales.

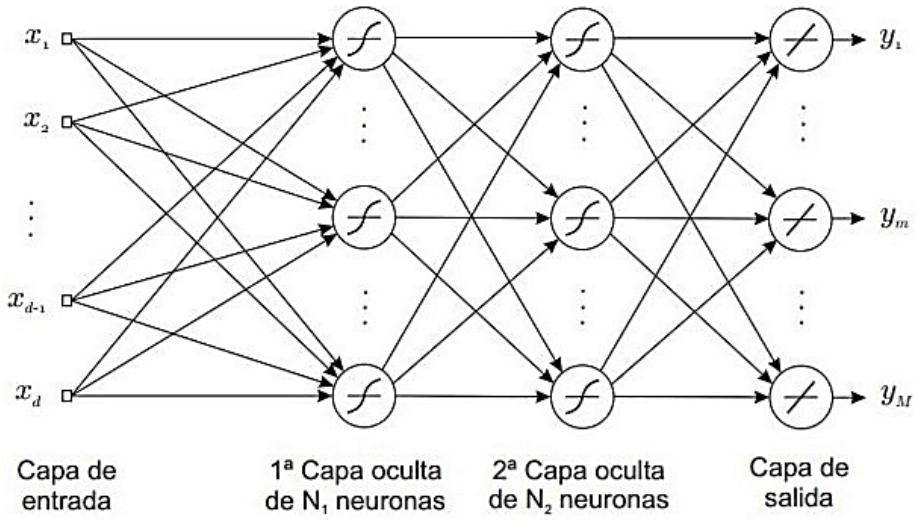


Figura 34. Arquitectura perceptrón multicapa

Las redes neuronales convolucionales (CNN por sus siglas en inglés, *Convolutional Neural Networks*) suelen utilizarse para el reconocimiento de imágenes o la visión por ordenador. Estas redes aprovechan los principios del álgebra lineal para identificar patrones dentro de una imagen.

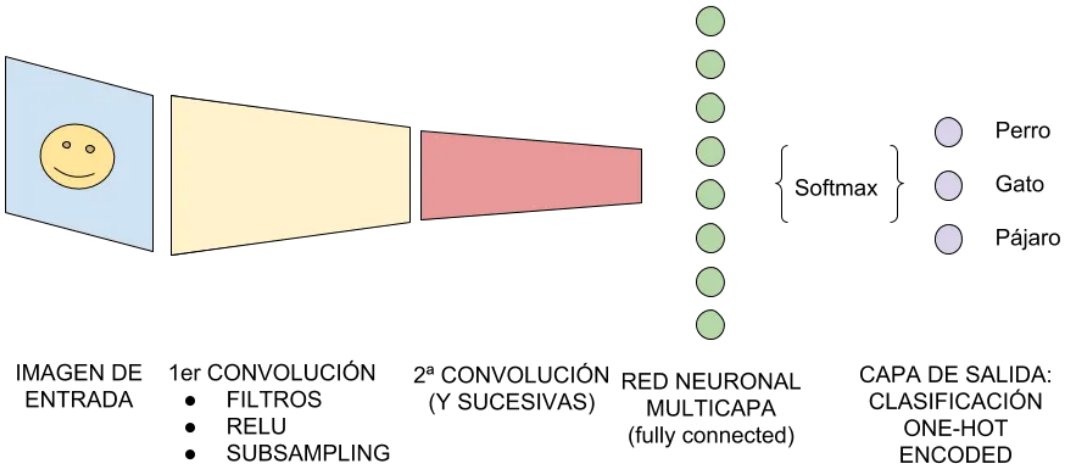


Figura 35. Arquitectura red neuronal convolucional

Las redes neuronales recurrentes (RNN por sus siglas en inglés *Recurrent Neural Networks*) se identifican por sus bucles de retroalimentación. Estos algoritmos de aprendizaje se aprovechan principalmente cuando se utilizan datos de series temporales para hacer predicciones sobre

resultados futuros, como la previsión de ventas o en nuestro caso, la previsión de una enfermedad. [22]

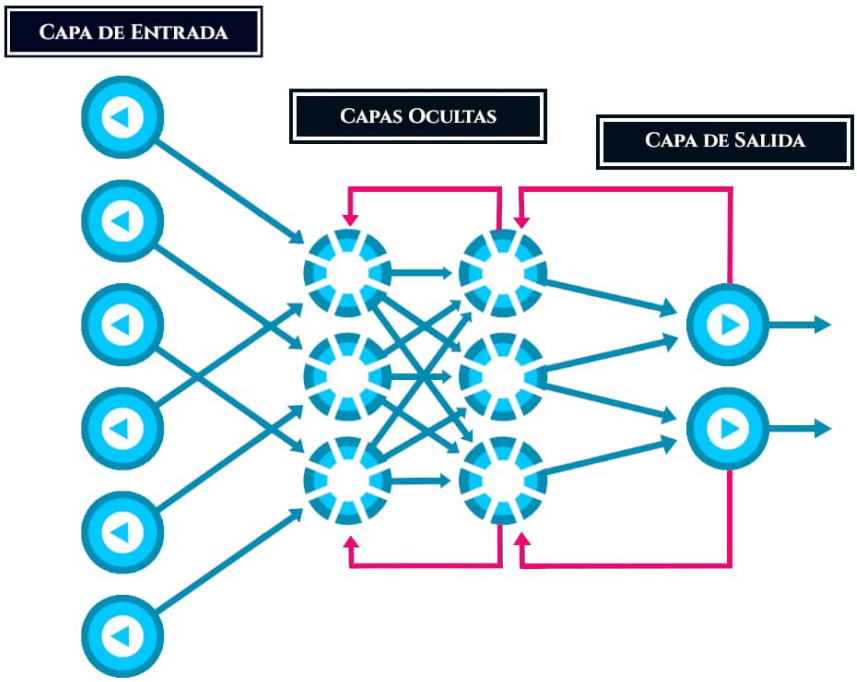


Figura 36. Estructura redes neuronales recurrentes

5.5 Aprendizaje profundo

El aprendizaje profundo es una técnica avanzada de *machine learning* y se basa principalmente en una mejora de técnica de redes neuronales.

Es como una red neuronal, pero con muchas capas de abstracción en lugar de una relación directa de entrada a salida. Los avances en la potencia de cálculo, la disponibilidad de más y mayores conjuntos de datos, y la introducción de nuevos formatos de datos han hecho posible esto. [1]

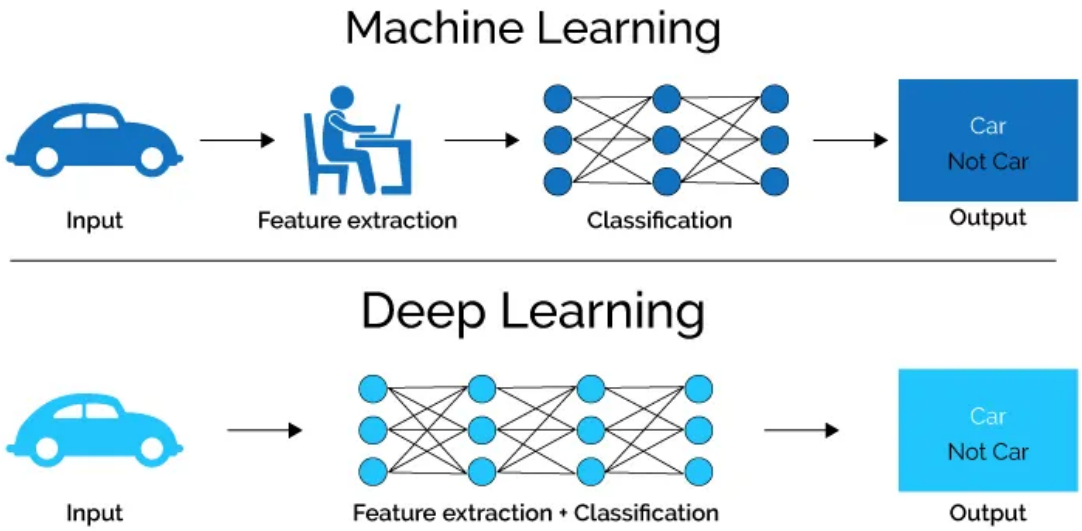


Figura 37. Diferencia machine learning y deep learning

Se puede optimizar la exploración de patrones no lineales complejos mediante esta técnica. Este tipo de aprendizaje se suele caracterizar por la existencia de numerosas capas ocultas que permiten el manejo de datos muy complejos y con estructuras diferentes.

Los algoritmos utilizados son especiales porque pueden generar autónomamente las características de interés en los datos de entrada una vez que se han introducido suficientes observaciones de entrenamiento.

El aprendizaje profundo es particularmente adecuado para los datos secuenciales y no estructurados, como las imágenes, el audio y el vídeo, donde puede descubrir la estructura intrincada en grandes conjuntos de datos.

Desempeña un papel importante en el reconocimiento de imágenes (diagnósticos médicos, reconocimiento facial, etc.), el reconocimiento del habla (asistentes de voz), las aplicaciones móviles, la visión artificial y los robots. Los algoritmos utilizados habitualmente para la asistencia sanitaria incluyen redes neuronales convolucionales (RNC), redes neuronales recurrentes (RNR) y aprendizaje por refuerzo profundo. [1]

6. Fases del proyecto

A la hora de realizar un proyecto de *machine learning*, es conveniente seguir un proceso, en el que están involucradas distintas fases con distintas prioridades.

En el siguiente diagrama se pueden visualizar dichas fases, y el esfuerzo que suele requerir cada una. [23]

Cada caso individual estará sujeto a sus propias particularidades, pero a nivel general se sigue un esquema como el de la Figura 38:

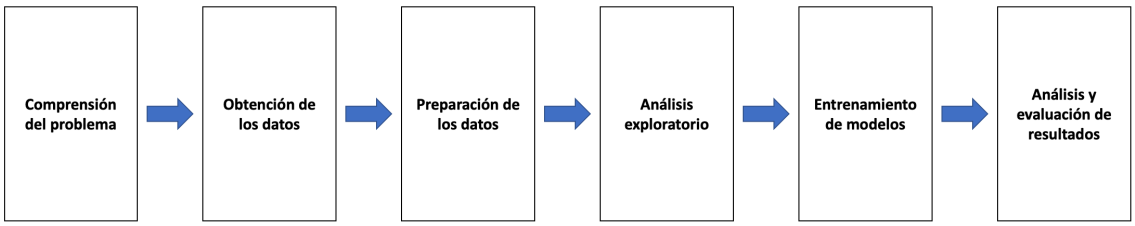


Figura 38. Fases de un proyecto de machine learning

6.1 Comprensión del problema

Es importante informarse bien de lo que trata el problema a abordar, dedicarle todo el tiempo necesario hasta el punto de la total comprensión del problema, y, por ende, de los datos a tratar.

Esta parte consiste en la comprensión del marco teórico en el que se desarrolla el problema. Esto es algo fundamental para la redacción de objetivos y la obtención de unas conclusiones coherentes. [24]

En nuestro caso se ha definido en apartados anteriores el marco teórico cuando se trata el estado del arte de la inteligencia artificial en el ámbito de la salud, así como cuando se explica el cáncer de mama, la resonancia magnética y la obtención de los biomarcadores de imagen.

Gracias a ello, hemos llegado al punto de conseguir entender el problema y poder abordarlo apropiadamente.

6.2 Obtención de los datos

En este tipo de proyectos, la materia prima son los datos, el objetivo es exprimir los datos para extraer de ellos la mayor información posible, los datos pueden proporcionar mucho, pero sin ellos no hay nada.

Para que un algoritmo de *machine learning* funcione debemos proporcionarle una gran cantidad de datos. A pesar de ser más importante la calidad que la cantidad, lo idóneo es plantear un equilibrio, es decir, cuantos más y mejores datos tengamos para empezar, mejor será el rendimiento del modelo. [25]

Como vemos en la Figura 1, hay muchas maneras de obtener los datos, ya sea de un repositorio de bases de datos, mediante *data scrapping* (método muy utilizado para recopilar datos de fuentes diversas, como por ejemplo una página web, un blog o una hoja de cálculo), o como en nuestro caso, los datos se han obtenido tras el análisis de las imágenes de 24 casos reales de RMN de mama donde se tiene certeza de la localización del tumor, constatados por radiólogos expertos y biopsiados por expertos en la materia. Estos datos han sido proporcionados por el grupo de Ingeniería Estadística Multivariante (GIEM-UPV) fruto de su colaboración científica con centros clínicos de renombre a nivel nacional. Todos los datos derivados de los casos de RMN han sido correctamente anonimizados asegurando la privacidad de los pacientes.

Es importante también plantearse donde almacenar los datos, ya que pueden ser de enormes dimensiones. Se pueden almacenar en simples archivos csv (como es nuestro caso), en la nube, en archivos comprimidos, en formato matricial (numpy de Python), etc.

6.3 Preparación de los datos y análisis exploratorio

Una vez se ha estudiado el marco teórico, y se tienen los datos a disposición, se pueden comenzar su análisis, y el primer paso es hacer un preprocesamiento y un análisis exploratorio de los mismos.

Lo primero es hacer una previsualización de los datos para obtener una primera impresión. En nuestro caso, se disponía de varias bases de datos:

- Base de datos biomarcadores **difusión** de píxeles de tejido **sano**
- Base de datos biomarcadores **difusión** de píxeles de tejido **tumoral**
- Nombres variables **difusión**
- Base de datos biomarcadores **perfusión** de píxeles de tejido **sano**
- Base de datos biomarcadores **perfusión** de píxeles de tejido **tumoral**
- Nombres variables **perfusión**

El primer objetivo del preprocesamiento de datos es elaborar una base de datos única de perfusión y otra de difusión.

Para ello, se fusionan las bases de datos correspondientes a perfusión, y se añade una nueva variable *"dummy"* en la que 0 indica que el píxel es lesión tumoral y 1 que es un píxel de un tejido sano. Las variables son añadidas como los nombres de las columnas para una mejor comprensión de las tablas y futuras visualizaciones.

El mismo procedimiento se lleva a cabo con la base de datos de difusión.

Una vez las bases de datos están listas, se pasa a la parte de tratamiento de las mismas En nuestro caso, al haber recibido los datos de un previo estudio, los datos estaban ya tratados y procesados, lo único que había que realizar era eliminar aquellos píxeles de la base de datos de difusión que tuviesen el biomarcador $\lambda 3$ negativo, ya que en esos casos el tensor de difusión no es válido debido a que no tiene sentido físico.

Cabe destacar que nuestras bases de datos estaban bastante balanceadas, teniendo prácticamente el 50% de los casos de lesión y el otro 50% sanos.

Llegado a este punto, se puede comenzar a realizar el análisis exploratorio que nos ayuda tanto a comenzar a familiarizarse y visualizar los datos, como para la detección de anomalías. Inicialmente se han calculado varios estadísticos muestrales (media, desviación típica, mínimo, máximo y percentiles 25, 50 y 75):

	L1-L3	1-L3/L1	L1	L2	L3	Bio1	Bio2	Bio3	Bio4	Bio5	Bio6
count	8222.000000	8222.000000	8222.000000	8222.000000	8222.000000	8220.000000	8220.000000	8.220000e+03	8220.000000	8220.000000	8220.000000
mean	1.382388	1.820574	1.335745	1.212649	0.987412	0.003455	0.000005	2.162383e-09	0.000005	0.001152	0.001122
std	88.716109	85.578691	85.585939	77.748197	64.645888	0.001471	0.000004	2.782308e-09	0.000004	0.000490	0.000496
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	0.000418	0.306641	0.001114	0.000777	0.000417	0.002377	0.000002	3.560355e-10	0.000002	0.000792	0.000758
50%	0.000649	0.456584	0.001481	0.001100	0.000748	0.003318	0.000003	1.147391e-09	0.000004	0.001106	0.001079
75%	0.000964	0.653199	0.001878	0.001488	0.001124	0.004442	0.000006	2.935263e-09	0.000007	0.001481	0.001458
max	5992.000000	5662.000000	5662.000000	5199.000000	4911.000000	0.010106	0.000032	3.157295e-08	0.000039	0.003369	0.003282

	Bio7	Bio8	Bio9	Bio10	Bio11	Bio12	Bio13	Bio14	Bio15	Bio16	Bio17
count	8220.000000	8220.000000	8.220000e+03	8220.000000	8220.000000	8220.000000	8220.000000	8220.000000	8220.000000	8220.000000	8220.000000
mean	0.001083	0.001206	3.744827e-07	0.003282	0.001023	0.335024	0.212483	0.184511	0.033810	0.079571	0.051251
std	0.000512	0.000488	4.545287e-07	0.001504	0.000532	0.190867	0.138696	0.221487	0.045748	0.121760	0.097044
min	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000709	0.000858	9.311128e-08	0.002174	0.000639	0.183755	0.107306	0.034543	0.005774	0.011649	0.005858
50%	0.001047	0.001160	2.232117e-07	0.003159	0.000997	0.295604	0.175867	0.091660	0.015586	0.031537	0.016141
75%	0.001432	0.001528	4.882476e-07	0.004311	0.001393	0.463689	0.289242	0.244589	0.042744	0.089260	0.048200
max	0.003161	0.003595	6.592055e-06	0.009591	0.002932	0.975447	0.931321	0.999545	0.635800	0.923111	0.872243

	Bio18	Bio19	Bio20	Paciente	Sano
count	8220.000000	8220.000000	8220.000000	8220.000000	8222.000000
mean	0.241846	0.000967	1.782811	12.551338	0.401970
std	0.172057	0.000477	1.001372	5.635557	0.490326
min	0.000000	0.000000	0.000000	1.000000	0.000000
25%	0.108760	0.000606	1.295559	8.000000	0.000000
50%	0.191493	0.000922	1.518756	13.000000	0.000000
75%	0.339348	0.001291	1.979447	18.000000	1.000000
max	0.963472	0.002952	41.253558	24.000000	1.000000

Tabla 5. Resumen estadístico base de datos de difusión

	C1	C2	C3	RSS	C3_C2	C1_C2	C1C3_C2	Paciente	Sano
count	7576.000000	7576.000000	7576.000000	7576.000000	7576.000000	7576.000000	7576.000000	7576.00000	7576.000000
mean	234.986262	99.805182	138.414499	453.652352	53.544863	135.424669	87.497086	11.94179	0.538279
std	85.301000	64.657858	59.232703	681.896931	67.844517	79.259090	72.817745	6.39027	0.498565
min	19.710911	0.000000	0.000000	0.382078	0.000000	0.000000	0.000000	1.00000	0.000000
25%	170.177934	44.119330	99.456084	103.863688	0.000000	77.235620	30.034738	7.00000	0.000000
50%	237.748158	70.137903	137.783324	230.571250	10.598480	128.829274	58.258638	11.00000	1.000000
75%	305.343645	165.573022	172.694176	504.497060	106.071797	179.228976	143.049620	19.00000	1.000000
max	488.915037	277.007884	397.577457	10058.936390	335.039743	432.957750	337.275745	24.00000	1.000000

Tabla 6. Resumen estadístico base de datos de perfusión

A continuación hemos llevado a cabo un análisis de componentes principales.

Para ello ha sido necesario centrar y escalar los datos, al tratarse de distintas métricas.

6.3.1 PCA Difusión

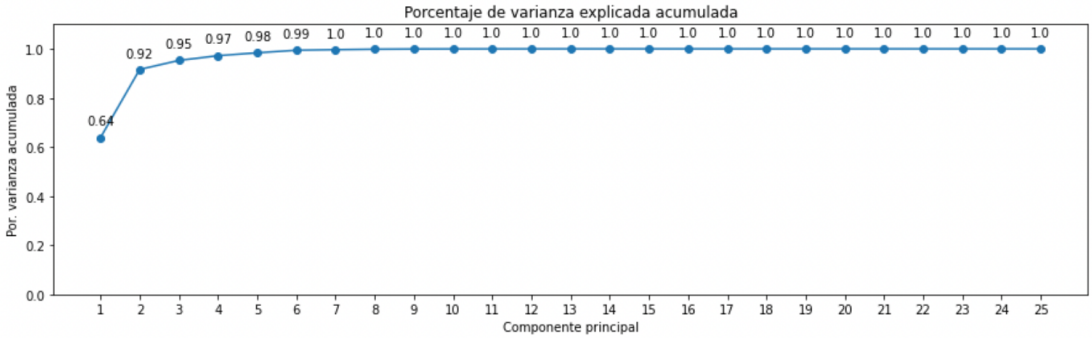


Figura 39. Porcentaje de variabilidad total explicada con cada componente principal adicional en la base de datos de difusión

Se lleva a cabo la reducción de dimensionalidad de nuestro espacio de datos, y vemos en la Figura 39 que, con las dos primeras componentes, se explica un 92% de la variabilidad total de los datos es explicada, un valor considerablemente alto del que podemos obtener interesantes conclusiones.

Nuestra variable objetivo es conocer si el píxel en cuestión corresponde a un tejido sano o tumoral, por lo que podemos visualizar ambos tipos de observaciones por separado en las dos primeras componentes, de tal manera que podremos tratar de explorar qué variables están más asociadas a la separación entre ambos grupos de píxeles.

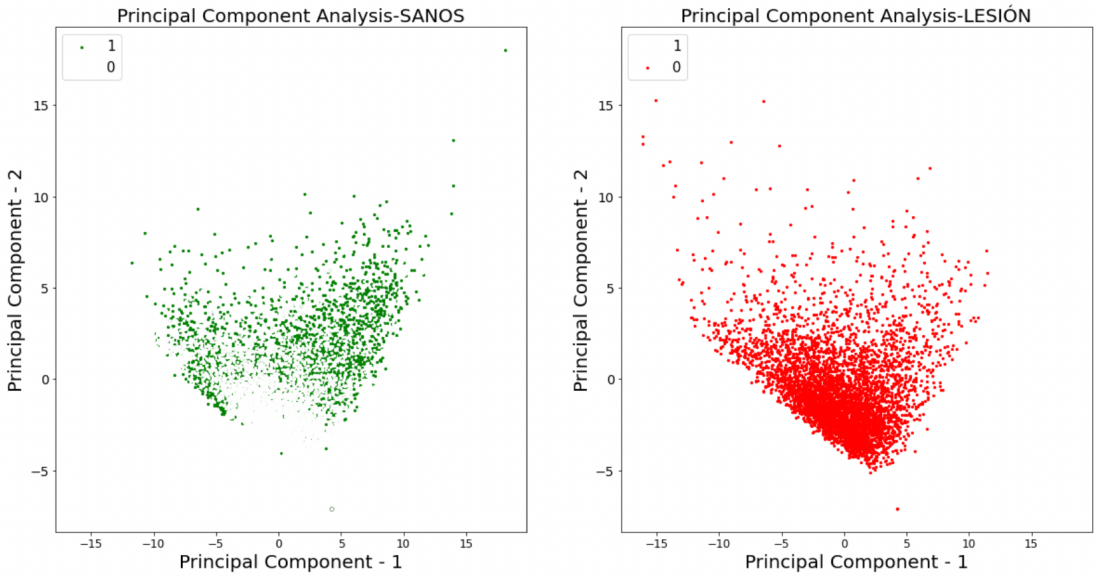


Figura 40. Píxeles correspondientes a tejido sano (izquierda) y tumoral (derecha) en las dos primeras componentes principales de la base de datos de difusión

Podemos ver en la figura 40 que es la segunda componente principal la variable latente que distingue de una mejor manera entre las clases de tejido. Visualizando los loadings podremos ver qué variables son más influyentes en dicha componente principal, y por tanto tienen comportamientos distintos en ambos tipos de tejido.

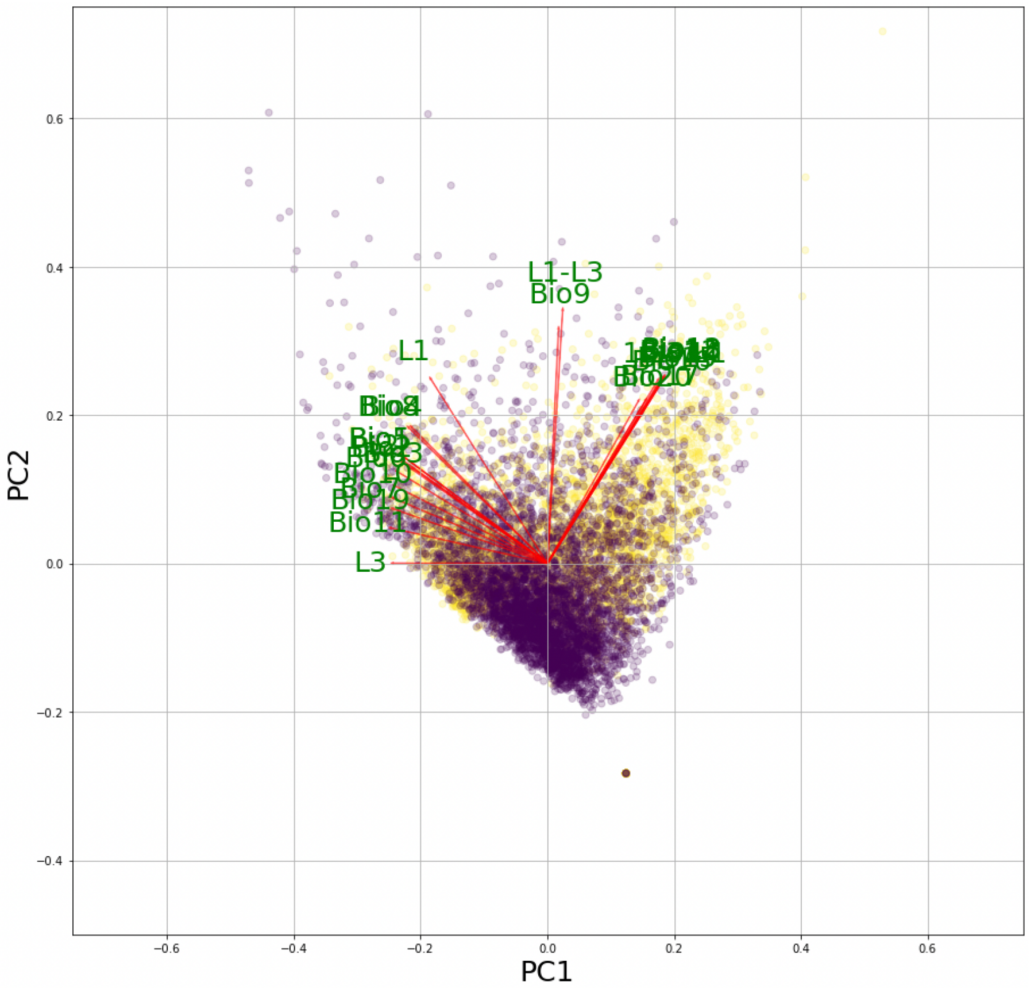


Figura 41. Loadings de las variables originales visualizadas en las dos primeras componentes principales de la base de datos de difusión

Esta segunda componente es una variable latente creada a partir de las variables originales, donde algunas han tenido más importancia que otras, siendo son dos las que destacan entre las demás: el biomarcador λ_1-λ_3 y la traza D_{an}. Por tanto, al ser la 2ª componente la que mejor distingue entre tipos de tumores, se podría concluir que estos son los dos biomarcadores más importantes a tener en cuenta cuando estamos hablando de la sanidad del tejido del píxel en cuestión.

6.3.2 PCA Perfusión

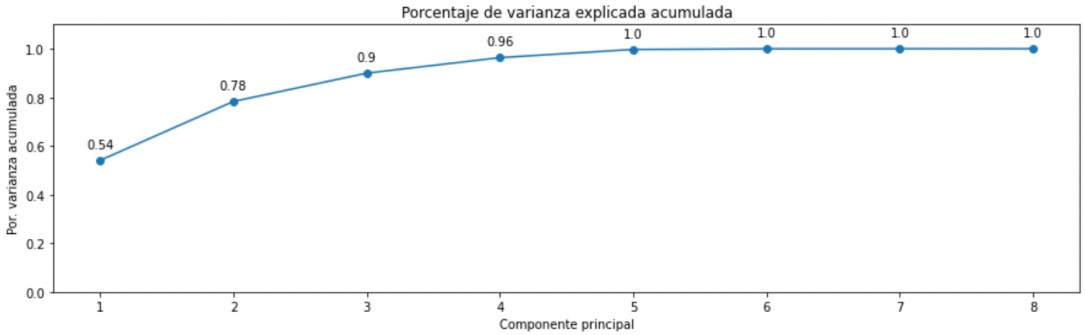


Figura 42. Porcentaje de variabilidad total explicada con cada componente principal adicional en la base de datos de perfusión

En este caso, la variabilidad explicada por cada componente adicional es menor que en perfusión. En la Figura 42 se puede ver que con las dos primeras componentes principales se explica un 78% de la variabilidad total de los datos. Es un valor considerablemente alto también y del que se pueden obtener útiles conclusiones.

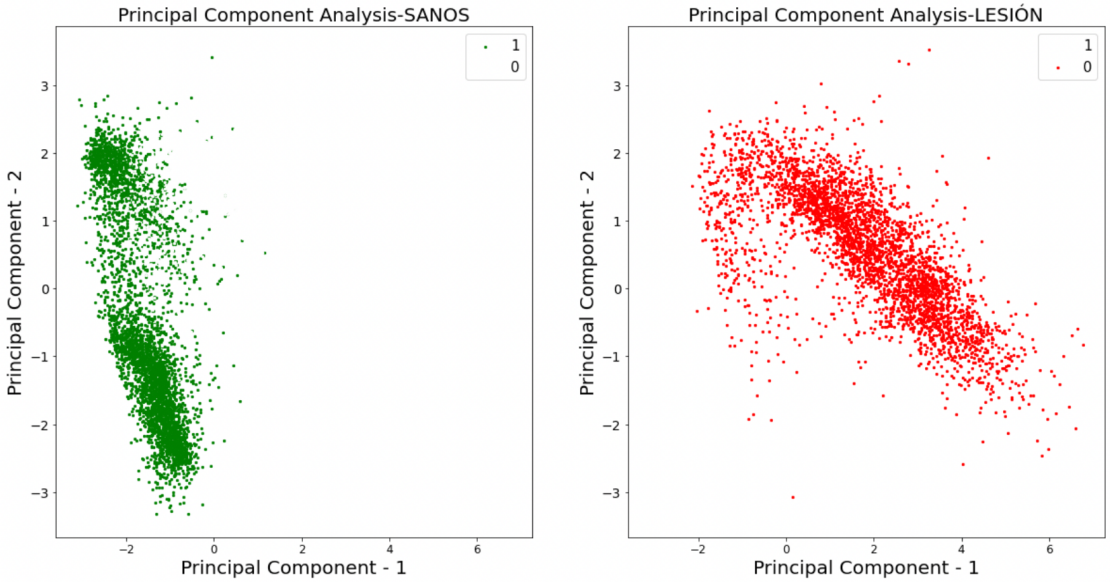


Figura 43. Píxeles correspondientes a tejido sano (izquierda) y tumoral (derecha) en las dos primeras componentes principales de la base de datos de perfusión

De nuevo se ha procedido de la misma forma que en difusión, distinguiendo entre tejido sano y tumoral. La Figura 43 muestra que en este caso la separación entre tipos de tejido se aprecia en la primera componente, y es mucho más clara que en el caso de la base de datos de difusión. Fijándonos en los *loadings* podremos observar qué variables son las que contribuyen principalmente a crear esta variable latente y por tanto, cuáles serán las que habrá que tener más en cuenta cuando estemos prediciendo la clase de tejido en cuestión.

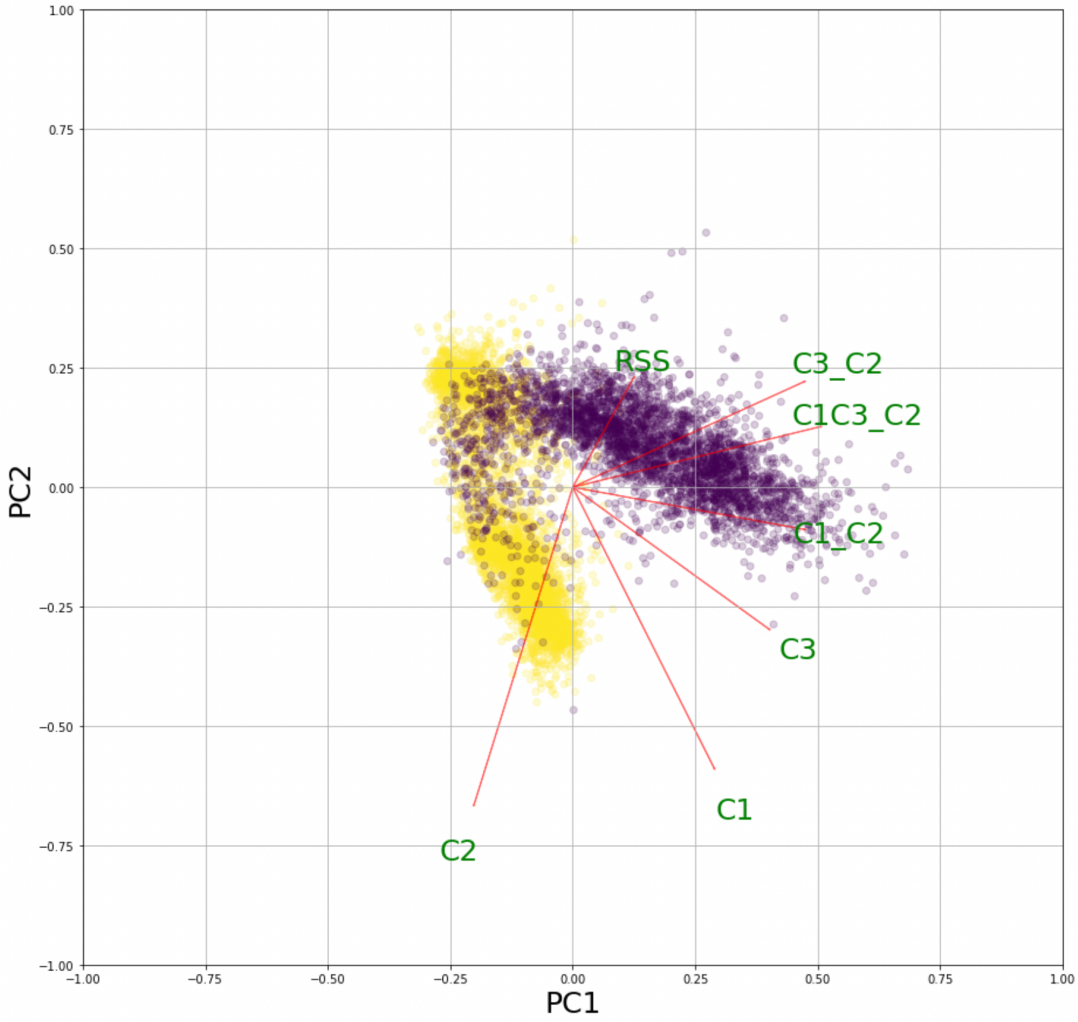


Figura 44. Loadings de las variables originales visualizadas en las dos primeras componentes principales de la base de datos de difusión

Recordemos que esta base de datos estaba formada por 3 biomarcadores y por combinaciones lineales de los mismos. Teniendo en cuenta que la primera componente principal es la que mejor distingue entre las clases de tejido del píxel, viendo los *loadings* observamos que las combinaciones lineales de los biomarcadores son las que tienen una mayor influencia en esta variable latente. Fijándonos en los tres biomarcadores por separado, es el C_3 el que tiene más importancia en esta primera componente principal, por lo que se puede concluir, que es este biomarcador, y las combinaciones lineales de todos, en los que tenemos que centrar nuestra atención cuando queremos distinguir el tipo de tejido. Es un resultado lógico debido a que el C_3 (VT) está directamente relacionado con una dinámica de un comportamiento vascularizado, que son los procesos fisiológicos relacionados con el cáncer en etapas precoces que queremos detectar con la técnica de la perfusión. Las combinaciones lineales de los biomarcadores permiten localizar de una manera más fácil los tumores (véase Figura 8) por lo que también es lógico pensar que sean las variables clave en la separación de las clases de tejido (tumoral vs sano).

6.4 Entrenamiento de modelos

Cuando la recolección y preparación de los datos está lista, es momento de elegir qué tipo de modelo o algoritmo queremos utilizar. En general, se puede clasificar en tres modelos:

- Clasificación binaria: respuesta dicotómica: sí o no
- Clasificación multiclase: respuesta categórica: clase 1, clase 2, ….
- Regresión: respuesta numérica.

En nuestro caso, es un problema de clasificación binaria, ya que queremos predecir si un píxel forma parte de un tejido tumoral o de un tejido sano. Por lo que tenemos que usar modelos de machine learning acordes a ese objetivo.

Por tanto nuestra variable objetivo Y será la variable *dummy* que indica de qué tejido se trata (tumoral o sano).

Nuestro conjunto de datos **X** estará formado por el resto de variables.

6.4.1 División de los datos

Para entrenar el modelo y evaluar posteriormente su capacidad predictiva, es necesario dividir el conjunto de datos. Se puede hacer de varias formas, pero las más utilizadas son las dos siguientes:

- División en conjunto de entrenamiento, validación y test:

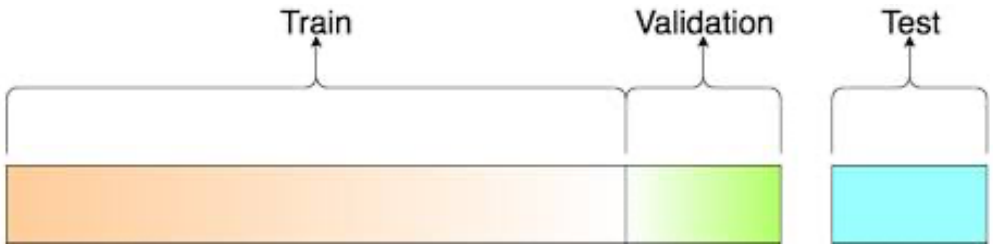


Figura 45. División de datos en entrenamiento, validación y test

El conjunto de entrenamiento (*Train*) son los datos que utilizamos para que nuestro modelo se entrene y aprenda.

El conjunto de validación (*Validation*) es la muestra de datos utilizada para proporcionar una evaluación insesgada del ajuste de un modelo en el conjunto de datos de entrenamiento mientras se ajustan los hiperparámetros del modelo.

El conjunto de test es la muestra de datos utilizada para proporcionar una evaluación insesgada del ajuste de un modelo final en el conjunto de datos de entrenamiento.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, stratify = y)
```

Figura 46. Código para dividir los datos

- Validación cruzada:

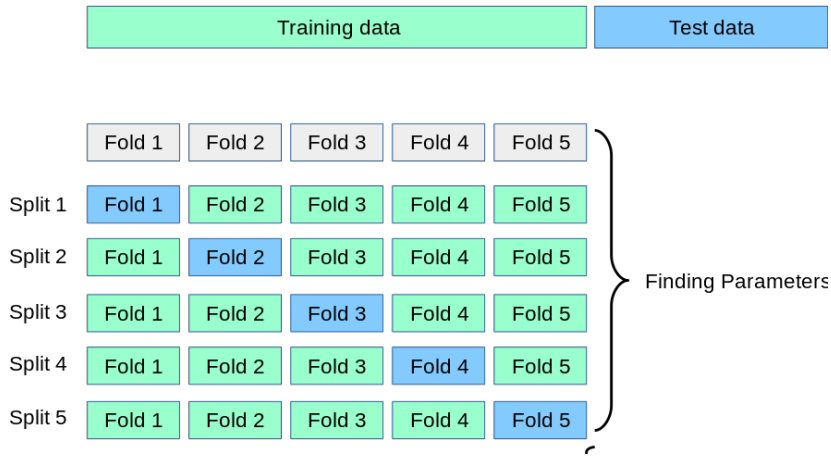


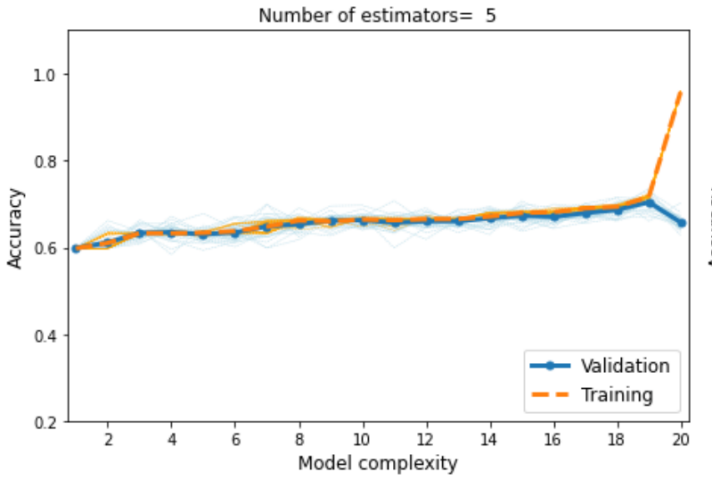
Figura 47. Funcionamiento de la validación cruzada

La validación cruzada o *"cross-validation"* es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y test. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Se utiliza en entornos donde no se dispone de bases de datos extensas.

Después de preprocesar los datos recogidos y dividirlos en tres subconjuntos, se puede proceder al entrenamiento del modelo. Este proceso implica "alimentar" al algoritmo con datos de entrenamiento. Un algoritmo procesará los datos y producirá un modelo capaz de encontrar un valor objetivo (atributo) en los nuevos datos, una respuesta que se desea obtener con el análisis predictivo.

6.4.2 Entrenamiento

El objetivo del entrenamiento del modelo es desarrollar un modelo con las mejores capacidades predictivas posibles, para ello, hay que saber de qué manera comenzar a buscar los mejores hiperparámetros para optimizar el tiempo de cálculo.




```
min_leaf_values = list(np.linspace((len(X_train)/3), 1,20, dtype=int))
```

Figura 48. Comportamiento de un modelo según la complejidad del mismo

En la figura 48 vemos un ejemplo de cómo se comporta el modelo Random Forest cuando aumentamos la complejidad del modelo, para ello utilizamos un parámetro (min_leaf_values) que viene dado por la fórmula de la Figura 48. Se visualiza el comportamiento del modelo cuando la complejidad del mismo varía. Con los primeros valores prueba un algoritmo altamente regularizado. Con el valor final (min_leaf_value = 1) prueba un algoritmo totalmente no regularizado. Vemos como se produce el *overfitting* (momento en el que el algoritmo aprende "demasiado bien" los datos de entrenamiento y empeora sus capacidades para datos nuevos).

Todo esto se traza para entender aproximadamente dónde hacer una búsqueda específica en la de los hiperparámetros en la futura *Grid Search*. Así se ahorra tiempo.

6.4.3 Grid Search. Búsqueda de los hiperparámetros óptimos.

La forma tradicional de realizar la optimización de hiperparámetros ha sido Grid Search, que es simplemente una búsqueda exhaustiva a través de un subconjunto especificado manualmente del espacio de hiperparámetros de un algoritmo de aprendizaje.

La Grid Search debe ser guiada por alguna métrica de rendimiento, normalmente medida por validación cruzada en el conjunto de entrenamiento.

```
In [42]: def create_model(dropout_rate=0.0, weight_constraint=0):
             # create model
             model = Sequential()
             model.add(Dense(12, input_dim=7, kernel_initializer='uniform', activation='linear', kernel_constraint=MaxNorm(w
             model.add(Dropout(dropout_rate))
             model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))
             model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
             return model

         seed = 7
         numpy.random.seed(seed)
         model = KerasClassifier(build_fn=create_model, epochs=100, batch_size=10, verbose=0)

         weight_constraint = [1, 2, 3, 4, 5]
         dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

         param_grid = dict(dropout_rate=dropout_rate, weight_constraint=weight_constraint)
         grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
         grid_result = grid.fit(X, Y)

         print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
         means = grid_result.cv_results_['mean_test_score']
         stds = grid_result.cv_results_['std_test_score']
         params = grid_result.cv_results_['params']
         for mean, stdev, param in zip(means, stds, params):
             print("%f (%f) with: %r" % (mean, stdev, param))

         Best: 0.880808 using {'dropout_rate': 0.2, 'weight_constraint': 1}
         0.873155 (0.041556) with: {'dropout_rate': 0.0, 'weight_constraint': 1}
         0.866557 (0.043681) with: {'dropout_rate': 0.0, 'weight_constraint': 2}
         0.869987 (0.042608) with: {'dropout_rate': 0.0, 'weight_constraint': 3}
         0.872891 (0.040832) with: {'dropout_rate': 0.0, 'weight_constraint': 4}
         0.862730 (0.049052) with: {'dropout_rate': 0.0, 'weight_constraint': 5}
         0.868537 (0.044097) with: {'dropout_rate': 0.1, 'weight_constraint': 1}
         0.873023 (0.041173) with: {'dropout_rate': 0.1, 'weight_constraint': 2}
         0.873287 (0.042040) with: {'dropout_rate': 0.1, 'weight_constraint': 3}
         0.864709 (0.047576) with: {'dropout_rate': 0.1, 'weight_constraint': 4}
         0.863917 (0.047168) with: {'dropout_rate': 0.1, 'weight_constraint': 5}
         0.880808 (0.037399) with: {'dropout_rate': 0.2, 'weight_constraint': 1}
         0.867480 (0.042799) with: {'dropout_rate': 0.2, 'weight_constraint': 2}
         0.872627 (0.041538) with: {'dropout_rate': 0.2, 'weight_constraint': 3}
         0.863522 (0.047463) with: {'dropout_rate': 0.2, 'weight_constraint': 4}
         0.867349 (0.045379) with: {'dropout_rate': 0.2, 'weight_constraint': 5}
         0.869591 (0.042576) with: {'dropout_rate': 0.3, 'weight_constraint': 1}
         0.863786 (0.048323) with: {'dropout_rate': 0.3, 'weight_constraint': 2}
         0.867480 (0.045258) with: {'dropout_rate': 0.3, 'weight_constraint': 3}
         0.874871 (0.040752) with: {'dropout_rate': 0.3, 'weight_constraint': 4}
```

Figura 49. Grid Search red neuronal

En el ejemplo de la Figura 49, se realiza una Grid Search para encontrar los mejores valores de dos hiperparámetros de una red neuronal, en concreto *dropout_rate* y *weight_constraing*.

6.5 Análisis de resultados

Se han aplicado distintos algoritmos a ambas bases de datos con el objetivo de predecir la clase de tejido del que se trata cada píxel. Se han ordenado según su *accuracy,* pero también se han aplicado la curva ROC y el coeficiente de correlación de Matthews.

Se define:

Real/Predicho	P	N
P	TP	FN
N	FP	TN

Tabla 7. Matriz de confusión

- TP = Verdaderos Positivos (*true positives*)
- TN = Verdaderos Negativos (*true negatives*)
- FP = Falsos Positivos (*false positives*)
- FN = Falsos Negativos (*false negatives*)

Dichas medidas de evaluación de modelos se definen de la siguiente manera:

- **ACCURACY:**

  El porcentaje de predicciones que nuestro modelo ha realizado exitosamente

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

- **MCC (*Matthews correlation coefficient*):**

  Cuando disponemos de bases de datos cuyas clases están desbalanceadas (hay muchas más observaciones de un tipo que del otro), la accuracy no sería una buena medida, ya que si un 80% fuese de un tipo, un modelo que siempre predijese dicho tipo, obtendría un 80% de acierto. Para esos casos es interesante recurrir a otras medidas que tienen en cuenta el desbalanceo entre clases, como el coeficiente de correlación de Matthews. Un coeficiente de +1 sería una predicción perfecta y un coeficiente de -1 indica un desacuerdo total entre la predicción y el valor real. Si se obtiene un coeficiente de 0, con una predicción aleatoria se obtendrían resultados similares.

$$MCC = \frac{TP * TN - FN * FP}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

- **AUROC (*area under the receiver operating characteristic*):**

  La curva ROC la capacidad de nuestro modelo de distinguir entre dos predicciones, en nuestro caso, si el píxel es de tejido sano o tumoral. Esta curva representa dos parámetros:

  - **El ratio de verdaderos positivos TPR (*True Positive Rate*):**

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

- **El ratio de falsos positivos FPR (*False Positive Rate*):**

$$TPR = \frac{FP}{FP + TN}$$

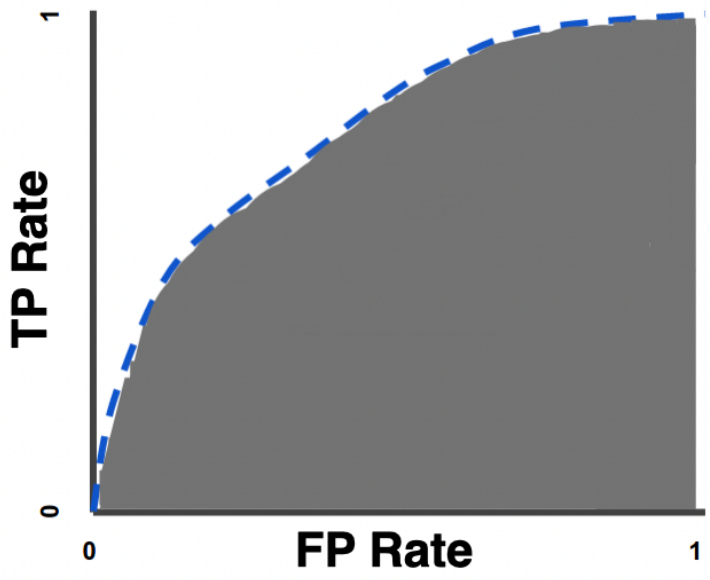


Figura 50. AUC

En la figura 50, lo representado por el color gris es el área bajo la curva (AUC). Esta métrica proporciona una medida agregada del rendimiento en todos los umbrales de clasificación posibles. Interesa que el AUC sea lo más próximo a uno, lo que indicaría una clasificación perfecta (FP=0 y FN=0). [26]

6.5.1 Difusión

Modelos basados en árboles	Hiperparámetros				Accuracy		AUROC		MCC	
	criterio	min_sample_split	min_sample_leaf	n_estimators	Train	Test	Train	Test	Train	Test
Random Forest	gini	2	97	5	0.731	0.725	0.702	0.696	0.418	0.392
	gini	2	120	5	0.736	0.715	0.707	0.690	0.422	0.402
AdaBoost	gini	2	400	5	0.724	0.719	0.710	0.643	0.425	0.349
	gini	6	400	5	0.724	0.719	0.710	0.643	0.421	0.315
Decision Tree	gini	4	130	-	0.719	0.701	0.699	0.682	0.407	0.370

Tabla 8. Resultados de los modelos basados en árboles de decisión en difusión

```
In [11]: clf = RandomForestClassifier(criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=97,
                                     max_features=None, n_estimators=5)
         clf = clf.fit(X_train, y_train)
         y_pred = clf.predict(X_test)
         y_pred_tr = clf.predict(X_train)
         acc_train= round(accuracy_score(y_test, y_pred),3)
         acc_test = round(accuracy_score(y_train, y_pred_tr), 3)
         print("Train accuracy: ", acc_train)
         print("Test accuracy", acc_test)

         Train accuracy:  0.731
         Test accuracy 0.725
```

Figura 51. Código del test correspondiente al mejor modelo basado en árbol para difusión

En difusión, se observa que los mejores resultados se obtienen con un Random Forest, obteniendo un 72.5% de *accuracy* en el test.

Cabe destacar que el modelo AdaBoost proporciona unos resultados muy buenos también.

	Hiperparámetros			Accuracy		AUROC		MCC	
	Kernel	C	Gamma	Train	Test	Train	Test	Train	Test
SVM	rbf	100	auto	0.685	0.692	0.639	0.632	0.322	0.305
	rbf	50	scale	0.669	0.653	0.608	0.603	0.279	0.259

Tabla 9. Resultados de las máquinas de vectores soporte en difusión

```
In [14]: final_model =  SVC(C = 100, kernel='rbf', gamma= 'auto')
         final_model.fit(X,y)
         y_pred_tr = final_model.predict(X_train)
         y_pred_ts = final_model.predict(X_test)

         tr_acc = accuracy_score(y_train ,y_pred_tr)
         ts_acc = accuracy_score(y_test,y_pred_ts)
         print('train accuracy: ', tr_acc)
         print('test accuracy: ', ts_acc)

         train accuracy:  0.6845093268450932
         test accuracy:  0.6922141119221411
```

Figura 52. Código del test correspondiente al mejor modelo de SVM para difusión

En cuanto a las máquinas de vectores soporte, un 62,9% de *accuracy* es lo máximo obtenido tras la búsqueda de los mejores hiperparámetros.

Hiperparámetros red neuronal artificial							
n	**epochs**	**Batch size**	**Optimizer**	**Learn rate**	**mom**	**Init mode**	**Weight const**
64	60	10	'nadam'	0.3	0.9	'he_uniform'	1

Tabla 10. Hiperparámetros red neuronal artificial en difusión

Métricas de evaluación red neuronal					
Accuracy		**AUROC**		**MCC**	
Train	Test	Train	Test	Train	Test
0.716	0.690	0.702	0.676	0.407	0.353

Tabla 11. Resultados de la red neuronal artificial en difusión

```
In [91]: model = Sequential()
         model.add(Dense(64, activation=tf.nn.relu, kernel_initializer='uniform',
                         input_dim = 25))
         model.add(Dropout(0.1))
         model.add(Dense(64, kernel_initializer='uniform', activation=tf.nn.relu))
         model.add(Dense(1, kernel_initializer='he_uniform', activation='sigmoid'))
         model.compile(loss='binary_crossentropy',
                       optimizer='nadam',
                       metrics=['acc'])
         model_history = model.fit(X_train, y_train,
                               batch_size=10,
                               epochs=60,
                               verbose=1,
                               validation_data=(X_test, y_test))

         model.fit(X_train, y_train, epochs=60, batch_size=10)
         y_pred_ts = model.predict(X_test)
         y_pred_tr = model.predict(X_train)

         for i in range(len(y_pred_ts)):
             if y_pred_ts[i]<0.5:
                 y_pred_ts[i] = 0
             else:
                 y_pred_ts[i] = 1

         for i in range(len(y_pred_tr)):
             if y_pred_tr[i]<0.5:
                 y_pred_tr[i] = 0
             else:
                 y_pred_tr[i] = 1
         tr_acc = accuracy_score(y_train ,y_pred_tr)
         ts_acc = accuracy_score(y_test,y_pred_ts)
         print('train accuracy: ', tr_acc)
         print('test accuracy: ', ts_acc)
         7397/7397 [==============================] - 3s 405us/sample - loss: 0.5310 - acc: 0.7154
         Epoch 54/60
         7397/7397 [==============================] - 4s 475us/sample - loss: 0.5310 - acc: 0.7131
         Epoch 55/60
         7397/7397 [==============================] - 4s 547us/sample - loss: 0.5312 - acc: 0.7184
         Epoch 56/60
         7397/7397 [==============================] - 6s 764us/sample - loss: 0.5308 - acc: 0.7168
         Epoch 57/60
         7397/7397 [==============================] - 1003s 136ms/sample - loss: 0.5313 - acc: 0.7170
         Epoch 58/60
         7397/7397 [==============================] - 6s 860us/sample - loss: 0.5306 - acc: 0.7169
         Epoch 59/60
         7397/7397 [==============================] - 224s 30ms/sample - loss: 0.5312 - acc: 0.7160
         Epoch 60/60
         7397/7397 [==============================] - 5s 735us/sample - loss: 0.5313 - acc: 0.7146
         WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle in
         put: <class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>
         WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle in
         put: <class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>
         train accuracy:  0.7179937812626741
         test accuracy:  0.7019464720194647
```

Figura 53. Código red neuronal artificial en difusión

Las redes neuronales nos proporcionan un mejor resultado que las máquinas de vectores soporte, pero no llegan a los valores de los métodos basados en árboles, a pesar de acercarse considerablemente. El mejor resultado ha sido de un 69% de *accuracy* en el conjunto de test.

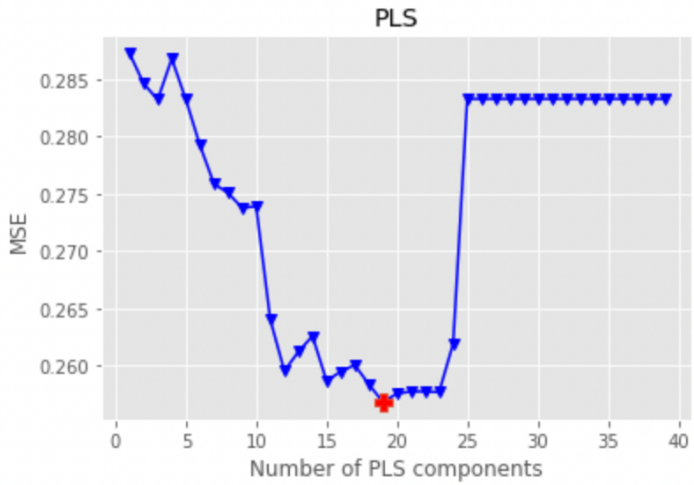


Figura 54. Búsqueda del número óptimo de componentes PLS

```
In [32]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt

         from scipy.signal import savgol_filter

         from sklearn.cross_decomposition import PLSRegression
         from sklearn.model_selection import KFold, cross_val_predict, train_test_split
         from sklearn.metrics import accuracy_score

         pls_binary = PLSRegression(n_components=19)
         pls_binary.fit(X_train, y_train)

         y_pred = pls_binary.predict(X_test)[:,0]


         y_pred_ts = pls_binary.predict(X_test)
         y_pred_tr = pls_binary.predict(X_train)
         for i in range(len(y_pred_tr)):
             if y_pred_tr[i]<0.5:
                 y_pred_tr[i] = 0
             else:
                 y_pred_tr[i] = 1
         for i in range(len(y_pred_ts)):
             if y_pred_ts[i]<0.5:
                 y_pred_ts[i] = 0
             else:
                 y_pred_ts[i] = 1

         tr_acc = accuracy_score(y_train,y_pred_tr)
         ts_acc = accuracy_score(y_test,y_pred_ts)
         print('train accuracy: ', tr_acc)
         print('test accuracy: ', ts_acc)

         train accuracy:  0.6804109774232797
         test accuracy:  0.6739659367396593
```

Figura 55. Código PLS difusión

Métricas de evaluación PLS					
Accuracy		**AUROC**		**MCC**	
Train	Test	Train	Test	Train	Test
0.680	0.674	0.659	0.653	0.325	0.312

Tabla 12. Resultados PLS en difusión

Cabe destacar que se ha llevado a cabo también un PLS, obteniendo un 67% de *accuracy* en el conjunto de test.

Se visualizan a los resultados de todos los modelos en conjunto en la Figura 56.

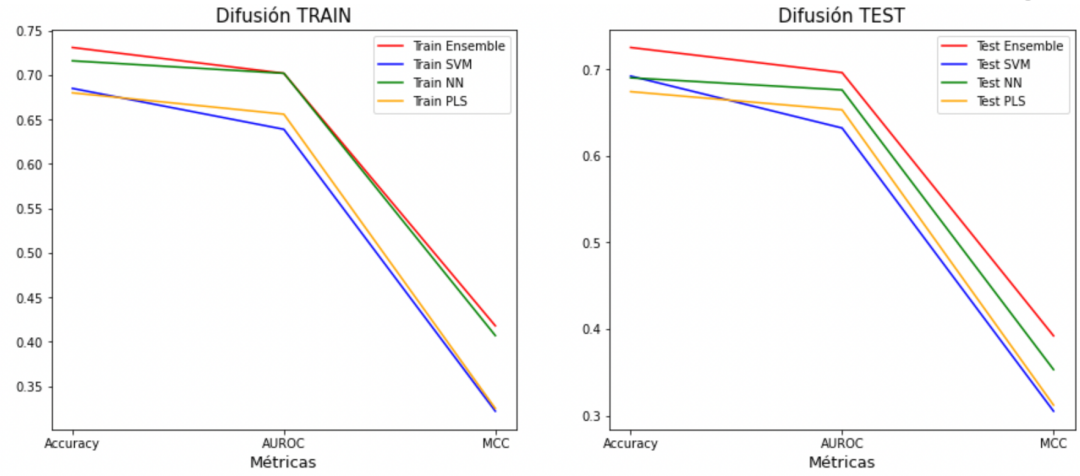


Figura 56. Resultados difusión

6.5.2 Perfusión

Modelos basados en árboles	Hiperparámetros				Accuracy		AUROC		MCC	
	criterio	min_sample_split	min_sample_leaf	n_estimators	Train	Test	Train	Test	Train	Test
AdaBoost	gini	8	1	5	0.960	1	1	0.969	1	0.939
	gini	6	1	5	0.956	1	1	0.969	1	0.939
Random Forest	gini	8	1	5	0.966	0.989	0.989	0.968	0.979	0.936
	gini	4	1	5	0.954	0.993	0.994	0.970	0.988	0.942
Decision Tree	gini	10	1	-	0.990	0.962	0.990	0.962	0.981	0.923

Tabla 13. Resultados de los modelos basados en árboles de decisión en perfusión

```
In [14]:
clf = AdaBoostClassifier(DecisionTreeClassifier(min_samples_leaf=1, min_samples_split = 8, criterion = 'gini'),
                         n_estimators=5)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_pred_tr = clf.predict(X_train)
acc_train= round(accuracy_score(y_test, y_pred),3)
acc_test = round(accuracy_score(y_train, y_pred_tr), 3)
print("Train accuracy: ", acc_train)
print("Test accuracy", acc_test)

Train accuracy:  0.96
Test accuracy 1.0
```

Figura 57. Código del test correspondiente al mejor modelo basado en árbol para perfusión

En el caso de los modelos basados en árboles, es el AdaBoost el que ha conseguido predicción perfecta, siendo de un 100% en el conjunto de test. Seguido de Random Forest con excelentes resultados también.

	Hiperparámetros			Accuracy		AUROC		MCC	
	Kernel	C	Degree	Train	Test	Train	Test	Train	Test
SVM	'poly'	100	2	0.932	0.941	0.927	0.946	0.864	0.898
	'poly'	50	2	0.930	0.940	0.925	0.943	0.862	0.923

Tabla 14. Resultados de las máquinas de vectores soporte en perfusión

```
In [8]: final_model =  SVC(C = 100, kernel='poly', degree= 2)
        final_model.fit(X,y)
        y_pred_tr = final_model.predict(X_train)
        y_pred_ts = final_model.predict(X_test)

        tr_acc = accuracy_score(y_train ,y_pred_tr)
        ts_acc = accuracy_score(y_test,y_pred_ts)
        print('train accuracy: ', tr_acc)
        print('test accuracy: ', ts_acc)

        train accuracy:  0.9316515107069522
        test accuracy:  0.9406332453825857
```

Figura 58. Código del test correspondiente al mejor modelo de SVM para perfusión

En cuanto a las máquinas de vectores soporte, se ha conseguido un 94.1% de accuracy en el conjunto de test, siendo también un muy buen resultado.

Hiperparámetros red neuronal artificial							
n	**epochs**	**Batch size**	**Optimizer**	**Learn rate**	**mom**	**Init mode**	**Weight const**
30	10	20	‘adagrad’	0.3	0.9	‘he_uniform’	1

Tabla 15. Hiperparámetro de la red neuronal artificial en perfusión

Métricas de evaluación red neuronal					
Accuracy		**AUROC**		**MCC**	
Train	Test	Train	Test	Train	Test
0.924	0.917	0.893	0.900	0.785	0.801

Tabla 16. Resultados de la red neuronal en perfusión

```
In [43]: model = Sequential()
model.add(Dense(30, input_dim=7, kernel_initializer='uniform', kernel_constraint = MaxNorm(1)))
model.add(Dropout(0))
model.add(Dense(1, kernel_initializer='he_uniform', activation='sigmoid'))
optimizer = SGD(lr=0.3, momentum=0.9)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=20)
y_pred_ts = model.predict(X_test)
y_pred_tr = model.predict(X_train)
for i in range(len(y_pred_tr)):
    if y_pred_tr[i]<0.5:
        y_pred_tr[i] = 0
    else:
        y_pred_tr[i] = 1
for i in range(len(y_pred_ts)):
    if y_pred_ts[i]<0.5:
        y_pred_ts[i] = 0
    else:
        y_pred_ts[i] = 1
tr_acc = accuracy_score(y_train,y_pred_tr)
ts_acc = accuracy_score(y_test,y_pred_ts)
print('train accuracy: ', tr_acc)
print('test accuracy: ', ts_acc)
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle in
put: <class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>
Train on 6818 samples
Epoch 1/10
6818/6818 [==============================] - 1s 92us/sample - loss: 0.2957 - accuracy: 0.9141
Epoch 2/10
6818/6818 [==============================] - 0s 53us/sample - loss: 0.2430 - accuracy: 0.9193
Epoch 3/10
6818/6818 [==============================] - 0s 53us/sample - loss: 0.2392 - accuracy: 0.9190
Epoch 4/10
6818/6818 [==============================] - 0s 53us/sample - loss: 0.2342 - accuracy: 0.9220
Epoch 5/10
6818/6818 [==============================] - 0s 58us/sample - loss: 0.2397 - accuracy: 0.9218
Epoch 6/10
6818/6818 [==============================] - 0s 56us/sample - loss: 0.2385 - accuracy: 0.9202
Epoch 7/10
6818/6818 [==============================] - 0s 54us/sample - loss: 0.2402 - accuracy: 0.9182
Epoch 8/10
6818/6818 [==============================] - 0s 56us/sample - loss: 0.2411 - accuracy: 0.9199
Epoch 9/10
6818/6818 [==============================] - 0s 53us/sample - loss: 0.2458 - accuracy: 0.9174
Epoch 10/10
6818/6818 [==============================] - 0s 54us/sample - loss: 0.2432 - accuracy: 0.9185
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle in
put: <class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle in
put: <class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>
train accuracy:  0.9240246406570842
test accuracy:  0.91688654353562
```

Figura 59. Código red neuronal artificial en perfusión

Con las redes neuronales se ha alcanzado un 91.4% en el conjunto de test.

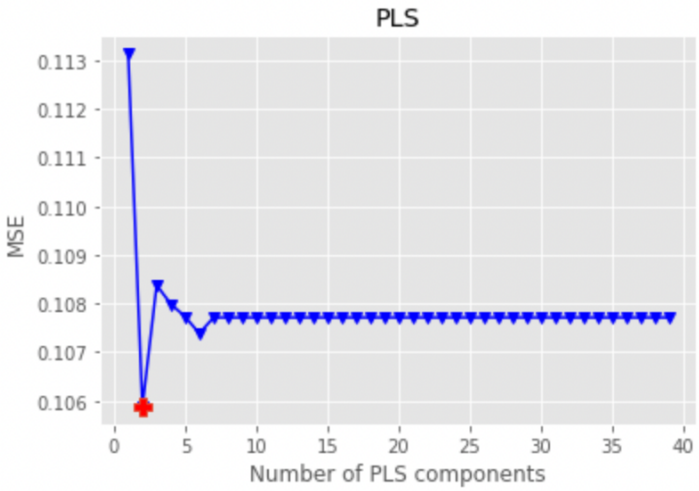


Figura 60. Búsqueda del número óptimo de componentes PLS

```
In [48]:
pls_binary = PLSRegression(n_components=2)
pls_binary.fit(X_train, y_train)
y_pred = pls_binary.predict(X_test)[:,0]
y_pred_ts = pls_binary.predict(X_test)
y_pred_tr = pls_binary.predict(X_train)
for i in range(len(y_pred_tr)):
    if y_pred_tr[i]<0.5:
        y_pred_tr[i] = 0
    else:
        y_pred_tr[i] = 1
for i in range(len(y_pred_ts)):
    if y_pred_ts[i]<0.5:
        y_pred_ts[i] = 0
    else:
        y_pred_ts[i] = 1

In [49]:
tr_acc = accuracy_score(y_train,y_pred_tr)
ts_acc = accuracy_score(y_test,y_pred_ts)
print('train accuracy: ', tr_acc)
print('test accuracy: ', ts_acc)

train accuracy:  0.9222645937224992
test accuracy:  0.9129287598944591
```

Figura 61. Código PLS perfusión

Métricas de evaluación PLS					
Accuracy		**AUROC**		**MCC**	
Train	Test	Train	Test	Train	Test
0.927	0.927	0.916	0.923	0.843	0.856

Tabla 17. Resultados PLS perfusión

El PLS también se ha implementado, obteniendo un 91% de *accuracy* en el conjunto de test.

Al igual que en la base de datos de perfusión, ha sido con los modelos basados en árboles con los que se han obtenido los mejores resultados.

Para finalizar, en la Figura 62 visualizan los resultados en conjunto.

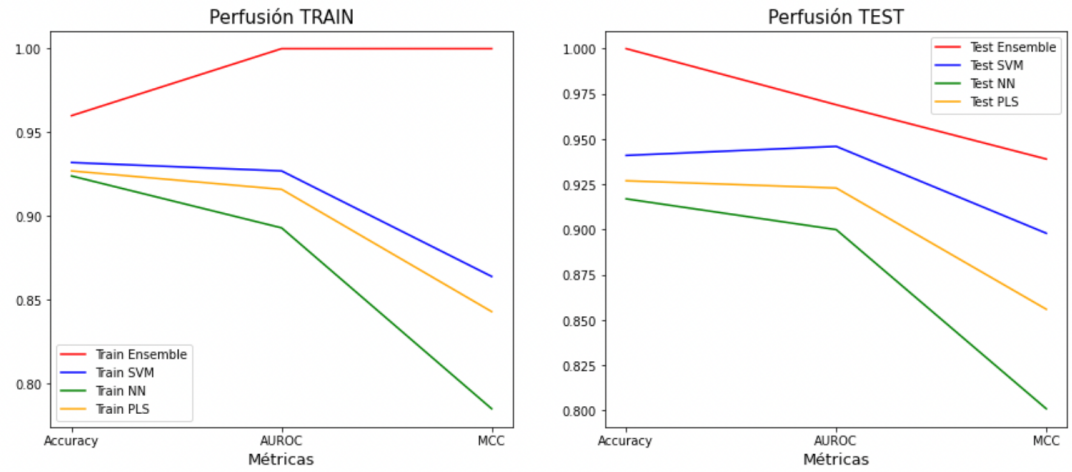


Figura 62. Resultados perfusión

7. Conclusiones

Cada proyecto de *machine learning* es único, ya que depende mucho del marco en el que se desarrolla, y de los datos en cuestión. A pesar de tener mucha experiencia desarrollando modelos y resolviendo problemas con ellos, al comenzar un proyecto nuevo, nuevos retos se presentan y hay que lidiar con ellos.

Tras la visualización de los resultados de los modelos, se presentan las conclusiones extraídas del estudio.

En primer lugar, la base de datos de perfusión es mucho más clara en la diferenciación, dispone de unos biomarcadores muy apropiados para la distinción entre tejido tumoral y sano. Es algo que se va intuyendo desde el principio del proyecto cuando se realiza el análisis exploratorio. La base de datos de difusión no llega a los niveles de capacidad de predicción de perfusión, pero un 72% es un resultado positivo e interesante, ya que puede servir como modelo complementario cuando se tienen dudas o se trata con lesiones donde la perfusión te muestra vascularización clara pero no tiene por qué ser lesión tumoral. De esta manera podríamos evitar falsos positivos en la perfusión proporcionando un modelo predictivo con el DTI.

Por otro lado, se cree conveniente destacar dos fases del proyecto, que han resultado cruciales para la comprensión del proyecto:

- La reducción de dimensionalidad como método de análisis exploratorio, ya que proporciona información importante que ayuda a la toma de decisiones en el resto del estudio.
- La capacidad de modelos simples y rápidos como son los modelos basados en árboles, debido a que computacionalmente funcionan bien con grandes datasets, son sencillos de comprender y se pueden explicar muy bien el por qué de sus resultados.

Al haber sido los modelos basados en árboles los más acertados, se ha querido conocer qué biomarcadores han sido más importantes en dichos modelos.

In [10]:
```
importancia_predictores = pd.DataFrame(
                            {'predictor': X_train.columns,
                             'importancia': clf.feature_importances_}
                            )
print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores.sort_values('importancia', ascending=False)
```

```
Importancia de los predictores en el modelo
-------------------------------------------
```

Out[10]:

	predictor	importancia
2	L1	0.224992
0	L1-L3	0.169897
19	Bio15	0.125990
4	L3	0.090024
21	Bio17	0.076568
14	Bio10	0.044115
15	Bio11	0.038150
10	Bio6	0.035871
3	L2	0.034871
16	Bio12	0.034630
23	Bio19	0.034627
24	Bio20	0.029117
1	1-L3/L1	0.019006
20	Bio16	0.012833
5	Bio1	0.012000

Figura 63. Importancia biomarcadores en difusión

En el caso de la base de datos de difusión, el biomarcador más determinante en el modelo con mejores resultados ha sido λ_1, recordemos que este biomarcador representa la dirección de mayor difusión, la orientación del tejido donde las moléculas de agua se mueven con mayor libertad, por tanto, este factor sería aquel que se debería tener más en cuenta a la hora de intentar predecir el tipo de tejido mediante este modelo. El biomarcador con más importancia después de este es el correspondiente a la resta de ese biomarcador con λ_3, que representa el autovalor más pequeño de los vectores ortogonales a la dirección de máxima difusión, siendo un parámetro clave para diferenciar tensores elipsoidales (sanos) de tensores esféricos (sospecha de tumor).

Cabe destacar que biomarcadores como la anisotropía relativa escalada o el determinante $\lambda_1\lambda_2\lambda_3$, ()no son relevantes en este modelo.

Son resultados similares a los concluidos con el método de componentes principales (Figura 41), donde veíamos que λ_1 y λ_1-λ_3 eran más determinantes al distinguir entre tipos de tejido.

In [34]:
```
importancia_predictores = pd.DataFrame(
                            {'predictor': X_train.columns,
                             'importancia': clf.feature_importances_}
                            )
print("Importancia de los predictores en el modelo")
print("-------------------------------------------")
importancia_predictores.sort_values('importancia', ascending=False)
```

```
Importancia de los predictores en el modelo
-------------------------------------------
```

Out[34]:

	predictor	importancia
2	C3	0.233345
1	C2	0.166870
6	C1C3_C2	0.117961
0	C1	0.083808
5	C1_C2	0.034330
3	RSS	NaN
4	C3_C2	NaN

Figura 64. Importancia biomarcadores en perfusión

En cuanto al mejor modelo aplicado a la base de datos de perfusión, el biomarcador correspondiente al tipo VT (C_3) el más determinante. Este biomarcador representa signos de inicio de procesos tumorales, neovasculización y angiogénesis. Seguido de dicho biomarcador, el tipo CMA (C_2) es el segundo más importante, que se corresponde con un ruido estructural (artefacto) provocado por la llegada del contraste al tejido. Esto es debido a que en el tipo CMA, los tejidos vascularizados (potencialmente tumorales) tienen unos scores cercanos a 0, mientras que el resto del tejido mantienen valores constantes por lo que es un parámetro clave en la detección de lesiones. Si nos apoyamos en este modelo para la predicción del tipo de tejido en cuestión, estos serían los biomarcadores más determinantes.

Es importante mencionar que biomarcadores como el correspondiente a la resta entre los dos de más importancia (C_3-C_2), no son relevantes en este modelo. Esto es debido a que esa información ya está presente en los biomarcadores individuales (C_2 y C_3) y, por lo tanto, al tratarse de combinaciones lineales de los anteriores no son necesarios para mejorar la capacidad de predicción.

Estos resultados se corresponden a lo analizado anteriormente con el método de componentes principales, (Figura 44), el biomarcador C_3 también era el más determinante a la hora de distinguir entre tipos de tejidos.

Como futuros avances, estudiar las bases de datos en conjunto sería interesante aprovechando los biomarcadores de todas las secuencias, para ello se debe realizar previamente un alineamiento automático y corrección de estas, algoritmo que se tiene que desarrollar ya que no es trivial. Se pretende añadir nuevos biomarcadores interpretables en ambas técnicas de análisis de imagen de RM con nuevos modelos a desarrollar. Además, el análisis de una nueva técnica de perfusión llamada *ultrafast* (UF) sería una buena opción para continuar el trabajo.

En cuanto al despliegue del modelo, se cree conveniente la posibilidad de realización de una aplicación, que sirva de apoyo al diagnóstico dado por el médico cuando tiene que tratar con un paciente que sufra, o parezca sufrir esta enfermedad. En aquellos casos en los que el médico tenga muy claro el diagnóstico, será posible indicárselo al modelo, para que pueda reentrenarse y mejorar sus capacidades predictivas.

```
In [2]: xl = pd.read_excel('XL.xlsx')
        var = pd.read_excel('var_DTI.xlsx')
        xh = pd.read_excel('XH.xlsx')
```

Figura 65. Carga de bases de datos

```
In [73]: variab.append('Paciente')
         variab

Out[73]: ['L1-L3',
          '1-L3/L1',
          'L1',
          'L2',
          'L3',
          'Bio1',
          'Bio2',
          'Bio3',
          'Bio4',
          'Bio5',
          'Bio6',
          'Bio7',
          'Bio8',
          'Bio9',
          'Bio10',
          'Bio11',
          'Bio12',
          'Bio13',
          'Bio14',
          'Bio15',
          'Bio16',
          'Bio17',
          'Bio18',
          'Bio19',
          'Bio20',
          'Paciente']

In [74]: xl.columns = variab
         xh.columns = variab
```

Figura 66. Asignación nombre de variables en difusión

```
In [77]: xh = xh.assign(Sano=1)

In [78]: xl = xl.assign(Sano=0)

In [80]: from pandas import concat
         data = concat([xh,xl])
         data
```

Out[80]:

	L1-L3	1-L3/L1	L1	L2	L3	Bio1	Bio2	Bio3	Bio4	Bio5	...	Bio13	Bio14	E
0	0.001273	1.595930	0.000798	0.000231	-0.000475	0.000553	-3.045519e-07	-8.759618e-11	9.153102e-07	0.000184	...	1.995944	14.958140	1.00
1	0.000548	6.768052	0.000081	-0.000083	-0.000467	-0.000470	-5.695144e-09	3.152882e-12	2.318954e-07	-0.000157	...	1.038019	1.822136	1.00
2	0.000623	0.928957	0.000671	0.000398	0.000048	0.001117	3.180362e-07	1.272676e-11	6.106928e-07	0.000372	...	0.484492	0.753166	0.12
3	0.001626	1.157109	0.001405	0.000784	-0.000221	0.001968	6.180884e-07	-2.430648e-10	2.636913e-06	0.000656	...	0.721972	1.860993	0.30
4	0.001289	1.017770	0.001267	0.000054	-0.000023	0.001298	3.848923e-08	-1.535467e-12	1.608000e-06	0.000433	...	0.965128	1.018955	0.73
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5371	0.001011	0.645881	0.001565	0.001275	0.000554	0.003394	3.569032e-06	1.105633e-09	4.381631e-06	0.001131	...	0.265594	0.236492	0.03
5372	0.001127	0.802068	0.001406	0.000879	0.000278	0.002563	1.871204e-06	3.437531e-10	2.825804e-06	0.000854	...	0.381230	0.448634	0.07
5373	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
5374	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
5375	5368.000000	5308.000000	5308.000000	5199.000000	4911.000000	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

11476 rows × 27 columns

Figura 67. Unión de bases de datos tras añadir la variable "*dummy*" objetivo en difusión

```
In [81]: DTI = data[(data.L3 >= 0)]

In [82]: DTI
```

Out[82]:

	L1-L3	1-L3/L1	L1	L2	L3	Bio1	Bio2	Bio3	Bio4	Bio5	...	Bio13	Bio14	Bio15
2	0.000623	0.928957	0.000671	0.000398	0.000048	0.001117	3.180362e-07	1.272676e-11	6.106928e-07	0.000372	...	0.484492	0.753166	0.125204
7	0.001425	0.828675	0.001720	0.000488	0.000295	0.002503	1.490122e-06	2.473707e-10	3.282844e-06	0.000834	...	0.535010	0.573883	0.155155
8	0.001211	0.695278	0.001742	0.001304	0.000531	0.003578	3.890025e-06	1.206502e-09	5.018692e-06	0.001193	...	0.296961	0.288556	0.045110
9	0.002079	0.914340	0.002274	0.000897	0.000195	0.003366	2.656756e-06	3.972336e-10	6.014137e-06	0.001122	...	0.544415	0.718680	0.161184
10	0.001055	0.779737	0.001353	0.000634	0.000298	0.002286	1.451184e-06	2.559587e-10	2.323171e-06	0.000762	...	0.408498	0.421455	0.087241
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5369	0.000431	0.260884	0.001653	0.001599	0.001222	0.004473	6.615468e-06	3.228725e-09	6.780894e-06	0.001491	...	0.090920	0.026212	0.004142
5370	0.000679	0.402885	0.001685	0.001498	0.001006	0.004190	5.728128e-06	2.540643e-09	6.097031e-06	0.001397	...	0.144969	0.067240	0.010564
5371	0.001011	0.645881	0.001565	0.001275	0.000554	0.003394	3.569032e-06	1.105633e-09	4.381631e-06	0.001131	...	0.265594	0.236492	0.035915
5372	0.001127	0.802068	0.001406	0.000879	0.000278	0.002563	1.871204e-06	3.437531e-10	2.825804e-06	0.000854	...	0.381230	0.448634	0.075520
5375	5368.000000	5308.000000	5308.000000	5199.000000	4911.000000	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN

8222 rows × 27 columns

Figura 68. Eliminación de observaciones de difusión con el biomarcador λ3 menor que 0 para que tengan sentido físico

```
In [232]: DTI.describe()
```

Out[232]:

	L1-L3	1-L3/L1	L1	L2	L3	Bio1	Bio2	Bio3	Bio4	Bio5	...	Bio13	
count	8222.000000	8222.000000	8222.000000	8222.000000	8222.000000	8220.000000	8220.000000	8.220000e+03	8220.000000	8220.000000	...	8220.000000	8
mean	1.382388	1.820574	1.335745	1.212649	0.987412	0.003455	0.000005	2.162383e-09	0.000005	0.001152	...	0.212483	
std	88.716109	85.578691	85.585939	77.748197	64.645888	0.001471	0.000004	2.782308e-09	0.000004	0.000490	...	0.138696	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	...	0.000000	
25%	0.000418	0.306641	0.001114	0.000777	0.000417	0.002377	0.000002	3.560355e-10	0.000002	0.000792	...	0.107306	
50%	0.000649	0.456584	0.001481	0.001100	0.000748	0.003318	0.000003	1.147391e-09	0.000004	0.001106	...	0.175867	
75%	0.000964	0.653199	0.001878	0.001488	0.001124	0.004442	0.000006	2.935263e-09	0.000007	0.001481	...	0.289242	
max	5992.000000	5662.000000	5662.000000	5199.000000	4911.000000	0.010106	0.000032	3.157295e-08	0.000039	0.003369	...	0.931321	

8 rows × 27 columns

Figura 69. Análisis estadístico base de datos difusión

```
In [5]: var = pd.read_excel('var.xlsx')

In [6]: var
```

Out[6]:

	0
0	C1
1	C2
2	C3
3	RSS
4	C3_C2
5	C1_C2
6	C1C3_C2
7	C1
8	C2
9	C3
10	RSS
11	C3_C2
12	C1_C2
13	C1C3_C2

Figura 70. Asignación nombre de variables en perfusión

```
In [45]: xl = xl.assign(Sano=0)

In [43]: xh = xh.assign(Sano=1)

In [47]: from pandas import concat
         data = concat([xh,xl])
         data
```

Out[47]:

	C1	C2	C3	RSS	C3_C2	C1_C2	C1C3_C2	Paciente	Sano
0	359.134206	202.266992	154.802201	349.082202	0.000000	156.867215	54.701212	1	1
1	362.748346	184.425396	132.843883	290.497417	0.000000	178.322950	63.370719	1	1
2	338.771623	158.453906	132.980731	21.224631	0.000000	180.317717	77.422271	1	1
3	312.529493	164.226614	157.393083	60.262141	0.000000	148.302880	70.734675	1	1
4	312.282653	177.782607	156.181376	169.098622	0.000000	134.500046	56.449407	1	1
...	...	...	...	...	...	...	...	...	...
3493	412.120807	55.294206	268.692148	455.142658	213.397942	356.826601	285.112271	24	0
3494	305.532491	103.760141	143.154142	523.246687	39.394001	201.772350	120.583175	24	0
3495	335.098320	109.224360	145.229861	1661.911072	36.005501	225.873961	130.939731	24	0
3496	374.654512	126.033480	158.113020	1483.942050	32.079541	248.621032	140.350287	24	0
3497	282.977570	119.260323	84.299174	1429.872148	0.000000	163.717246	64.378049	24	0

7576 rows × 9 columns

Figura 71. Unión de bases de datos tras añadir la variable "*dummy*" objetivo en perfusión

```
In [6]: PERF.describe()
```

Out[6]:

	C1	C2	C3	RSS	C3_C2	C1_C2	C1C3_C2	Paciente	Sano
count	7576.000000	7576.000000	7576.000000	7576.000000	7576.000000	7576.000000	7576.000000	7576.00000	7576.000000
mean	234.986262	99.805182	138.414499	453.652352	53.544863	135.424669	87.497086	11.94179	0.538279
std	85.301000	64.657858	59.232703	681.896931	67.844517	79.259090	72.817745	6.39027	0.498565
min	19.710911	0.000000	0.000000	0.382078	0.000000	0.000000	0.000000	1.00000	0.000000
25%	170.177934	44.119330	99.456084	103.863688	0.000000	77.235620	30.034738	7.00000	0.000000
50%	237.748158	70.137903	137.783324	230.571250	10.598480	128.829274	58.258638	11.00000	1.000000
75%	305.343645	165.573022	172.694176	504.497060	106.071797	179.228976	143.049620	19.00000	1.000000
max	488.915037	277.007884	397.577457	10058.936390	335.039743	432.957750	337.275745	24.00000	1.000000

Figura 72. Análisis estadístico base de datos de perfusión

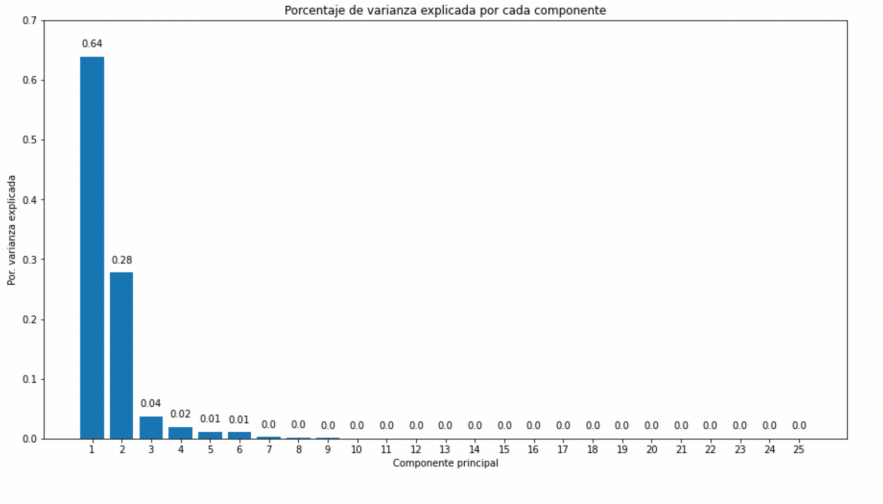


Figura 73. Porcentaje de varianza explicada por cada componente principal difusión

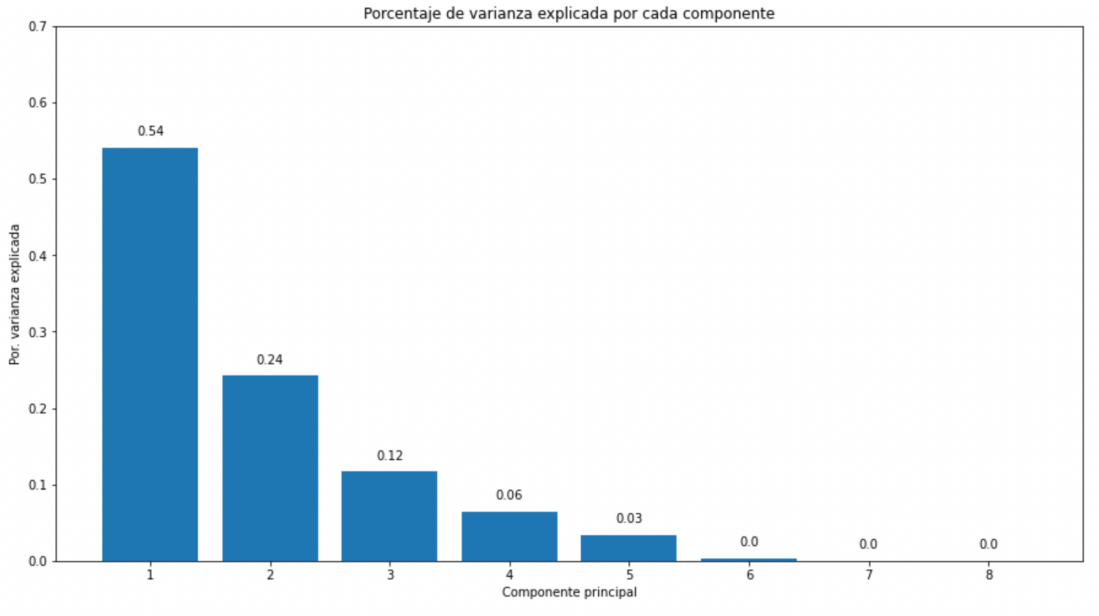


Figura 74. Porcentaje de varianza explicada por cada componente principal perfusión

```
In [233]: print('--------------------------------------------------')
          print('Porcentaje de varianza explicada por cada componente')
          print('--------------------------------------------------')
          print(modelo_pca.explained_variance_ratio_)

          fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15, 8))
          ax.bar(
              x      = np.arange(modelo_pca.n_components_) + 1,
              height = modelo_pca.explained_variance_ratio_
          )

          for x, y in zip(np.arange(len(datos.columns)) + 1, modelo_pca.explained_variance_ratio_):
              label = round(y, 2)
              ax.annotate(
                  label,
                  (x,y),
                  textcoords="offset points",
                  xytext=(0,10),
                  ha='center'
              )

          ax.set_xticks(np.arange(modelo_pca.n_components_) + 1)
          ax.set_ylim(0, 0.7)
          ax.set_title('Porcentaje de varianza explicada por cada componente')
          ax.set_xlabel('Componente principal')
          ax.set_ylabel('Por. varianza explicada');
```

Figura 75. Código para el porcentaje de varianza explicada

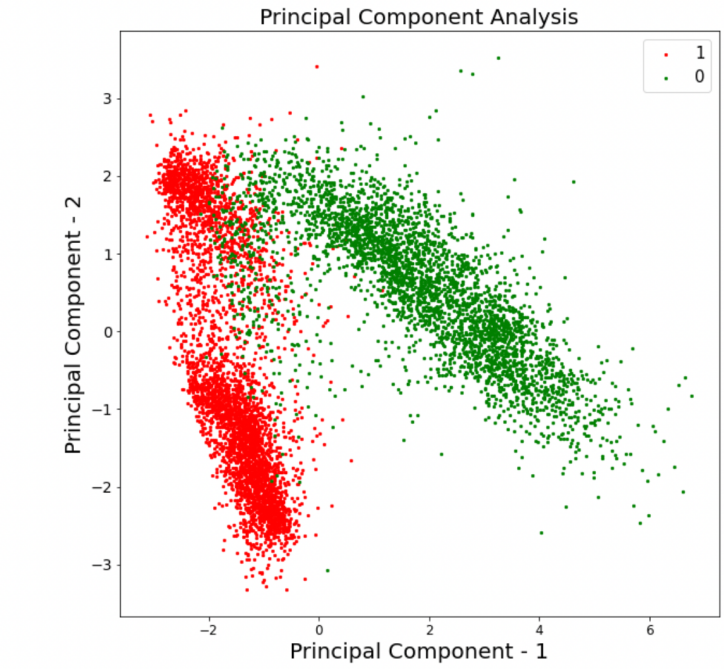


Figura 76. Píxeles correspondientes a tejido sano y tumoral en las dos primeras componentes principales de la base de datos de perfusión. Visualización conjunta

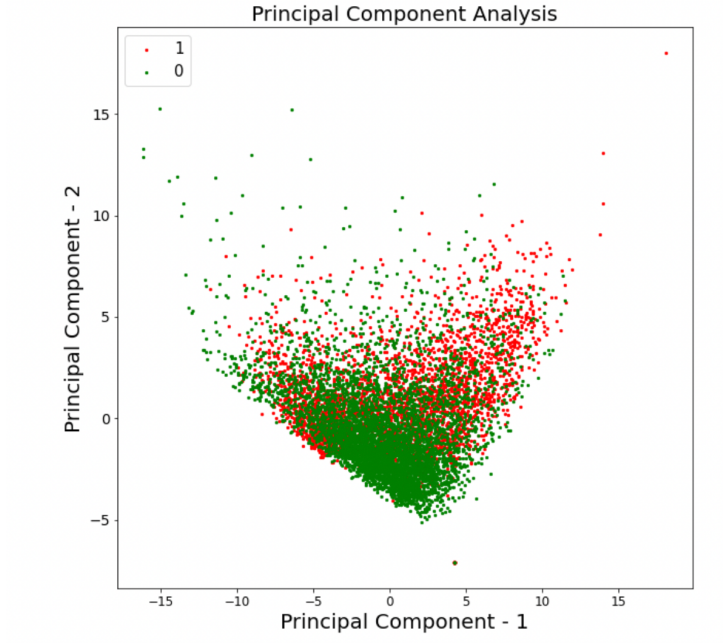


Figura 77. Píxeles correspondientes a tejido sano y tumoral en las dos primeras componentes principales de la base de datos de perfusión. Visualización conjunta

```
In [40]: plt.figure(figsize=(20,10))
         plt.subplot(1, 2, 1)
         plt.xticks(fontsize=12)
         plt.yticks(fontsize=14)
         plt.xlabel('Principal Component - 1',fontsize=20)
         plt.ylabel('Principal Component - 2',fontsize=20)
         plt.title("Principal Component Analysis-SANOS",fontsize=20)
         targets = [1, 0]
         colors = ['g', 'w']
         for target, color in zip(targets,colors):
             indicesToKeep = datos['Sano'] == target
             plt.scatter(principal_Df.loc[indicesToKeep, 'principal component 1']
                         , principal_Df.loc[indicesToKeep, 'principal component 2'], c = color, s = 5)

         plt.legend(targets,prop={'size': 15})


         plt.subplot(1, 2, 2)
         plt.xticks(fontsize=12)
         plt.yticks(fontsize=14)
         plt.xlabel('Principal Component - 1',fontsize=20)
         plt.ylabel('Principal Component - 2',fontsize=20)
         plt.title("Principal Component Analysis-LESIÓN",fontsize=20)
         targets = [1, 0]
         colors = ['w', 'r']
         for target, color in zip(targets,colors):
             indicesToKeep = datos['Sano'] == target
             plt.scatter(principal_Df.loc[indicesToKeep, 'principal component 1']
                         , principal_Df.loc[indicesToKeep, 'principal component 2'], c = color, s = 5)

         plt.legend(targets,prop={'size': 15})


         plt.show()
```

Figura 78. Código para las visualizaciones los píxeles tumorales y sanos en las dos primeras componentes.

```
In [22]:
         def myplot(score,coeff,labels=None):
             xs = score[:,0]
             ys = score[:,1]
             n = coeff.shape[0]
             scalex = 1.0/(xs.max() - xs.min())
             scaley = 1.0/(ys.max() - ys.min())
             plt.scatter(xs * scalex,ys * scaley, c = y)
             for i in range(n):
                 plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
                 if labels is None:
                     plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, "Var"+str(i+1), color = 'g', ha = 'center', va = 'center'
                 else:
                     plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], color = 'g', ha = 'center', va = 'center')
             plt.xlim(-1,1)
             plt.ylim(-1,1)
             plt.xlabel("PC{}".format(1))
             plt.ylabel("PC{}".format(2))
             plt.grid()

         #Call the function. Use only the 2 PCs.
         plt.figure(figsize=(20,20))
         myplot(x_new[:,0:2],np.transpose(pca.components_[0:2, :]))

         plt.show()
```

Figura 79. Código para visualización de loadings

```
In [27]: tuned_parameters = [
             {"kernel": ["rbf"], "gamma": ["scale","auto"], "C": [1, 10, 50, 100]},
             {"kernel": ["linear"], "C": [1, 10, 50, 100]},
             {"kernel": ["poly"], "degree": [2,3,5,8,10], "C": [1, 10, 50, 100]},
         ]

         gridsearch = GridSearchCV(SVC(), tuned_parameters, scoring = "accuracy", cv=5)
         gridsearch.fit(X_train_test, y_train_test)
```

Figura 80. Código para Grid search SVM

In [64]:
```
n_est = [5,10,50,100]
d = {}
for est in tqdm((n_est)):
  d[f'r_mean_{est}'] = {}
  d[f'r_all_{est}'] = {}
  for c in range(20):
    temp_mt = []
    temp_mr = []
    for _ in range(30):
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, stratify
      #clf = DecisionTreeClassifier(criterion='gini', max_depth=maxDepth_dic.get(c), mi
      clf = BaggingClassifier(DecisionTreeClassifier(criterion='gini', max_depth=None,
                              n_estimators=est)
      clf = clf.fit(X_train, y_train)
      y_pred = clf.predict(X_test)
      y_pred_tr = clf.predict(X_train)
      temp_mt.append(round(accuracy_score(y_test, y_pred),3))
      temp_mr.append(accuracy_score(y_train, y_pred_tr))
    d[f'r_all_{est}'][c] = [temp_mr, temp_mt]
    d[f'r_mean_{est}'][c] = [np.array(temp_mr).mean(), np.array(temp_mt).mean()]
d_bagging = d.copy()
```

```
100%|██████████| 4/4 [36:10<00:00, 542.51s/it]
```

In [65]:
```
fig, ax = plt.subplots(2,2,figsize=(13,9))
for est, n, n1 in zip(n_est, [0,0,1,1], [0,1,0,1]):
  plt.sca(ax[n,n1])
  plt.plot(list(range(1,21)), [_[1] for _ in d_bagging[f'r_all_{est}'].values()], lines
  plt.plot(list(range(1,21)), [_[0] for _ in d_bagging[f'r_all_{est}'].values()], lines
  plt.plot(list(range(1,21)), [_[1] for _ in d_bagging[f'r_mean_{est}'].values()], labe
  plt.plot(list(range(1,21)), [_[0] for _ in d_bagging[f'r_mean_{est}'].values()], labe
  plt.locator_params(axis='x', nbins=11)
  plt.ylim([0.2,1.1])
  plt.xlim([0.75,20.25])
  plt.xlabel('Model complexity', fontsize='large')
  plt.ylabel('Accuracy', fontsize='large')
  plt.legend(loc='lower right', fontsize='large')
  plt.title(f'Number of estimators=  {est}')
plt.tight_layout()
plt.show()
```

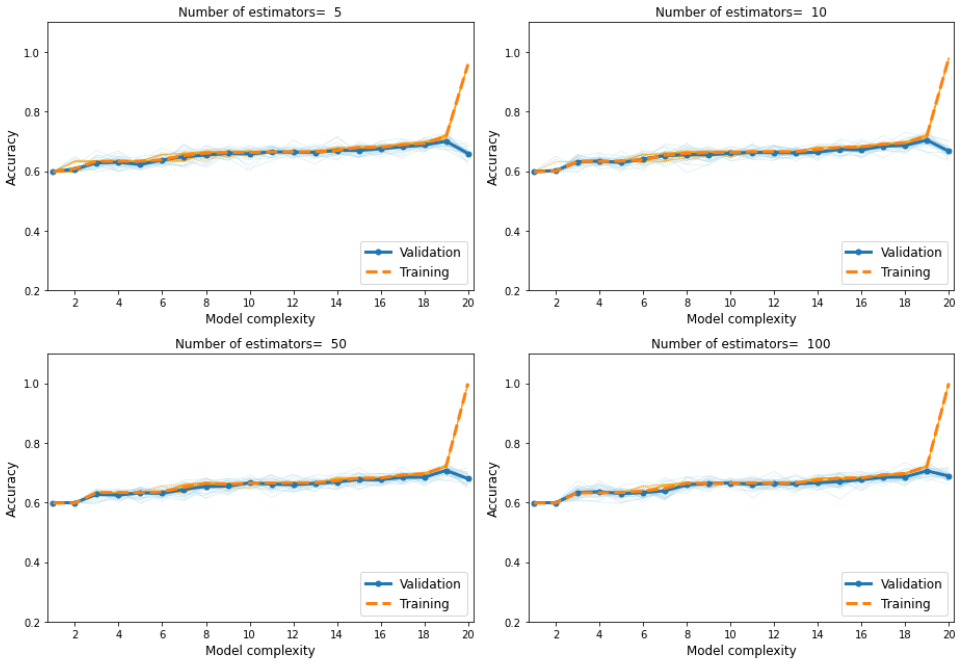


Figura 81. Ejemplo visualización del efecto de la generalización en modelos basados en árboles. Se puede ver como se sobre entrena al final partir del valor 18 del min_leaf_values explicado anteriormente

In [55]:

```
r_mean_box = {}
r_all_box = {}
for c in tqdm([1,2,5,7,10,20,50]):
  temp_mt = []
  temp_mr = []
  for _ in range(30):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, stratify=y)
    clf = AdaBoostClassifier(DecisionTreeClassifier(criterion='gini', max_depth=None, min_samples_split=20, mi
                             n_estimators=c)
    clf = clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    y_pred_tr = clf.predict(X_train)
    temp_mt.append(round(accuracy_score(y_test, y_pred),3))
    temp_mr.append(accuracy_score(y_train, y_pred_tr))
  r_all_box[c] = [temp_mr, temp_mt]
  r_mean_box[c] = [np.array(temp_mr).mean(), np.array(temp_mt).mean()]
```

```
100%|██████████████████████████████████| 7/7 [18:12<00:00, 156.07s/it]
```

In [56]:

```
fig, ax = plt.subplots(figsize=(10,5))
plt.boxplot([_[1] for _ in r_all_box.values()], labels=list(r_all_box.keys()), showmeans=True)
plt.grid(color='grey', linestyle='dashed', axis='y', linewidth=0.15)
plt.xlabel('Number of trees', fontsize='large')
plt.ylabel('Accuracy', fontsize='large')
plt.show()
```

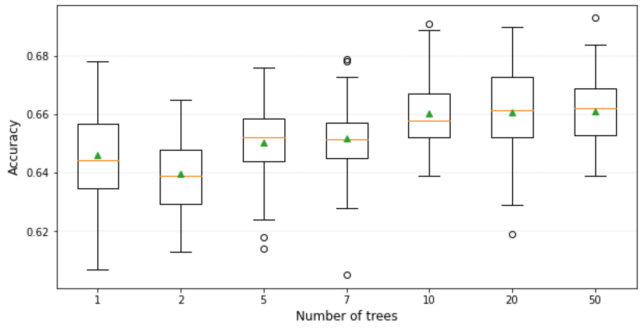


Figura 82. Efecto de la elección del número de árboles

In [26]:
```
params = {'min_samples_split': [2,4,6,8],
              'min_samples_leaf': [400],
              'max_features': [None],
              'n_estimators': [5]}
advanced_search = { 'min_samples_leaf':[20,'int']}

create_combination_df()
for loop in range(10):
  print(f'loop {loop}')
  launch_search(crossval_repetitions=8+(loop*2))
  if previous_accuracy_higher_than_new(params_df.loc[temp_index,'mean_accuracy'].max())
    break
  else:
    update_parameters(percentage_update=20-(loop*2))
```

```
# of initial combinations:  4
loop 0
# of crossval_repetitions:  8
{'min_samples_split': 2, 'min_samples_leaf': 400, 'max_features': None, 'n_estimators'
: 5, 'mean_accuracy': 0.6958029197080293}
{'min_samples_split': 4, 'min_samples_leaf': 400, 'max_features': None, 'n_estimators'
: 5, 'mean_accuracy': 0.6947232360097323}
{'min_samples_split': 6, 'min_samples_leaf': 400, 'max_features': None, 'n_estimators'
: 5, 'mean_accuracy': 0.694875304136253}
{'min_samples_split': 8, 'min_samples_leaf': 400, 'max_features': None, 'n_estimators'
: 5, 'mean_accuracy': 0.6940085158150853}
# of new combinations:  2
loop 1
# of crossval_repetitions:  10
{'min_samples_split': 2, 'min_samples_leaf': 320, 'max_features': None, 'n_estimators'
: 5, 'mean_accuracy': 0.692712895377129}
{'min_samples_split': 2, 'min_samples_leaf': 480, 'max_features': None, 'n_estimators'
: 5, 'mean_accuracy': 0.6948418491484184}
Stopping search because no improvements
previous_best =  0.6958029197080293
new_best =  0.6948418491484184
```

In [27]:
```
best_performers_df = params_df.sort_values(by=['mean_accuracy','#crossval_repetitions']
best_performers_df
```

Out[27]:

	index	min_samples_split	min_samples_leaf	max_features	n_estimators	mean_accuracy	#crossva
0	0	2	400	None	5	0.695803	
1	2	6	400	None	5	0.694875	
2	5	2	480	None	5	0.694842	
3	1	4	400	None	5	0.694723	
4	3	8	400	None	5	0.694009	
5	4	2	320	None	5	0.692713	

Figura 83. Código para Grid search para AdaBoost

In [20]:

```
# Function to create model, required for KerasClassifier
def create_model(init_mode='uniform'):
    # create model
    model = Sequential()
    model.add(Dense(12, input_dim=7, kernel_initializer=init_mode, activation='relu'))
    model.add(Dense(1, kernel_initializer=init_mode, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load dataset

# create model
model = KerasClassifier(build_fn=create_model, epochs=100, batch_size=10, verbose=0)
# define the grid search parameters
init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_unif
param_grid = dict(init_mode=init_mode)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X, Y)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle input:
<class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to find data adapter that can handle input:
<class 'pandas.core.frame.DataFrame'>, <class 'NoneType'>

2022-05-15 12:24:03.524418: I tensorflow/core/platform/cpu_feature_guard.cc:145] This TensorFlow binary is optimiz
ed with Intel(R) MKL-DNN to use the following CPU instructions in performance critical operations:  SSE4.1 SSE4.2
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2022-05-15 12:24:03.525183: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with de
fault inter op setting: 8. Tune using inter_op_parallelism_threads for best performance.

Best: 0.864310 using {'init_mode': 'he_uniform'}
0.863651 (0.031515) with: {'init_mode': 'uniform'}
0.863386 (0.027309) with: {'init_mode': 'lecun_uniform'}
0.858239 (0.028551) with: {'init_mode': 'normal'}
0.128449 (0.181654) with: {'init_mode': 'zero'}
0.857052 (0.030630) with: {'init_mode': 'glorot_normal'}
0.862991 (0.024501) with: {'init_mode': 'glorot_uniform'}
0.859295 (0.020831) with: {'init_mode': 'he_normal'}
0.864310 (0.030079) with: {'init_mode': 'he_uniform'}
```

Figura 84. Código para Grid search red neuronal. En concreto para el parámetro 'init_mode'

In [42]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import matthews_corrcoef
print('roc train',roc_auc_score(y_train, y_pred_tr))
print('roc test',roc_auc_score(y_test, y_pred_ts))
print('mcc train', matthews_corrcoef(y_train, y_pred_tr))
print('mcc train', matthews_corrcoef(y_test, y_pred_ts))
```

```
roc train 0.8925709502854631
roc test 0.9001680672268907
mcc train 0.7849509712984066
mcc train 0.8008494482924053
```

Figura 85. Código para métricas adicionales

In [52]:
```
fig, ax = plt.subplots(figsize=(15, 6))

plt.subplot(1, 2, 1)


plt.plot(variables,train_ensemble,  label='Train Ensemble',color='red')
plt.plot(variables,train_svm, label='Train SVM',color='blue')
plt.plot(variables,train_nn,label='Train NN', color= 'green')
plt.plot(variables, train_pls,  label='Train PLS', color = 'orange')

leg = plt.legend()
plt.title("Difusión TRAIN",fontsize=15)
plt.xlabel("Métricas",fontsize=13)

plt.subplot(1, 2, 2)

plt.plot(variables,test_ensemble, label='Test Ensemble',color='red')
plt.plot(variables,test_svm,label='Test SVM',color='blue')
plt.plot(variables,test_nn, label='Test NN', color='green')
plt.plot(variables, test_pls, label='Test PLS', color = 'orange')
leg = plt.legend()
plt.title("Difusión TEST",fontsize=15)
plt.xlabel("Métricas",fontsize=13)
plt.show()
```

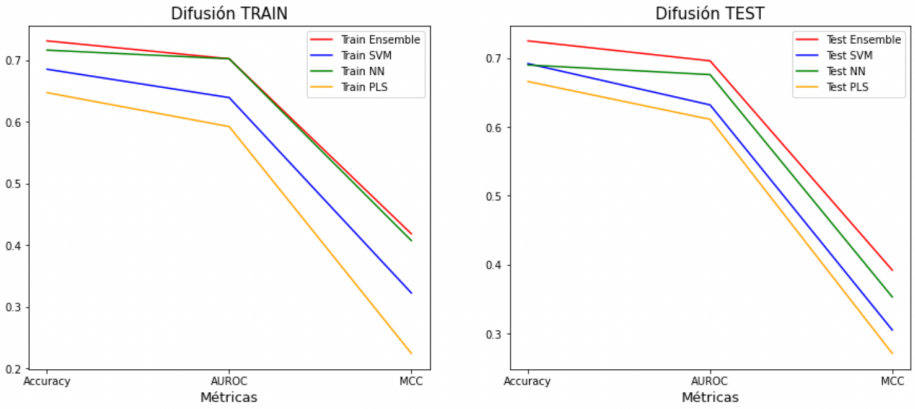


Figura 86. Código para visualización de resultados.

REFERENCIAS

[1] Adam Bohr and Kaveh Memarzadeh, *Inteligencia Artificial en el ámbito de la salud*. 2020.

[2] Cardoso. F, Kyriakides. S, and Ohno. S, *International consensus guidelines for advanced breast cancer*. 2019.

[3] J. Costa Subias, "Resonancia magnética dirigida a técnicos superiores en imagen para el diagnóstico." doi: 10.1016/B978-84-9022-745-9/00020-7.

[4] E. O. Stejskal and J. E. Tanner, "Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient," *The Journal of Chemical Physics*, vol. 42, no. 1, p. 288, Jul. 2004, doi: 10.1063/1.1695690.

[5] D. le Bihan *et al.*, "Diffusion tensor imaging: Concepts and applications," *Journal of Magnetic Resonance Imaging*, vol. 13, no. 4, pp. 534–546, Apr. 2001, doi: 10.1002/JMRI.1076.

[6] P. J. Basser and D. K. Jones, "Diffusion-tensor MRI: theory, experimental design and data analysis – a technical review," *NMR in Biomedicine*, vol. 15, no. 7–8, pp. 456–467, Nov. 2002, doi: 10.1002/NBM.783.

[7] P. B. Kingsley, "Introduction to diffusion tensor imaging mathematics: Part I. Tensors, rotations, and eigenvectors," *Concepts in Magnetic Resonance Part A*, vol. 28A, no. 2, pp. 101–122, Mar. 2006, doi: 10.1002/CMR.A.20048.

[8] R. Tauler, A. Smilde, and B. Kowalski, "Selectivity, local rank, three-way data analysis and ambiguity in multivariate curve resolution," *Journal of Chemometrics*, vol. 9, no. 1, pp. 31–58, Jan. 1995, doi: 10.1002/CEM.1180090105.

[9] R. Tauler, "Multivariate curve resolution applied to second order data," *Chemometrics and Intelligent Laboratory Systems*, vol. 30, no. 1, pp. 133–146, Nov. 1995, doi: 10.1016/0169-7439(95)00047-X.

[10] J.M. Prats-Montabán, E. Aguado-Sarrió, and A. Ferrer, "Multivariate curve resolution for magnetic resonance image analysis: applications in prostate cancer biomarkers development," *"Resolving Spectral Mixtures, with application from ultrafast spectroscopy to super-resolution imaging", Data Handling in Science and Technology*, 2022.

[11] J. M. Prats-Montalbán, R. Sanz-Requena, L. Martí-Bonmatí, and A. Ferrer, "Prostate functional magnetic resonance image analysis using multivariate curve resolution methods," *Journal of Chemometrics*, vol. 28, no. 8, pp. 672–680, 2014, doi: 10.1002/CEM.2585.

[12] "Machine learning, explained | MIT Sloan." https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained (accessed Jun. 18, 2022).

[13] F. E. Harrell , *Regression Modeling Strategies*, 2nd ed. Cham: New York; Springer, 2015. doi: 10.1007/978-3-319-19425-7.

[14] J. Tolles and W. J. Meurer, "Logistic Regression Relating Patient Characteristics to Outcomes," *JAMA*, vol. 316, no. 5, pp. 533–4, Aug. 2016, doi: 10.1001/jama.2016.7653.

[15] "Support Vector Machine (SVM) - MATLAB & Simulink." https://es.mathworks.com/discovery/support-vector-machine.html (accessed Jun. 18, 2022).

[16] "Arboles de decision, Random Forest, Gradient Boosting y C5.0." https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting#Introducci%C3%B3n (accessed Jun. 18, 2022).

[17] "Clustering in Machine Learning - GeeksforGeeks." https://www.geeksforgeeks.org/clustering-in-machine-learning/ (accessed Jun. 18, 2022).

[18] S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis. Chemometrics and Intelligent Laboratory Systems," 1987.

[19] "Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE." https://www.cienciadedatos.net/documentos/35_principal_component_analysis (accessed Jun. 18, 2022).

[20] "A Complete Guide to Principal Component Analysis – PCA in Machine Learning – Data Science Duniya." https://ashutoshtripathi.com/2019/07/11/a-complete-guide-to-principal-component-analysis-pca-in-machine-learning/ (accessed Jun. 18, 2022).

[21] "What are Neural Networks? | IBM." https://www.ibm.com/cloud/learn/neural-networks (accessed Jun. 18, 2022).

[22] "Qué son las redes neuronales y sus funciones | ATRIA Innovation." https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/ (accessed Jun. 18, 2022).

[23] "Steps to Complete a Machine Learning Project - Analytics Vidhya." https://www.analyticsvidhya.com/blog/2021/04/steps-to-complete-a-machine-learning-project/ (accessed Jun. 18, 2022).

[24] "Las 7 Fases del Proceso de Machine Learning - IArtificial.net." https://www.iartificial.net/fases-del-proceso-de-machine-learning/ (accessed Jun. 18, 2022).

[25] "Etapas del proceso de Machine Learning - Agencia B12." https://agenciab12.com/noticia/etapas-proceso-machine-learning (accessed Jun. 18, 2022).

[26] "Classification: ROC Curve and AUC | Machine Learning Crash Course | Google Developers." https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (accessed Jun. 27, 2022).

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**		X		
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

En 2015 todos los Estados Miembros de las Naciones Unidas aprobaron 17 objetivos, los cuales son parte de la Agenda 2030. Estos son los denominados objetivos de desarrollo sostenible (ODS), cuyo objetivo es acabar con la pobreza, proteger el planeta y mejorar tanto las vidas como las perspectivas de las personas de todo el mundo. Con este trabajo se quiere contribuir a esta gran iniciativa, por ello se centra en dos de los objetivos.

Salud y bienestar

El tercer objetivo de desarrollo sostenible es el relacionado con la salud y el bienestar, concretamente en garantizar una vida sana y promover el bienestar para todas las edades.

En este proyecto se contribuye a ello, al aplicar una de las herramientas más potentes de hoy en día, la inteligencia artificial, con el objetivo de reducir lo máximo posible el número de fallecimientos a causa del cáncer de mama.

Como se ha explicado anteriormente, la tasa de mortalidad del cáncer de mama es mucho más alta a medida que progresa a estadios más avanzados, si con la inteligencia artificial se consigue una detección más acertada y más temprano, tendrá unas consecuencias realmente positivas en lo que es la mortalidad provocada con la enfermedad.

Además, gracias al avance y mejora continua de los algoritmos, se podrá elaborar un diagnóstico personalizado a cada paciente, de forma que el cliente sufrirá durante menos tiempo, y en un menor nivel, las consecuencias del tratamiento de esta enfermedad.

Trabajo decente y crecimiento económico

El octavo objetivo de desarrollo sostenible se centra en el trabajo decente y crecimiento económico, concretamente en promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todos.

Con este proyecto se propone una idea novedosa de enfocar el diagnóstico de enfermedades, de tal manera que el médico tiene un nuevo elemento de apoyo a la hora de trabajar. En un futuro, con más datos y con un algoritmo mejorado, se puede convertir en algo más que en un elemento de apoyo, por lo que se pueden llegar más lejos en este campo.

Gracias al avance tecnológico y sobre todo a la inteligencia artificial, podremos diseñar mejores estrategias, destacar sobre la competencia, automatizar procesos costosos, de tal manera que los trabajadores, en nuestro caso, los médicos, puedan aprovechar mejor su tiempo. También serán los clientes los que serán beneficiados, al recibir unos diagnósticos y un producto mejor elaborado, al tener la certeza de que ciertos errores que se pueden producir por fallos humanos se han mitigado.