UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE QUIMICA

VALIDACIÓN DEL MÉTODO MODIFICADO DE DUPLICACIÓN DE J A TRAVÉS DE ALGORITMOS DE INTELIGENCIA ARTIFICIAL

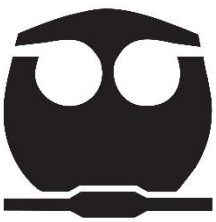
T E S I S

PARA OBTENER EL GRADO DE:

QUÍMICO

P R E S E N T A :

COLMENARES VILLAGARCIA LIAT



DIRECTOR DE TESIS:
DR. JOSÉ FEDERICO DEL RIO PORTILLA

CIUDAD DE MEXICO 2022

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

MEMORANDUM DE ASIGNACION DE JURADO

NOMBRE DEL ALUMNO(A): COLMENARES VILLAGARCIA LIAT

NUMERO DE CUENTA: 314047133

CARRERA: QUÍMICA

JURADO ASIGNADO

PRESIDENTE: DEL RIO PORTILLA JOSE FEDERICO

VOCAL: GAVIÑO RAMIREZ RUBEN LUIS

SECRETARIO: SANCHEZ MENDOZA ERNESTO

SUPLENTE 1: ENRIQUEZ VILLEDA MANUEL ESTEBAN

SUPLENTE 2: PEREZ GONZALES OSCAR ENRIQUE

Sitio Donde se Desarrolló el Tema: Laboratorio 1. Departamento de Química de Biomacromoléculas. Instituto de Química, UNAM

SUSTENTANTE:

_____
Liat Colmenares Villagarcía

ASESOR DEL TEMA:

_____
Dr. José Federico del Río Portilla

SUPERVISOR TÉCNICO:

_____
Dr. Ehecatl Antonio del Río Chanona

ÍNDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

ABREVIATURAS

D : Diferencia en (Hz)

FID: Decaimiento de inducción libre

J Doubling: Método modificado de duplicación de J

J*: Constante de acoplamiento de prueba (Hz)

J: Constante de acoplamiento (Hz)

J_{real}: Constante de acoplamiento real (Hz)

MAE: Error absoluto medio

MSE: Error cuadrático medio

RMN: Resonancia magnética nuclear

RMSE: Raíz del error cuadrático medio

Sh: Primer subarmónico (Hz)

SNR: Relación señal-ruido

W: Ancho de señal (Hz)

RESUMEN

En el presente trabajo, se propone la validación del método modificado de duplicación de J o “J Doubling”, por medio de tres algoritmos de inteligencia artificial: redes neuronales, procesos gaussianos y máquinas de vector soporte.

El objetivo principal del proyecto es desarrollar una metodología de validación completa, basada en tres algoritmos de aprendizaje automático: redes neuronales, máquinas de vector soporte y procesos gaussianos.

La primera parte del texto, sección 1 “marco teórico”, aborda de manera general, los fundamentos de resonancia magnética nuclear o RMN, con el propósito de comprender el fenómeno físico detrás de la generación de los espectros. En la misma sección, también se discute de manera breve, acerca de las bases de la inteligencia artificial y el aprendizaje automático, además de los principios de los tres algoritmos mencionados anteriormente.

Por otra parte, en la sección 4 “desarrollo experimental”, se describen a profundidad las especificaciones computacionales de J doubling, la creación de la base de datos y las variables a trabajar, además de cada uno de los algoritmos de inteligencia artificial utilizados.

Finalmente, en la sección 5 y 6 resultados y análisis de resultados, se discute de manera profunda, acerca del desempeño del entrenamiento de cada uno de los algoritmos utilizados. Para terminar, aplicando el método ya validado, a 5 señales experimentales de RMN, correspondientes a los aminoácidos de la proteína conocida como tamapina.

Cabe mencionar que en la sección 9 Anexo I, se podrán encontrar palabras clave, en donde se definen conceptos fundamentales de aprendizaje automático y RMN.

1. INTRODUCCIÓN

1.1 Problema a resolver

En el campo de la resonancia magnética nuclear existen problemas para determinar constantes de acoplamiento. Una de las problemáticas es cuando las señales son anchas y los máximos de multiplicidad no se distinguen fácilmente, aunado a que puede existir una alta cantidad de ruido, haciendo de este un problema más complejo de resolver. Debido a estas dificultades se publica el articulo "*Measurement of poorly resolved splittings by J Doubling in the frequency domain*", el cuál propone el método modificado de duplicación de J en el dominio de las frecuencias. Dicho método está basado en la deconvolución de las señales, con el objetivo de obtener la constante de acoplamiento por medio de las denominadas "gráficas de integral", donde el mínimo a mayor J* representa la J_{real}. (Figura 1) (Del Rio Portilla, Blechta, & Freeman, 1994)

Varios años después, y luego de incontables publicaciones que exploran el alcance del método (Borceguí Rubio, Chávez, & del Rio Portilla, 2001), en el presente trabajo, se propone dar confiabilidad a los resultados aprovechando el auge de la inteligencia artificial. Se propone realizar una validación completa, planteándolo desde una perspectiva moderna, brindando así una interpretación novedosa de conceptos tales como; robustez, límites de detección, sensibilidad, etc. (España, P.P, & colaboradores, 2016)

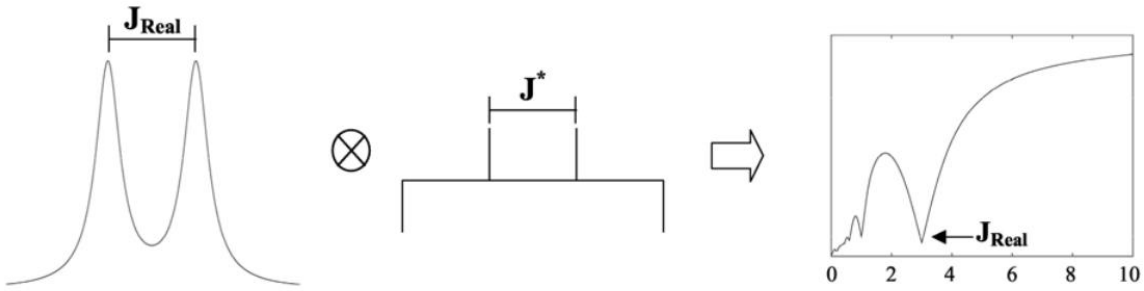


Figura 1. Señal deconvolucionada de RMN, donde la J* coincide exactamente con la J_{real}, dando como resultado las gráficas de integral (derecha).

1.2 Aplicaciones de inteligencia artificial en espectroscopia

Posterior a la expansión, en el uso de tecnologías digitales, no es de sorprenderse que la inteligencia artificial llegara al alcance de la química y más en concreto, a la resonancia magnética nuclear. Uno de sus campos de estudio más amplios, está en el procesamiento de señales, donde se ha visto que las redes neuronales artificiales son de gran utilidad para analizar problemáticas tales como la influencia de ruido, reconstrucción de espectros, determinación de estructuras de proteínas, etc. En la tabla 1 se reportan algunos ejemplos concretos de la aplicación de estos algoritmos siendo uno de los más recurrentes, las ya mencionadas redes neuronales artificiales.

Tabla 1. Ejemplos de aplicación de redes neuronales en resonancia magnética nuclear.

Referencia	**Campo de aplicación**	**Resumen**
Kong, X. *et al.* Artificial intelligence enhanced two-dimensional nanoscale nuclear magnetic resonance spectroscopy. *npj Quantum Inf* **6,** 79 (2020). https://doi.org/10.1038/s41534-020-00311-z	Redes Neuronales, RMN	Aplicación de redes neuronales para el mejoramiento de espectros de resonancia magnética, nuclear bidimensional, a nano escala Mediante aprendizaje profundo, suprime ruido y, por tanto, mejora la sensibilidad del método.
F. Fricke *et al*., "Artificial Intelligence for Mass Spectrometry and Nuclear Magnetic Resonance Spectroscopy," *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 615-620, doi: 10.23919/DATE51398.2021.9473958.	Redes Neuronales, Simulación de Señales. RMN	Simulación de Señales como datos de entrada para redes neuronales entrenadas, y su posterior complementación con señales experimentales para así, evaluar el comportamiento de dichas redes y su posterior aplicación.
Qu, X. *et al.* (2019). Accelerated Nuclear Magnetic Resonance Spectroscopy with Deep Learning. Angewandte Chemie International Edition. doi:10.1002/anie.201908162	Redes Neuronales Convolucionales, Simulación de Señales. RMN	Entrenamiento de CNN a partir de un conjunto de datos simulado con el objetivo de reconstruir espectros de RMN a partir de señales FID.
Zimmerman, D. *et al.* (1997). Automated analysis of protein NMR assignments using methods from artificial intelligence. Journal of Molecular Biology, 269(4), 592–610. doi:10.1006/jmbi.1997.1055	Sistema de Conocimiento (Knowledge-Based System). Proteínas. RMN	Realiza una determinación de la estructura de una proteína teniendo como datos de entrada, la secuencia de aminoácidos de esta, un espectro bidimensional de correlación heteronuclear y algunos espectros tridimensionales.

1.3 Contribución de la Investigación

El objetivo principal de este proyecto es dar confiabilidad y validar el método modificado de duplicación de J. Generar un software de libre acceso, diseñado para el cómodo uso de la comunidad científica, ya que, en un futuro, se podrá abrir una puerta, para resolver problemas relacionados con la determinación de constantes de acoplamiento.

A su vez, cada uno de los resultados tendrá validez estadística, respaldada con una base de datos de más de cincuenta mil elementos, simulados por computadora. Dicha base de datos pretende emular las diferentes variables que pueden afectar el funcionamiento de J Doubling como lo es la SNR, Sh y W.

2. MARCO TEÓRICO

2.1 Breve introducción a la resonancia magnética nuclear (RMN)

La RMN es una de las técnicas analíticas instrumentales, más poderosa al día de hoy. Esto se debe, a que permite determinar la conectividad de los átomos en las moléculas, tanto en medios solubles como en estado sólido. Cabe destacar, que es una técnica no destructiva y que no solo ofrece resultados de elucidación estructural, sino que a su vez puede cuantificar a los analitos en estudio. (Hilzgrade, Wawer, & Diehl, 2008)

Históricamente, la primera señal de RMN, fue observada de manera independiente por dos físicos: Bloch y Purcell, los cuáles recibieron el premio Nobel de 1952 por este hallazgo, sin embargo, no fue hasta que la comunidad química tomó las riendas de dicha investigación que se empezó a aplicar para la elucidación estructural de moléculas. En 1953 se puso a la venta el primer aparato de RMN de baja resolución y para la década de 1970, ya se había avanzado tanto en el campo, que los primeros aparatos con magnetos superconductores, fueron mandados al mercado, los cuales alcanzaban una gran resolución y trabajaban con la ayuda de helio liquido alcanzando a temperaturas de 4 K. Actualmente, los equipos de RMN pueden llegar hasta 1.2 GHz. (buscar referencia)

El espín, es una propiedad de las partículas subatómicas qué posee una magnitud y dirección definidas. Por tanto, los electrones, los protones, los neutrones entre otros poseen espín. Esta propiedad se manifiesta únicamente cuando existe un campo magnético ya que en estado natural el momento total de espín nuclear es cero. Este campo magnético externo denominado como “B” provocara que la componente paralela (del espín nuclear) se alinee en contra o a favor del campo debido a la cuantización de esta coordenada. La energía provocada por esta cuantización está dada por:

$$\Delta E_m = \frac{\gamma \mathrm{hB}}{2\pi}$$

Donde la energía de separación entre ambos estados ΔE_m es función del campo aplicado B, la constante de Planck h y γ la constante giromagnética característica de cada elemento. La RMN consiste en determinar la diferencia de energía ΔE_m representado en la Figura 2.

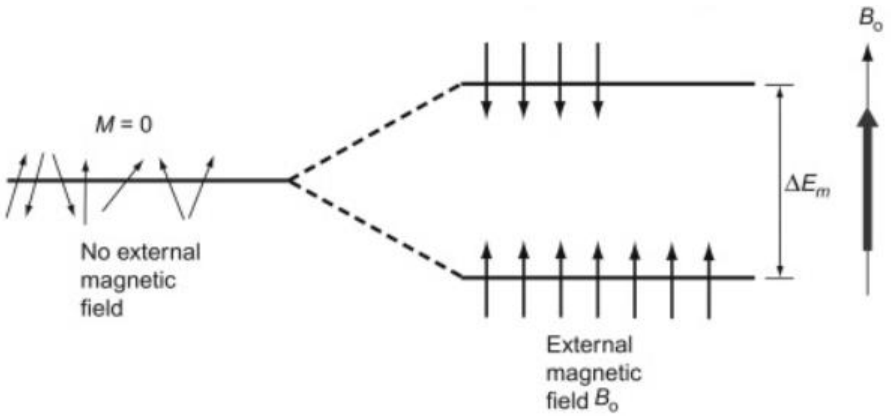


Figura 2. Representación a través de diagrama de niveles de energía, del comportamiento de los espines nucleares antes y después de aplicado el campo magnético externo B0. (Davidovits P. , 2018)

El espaciamiento de energía de manera discreta vuelve a este fenómeno un sistema resonante, y la frecuencia a la que corresponde esta diferencia de energía entre los dos estados se conoce como frecuencias de Larmor: (Davidovits P. , 2018)

$$f_L = \frac{\Delta E_m}{h} = \frac{\gamma B}{2\pi}$$

Una descripción clásica del experimento de RMN, es considerar una interacción clásica entre una partícula con spin y un campo magnético B_0. Este campo intenta alinear el momento magnético μ de la partícula con la dirección del campo aplicado, pero su momento angular provoca, en cambio, un movimiento de precesión de μ alrededor del eje del campo representado en la Figura 3, por lo que entonces μ se comporta como un giroscopio bajo la fuerza impuesta por un momento angular.

La magnetización longitudinal a través de la componente M_z no cambia con respecto al tiempo. Sin embargo, para la detección de la señal de RMN, es necesaria una dependencia con el tiempo, lo cual se logra obtener transfiriendo esta magnetización longitudinal al plano (xy) transversal a z.

Cundo se aplica el pulso de radiofrecuencia, a la frecuencia de Larmor w_0 causa que todos los espines que están precesando de forma incoherente (aleatoriamente) lo comiencen a hacer en fase resultando en una magnetización transversal neta, tal y como se observa en la Figura 3. (Gunther, 2013)

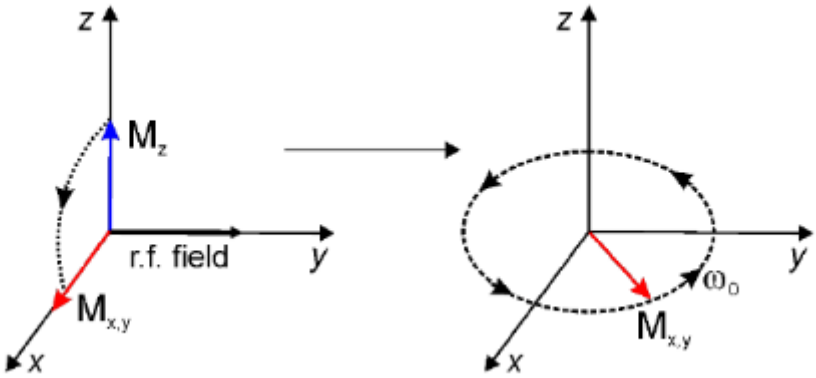


Figura 3. Representación del vector de magnetización longitudinal Mz, rotando por el plano transversal, después de aplicado el pulso de radiofrecuencia (Gunther, 2013)

Este pulso de radiofrecuencia, es generado a través de una bobina en el plano xy. La precesión del vector de magnetización M_{xy} , causará una oscilación en el campo magnético, el cual inducirá una corriente que será detectada por la propia bobina que fue usada para aplicar el pulso. Y una vez aplicado el pulso, este vector de magnetización M_{xy} , regresará gradualmente a su estado de equilibrio, a través del tiempo, mediante un proceso denominado como “tiempo de relajación”.

En una muestra real los diferentes espines se encuentran interactuando entre si, por lo que el mecanismo de relajación ocurre por 2 vías: la longitudinal o T_1 y la transversal o T_2. Este mecanismo se puede describir por medio de una hélice colapsada tal y como se muestra en la Figura 4. Mientras la relajación transversal decae con una trayectoria en espiral dominada por T_2. La relajación longitudinal está dada por T_1, dando una trayectoria helicoidal. Es esta corriente la que genera esta oscilación, lo que da como resultado la señal de RMN, que se conoce como FID o decaimiento de inducción libre, tal como se observa en la Figura 4(c). (Rudraksha, 2015)

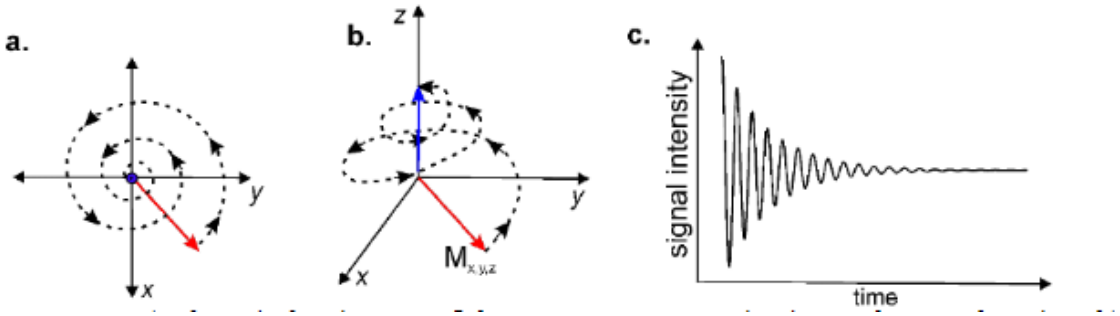


Figura 4. (a) trayectoria espiral de relajación transversal mediante la relajación T2. (b) trayectoria helicoidal combinando las relajaciones T1 y T2. (c) FID resultante. (Rudraksha, 2015)

2.1.1 Ecuaciones de Bloch y transformada de Fourier

Felix Bloch, describió el movimiento de magnetización de los núcleos por medio de tres ecuaciones diferenciales, comúnmente conocidas como “las ecuaciones de Bloch”. (Bloch, 1946). En ausencia de relajación, las rotaciones del vector de magnetización macroscópica $M = (M_x, M_y, M_z)$ están descritas por:

$$\frac{dM(t)}{dt} = M(t) \times \gamma B(t) \quad (1)$$

Donde $B = (B_x, B_y, B_z)$ representa los tres campos magnéticos ortogonales. B_x y B_y son parte del campo magnético oscilatorio del pulso de radiofrecuencia y B_z, representa el campo magnético estático B_0. γ es la constante giromagnética, característica de cada elemento. Desarrollando el producto cruz de la ecuación (1) se obtienen tres ecuaciones diferenciales acopladas para las 3 componentes de M :

$$\frac{dM_x(t)}{dt} = \gamma M_y(t) B_0 - \gamma M_z(t) B_{1y} \quad (2)$$

$$\frac{dM_y(t)}{dt} = \gamma M_z(t) B_{1x} - \gamma M_x(t) B_0 \quad (3)$$

$$\frac{dM_z(t)}{dt} = \gamma M_x(t) B_{1y} - \gamma M_y(t) B_{1x} \quad (4)$$

Las ecuaciones dos a cuatro, describen la precesión de Larmor de la magnetización nuclear M, en un campo magnético externo B_0, así como la rotación de M por un campo de radiofrecuencia variante con componentes B_{1x} y B_{1y}. La presencia de relajación, requiere un término extra a estas ecuaciones, el cuál describe la desaparición de la magnetización del plano transversal debido a la relajación T_2 y la reaparición de la magnetización del equilibrio M_o debido a la relajación T_1:

$$\frac{dM_x(t)}{dt} = \frac{M_x(t)}{T_2} \quad (5)$$

$$\frac{dM_y(t)}{dt} = \frac{M_y(t)}{T_2} \quad (6)$$

$$\frac{dM_z(t)}{dt} = \frac{M_z(t) - M_0}{T_1} \quad (7)$$

Añadiendo estos términos a las ecuaciones dos a cuatro, se obtiene un modelo completo de las ecuaciones de Bloch, en un ámbito no giratorio.

Podemos deducir, de manera simple, sus soluciones para la magnetización transversal después de la excitación y relajación T_1 y T_2 en ausencia de un campo de radiofrecuencia.

En muchas circunstancias, los experimentos de RMN se describen de manera más efectiva en un marco cartesiano, que gira alrededor del campo magnético principal B_0 a la frecuencia v del campo B_1. La transformación de las ecuaciones de Bloch al marco giratorio produce:

$$\frac{dM'_x(t)}{dt} = (\omega_0 - \omega)M_y(t) - \frac{dM'_x(t)}{T_2} \quad (8)$$

$$\frac{dM'_y(t)}{dt} = (\omega_0 - \omega)M'_x(t) + \gamma B_1 M'_z(t) - \frac{dM'_y(t)}{T_2} \quad (9)$$

$$\frac{dM'_z(t)}{dt} = \gamma B_1 M'_y(t) - \frac{M'_z(t) - M_0}{T_1} \quad (10)$$

Estas ecuaciones describen cuantitativamente el movimiento descrito en la Figura 4. (De Graaf, 2019)

Las ecuaciones de Bloch proporcionan una descripción cuantitativa de cualquier experimento de RMN, que involucra pulsos de radiofrecuencia y relajación en una muestra de espines que no interactúan, es decir, sin acoplamiento escalar. Algunas modificaciones de las ecuaciones de Bloch para incluir efectos de difusión e intercambio químico dan lugar a las ecuaciones de Bloch-Torrey (H.C, 1956) y Bloch-McConnell (McConnell, 1958) respectivamente.

De forma paralela, los espines nucleares magnetizados en una muestra son detectadas por su FID, cuyas siglas en español significan *decaimiento de inducción libre*. Este se induce, permitiéndoles alcanzar el equilibrio, en un campo magnético mediante un pulso de radiofrecuencia detectando y amplificando la señal (débil), que se emite cuando los espines reanudan su movimiento de precesión en el campo magnético. (H. Levvit, 2008)

La FID de una muestra con una sola frecuencia de Larmor, después de la excitación en el plano transversal, se puede obtener resolviendo las ecuaciones de Bloch 2 a 7 lo cual resulta en una función compleja con dos componentes: una parte real *R(t)* y otra imaginaria *iI(t):*

$$R(t) = M_x(t) = M_0 \cos(2\pi\upsilon_0 t + \phi)e^{-t/T_2} \quad (11)$$

$$I(t) = M_x(t) = -M_0 \sin(2\pi\upsilon_0 t + \phi)e^{-t/T_2} \quad (12)$$

Donde ϕ representa la fase de la magnetización transversal, relativa al eje *x* inmediatamente, después de la excitación.

Para un pulso de excitación a lo largo del eje *x*, la magnetización transversal es excitada a lo largo del eje y haciendo la fase ϕ igual a -90°. De ello se deduce que la FID, se describe por cuatro parámetros independientes*: M_0, v_0, T_2 y* φ (magnetización, frecuencias características de cada núcleo, tiempo de relajación transversal y fase de magnetización, respectivamente) por lo que supone, que la muestra estaba en un estado de equilibrio, antes de la excitación.

El parámetro M_0 es de gran importancia en muchas aplicaciones de RMN, ya que, está directamente relacionada con la concentración, y para, una señal de frecuencia única M_0 es proporcional a la intensidad FID, inmediatamente después de la excitación. (De Graaf, 2019)

En el caso de señales con más de una frecuencia, la deducción de estos cuatro parámetros es complicada, pero no imposible para una FID, en el dominio del tiempo (Figura 5A). Afortunadamente existe un proceso estándar, que permite extraer y separar diferentes frecuencias de una FID en el dominio del tiempo,

presentándolo como una señal en el dominio de las frecuencias. La Transformada de Fourier, es la responsable de calcular este espectro *F(v)* a partir de una señal en el dominio del tiempo *f(t)* de acuerdo a:

$$F(\upsilon) = \int_{-\infty}^{+\infty} f(t)\, e^{-2\pi i \upsilon t} dt \qquad (13)$$

La integral de la ecuación 13 involucra una exponencial compleja, y el espectro en el dominio de las frecuencias, se puede describir por medio de la ecuación:

$$F(v) = R(v) - iI(v)$$

Su parte real e imaginaria (R*(v)* y *i*I*(v)*), respectivamente, están dadas por las ecuaciones:

$$\mathrm{R}(\upsilon) = \mathrm{A}(\upsilon)\cos\phi - \mathrm{D}(\upsilon)\sin\phi \qquad (14)$$

$$\mathrm{I}(\upsilon) = \mathrm{D}(\upsilon)\cos\phi + \mathrm{A}(\upsilon)\sin\phi \qquad (15)$$

Los términos A*(v)* y D(*v)* representan la absorción y la dispersión de los componentes espectrales, respectivamente (no olvidar que φ representa la fase de la magnetización al eje x y v_0 las frecuencias características de cada núcleo) y están dados por:

$$\mathrm{A}(\upsilon) = \frac{\mathrm{M}_0\mathrm{T}_2}{1 + 4\pi^2(\upsilon_0 - \upsilon)^2\mathrm{T}_2^2} \qquad (16)$$

$$\mathrm{D}(\upsilon) = \frac{2\pi\mathrm{M}_0(\upsilon_0 - \upsilon)\mathrm{T}_2^2}{1 + 4\pi^2(\upsilon_0 - \upsilon)^2\mathrm{T}_2^2} \qquad (17)$$

Estas curvas son comúnmente conocidas como Lorentzianas, y la Figura 5B esquematiza la parte real e imaginaria de los espectros obtenidos, después de aplicar la transformada de Fourier a una señal FID en el dominio del tiempo.

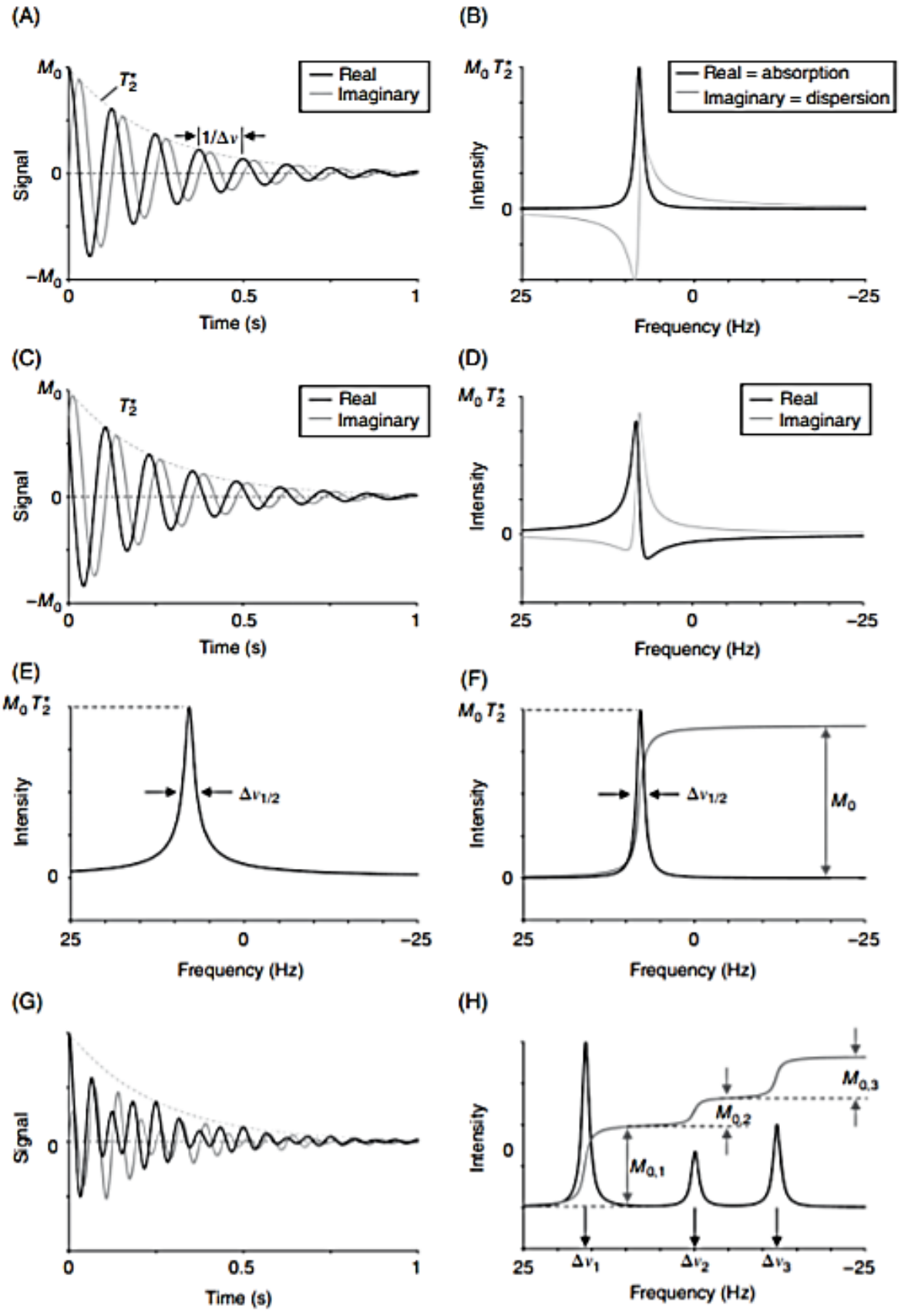


Figura 5. Transformada de Fourier para señales FID en el dominio del tiempo y frecuencia. (De Graaf, 2019)

Cuando φ=0, el espectro de dispersión, es igual al espectro de absorción, y en el caso de que φ≠0, los espectros real e imaginarios, representan una mezcla de líneas absorbentes y dispersivas de acuerdo con las ecuaciones 14 y 15 (Figura 3C y 3D). Por último, cuando el alto de pico y la integral del espectro de absorción son proporcionales a M_0 y T_2, hacen que el ancho de línea $\Delta_{v1/2}$ sea igual a $1/(\pi T_2)$ representado en la Figura 3E y 3F.
Mientras que la transformada de Fourier, para una señal multifrecuencia en el dominio del tiempo, revela sus diferentes componentes al convertirlo al dominio de las frecuencias (Figura 3G y 3H). (De Graaf, 2019)

2.1.2 Ancho, forma y relación señal/ruido en espectros de RMN

Una señal típica de RMN en el dominio de las frecuencias es caracterizada por propiedades específicas. Una de las más importantes es su frecuencia (posición en el espectro Δv_i) y su intensidad o altura de pico (o más específico, área bajo el pico), además de su forma y ancho a media altura $v_{1/2}$ cuyas unidades son los Hertz.

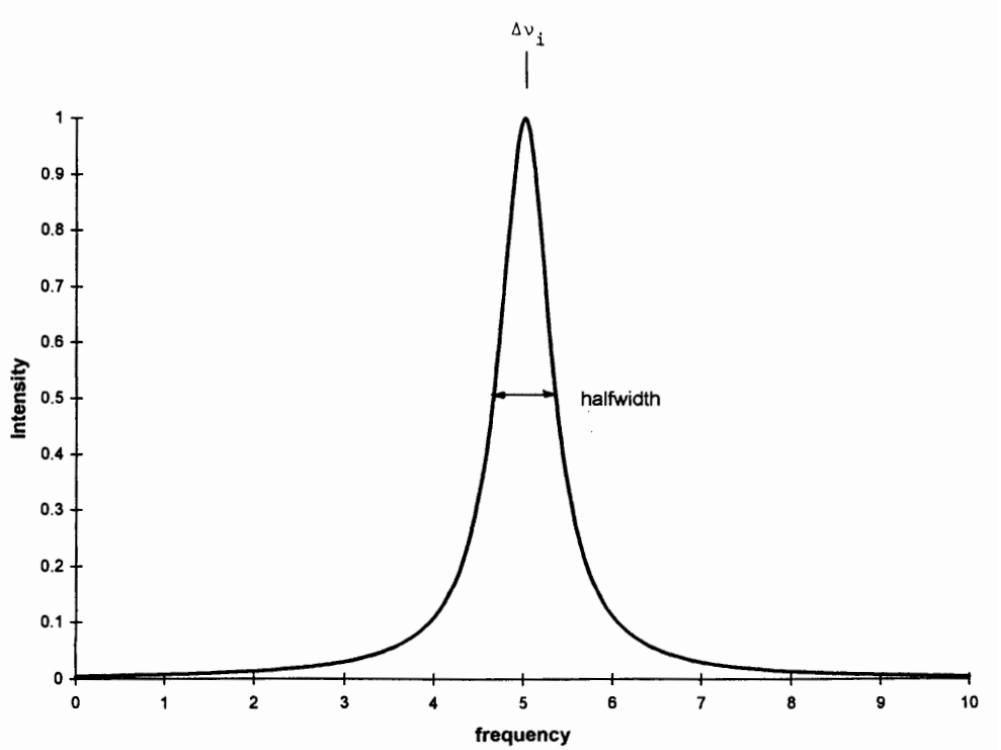


Figura 6. Señal de RMN con comportamiento Lorentziano, ancho de pico a media altura v1/2 y centrado (máximo de señal a Δvi)

Como se mencionó anteriormente, su forma es el resultado de la transformada de Fourier, la cual se describe por una Lorentziana. (W. & K.R, 1989-1990). Un modelo simplificado de las ecuaciones 16 y 17 permitirán abordar el problema de manera más sencilla, resaltando las características más importantes:

$$L_v = \frac{a\Delta v_i^2}{\Delta v_i^2 + b(v - \Delta v_i)^2} \tag{18}$$

Donde *a* es la altura máxima del pico al centro de la señal, Δv_i es el valor particular de frecuencia al máximo del pico (Figura 6) , v es cada punto a lo largo del eje de la frecuencia y b es un parámetro adimensional, que está inversamente relacionado al cuadrado del ancho del pico de acuerdo con:

$$b = (2\frac{\Delta\upsilon_i}{\upsilon_{1/2}})^2 \tag{19}$$

Un concepto importante a tratar en el caso del ancho de la señal, es el principio de incertidumbre para escalas de tiempo espectroscópicas, ya que, se ve afectado directamente. Nos dice que la incertidumbre en la frecuencia de precesión de un núcleo, está inversamente relacionada con la incertidumbre en el tiempo de vida del estado del espín. Si se toma $v_{1/2}$ como el mensurando de la incertidumbre en la frecuencia, esta será inversamente proporcional al tiempo de vida *T,* del estado del espín, por lo que entre más rápido se relaje, se obtendrán señales más anchas y de manera contraria, entre más lento sea el movimiento, se obtendrán señales más finas*:*

$$\upsilon_{1/2} \propto \frac{1}{T} \tag{20}$$

Es importante no confundir el ancho del pico a media altura, el cual es controlado por la tasa de relajación espín-espín con la intensidad de la señal que, en un espectro de modo pulsado, aumenta a medida que disminuye el tiempo de relajación longitudinal. (Rogers, 1998)

Por otra parte, la calidad de los resultados en un experimento de RMN, depende en gran medida de la relación señal/ruido o SNR de acuerdo a sus siglas en inglés. Un buen espectro se puede obtener de varias maneras diferentes: la primera es que el ruido y la interferencia, puedan ser reducidas a un valor lo suficientemente pequeño aumentando el número de experimentos promediados, la segunda es utilizar herramientas especializadas a la mejora de este parámetro. (Andris & Frollo, 2016)

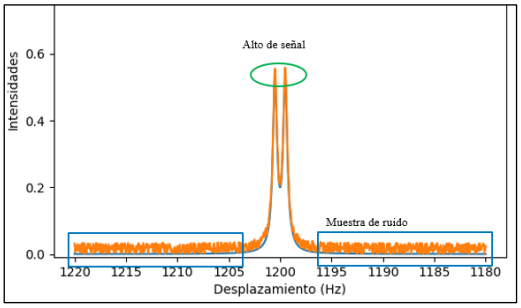


Figura 7. Simulación de un doblete J = 1.0 Hz, W= 0.5 Hz, SNR=53.9 y resolución digital = 0.04 Hz

De acuerdo con (Bentz, Baudzuz, & Krummrich, 2014), una aproximación simple y efectiva para determinar la SNR, de manera experimental, es:

$$SNR = \frac{A_{señal}}{\sigma_{Ruido}} \tag{21}$$

Donde $A_{señal}$ representa el máximo del pico de señal y σ_{Ruido} es la desviación estándar del ruido, la cual es calculada de una parte del espectro donde no exista señal. De acuerdo a la Figura 7 la desviación estándar (muestral para este caso) se calcula de acuerdo a:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{22}$$

x_i es cada punto de la señal de RMN en estudio, $\bar{x}$ es el promedio de todos los puntos de la señal de RMN y n es el total de puntos de la señal de RMN a trabajar. (Rouaud, 2017)

2.1.3 Acoplamientos espín-espín

Existen dos mecanismos diferentes de acoplamientos espín-espín, al más fuerte se le conoce como acoplamiento dipolo-dipolo directo, el cual implica la influencia directa de cada espín sobre su núcleo vecino a través de campos magnéticos que emanan en el espacio, sin embargo, este no es el responsable de la estructura de los multipletes en líquidos isotrópicos, ya que, este acoplamiento directo dipolo-dipolo es eliminado por el rápido volteo molecular. (H. Levvit, 2008)

El mecanismo responsable de la estructura de los multipletes (en líquidos isotrópicos[1]) se denomina Acoplamiento J o acoplamiento indirecto dipolo-dipolo. Este término, indica que los espines nucleares están acoplados con la ayuda de los electrones, cada protón magnetiza débilmente los electrones vecinos , que generan un campo magnético. Este campo transmitido permite que cada espín de los núcleos de hidrógeno detecte la presencia de los protones vecinos. (H. Levvit, 2008)

2.1.4 Método modificado de duplicación de J

De manera general, el método de duplicación de J o J Doubling, consiste en la deconvolución del multiplete en estudio mediante funciones delta, igualmente espaciadas y en antifase, para así lograr la duplicación del acoplamiento, (mejor representado en la Figura 8).

Una característica especial en las funciones delta, es que el espaciamiento entre los máximos/mínimos, es equivalente a los posibles candidatos de la constante de acoplamiento que podrían ser.

A esta separación, se le conoce como constante de acoplamiento de prueba *J*,* la cual se modifica dentro del intervalo donde se espera obtener la constante de acoplamiento “real” denominado como J_{real}

Como resultado de este proceso de deconvolución, se obtiene la integral absoluta para cada J* utilizada, cuyo conjunto da como resultado la denominada “gráfica de integral” (Figura 1) en donde los mininos más profundos,

[1] Isotropía: Homogeneidad de propiedades en todas las direcciones

corresponden al valor de las constantes de acoplamiento presentes en el multiplete. (Borceguí Rubio, Chávez, & del Rio Portilla, 2001)

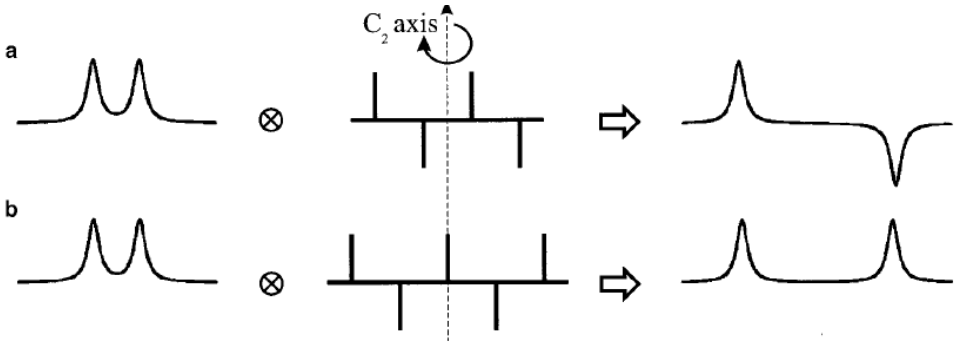


Figura 8. Representación gráfica del proceso de deconvolución de un doblete, mediante una función delta (+1, -1, +1, -1) y (+1, -1, +1, -1, +1) donde C2 corresponde al eje de simetría de las funciones delta. (Borceguí Rubio, Chávez, & del Rio Portilla, 2001)

2.1.5 Limitaciones

Este método tiene varias limitaciones, algunas de ellas se pueden observar en simulaciones, donde se obtienen las constantes de acoplamiento en función del ancho de la señal. La Figura 9a muestra los resultados de un multiplete con tres J = (1.0, 1.5 y 2.0) Hz y una vez que el ancho de línea es superior a 1.2 Hz, es posible ver solo un pico en lugar de ocho. Sin embargo, J Doubling aún puede medir las tres constantes hasta un ancho de línea de 4.6 Hz.

A este valor, varios mínimos que no corresponden a subarmónicos, comienzan a emerger cerca de la J más pequeña. Aunque todavía está presente un mínimo de 1 Hz en un ancho de línea superior a 5.4 Hz, ya no es posible verificar objetivamente que se trata de un acoplamiento real. En constantes de 2.0 Hz todavía se puede medir sin ambigüedades hasta los 6.0 Hz, que corresponde a cinco veces el límite del método directo. (Garza García, Ponzanelli Velázquez, & del Rio Portilla, 2001)

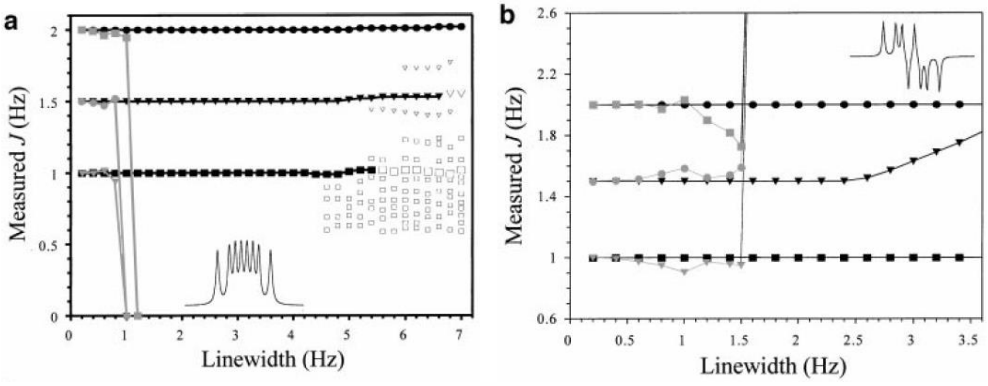


Figura 9. (a) Determinación de las constantes de acoplamiento con el método directo y modificado de J en función del ancho de línea para un multiplete simulado con tres constantes en fase de (1, 1.5 y 2.0) Hz. (b) Medición de las constantes de acoplamiento con el método directo y modificado de J como función del ancho de línea. Se utiliza un multiplete simulado con dos constantes en fase de (1 y 2) Hz y una constante antifase de 1.5 Hz. Las medidas directas se indican mediante J=2.0 Hz, ▄; J=1.5 Hz, ●; J = 1.0 Hz,▼. Las medidas de duplicación de J se indican mediante J=2.0 Hz,●; J=1.5 Hz▼; J=1.0 Hz▄. Los cuadrados vacíos pequeños (□) y triángulos (∆) indican mínimos que no corresponden a subarmónicos. Los cuadrados grandes vacíos y los triángulos indican mínimos que pueden corresponder al JReal. (Garza García, Ponzanelli Velázquez, & del Rio Portilla, 2001)

2.1.6 Convolución/deconvolución

La convolución *h(t)* de dos funciones *f(t) y g(t),* se puede explicar como la ampliación de una función por la otra. En este caso es utilizada para amplificar la señal en la posición correspondiente a la J_{real} , la cual contiene al multiplete en estudio *f(t),* mientras que *g(t)* representa lo que se conoce como función de ponderación, y para este caso en particular, las funciones delta se pueden describir matemáticamente como:

$$h(t) = \int_{-\infty}^{\infty} f(t) \cdot g(t-\tau)d\tau \; = \; f(t) \times \; g(t) \tag{23}$$

Poseen varias propiedades como la conmutatividad, asociatividad y leyes distributivas las cuales en combinación con la transformada de Fourier conducen a:

$$FT(f(t) \cdot g(t)) = FT(f(t)) \times FT(g(t)) \tag{24}$$
$$FT(f(t) \times g(t)) = FT(f(t)) \cdot FT(g(t)) \tag{25}$$

De igual manera se pueden derivar ecuaciones para la transformada de Fourier inversa. Las ecuaciones 24 y 25, indican que la convolución en un dominio es una simple multiplicación en el otro. Uno de los usos principales de esta operación es hacer mejoras en la FID, para señales que contienen mucho ruido en la cola final multiplicando esta por una función que cae exponencialmente, dando como resultado una convolución en el dominio de las frecuencias con el ruido atenuado. (De Graaf, 2019)

2.2 Aspectos básicos de la inteligencia artificial

El termino inteligencia artificial o IA, es un tema de gran polémica para todo aquel que se encuentra con este campo del conocimiento por primera vez. Con el paso del tiempo se ha creado una idea muy alejada de lo que en realidad significa, surgiendo la pregunta ¿Qué es la Inteligencia Artificial? Por lo que nos conduce a un complejo problema, derivado de la dificultad para su definición de manera clara y concisa.

Fue en 1955 cuando se aceptó formalmente este término a partir de las investigaciones de (McCarthy, Minsky, & Shannon, 2006). Ellos concluyeron que se podía imitar la manera en que aprende el ser humano por medio de una computadora. McCarthy, afirmaba que la meta de la inteligencia artificial es desarrollar maquinas de comportamiento inteligente.

Sin embargo, esta definición es demasiado general y por ende tiene varias carencias. Otra propuesta hecha por (Britannica, 1991) nos dice que la IA es la capacidad de las tecnologías digitales o robots controlados por computadora, para resolver problemas que se asocian normalmente con capacidades de proceso intelectual en seres humanos. De dicha definición surgen debilidades, ya que, admitiría que computadoras con gran capacidad de memoria y acceso a dicha información, se considerarían inteligentes. Por lo que, la manera más elegante de definir este concepto de manera sencilla seria:

> *“La inteligencia artificial es el estudio de cómo hacer que las computadoras hagan cosas que, hasta el momento, los humanos son mejores”.* (Rich, 1983)

Es importante mencionar que una de las ventajas más grandes de la inteligencia humana es la adaptabilidad, por qué somos capaces de ajustarnos a diferentes entornos según las condiciones y por ende, capaces de modificar nuestra conducta por medio del aprendizaje. Es precisamente por esta razón que la habilidad humana es vastamente superior a las computadoras. De esta problemática y citando directamente a la definición de Rich es que surge el subcampo del Aprendizaje automático o Machine Learning. (Wolfgang, 2017)

2.3 Aprendizaje automático o machine learning

La primera máquina formalmente registrada, de acuerdo a varios autores, se trata de la eolipila inventada por Herón de Alejandría en el siglo I d.C. (White, 1974), (García & Scherer, 1997). Considerada como la primera máquina de vapor, consiste de una esfera metálica hueca rellena de agua y que puede girar alrededor de un eje horizontal, hallándose provisto de dos tubos curvos, que parten de los extremos del diámetro perpendicular al eje de rotación. Cuando hay hervor, se produce cierta cantidad de vapor y así genera un movimiento de rotación. (Giro, 2020)

A lo largo de los años, las maquinas nos han servido para cumplir con un propósito en específico y poseen una entrada, una salida o resultado, teniendo en el medio un proceso, que hasta hace algunas décadas, era estático o no adaptativo, provocando así la gran diferencia con la inteligencia humana. Esta última, sí posee adaptabilidad con el medio que lo rodea (Bonaccorso, 2017) y no fue sino hasta 1946, que derivado de la Segunda Guerra Mundial surge la primera computadora. Dicha computadora pesaba 30 toneladas y estaba construida sobre módulos de metal de casi 3 metros de alto, 70000 resistencias y 18000 válvulas ocupando el área de todo un gimnasio. (Castells, 1999)

Las computadoras, ofrecen grandes beneficios pues son programables, flexibles y muy rápidas, lo que las hace especialmente eficientes para el aprendizaje automático. Este aprendizaje automático, se puede definir de manera generalizada como algoritmos que pueden extrapolar leyes y aprender su estructura con una alta precisión. Si los datos de entrenamiento son los correctos, puede generar predicciones. (Bonaccorso, 2017)

2.3.1 Taxonomía de la inteligencia artificial y tipos de aprendizaje

La subcategoría más importante de la inteligencia artificial, es el aprendizaje automático cuya estructura interna tiene más campos, donde cada uno cumple con objetivos y enfoques en específico. Un esquema general está representado en la Figura 10 donde se observa que una parte importante está basada en el aprendizaje profundo o redes neuronales artificiales de las cuales se hablará más adelante. (Zahangir, 2019)

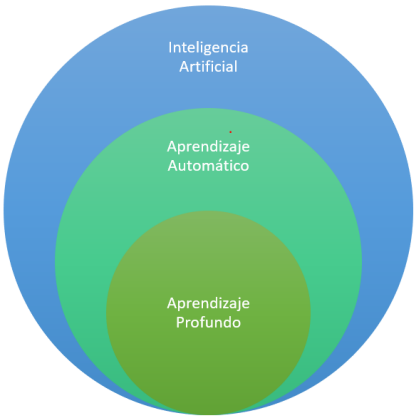


Figura 10. Representación del universo de la Inteligencia Artificial y sus subcategorías

El aprendizaje profundo o *deep learning,* surgió en 2006 y utiliza arquitecturas profundas o en capas de aprendizaje (enfoques jerárquicos). Su fundamento consiste en el procedimiento de estimar los pesos del modelo para que pueda realizar una tarea especifica. Consta de varias capas entre la entrada y la salida (Figura 11) lo cual permite muchas etapas de procesamiento de información no lineal, con arquitecturas jerárquicas existentes, las cuáles emplean para el aprendizaje de características especificadas por el usuario, y la clasificación de patrones.

Una diferencia clave entre el aprendizaje automático tradicional y el aprendizaje profundo, radica en cómo se extraen dichas características. Estos enfoques tradicionales utilizan ingeniería mediante la aplicación de varios algoritmos de extracción, mientras que el aprendizaje profundo, entrena automáticamente mediante representaciones jerárquicas. (Salgueiro, Rodríguez, Rodríguez, & Mateo, 2020)

Los tipos de aprendizaje que se utilizan en la inteligencia artificial se pueden clasificar en tres subcategorías: supervisados/semi supervisados, no supervisados y reforzados. (Salgueiro, Rodríguez, Rodríguez, & Mateo, 2020). El aprendizaje supervisado consiste en un conjunto de datos de entrenamiento, el cual contiene una entrada y una salida, comúnmente, denominados en informática como input y output, respectivamente. Derivado de estos datos, el agente inteligente (el algoritmo) puede corregir sus parámetros para reducir la magnitud global de la función de pérdida a evaluar y después de cada iteración. También puede cambiar para ajustarse a los datos de manera coherente y flexible, haciendo así que la diferencia entre el valor predicho y el real sea lo más cercana al cero. (Bonaccorso, 2017) .

El aprendizaje no supervisado está basado en la ausencia de la evaluación del modelo mediante métricas de desempeño y, por ende, no se puede medir la

diferencia entre un valor predicho y el real, sin embargo, es muy útil en tareas de agrupación de datos. Por último, está el aprendizaje reforzado, el cual tampoco posee supervisores si no que está basado en una retroalimentación proporcionada por el entorno que lo rodea. En este caso la información es más cualitativa y no ayuda al agente inteligente, a determinar una predicción precisa de su error. Su retroalimentación en este caso, se le conoce como recompensa (y si es negativa se le conoce como castigo) y ayuda a entender de mejor manera hacia dónde va el sentido del aprendizaje. (Bonaccorso, 2017)

2.4 Redes neuronales artificiales

2.4.1 Primeros años y símil con redes neuronales biológicas

El aprendizaje profundo, ha sido una de las subramas más desarrolladas del aprendizaje automático en los últimos años. Uno de los principales detonantes de este crecimiento es en el reconocimiento de imágenes, ya que, hasta hace algunos años parecía imposible que una maquina tuviera esa habilidad (He, Zhang, Ren, & Sun., 2016).

Imitar el mecanismo de aprendizaje en organismos biológicos, como el humano, es la consecuencia directa de este tipo de aprendizaje. Este tipo de sistemas contienen un tipo especial de células llamadas neuronas, las cuales están interconectadas entre sí por medio de los axones y dendritas siendo la región donde se lleva a cabo esa conexión conocida como sinápsis, la cual responde fuertemente al tipo de estímulo y su entorno. (Aggarwal, 2018)

Estos mecanismos biológicos son simulados a través de *redes neuronales artificiales,* las cuales contienen unidades computacionales referidas como neuronas. Estas neuronas, se interconectan entre si a través de *pesos* los cuales juegan el mismo rol que la fuerza de una conexión sináptica en organismos biológicos. Cada entrada (símil a un estímulo) es escalada a través de los pesos, mismos que afectan el funcionamiento de estas unidades computacionales. Las redes neuronales, artificiales funcionan a través de las *entradas* estas se propagan a través de la red, por medio de los pesos asignados, y el aprendizaje ocurre ajustando los pesos que conectan a las neuronas. (Cun, Vengio, & Hinton, 2015)

2.4.2 Tipos de redes neuronales y arquitectura

Existen principalmente dos tipos de redes neuronales: las de una sola capa y las multicapas. La primera es conocida como “perceptrón”, y tiene como particularidad que sus entradas o inputs son procesadas directamente para producir la salida u output. En cambio, en las redes neuronales multicapa, las neuronas están acomodadas en un diseño en donde la entrada y salida de información, se separa por las capas ocultas tal y como se observa en la Figura 11. A este tipo de red neuronal, también se le conoce como Feed-Forward Network o por su traducción al español como redes neuronales hacia adelante. Su nombre también se debe a que las capas sucesivas se alimentan entre sí en la dirección de avance desde la entrada a la salida. (Aggarwal, 2018)

Un concepto fundamental en estas redes es la “neurona”, misma que se puede definir como la unidad básica de una red neuronal, en donde las entradas se someten a una suma ponderada y pasan a través de la función de activación. (de las cuales se hablará más adelante). (Kubat, 2017)

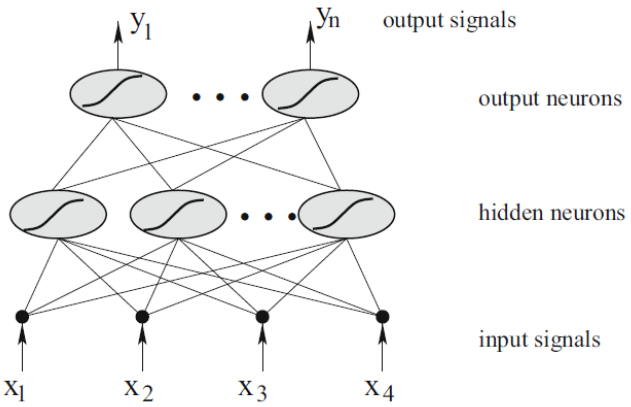


Figura 11. Representación de red neuronal hacia adelante con una capa oculta.

En la Figura 11, se observa de manera explícita una función de activación (función tipo sigmoide dentro de las neuronas de cada capa). Ésta es la encargada de realizar una operación específica sobre la información para así, transferirla a la capa siguiente.

2.4.3 Como funciona una red neuronal

El modelo más sencillo de redes neuronales es el perceptrón, y a partir de este se deducirán los conceptos clave para después, extrapolarlos a redes multicapa. Considerando una situación donde se tiene un conjunto de datos de entrenamiento (variables de entrada o características que aprenderá el algoritmo) de la forma *(X, y)* donde $X = [X_1, \ldots X_d]$ y contiene *d* variables diferentes, mientras que y representa el valor observado o a predecir. A partir de esto la capa de entrada va a contener *d* modos que transmitirán las *d* variables de entrada, cada una con *d* cantidad de pesos $W = [w_1, \ldots w_d]$ que conectarán al nodo de salida. Para esto se aplica la operación lineal $W \cdot X = \sum_{i=1}^{d} w_i x_i$,la cual, se puede describir como una suma ponderada de todos los pesos por todas las entradas en la red. A su vez esta suma es procesada por una función de activación, para dar como resultado la salida de la red, mejor conocida como valor predicho $\hat{y}$:

$$\hat{y} = activación\{W \cdot X\} = activación\{\sum_{i=1}^{d} w_i x_i\} \quad (26)$$

Es importante diferenciar el valor predicho $\hat{y}$ al valor real *y* ya que se calculará el error como $E(X) = (y - \hat{y})$. Si después de calcular dicho error, este es diferente de cero entonces los pesos en la red neuronal se actualizarán para así mejorar la predicción. Un caso especial está relacionado con modelos que contienen una parte sesgada, la cual se relaciona con una distribución muy desequilibrada en los datos, por lo que se agrega un parámetro extra conocido como "bias". La "bias" es una variable de sesgo denominada como *b* y se suma a la ecuación 26 para dar:

$$\hat{y} = activación\{W \cdot X + b\} = activación\{\sum_{i=1}^{d} w_i x_i + b\} \quad (27)$$

Una vez entendido como se procesan los datos a través de la red, surge de manera inmediata la pregunta ¿Como sabemos cuáles son los pesos adecuados para realizar la mejor predicción?. Es así como Rosenblatt (Rosenblatt, 1958) propone un método de optimización adecuado para encontrarlos. El objetivo es encontrar los pesos *W* que minimizan el error entre la predicción y el valor real para un conjunto de datos denominado *D*:

$$\min_W L = \sum_{(X,y)\in D} (y - \hat{y})^2 = \sum_{(X,y)\in D} (y - activación\{W \cdot X\})^2 \quad (28)$$

A este tipo de funciones también se les conoce como función de perdida/objetivo, y son ampliamente usadas en casi todos los algoritmos de redes neuronales. Para el caso del perceptrón, genera superficies suaves y utiliza una aproximación del gradiente para encontrar el mínimo de la función de perdida ∇L , la cual se define como:

$$\nabla L = \sum_{(X,y)\in D} (y - \hat{y})\, X \quad (29)$$

El algoritmo que entrena a la red trabaja de manera que va alimentando cada entrada de datos *X*, para así, ir calculando la predicción $\hat{y}$. Es entonces cuando los pesos son actualizados basados en el error de la predicción $E(X) = (y - \hat{y})$. Cuando cada dato de entrada *X* es alimentado a la red, el vector de peso *W* es actualizado de acuerdo a:

$$W \Leftarrow W + \alpha(y - \hat{y})X \quad (30)$$

El parámetro alfa α regula lo que se conoce como "taza de aprendizaje", y representa el recorrido que hace el algoritmo del perceptrón sobre todos los ejemplos de entrenamiento de manera aleatoria, para ir ajustando los pesos hasta que se alcanza la convergencia al mínimo de error. Cada uno de esos ciclos se le conoce como época o epoch. Este algoritmo también se le conoce como gradiente descendente estocástico o "*stochastic gradient-desent"* y

minimiza el error de la predicción para valores aleatorios de entrenamiento. (Aggarwal, 2018)

2.4.4 Funciones de activación y perdida

Cada modelo de neurona consta de un elemento de procesamiento, con conexiones de entrada sinápticas y una única salida, cuyo flujo de señales de entrada se considera unidireccional (Zurada, 2006). De igual manera, se puede describir a una neurona internamente de acuerdo al valor pre y post operación. El primero captura los datos de entrada en función de la suma ponderada de sus pesos, mientras que el post operación, procesa dichos valores de entrada por medio de las funciones de activación. Algunas de las funciones más comunes son:

- Lineal: Se utiliza en casos donde el valor de salida es un valor real con un dominio en las abscisas de -∞ a +∞
- Sigmoide: Se utiliza comúnmente en algoritmos de clasificación binaria, cuyo dominio en el eje de las *y* va de 0 a 1. También se relaciona ampliamente con modelos de probabilidad.
- Tanh: Tiene una forma muy similar a la sigmoide, sin embargo, ésta es rescalda de manera horizontal y vertical para obtener un dominio en el eje de las abscisas de -1 a 1
- ReLU: Ampliamente utilizada en años recientes cuyo dominio en la abscisa va de 0 a +∞ (Sibi & Siddarth, 2013)

Una representación gráfica de dichas funciones de activación se puede observar en la Figura 12.

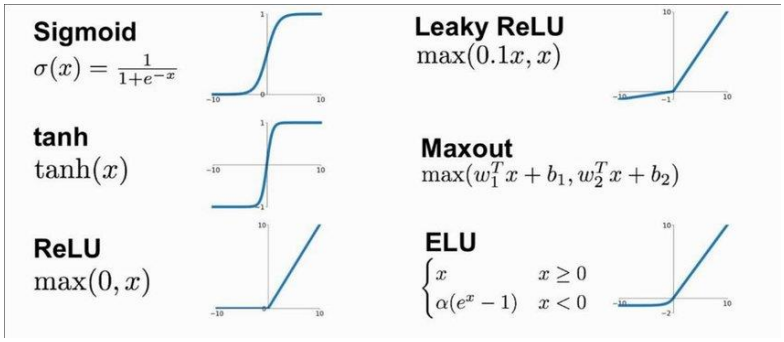


Figura 12. Representación de funciones de activación básicas para redes neuronales. (Jayawardana & Sameera Bandaranayake, 2021)

Por otra parte, las funciones de perdida, son las encargadas de minimizar el error con el objetivo de encontrar a los candidatos de peso adecuados para la red neuronal. (Goodfellow, Vengio, & Courville, 2016) Estas funciones también reducen las características de un sistema posiblemente complejo, a un solo valor escalar que permite clasificar y comparar las soluciones candidatas (Reed & MarksII, 1999). Uno de los más comunes es el error absoluto medio mejor conocido como MAE, junto con el error cuadrático medio o MSE por sus siglas en inglés.

Estas funciones de pérdida se pueden definir de acuerdo a los formalismos de inteligencia artificial como:

- MAE: Mide la magnitud promedio de las diferencias absolutas entre *N* vectores predichos $S=\{x_1,x_2,\ldots,x_N\}$ y las observaciones reales $S^*=\{y_1,y_2,\ldots,y_N\}$. Se define la función de perdida en términos del error absoluto medio como:

$$\mathcal{L}_{MAE}(S,S*) = \frac{1}{N}\sum_{i=1}^{N} ||x_i - y_i||_1 \tag{31}$$

  Donde $||\ ||_1$ representa la norma del vector $L_{1=}\ x_i - y_i$.

- MSE: Representa una regla cuadrática que mide la magnitud promedio de los *N* vectores predichos $S=\{x_1,x_2,\ldots,x_N\}$ y las observaciones reales $S^*=\{y_1,y_2,\ldots,y_N\}$. Su función de perdida corresponde a:

$$\mathcal{L}_{MAE}(S,S*) = \frac{1}{N}\sum_{i=1}^{N} ||x_i - y_i||_2^2 \tag{32}$$

Nuevamente $||\ ||_2$ representa la norma del vector $L_{2=}\ x_i - y_i$. (Qi, Siniscalchi, Ma, & Lee, 2020)

Por último, se ha demostrado (Qi, Siniscalchi, Ma, & Lee, 2020) que MAE es una función de perdida altamente eficiente y garantizada contra ruidos aditivos provocando una gran robustez en algoritmos entrenados. Por otro lado, MSE no ofrece esta opción, sin embargo, también ha demostrado ser lo suficientemente robusta para generar resultados confiables.

2.4.5 Algoritmo backpropagation, sobreajuste y early stopping

En redes neuronales multicapa, su arquitectura consiste de filas completamente interconectadas entre si llamadas nodos, los cuales se organizan de manera secuencial para formar las capas. Las entradas o capas de entrada, son las encargadas de recibir los datos, mientras que las salidas o capas de salida, son las responsables de producir el resultado final, mientras que las internas u ocultas, son las que proveen las conexiones entre la salida y la entrada (Figura 11). La entrada al nodo de la O_i capa *(O[j]),* es igual a la suma de las salidas de la anterior denominadas como *(g[y])*, cada entrada de un nodo anterior es multiplicado por un factor de peso *(w[jy])* asociado con una conexión en particular de acuerdo a (Erb, 1993):

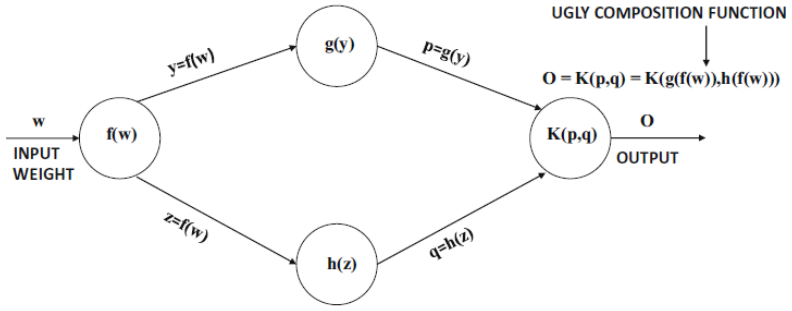
$$nueva\ entrada\colon i[j] = \sum_{i}\{w[jy]O[j]\} \tag{33}$$

En el caso de las redes neuronales perceptrón la elección de los pesos es relativamente sencilla, ya que la función de perdida es función directa de los propios pesos. Sin embargo, en el caso de las redes multicapa, el problema se vuelve mucho más complicado, debido a que la función de perdida es una complicada composición de funciones de los pesos de cada capa que compone a la red. (Aggarwal, 2018)

Para conocer el gradiente en funciones de pérdida compuestas, se utiliza la regla de la cadena comúnmente utilizada en calculo diferencial. A este algoritmo se le conoce como "*backpropagation",* y calcula los gradientes de error en términos de sumas de productos de gradiente local sobre las diversas trayectorias de un nodo a la salida (Goh, 1995). Este algoritmo contiene dos fases: hacia enfrente y hacia atrás, mejor conocidas en inglés como "*forward phase"* y "*backward phase*".

- Fase hacia adelante: las entradas que alimentan a la red neuronal van recorriendo la red en dirección hacia el nodo de salida, utilizando los pesos designados en dicha vuelta. Dan como resultado la salida final y se comparará con el valor de entrenamiento, para así, calcular la derivada de la función de perdida. Este gradiente se debe calcular con todos los pesos de todas las capas.
- Fase Hacia Atrás: El objetivo de esta fase es, aprender el gradiente de la función de perdida con el propósito de ir actualizando los pesos. Cabe mencionar que este gradiente se aprende en dirección hacia atrás, comenzando desde el nodo de salida hacia la entrada. (Hecht Nielsen, 1989)

En la Figura 13 se puede apreciar de manera esquemática la aplicación de la regla de la cadena hacia la salida de la neurona *K(p,q),* la cuál es el resultado de las neuronas *g(y)* y *h(z).*



$$\frac{\partial o}{\partial w} = \frac{\partial o}{\partial p} \cdot \frac{\partial p}{\partial w} + \frac{\partial o}{\partial q} \cdot \frac{\partial q}{\partial w} \quad \text{[Multivariable Chain Rule]}$$

$$= \frac{\partial o}{\partial p} \cdot \frac{\partial p}{\partial y} \cdot \frac{\partial y}{\partial w} + \frac{\partial o}{\partial q} \cdot \frac{\partial q}{\partial z} \cdot \frac{\partial z}{\partial w} \quad \text{[Univariate Chain Rule]}$$

$$= \underbrace{\frac{\partial K(p,q)}{\partial p} \cdot g'(y) \cdot f'(w)}_{\text{First path}} + \underbrace{\frac{\partial K(p,q)}{\partial q} \cdot h'(z) \cdot f'(w)}_{\text{Second path}}$$

Figura 13. Ejemplo de aplicación de la regla de la cadena en redes neuronales. (Aggarwal, 2018)

Un problema clásico en el entrenamiento de algoritmos de aprendizaje automático es el sobreajuste del modelo, y se refiere a cuando un algoritmo modela a un conjunto particular de datos de manera casi perfecta. Contrario al hecho de que se piense que puede resultar bueno, genera un gran problema, ya que no se puede garantizar un buen resultado en datos nuevos, ocasionando así, poca robustez del modelo. (Lawrence, Giles, & Tsoi, 1997)

La solución sencilla a este problema es utilizando un tipo especial de entrenamiento, conocido como "*early stopping*" o parada anticipada. Este proceso consiste en detener el descenso de gradiente después de unas cuantas iteraciones. La manera de decidir el punto de parada es dividiendo el conjunto de datos de entrenamiento en dos: "entrenamiento" y "prueba". Los datos de entrenamiento, son aquellos proporcionados por el usuario y a los cuales se desea ajustar el modelo, mientras que los datos de prueba son un subconjunto de los de entrenamiento y, serán con los que se compararán las predicciones obtenidas por el algoritmo. Se finalizará el entrenamiento una vez que el error en el conjunto de entrenamiento vs. prueba comience a aumentar o se mantenga constante. (Prechelt, 1998)

2.5 Máquinas de vector soporte

Las máquinas de vector soporte o SVM, son algoritmos computacionales que, aprenden por medio de ejemplos asignando etiquetas a objetos (Boser, Guyon, & Vapnik, 1992). En esencia, es una entidad matemática que maximiza una función en particular con respecto a una colección de datos de entrenamiento. (Noble & S, 2006)

Una de las grandes ventajas en el aprendizaje de este algoritmo, es que se hace con gran reproducibilidad y precisión. Su principal uso reside en resolver problemas de clasificación, aunque algunas veces también puede aplicarse a casos de regresión (Pisner & Schnyer, 2020). Su funcionamiento está sustentando en el campo de la teoría del aprendizaje estadístico (Vapnik & Lerner, 1963) y su propósito es, seleccionar el margen máximo de separación del hiperplano de clasificación (Figura 14d y Figura 14f).

En la Figura 14 se plantea un sistema simple, cuyo problema reside en separar un conjunto de datos de acuerdo a sus características. Uno perteneciente a la clase rojo y otro a la clase verde. El objetivo es obtener una recta que separe ambas características, sin embargo ¿Cuál será la mejor?. Para ello se definen vectores que, en realidad son los datos de entrenamiento, para encontrar así, los "vectores soporte" que maximizan la distancia de separación en esta clasificación binaria.

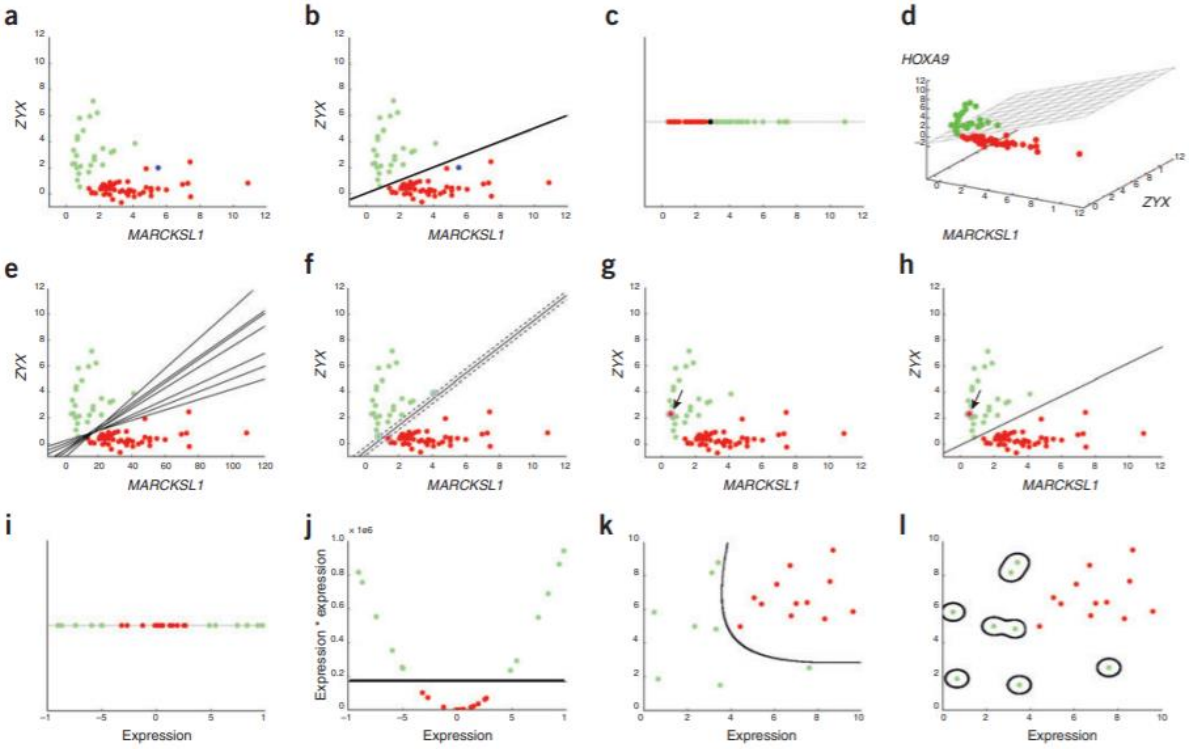


Figura 14. Ejemplificación del uso de SVM en un problema binario de clasificación entre dos clases, verde y rojo. (Noble & S, 2006)

Un concepto importante es el “margen suave”, el cual permite manejar situaciones donde aparezcan datos anómalos tal y como se refleja en la Figura 14g para así, entrenar al algoritmo de manera que, no se vea fuertemente afectado por un dato en específico. Dicho margen suave vuelve mucho más robusto al algoritmo. Es importante no confundir este concepto con el “margen máximo de separación”, que se puede apreciar gráficamente en la Figura 14f, y representa la distancia máxima entre los vectores soporte y los datos. (Kubat, 2017)

Es importante resaltar que existen casos donde la separación con hiperplanos no será posible, debido a la propia naturaleza de los datos. Una manera de solucionar esta problemática es a partir de los “kernel”. Los kernel son funciones que proveen soluciones adicionando dimensiones extra a los datos. En general, una función de este tipo proyecta los datos de un espacio de baja dimensión, a uno de dimensión superior. Sin embargo, hay que tener cuidado, ya que, si se eligen demasiadas dimensiones, se producirá un sobreajuste en los datos (Figura 14i), causando las mismas problemáticas que en aprendizaje profundo, y resultar en un algoritmo poco robusto ante nuevas entradas. (Wolfgang, 2017)

Dado que SVM se utiliza principalmente para resolver problemas de clasificación, la manera correcta de evaluar su desempeño en el entrenamiento, es a través de la “matriz de confusión”. Ésta consiste en una matriz de dimensión 2x2, donde cada lugar indica la cantidad de casos que caen en las categorías:

L: todos positivos o correctamente clasificados
I: todos negativos o erróneamente clasificados
J: falsos positivos
K: falsos negativos

Dicha clasificación se observa en la Tabla 2 y derivado de los postulados anteriores, se pueden asociar 4 métricas las cuales ayudarán a determinar si el modelo es adecuado o no:

1. Sensibilidad: Se refiere al porcentaje de casos positivos detectados, o interpretando de otra manera, es la proporción de los casos positivos clasificados con respecto al total de positivos reales. Se calcula como L/(L+K)

2. Especificidad: Representa al porcentaje de caos negativos detectados o, de otra manera, es la proporción de los casos negativos clasificados con respecto al total de negativos reales. Se calcula como I/(I+J)

3. Precisión: Se refiere al porcentaje de predicciones positivas acertadas y se calcula como L/(L+J)

4. Exactitud: Es el porcentaje total de predicciones correctas y se calcula como (I+L)/(I+J+K+L) (Visa, Ramsay, Ralescu, & Van Der Knaap, 2011)

Tabla 2. Ejemplificación de las posiciones en la Matriz de confusión.

		Estimados por el modelo	
		Negativos (N)	Positivos (P)
Valores reales	Negativos (N)	I	J
	Positivos (P)	K	L

2.6 Procesos gaussianos

Es bien sabido que la estadística y el aprendizaje automático están ampliamente relacionados. Sin embargo, éstos toman enfoques distintos ya que el primero consiste en, entender las relaciones entre los datos en términos de modelos que intentan aproximar la realidad. Por otro lado, el aprendizaje automático tiene como objetivo, realizar predicciones de la manera más acertada posible y con una alta precisión. (Rasmussen & Williams, 2006)

Un proceso estocástico[2] es una generalización de la distribución de un proceso gaussiano probabilístico, y describe una variable aleatoria de dimensión finita. Resulta que los cálculos para la inferencia y el aprendizaje se vuelven relativamente fáciles y, por lo tanto, los problemas de aprendizaje supervisado se pueden resolver a partir de ejemplos y, convertir directamente en el marco del proceso. (M. & Neal, 1998)

[2] Sistema que permite dar seguimiento a un fenómeno de variables aleatorias, las cuáles dependen de una variable determinista

2.6.1 Regresión Probabilística

Una de las aplicaciones más grandes de los procesos gaussianos, es la regresión probabilística tal y como se observa en la Figura 15. El modelo plantea un conjunto de datos de entrenamiento *D={(xi,yi), i=1,…,n}* con *n* pares de entradas x_i (vectores) y salidas y_i escalares. En este caso se asumirá que el ruido es aditivo, independiente y con comportamiento gaussiano tal que, las relaciones entre la función *f(x)* y los valores observados y_i están dados por:

$$y_i = f(x_i) + \varepsilon_i \quad \text{donde } \varepsilon_i \sim \mathcal{N}(0, \sigma^2_{ruido}) \tag{34}$$

ε_i representa el ruido gaussiano que tiene media cero y varianza σ^2_ε. Estos procesos son una aproximación bayesiana[3] que, asume que los valores de la función se comportan de acuerdo a:

$$p(f|x_1, x_2, \ldots, x_n) = \mathcal{N}(0, K) \tag{35}$$

Donde $f = [f_1, f_2, \ldots, f_n]^T$ es un vector de valores de funciones latentes $f_i=f(x_i)$ y, *K* es la matriz de covarianza, cuyas entradas están dadas por la función de covarianza $K_{ij}=k(x_i, x_j)$. Cabe resaltar que, los valores de las funciones latentes, son tratados como funciones aleatorias indexadas por su correspondiente entrada. La inferencia en el modelo es simple: se coloca un conjunto de funciones antes del entrenamiento y, se prueban los valores latentes *f* y *f** para combinarlos con la probabilidad *p(y|f),* usando la regla de Bayes para así obtener:

$$p(f, f^*|y) = \frac{p(f, f^*)p(y|f)}{p(y)} \tag{36}$$

El paso final para producir la distribución predictiva es, marginar las variables patentes del conjunto de entrenamiento no deseado:

$$p(f, f*|y) = \int p(f, f*|y)df = \frac{1}{p(y)} \int p(f|y)p(f, f*)df \tag{37}$$

La distribución predictiva es entonces, el marginal del conjunto re normalizado antes de la probabilidad de acuerdo al siguiente modelo:

$$p(f, f*) = \mathcal{N}\left(0, \begin{bmatrix} K_{f,f} & K_{f*,f} \\ K_{f,f*} & K_{f*,f*} \end{bmatrix}\right) \; y \; p(y|f) = \mathcal{N}(f, \sigma^2_{ruido} I) \tag{38}$$

Donde *K* es la matriz de covarianza entre las variables, e *I* es la matriz identidad. Derivado de que ambos factores de la integral (37) son gaussianos, esta puede ser evaluada para dar la distribución predictiva:

$$p(f*|y) = \mathcal{N}(K_{f*,f}(K_{f,f} + \sigma^2_{ruido} I)^{-1} y, K_{f*,f*} - K_{f*,f}(K_{f,f} + \sigma^2_{ruido} I)^{-1} K_{f,f*}) \tag{39}$$

El principal problema en la ecuación anterior, es que requiere la transpuesta de la matriz de tamaño *"n x n".* Esto implica $O(n^3)$ operaciones, donde *n* es el número

[3] Asume que, se puede hacer referencia a un parámetro por medio de distribuciones de probabilidad.

de datos de entrada. Por lo tanto, este algoritmo solo puede lidiar con unos cuantos cientos de datos. (Quinonero-Candela & Rasmussen, 2005)

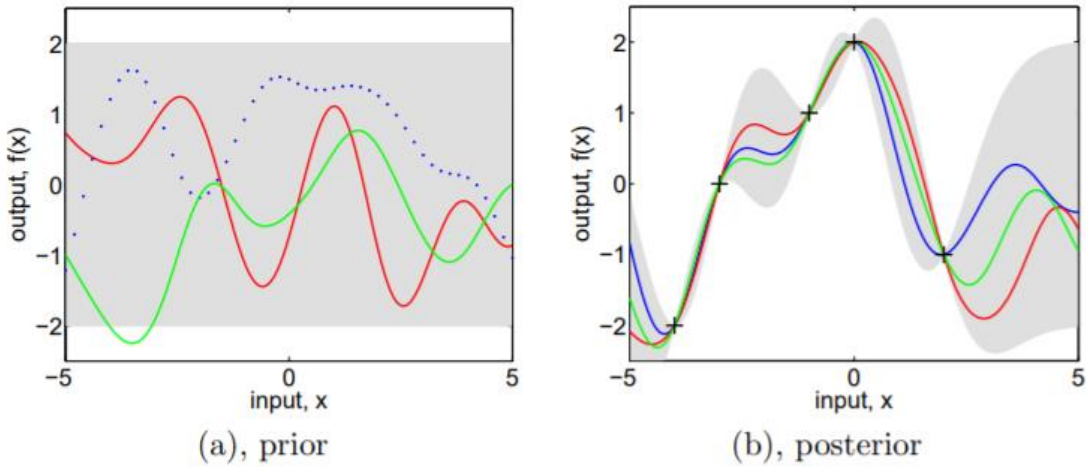


Figura 15. (a) Proceso Gaussiano a prior, las líneas de tendencia (azul, rojo, verde) representan las funciones antes del entrenamiento. El sombreado en gris representa la desviación estándar. (b) proceso Gaussiano posterior al entrenamiento, el área sombreada es la media puntual ± dos veces la deviación estándar, con una confianza del 95% para cada punto. (Rasmussen & Williams, 2006)

2.6.2 Funciones de covarianza o kernels

Los kernels no son más que las funciones de covarianza descritas anteriormente. Tienen la propiedad de ser definidas y positivas, con dos entradas *f* y *f** , las cuales son vectores en un espacio euclidiano (también pueden ser gráficos, e incluso texto). Sirven para determinar la generalización de propiedades del modelo.

Cada función de covarianza, tiene un número especifico de parámetros, que nos dará una forma particular como resultado. Comúnmente se conocen como “hiper-parámetros”, ya que pueden verse como especificaciones de una distribución, sobre los parámetros de la función. Estos son: la escala de longitud, la varianza de los datos y la varianza del ruido. (Kristjanson Duvenaud, 2014)

Como se puede observar en la Figura16a, se maximiza la probabilidad marginal (probabilidad de máxima separación entre los datos), lo que da como resultado un buen ajuste en las cifras experimentales. La Figura 16b contiene una varianza muy grande en comparación con la Figura 16a. Una escala pequeña en el kernel, provoca el comportamiento mostrado en la Figura 16c, mientras que un parámetro de escala grande da como resultado la figura 16d.

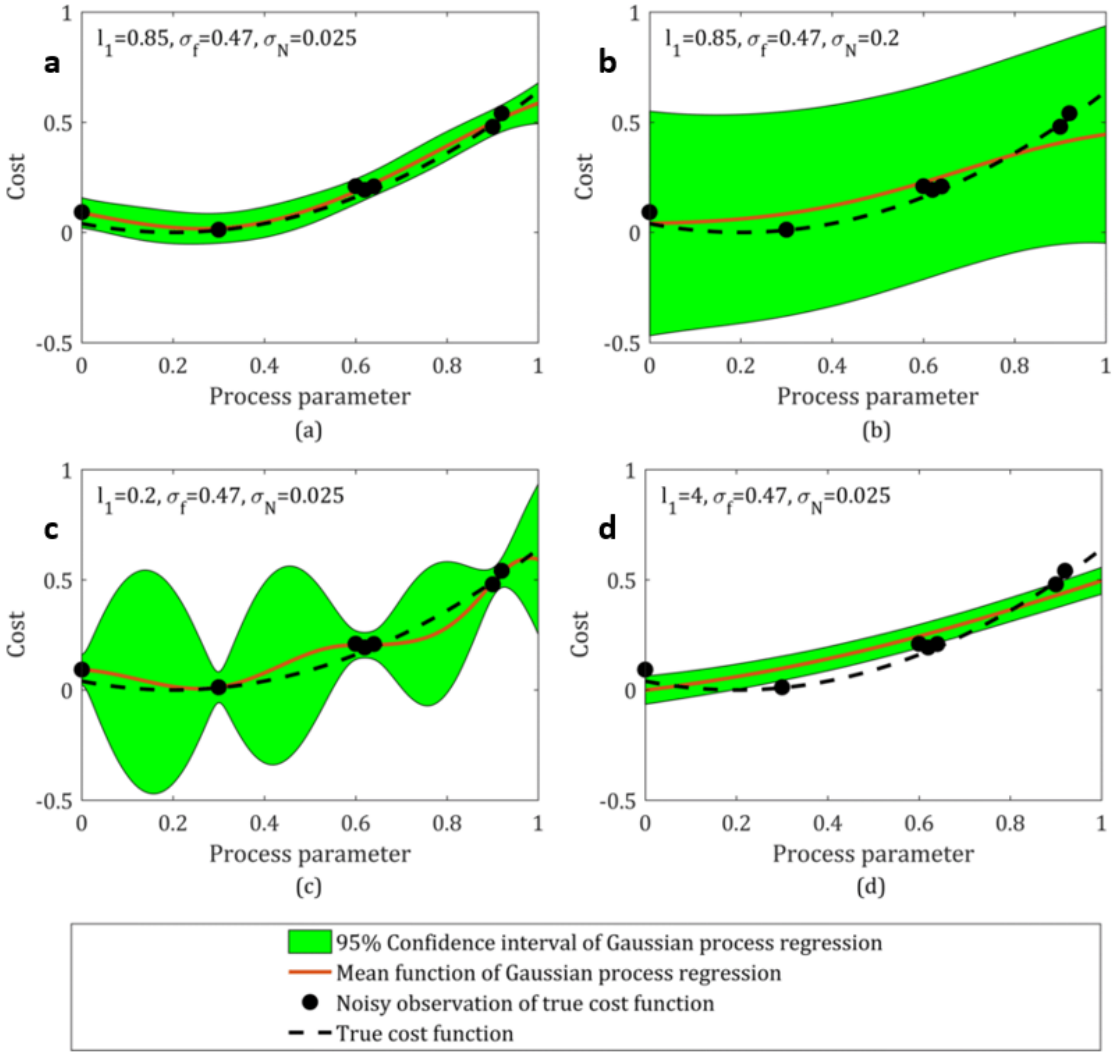


Figura 16. Ejemplo de regresión con proceso gaussiano variando diferentes hiper-parámetros. (Maier, Rupenyan, Bobst, & Wegener, 2020)

3. HIPOTESIS Y OBJETIVOS

3.1 Hipótesis

La implementación de modelos de aprendizaje automático, determinará los parámetros de validación del método, a través de, un conjunto de datos de referencia simulados por computadora. Estos algoritmos darán un símil para el límite de cuantificación, robustez, incertidumbre y error para el método modificado de duplicación de J en el dominio de las frecuencias.

3.2 Objetivo principal

Desarrollar una metodología de validación completa, basada en tres algoritmos de aprendizaje automático: redes neuronales, máquinas de vector soporte y procesos gaussianos.

3.3 Objetivos particulares

- Simular multipletes de RMN, que servirán como patrón de referencia en la creación de la base de datos. Estos datos serán las entradas del entrenamiento de los diferentes algoritmos.
- Diseñar un entorno de trabajo en donde sea posible procesar señales experimentales. Estas señales, serán previamente procesadas para obtener los parámetros correspondientes y, generar predicciones con inteligencia artificial.
- Reportar para cualquier determinación del método modificado de duplicación de J, una corrección y su correspondiente incertidumbre con factor de cobertura del 95% .

4. DESARROLLO EXPERIMENTAL

Las variables de entrada a los diferentes algoritmos son:

- Constante de acoplamiento determinada: J_{det}
- Constante de acoplamiento teórica: J_{real}
- Ancho de señal: W
- Subarmónico: Sh
- Distancia: D
- Relación señal ruido: SNR
- Error

Dichas variables fueron generadas a través de simulación de señales de RMN, cuyos datos se recopilaron en una base de datos. Posteriormente, con esta base de datos, se entrenaron tres algoritmos: redes neuronales hacia adelante, máquinas de vector soporte y procesos gaussianos de regresión. Finalmente, se evaluó su desempeño.

4.1 Método modificado de duplicación de J

El entorno de trabajo se desarrolló en el lenguaje Python, versión 3.8.6. La documentación utilizada fué NumPy, Pandas y MatplotLib. Para la sección 4.3 se utiliza la paquetería "NMRsim", la cual es una librería especializada en simulación de espectros de RMN, creada por Geoffrey M. Sametz, profesor en la Universidad de Delaware. El editor de código principalmente utilizado es Visual Studio Code para Windows 10, sin embargo, también se utilizaron entornos en línea, tal como Google Colaboratory.

J Doubling fue diseñado a partir de programación orientada a objetos. Se trata de una superclase abstracta, que servirá como base para aplicar todos los métodos y atributos propios del algoritmo, representado en lenguaje UML (Figura 17).

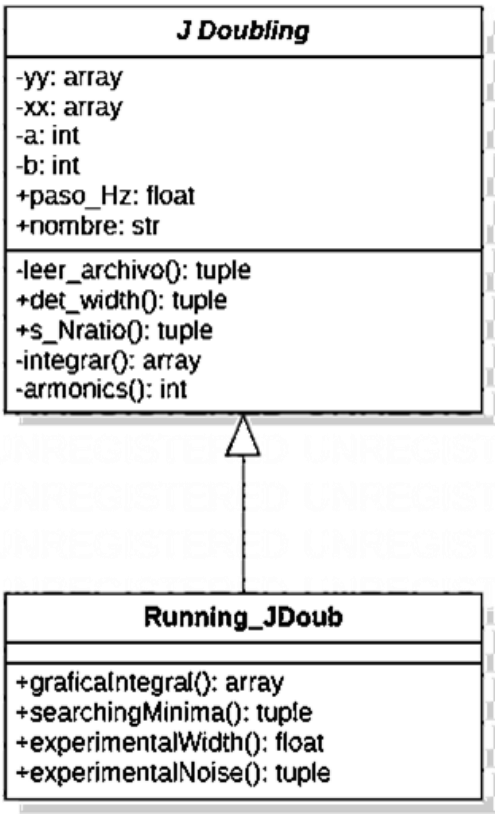


Figura 17. Diagrama UML de la superclase abstracta J Doubling, y la herencia de sus atributos a la subclase Running_JDoub, la cual procesa cualquier señal experimental.

Como se puede observar en la Figura 17, la subclase "*Running_JDoub*" es la encargada de generar todas las salidas necesarias, que después servirán como nuevas entradas para los algoritmos de inteligencia artificial.

Los datos de entrada para analizar cualquier señal son dos: *xx* y representa el intervalo en Hz donde se encuentra situada la señal, mientras que *yy* es la intensidad relativa de la misma. Adicionalmente se necesitan otros dos valores, denominados como *a* y *b,* los cuales son el primer y último valor del arreglo *xx*. Estos dan como resultado una de las variables más importantes, denominada como "resolución digital*",* e indica la taza del número de puntos en el intervalo donde se encuentra la señal.

4.2 Determinación de SNR y W en señales experimentales

Para determinar SNR y W se utilizaron dos algoritmos, representados en los diagramas de flujo de la Figura 18. Cabe resaltar que pertenecen a los métodos de la superclase "J Doubling"*,* y por tanto, heredan los atributos de cualquier señal *xx* y *yy,* siendo los datos de entrada para ejecutar el programa.

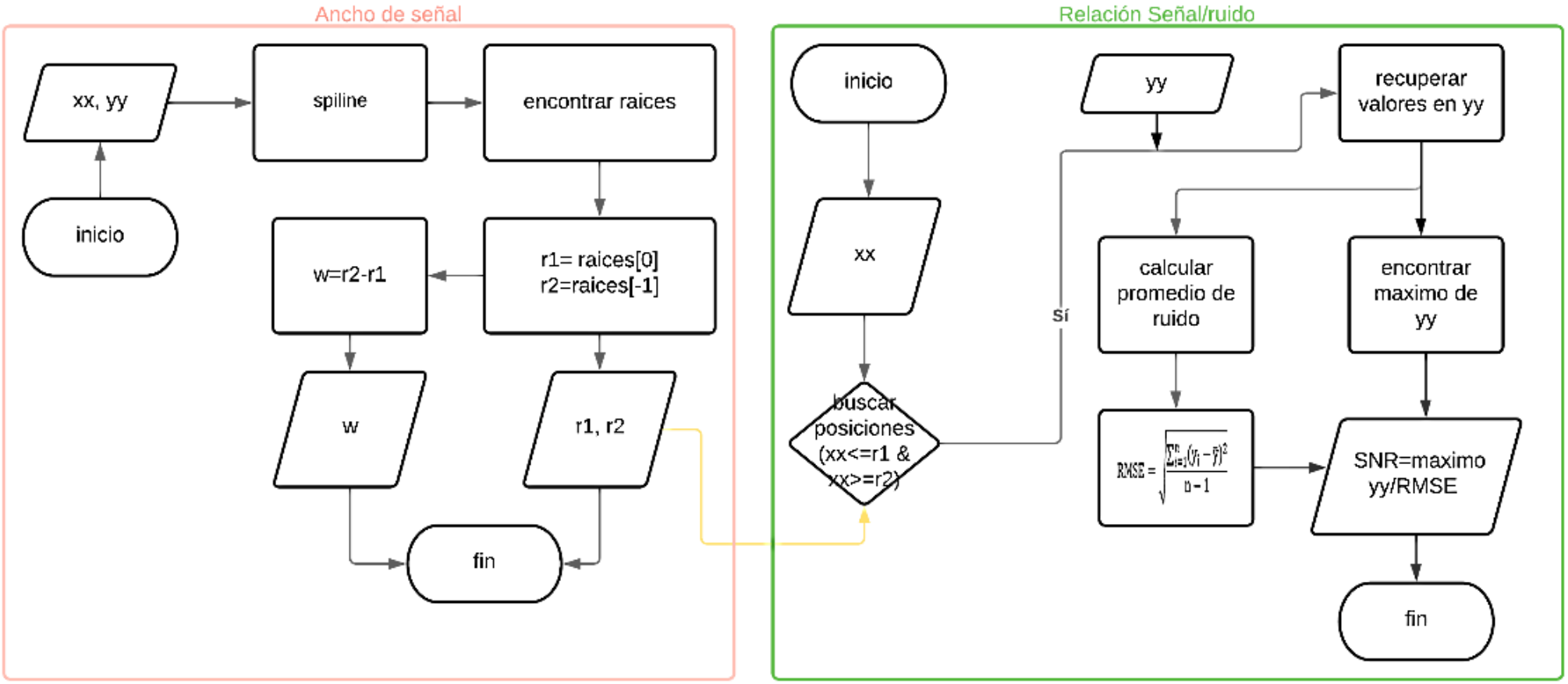


Figura 18. Diagramas de flujo para la determinación experimental del ancho de señal y relación señal ruido en cualquier multiplete.

El método “spline” hace referencia a “UnivariateSpline”, correspondiente a una subrama dedicada al análisis de señales de SciPy. El algoritmo consiste en, ajustar una función tipo spline y=spl(x) de grado k a los datos proporcionados, para después, encontrar las raíces del polinomio ajustado (SciPy, 2021). Un ejemplo se observa en la Figura 19, y muestra la determinación del ancho de señal a media altura.

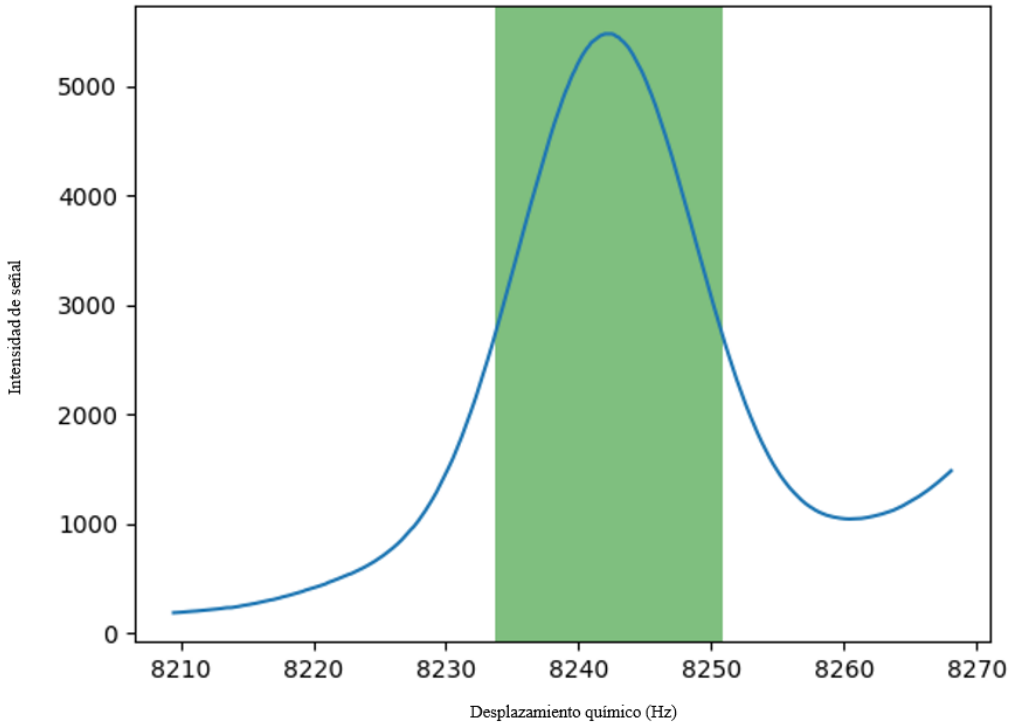


Figura 19. Determinación del ancho a media altura en señal experimental: W= 17.06 Hz señalada en color verde

Por último, el algoritmo de SNR, consiste en encontrar los valores mínimo y máximo que determina del ancho de señal, para luego seleccionar únicamente las colas, en donde se aplicará la ecuación:

$$\mathrm{SNR} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

descrita a detalle en la sección 2.1.2 del marco teórico.

4.3 Simulación de espectros de RMN y creación de base de datos

Todas las simulaciones fueron realizadas con ayuda del proyecto NMRsim (versión 0.5.2 beta) creado por Geoffrey M. Sametz. Este entorno provee, herramientas de Python para simulaciones de espectros de primer y segundo orden en sistemas de espín ½. Se utiliza la clase “Multiplet”, cuyos argumentos requieren los parámetros: frecuencia central v, intensidad de señal I y constante de acoplamiento J. Los argumentos se vuelven atributos del multiplete y, cada entrada en la lista es una tupla, la cual contiene: [J en Hz, número de núcleos

causando el acoplamiento]. (Sametz, 2021). Cabe destacar que, cada señal de RMN es simulada a partir de las Lorentzianas descritas en la sección 2.1.2

Un ejemplo sencillo es representado en la Figura 20, dicho código se puede adaptar a prácticamente cualquier señal que cumpla las características antes descritas.

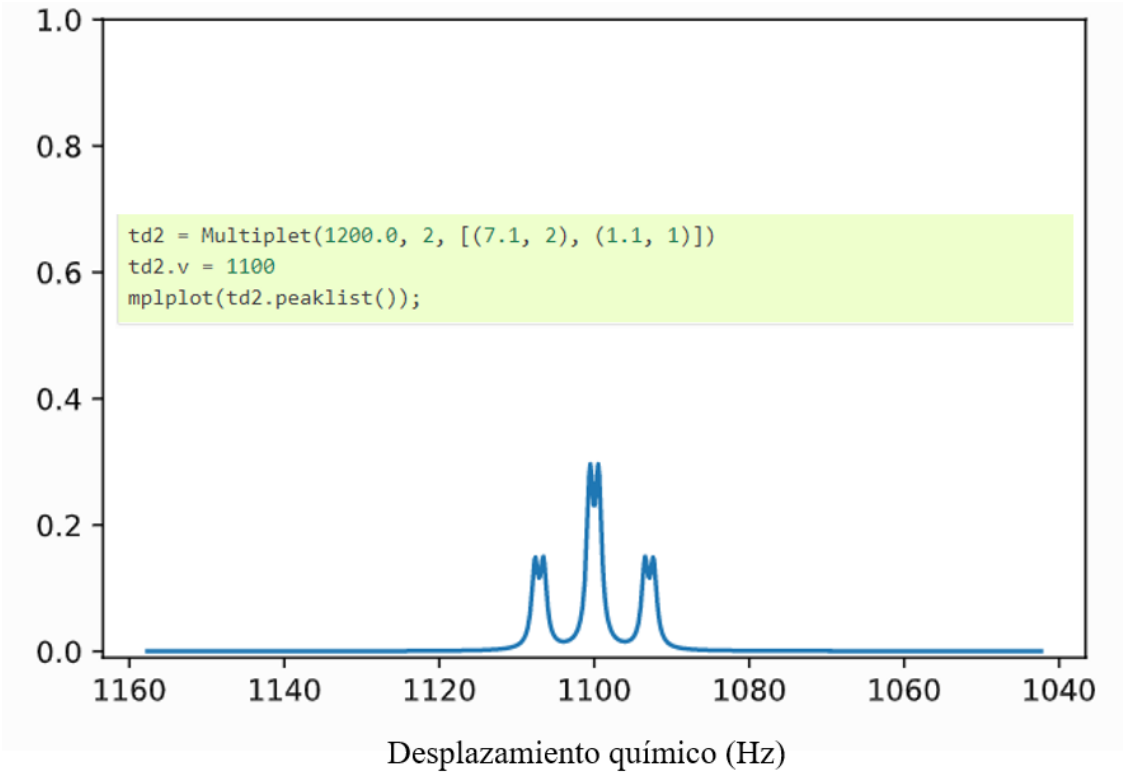


Figura 20. Simulación de multiplete a través de la clase Multiplet. Sus atributos pueden ser modificados como el desplazamiento químico, de 1200 Hz a 1100 Hz.

La base de datos fue creada con doscientos cincuenta mil señales, las cuales se emplearán como referencia, y entrenarán los diferentes algoritmos de aprendizaje automático. Toda la información se almacenó con el formato JSON o JavaScript Object Notation, debido a la alta compatibilidad entre lenguajes. (Taylor, 2014)

Las variables más influyentes a la hora de ejecutar J Doubling son 2: el ancho de la señal y la cantidad de ruido, por lo que, para cada constante de acoplamiento, se deben analizar dichos parámetros provenientes de la señal que la genera. Éstas dos variables afectan directamente a otras dos: el primer subarmónico Sh, y la distancia entre el mínimo de la constante de acoplamiento y el subarmónico D.

El parámetro que se utilizó para evaluar el desempeño de J Doubling es el error, ya que al conocer la referencia (J teórica), se puede saber bajo qué condiciones el método trabaja de manera adecuada o no.

El intervalo de trabajo usado en la simulación de señales, abarca una constante de acoplamiento de (0.5 a 12.0) Hz. Se utilizó una resolución digital de 0.04 Hz

para cada simulación, mientras que la variación en el intervalo de trabajo para las diferentes J a entrenar fue de, 0.4 Hz. Cabe destacar que, la división mínima para la generación de datos de entrenamiento, se eligió con base en dos criterios principales. El primero considera el coste computacional propio de J Doubling, ya que el proceso de deconvolución es matemáticamente iterativo, además de que, la integración para obtener la J, es la responsable del alto tiempo de ejecución. El segundo criterio se deriva directamente del sobre ajuste en los modelos de IA, ya que, es preferible tener menos datos los cuales abarquen todo el intervalo de trabajo, a demasiados de ellos y por ende, disminuya la robustez ante nuevas entradas.

4.4 Redes neuronales

4.4.1 Preprocesamiento de datos de entrenamiento

El objetivo de este algoritmo, es estudiar bajo qué condiciones J Doubling determinará correctamente la constante de acoplamiento, ya que, se conoce la constante de acoplamiento real $Jr_{eal.}$ Para esto, se busca predecir el error bajo los siguientes parámetros de entrada:

- Constante de acoplamiento determinada: J_{det}
- Ancho de señal: W
- Subarmónico: Sh
- Distancia: D
- Relación señal ruido: SNR

Como se mencionó anteriormente, es posible conocer el error en el método, ya que, al tratarse de datos de referencia simulados, se puede calcular dicho parámetro con total certeza. Se asume que un error pequeño en comparación con el mensurando, es una buena determinación, en cambio, un error grande comparado con el mensurando significa una mala determinación. La taza de diferencia no debe ser mayor al 10%.

Un caso especial en las variables de entrenamiento es, la distancia entre el subarmónico y la constante de acoplamiento. Se calcula de manera numérica como:

$$D = J - Sh$$

Sin embargo, existe la condición límite donde el algoritmo de J Doubling no es capaz de detectar el subarmónico, de acuerdo a:

$$\lim_{Sh \to 0} D(Sh) = J \qquad (40)$$

Cuando se cumple la condición anterior, se observa el fenómeno representado en la Figura 21, en donde dentro del comportamiento del subarmónico, se replica la tendencia propia de la constante de acoplamiento. Por ende, crea un falso positivo en los datos de entrenamiento, propiciando futuros errores. Por esta razón, es necesario realizar una limpieza previa, y retirar todos los casos donde se presente dicha anomalía.

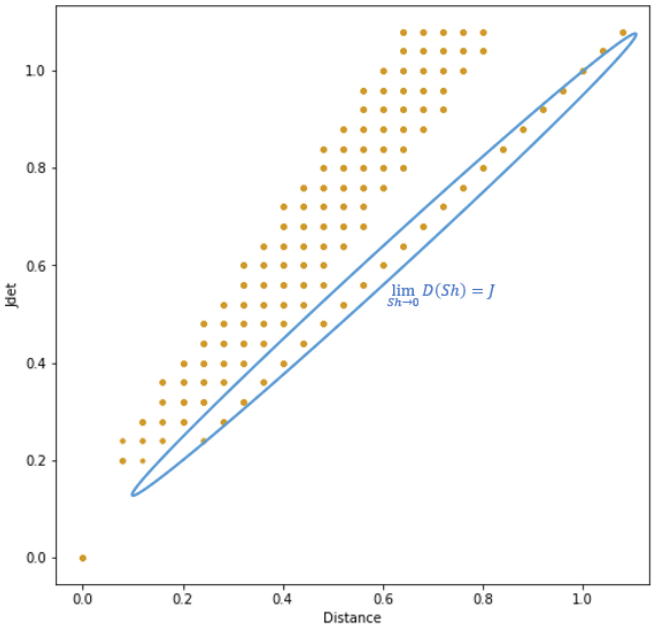


Figura 21. Influencia del subarmónico a la J determinada, con representación gráfica del caso límite.

Una vez filtrados los datos, se procedió a preprocesarlos por medio del algoritmo “Z-score”, el cual es simplemente, una normalización estándar de acuerdo a la ecuación:

$$N = \frac{x_i - \mu}{\sigma} \quad (41)$$

Donde xi representa cada dato a normalizar, μ es la media de los datos en cada columna de entrenamiento y σ la desviación estándar de cada columna. Posteriormente, se realizó una reducción de dimensionalidad con ayuda del algoritmo “Partial Leasts Squares” o PLS. Esto es con el objetivo de reducir la cantidad de variables de entrada, y a la vez, disminuir su varianza, para así, tener los datos listos para el entrenamiento.

Por último, los datos son divididos en dos categorías: entrenamiento y prueba, en una proporción 80:20. Dichos datos pasan a la red neuronal, siendo X la entrada y la salida o valor a predecir, y.

4.4.2 Arquitectura y Entrenamiento

Se utilizó una red neuronal FeedForward de regresión, ya que como se comentó anteriormente, el objetivo es realizar predicciones que corrijan las J determinadas por J doubling.

Un resumen de los parámetros de la red, se encuentra reportado en la Tabla 3. En ella, se tiene una capa de entrada con 6 neuronas y función de activación sigmoide, una primera capa oculta con 6 neuronas y función de activación sigmoide, una segunda capa oculta con 6 neuronas y función de activación ReLu, para dar como resultado, una única salida.

Tabla 3. Resumen de parámetros para red neuronal de regresión. Realizada con TensorFlow/Keras + neurona Bias

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 6)                 24
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 42
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 42
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 7
=================================================================
Total params: 115
Trainable params: 115
Non-trainable params: 0
_________________________________________________________________
```

Con el objetivo de evitar un sobreajuste en los datos, se realizó un entrenamiento tipo "*Early Stopping",* del cual se discutirá a fondo en la sección de resultados.

4.4.3 Evaluación de desempeño

El conjunto de datos de prueba (separado en la sección 4.4.1), tiene el objetivo de actuar como puntos independientes al entrenamiento, por lo que son una referencia en la evaluación del desempeño, ya que jamás han sido vistos por el algoritmo. Las métricas utilizadas son MAE y MSE.

Posteriormente, se compararon las predicciones de la red con los datos originales, y se realizó una inspección visual por medio del gráfico, "original/predicho". Se finaliza haciendo una confrontación entre las métricas de desempeño, del conjunto de datos de entrenamiento y prueba, con el propósito evaluar si hubo alguna anomalía u ocurrió un sobre ajuste del modelo.

4.5 Máquinas de vector soporte

4.5.1 Preprocesamiento de datos de entrenamiento

Este algoritmo, es utilizado con el propósito de analizar la influencia de SNR en el error propio de J Doubling, o en otras palabras, ver bajo qué condiciones de ruido, J Doubling se equivoca menos. Para logar dicho propósito, no es necesario realizar transformaciones previas en los datos, ya que solo se está trabajando en dos dimensiones: SNR y $J_{det.}$ Adicionalmente, se agruparán los datos bajo la condición de, buena determinación = 0 o mala determinación = -1.

El criterio para saber a qué categoría pertenece cada dato, es una tolerancia máxima del 10% de error en la constante de acoplamiento de referencia, por lo

que, si se tiene una J_{ref} =1.0 Hz y una SNR=54.3, J Doubling devuelve una J_{det} =0.5 Hz, cayendo dentro de la condición -1.

4.5.2 Especificaciones del modelo

Se utilizó la documentación correspondiente a “scikit-learn” de su módulo “*Support Vector Classifier”*, eligiendo un modelo de clasificación binaria, entre las dos clases descritas anteriormente. El kernel más adecuado es radial, ya que se espera obtener una función, la cual describa el comportamiento de los datos (no es posible separarlos con una simple línea de tendencia).

4.5.3 Evaluación del desempeño

La métrica de desempeño en este caso es el “accuracy”, o la exactitud con la que clasifica el algoritmo. También se utiliza la matriz de confusión discutida en la sección 2.6, con el objetivo de evaluar de manera más específica y, en función de los datos de prueba, su capacidad de clasificación.

4.6 Procesos gaussianos

4.6.1 Preprocesamiento de datos de entrenamiento

En este caso, se aplica la misma secuencia de la red neuronal: limpieza del caso límite en Sh, normalización estándar y reducción de dimensionalidad con PLS, para finalizar dividiendo los datos en entrenamiento y prueba. Un paso adicional después de PLS, es un muestreo aleatorio simple para poblaciones finitas, debido a la gran carga computacional que genera algoritmo (sección 2.7.1).

La diferencia radica en los datos de entrada X, los cuáles son:

- Ancho de señal: W
- Subarmónico: Sh
- Distancia: D
- Relación señal/ruido: SNR

Mientas que el valor a predecir y, es la propia constante de acoplamiento, ya que el objetivo de este algoritmo, es analizar la influencia de las diferentes variables hacia la J, para que con estos datos, se pueda reportar una incertidumbre estadística con factor de cobertura del 95%.

4.6.2 Especificaciones del modelo

Dentro de la documentación de TensorFlow, existe una subcategoría dedicada a los procesos gaussianos, denominada “GPflow”. Una de sus funciones principales es, realizar modelos de regresión denominados como “GPR”, los cuales serán utilizados para abordar el problema.

Dado que se realizó una reducción de dimensionalidad a 3 componentes, se utilizaron 3 kernel polinomiales, uno para cada componente. Cabe resaltar que, se eligió un polinomio en vez de un modelo lineal, ya que, estos pueden abordar más tipos de comportamiento, incluido el grado 1, y por lo tanto, abarcar más

casos por si los datos así lo requieren. Un resumen rápido del modelo antes de la optimización, se observa en la Tabla 4.

Tabla 4. Resumen de parámetros para proceso Gaussiano de regresión con tres kernel polinomiales. Las primeras 6 líneas corresponden a parámetros del kernel, mientras que la última, representa el parámetro de probabilidad de ruido agregado τ^2.

name	class	transform	prior	trainable	shape	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus		True	()	float64	1
GPR.kernel.kernels[0].offset	Parameter	Softplus		True	()	float64	1
GPR.kernel.kernels[1].variance	Parameter	Softplus		True	()	float64	1
GPR.kernel.kernels[1].offset	Parameter	Softplus		True	()	float64	1
GPR.kernel.kernels[2].variance	Parameter	Softplus		True	()	float64	1
GPR.kernel.kernels[2].offset	Parameter	Softplus		True	()	float64	1
GPR.likelihood.variance	Parameter	Softplus + Shift		True	()	float64	1

Existen varios optimizadores en GPflow, pero el más utilizado y por defecto es con “Scipy”, el cual implementa el algoritmo L-BFGS-B, siendo este el elegido. Para el entrenamiento, se necesita maximizar la probabilidad marginal logarítmica del modelo, siendo en este caso, la probabilidad marginal logarítmica negativa. Otro método de entrenamiento es con “Markov Chain Monte Carlo” ó MCMC, sin embargo, se encuentra por fuera de los objetivos del proyecto, optando por maximizar la probabilidad. (GPflow, 2020)

4.6.3 Evaluación del desempeño: prueba de normalidad Shapiro Wilk y t de Student para muestras independientes

El test de Shapiro Wilk se emplea, para contrastar la normalidad cuando un tamaño de muestra es menor a 50 observaciones. Este método, consiste en ordenar la muestra de menor a mayor, con el objetivo de obtener un nuevo vector muestral. Se rechaza la hipótesis nula de normalidad si el estadístico de prueba “W”, es menor que el valor crítico proporcionado por las tablas (reportadas para un tamaño de muestra n y nivel de significancia del 95%, para este caso). El estadístico se define como:

$$W = \frac{\sum_{i=1}^{n}(\alpha_i * Y_{último} - Y_{primero})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Donde Yi son los datos de la muestra ordenados por tamaño, Yúltimo & Yprimero son los datos previamente ordenados, αi son los coeficientes de Shapiro Wilk, y $\bar{Y}$ es la media de los datos. (Flores Tapia, 2021)

Para la prueba t de una muestra independiente, se realiza una prueba de hipótesis, con el objetivo de establecer si la media de la población desconocida, es igual a la de un valor específico y previamente conocido. La hipótesis nula se acepta cuando el estadístico de prueba, cae dentro del intervalo dado por el estadístico “t crítico” para un número de casos n, y cierto nivel de significancia. Se calcula como:

$$t = \frac{\bar{Y} - Y}{s/\sqrt{n}}$$

Donde $\bar{Y}$ es la media muestral, Y la media poblacional, s la desviación estándar y n el tamaño de la muestra. (Pinilla, 2011)

4.7 Entorno de trabajo completo para el entrenamiento de los algoritmos

Haciendo una recapitulación de los puntos anteriores, se creó una base de datos con señales simuladas, con el objetivo de entrenar tres algoritmos diferentes que validen J Doubling. Un esquema detallado se encuentra reportado en la Figura 22. De manera sencilla, se observa cada uno de los pasos en preprocesamiento de datos, y como estos se dividen para la evaluación de desempeño, junto con las salidas de cada algoritmo.

4.8 Entorno de trabajo completo para datos experimentales

Una vez entrenados todos los algoritmos, estos están listos para realizar predicciones en señales experimentales. Se trabajó con señales de proteínas, las cuales tienen la característica particular, de ser sumamente anchas, donde los máximos no son apreciables.

El archivo que contiene los datos del multiplete, debe contener las siguientes características, y estar en formato ".slc":

> Number of data points : 610 #no. de puntos que describen al multiplete
> Chemical shift range (ppm) : 7.170974 7.059341 #valor maximo/minimo en ppm
> Chemical shift range (Hz) : 6809.951172 6703.938965 #valor maximo/minimo en Hz
> 214 #puntos que describen al multiplete (intensidades)
> 216
> 219
> 221

Posteriormente, la señal será procesada por J Doubling, la cual determinará la constante de acoplamiento y otros parámetros descritos en la Figura 23, para posteriormente, entrar a los diferentes algoritmos de IA, y dar como salida final, los parámetros de validación: límite de cuantificación, incertidumbre tipo A y la corrección correspondiente a cada determinación.

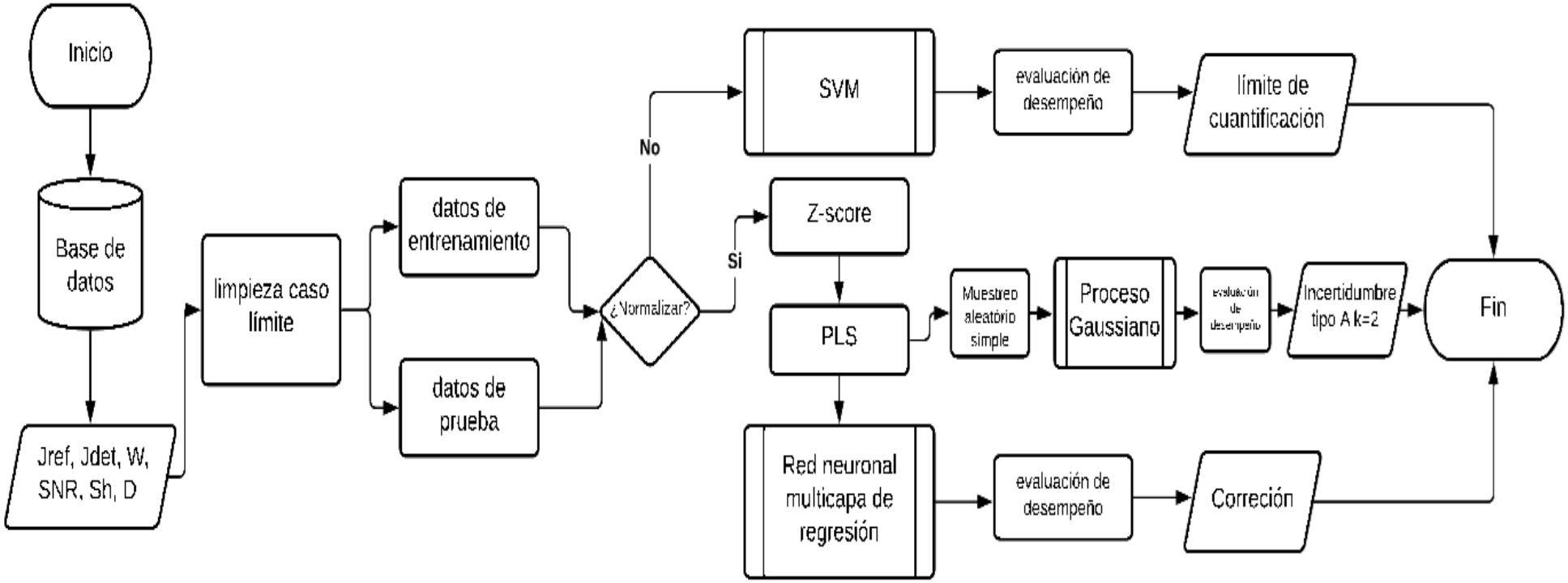


Figura 22. Diagrama de flujo para la representación del macro-algoritmo de validación de J Doubling.

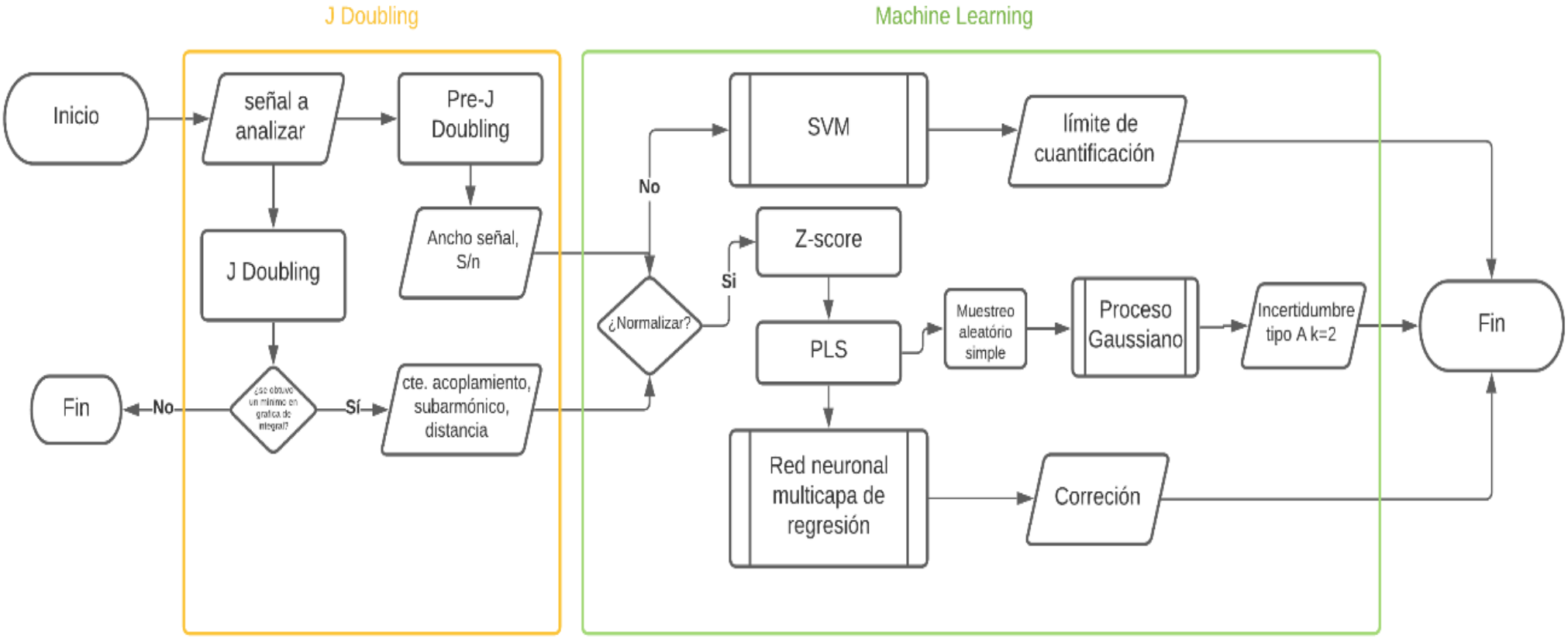


Figura 23. Diagrama de flujo para el procesamiento de señales experimentales en el entorno de J Doubling/Inteligencia Artificial para la validación de resultados.

5. RESULTADOS

5.1 Evaluación de desempeño de los algoritmos

5.1.1 PLS o mínimos cuadrados parciales

Se utilizó este algoritmo con el objetivo de, reducir la dimensionalidad de los datos de entrada, tanto para la red neuronal, como para el proceso gaussiano. La evaluación del desempeño de PLS o mínimos cuadrados parciales, se realiza mediante bucles, donde se determina con cuantas componentes es más adecuado trabajar el modelo. En este caso, se realizó un bucle de 1 a 6 componentes, haciendo uso de la validación cruzada con el método de "K-Fold", y 10 lotes. Dichos resultados se reportan en la Figura 24.

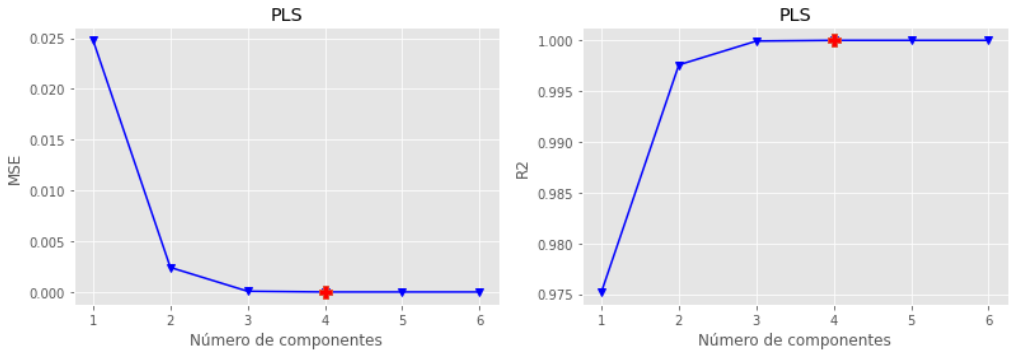


Figura 24. Monitoreo de MSE y r^2 , en función del número de componentes para el preprocesamiento de datos.

Después de ver que, el número más adecuado de componentes es cuatro, se procedió a observar el comportamiento de los datos de manera gráfica. Una muestra de las componentes después de ser reducidas al valor óptimo, se aprecia en la Figura 25. En dicha figura, se contrasta el comportamiento entre tres y cuatro componentes.

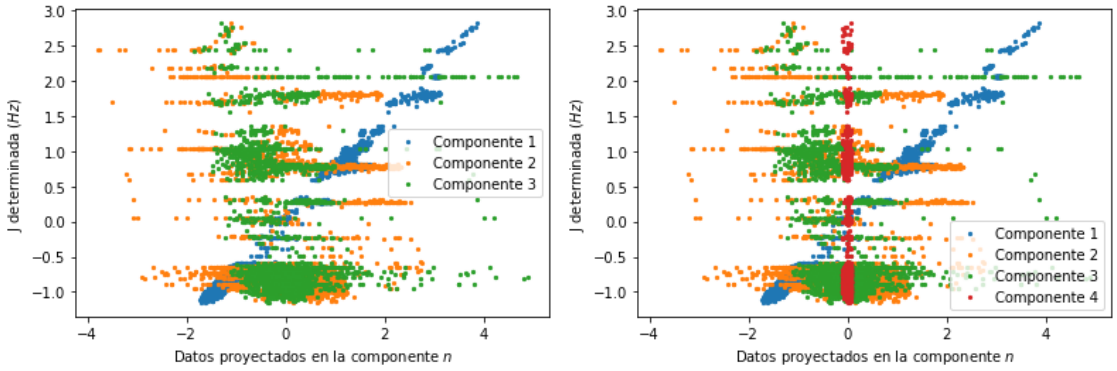


Figura 25. Comportamiento de las componentes después de la reducción de dimensionalidad en los datos de entrenamiento/prueba con PLS. (izquierda) tres componentes. (derecha) cuatro componentes.

Las métricas de desempeño para tres y cuatro componentes, se observa en la Tabla 5. Se puede observar que, el coeficiente de determinación para cuatro componentes es perfecto, lo cual provocaría un posterior sobre ajuste en los demás algoritmos.

Tabla 5. Métricas de desempeño en función del número de componentes límite

Componente/métrica	MSE	r^2
3	0.000072	0.999927
4	0.000001	1.000000

5.1.2 Red neuronal

Se evaluó el desempeño de este algoritmo realizando varias comparaciones. La primera de ellas es cotejar las métricas, MAE y MSE en función de los "epoch" o épocas, tanto para los lotes de entrenamiento y validación. Dichas gráficas se muestran en la Figura *26*.

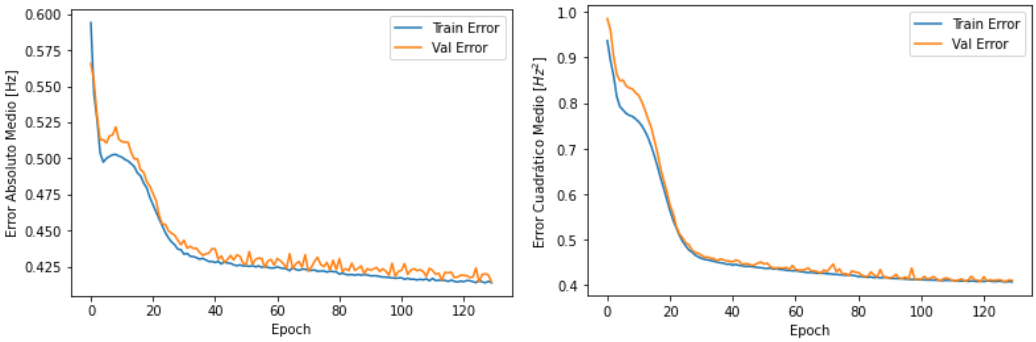


Figura 26 (derecha). Evaluación del desempeño del entrenamiento con MAE.(izquierda). Evaluación del desempeño del entrenamiento con MSE

Posteriormente, se recupera el conjunto de datos de prueba, con el objetivo de comparar los valores reales vs. los predichos con la red neuronal. En la Figura 27 se puede apreciar la diferencia entre estos.

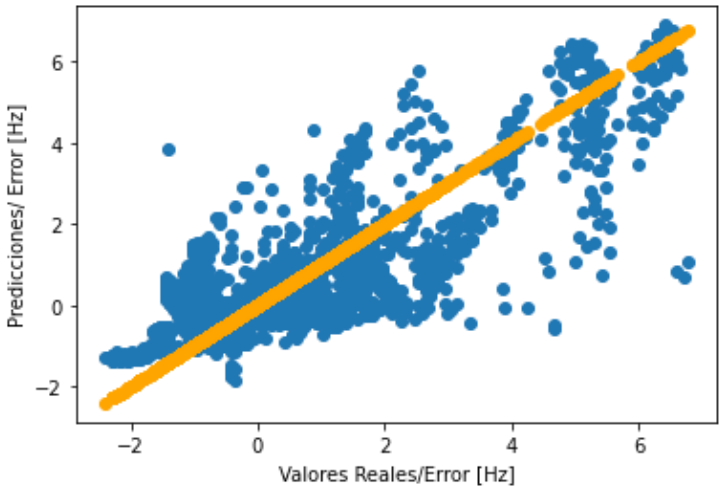


Figura 27. Comparación visual entre las predicciones de la red neuronal (naranja) y los valores reales del conjunto de pruebas (azul).

Por último, se comparan numéricamente las métricas totales MAE y MSE, en donde se confronta el conjunto de entrenamiento y prueba reportado en la Tabla *6*. Esto se hace con el objetivo de confirmar numéricamente que, no existe una diferencia significativa entre los datos reales y los predichos.

Tabla 6. Comparación de métricas de desempeño entre los conjuntos de datos de entrada para red neuronal multicapa.

Conjunto	**MAE (Hz)**	**MSE (Hz)**
Entrenamiento	0.4102	0.4063
Pruebas	0.4228	0.4255

5.1.3 Máquina de vector soporte

El algoritmo utilizado es “Soft Margin SVM” o “Support Vector Clasifier”, el cual permite un cierto margen de error al determinar el hiperplano de separación, observado en la Figura 28. Se trabajó con un kernel radial, ya que, dado el comportamiento visual de los datos, ajustar una recta sería poco acertado, derivado de la creciente cantidad de casos no permitidos en las vecindades de 1 Hz .

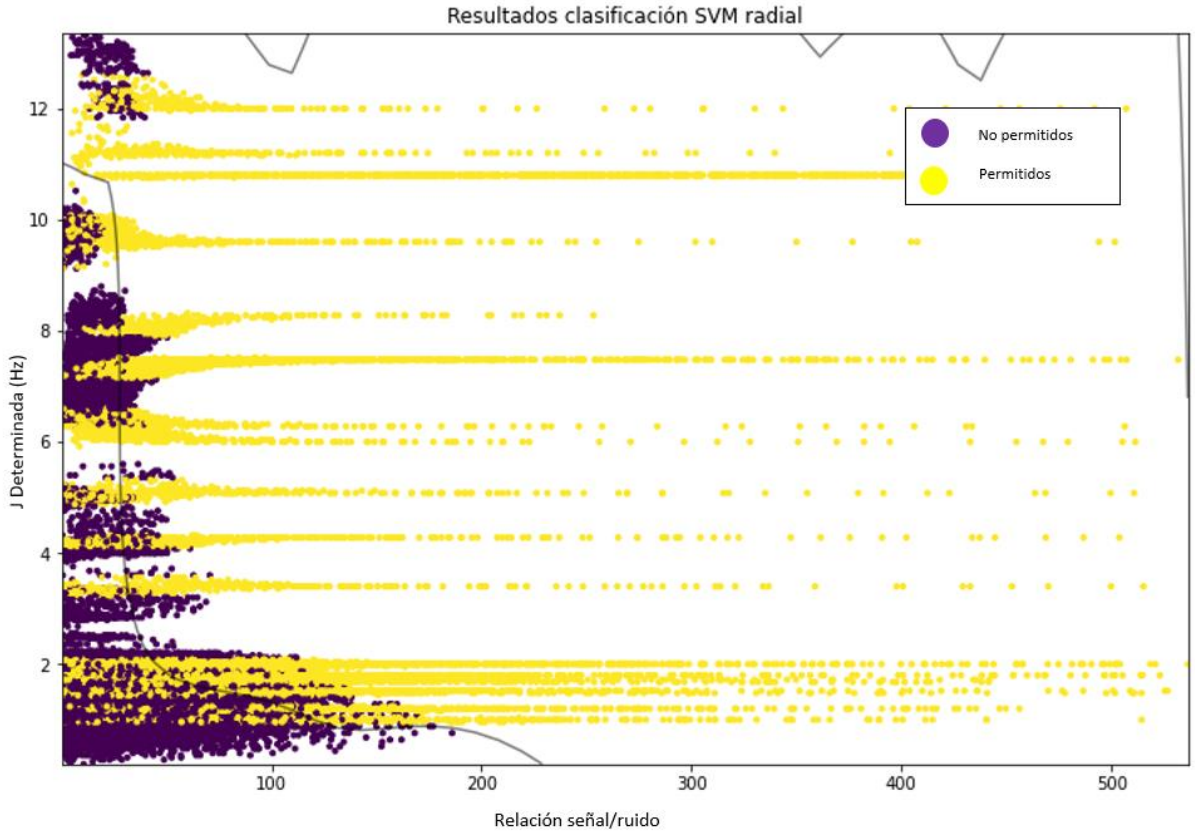


Figura 28. Clasificación con SVM con tolerancia máxima del 10% con respecto a la referencia original de SNR.

La métrica de desempeño a evaluar en este caso, es la precisión y el hiperparámetro C óptimo (penaliza la cantidad de errores aceptados dentro de la clasificación), mostrado en la Tabla *7*.

Tabla 7. Métricas de desempeño para SVM en el conjunto de datos de entrenamiento.

Conjunto de datos	*C*	*precisión*
Entrenamiento	$5.4555x10^5$	0.7629

Para evaluar el desempeño del modelo ante datos externos al entrenamiento, se recurre al conjunto de pruebas, en donde se calculará la matriz de confusión, reportada en la Tabla *8*. Su interpretación se discutirá a profundidad en la sección 6, análisis de resultados.

Tabla 8. Matriz de confusión obtenida para conjunto de datos de prueba.

		Estimados por el modelo	
		Negativos (N)	Positivos (P)
Valores reales	Negativos (N)	7236	689
	Positivos (P)	2795	3870

5.1.4 Proceso gaussiano

Recordando el proceso de entrenamiento, descrito en la sección 2.7.1, éste se inicializa con valores estándar de 1. Después de dicho paso, entra en acción el optimizador, cuya métrica de desempeño a evaluar es la varianza, que es individual para cada uno de los tres kernel elegidos. Los valores finales se encuentran reportados en la Tabla *9*.

Tabla 9. Valores post-entrenamiento para el proceso gaussiano multidimensional. Contiene tres kernel + ajuste por ruido.

name	class	transform	prior	trainable	shape	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus		True	()	float64	8.06048e-05
GPR.kernel.kernels[0].offset	Parameter	Softplus		True	()	float64	26.1013
GPR.kernel.kernels[1].variance	Parameter	Softplus		True	()	float64	6.43651e-05
GPR.kernel.kernels[1].offset	Parameter	Softplus		True	()	float64	8.6222
GPR.kernel.kernels[2].variance	Parameter	Softplus		True	()	float64	1.14395e-05
GPR.kernel.kernels[2].offset	Parameter	Softplus		True	()	float64	10.5227
GPR.likelihood.variance	Parameter	Softplus + Shift		True	()	float64	7.16309e-05

Posterior al primer entrenamiento sobre el conjunto de datos de entrada, se realizan otros cuatro. Cabe mencionar que, debido a que este algoritmo no es capaz de procesar grandes cantidades de datos, se realizan muestreos aleatorios simples con diferentes semillas para, al final, rectificar si la métrica de desempeño que se evalúa (varianza), cambia de manera significativa con respecto al entrenamiento inicial. Dichos resultados se reportan en la Tabla *10*.

Tabla 10. Comparación entre la varianza de los cuatro muestreos para los tres kernel + ajuste por ruido, después del entrenamiento.

semilla	Kernel 1-V	Kernel 1-O	Kernel 2-V	Kernel 2-O	Kernel 3-V	Kernel 3-O	Ruido-V
original	*8.06E-05*	*2.61E+01*	*6.44E-05*	*8.62E+00*	*1.14E-05*	*1.05E+01*	*7.16E-05*
5	8.84E-04	4.09E+01	5.65E-05	2.74E+01	6.93E-05	2.03E+01	7.31E-05
12	2.41E-04	3.49E+01	1.84E-05	1.42E+01	5.64E-05	9.79E+00	7.20E-05
26	4.43E-05	1.62E+01	1.76E-05	1.31E+01	2.13E-05	7.02E+00	7.17E-05
32	3.32E-05	7.87E+00	8.14E-06	7.04E+00	3.65E-06	4.63E+00	7.16E-05

Por último, se realizan dos pruebas, la primera es un test de normalidad Shapiro Wilk para el conjunto de 4 semillas, reportado en la Tabla 10, junto con una prueba t para muestras independientes. Las hipótesis propuestas y los resultados de ambas se reportan en la Tabla 11.

Pruebas de hipótesis:
Shapiro Wilk:
Nula. H_0 : Xi ≈ N(μ,σ^2)
Alternativa H_1 : Xi ≠ N(μ,σ^2)

t para una muestra independiente:
Nula. H_0 : μ = métrica del primer entrenamiento
Alternativa H_1 : μ ≠ métrica del primer entrenamiento

Tabla 11. Estadísticos obtenidos en el conjunto de datos para el segundo entrenamiento con 4 semillas diferentes

		Kernel 1-V	Kernel 1-O	Kernel 2-V	Kernel 2-O	Kernel 3-V	Kernel 3-O	Ruido-V
Shapiro Wilk	*Estadístico de prueba*	0.79	0.92	0.81	0.91	0.94	0.88	0.82
	p-valor	0.10	0.37	0.05	0.33	0.50	0.25	0.07
	$SW_{0.05,4}$	0.75	0.75	0.75	0.75	0.75	0.75	0.75
T de Student	*μ*	3.00E-04	24.90	2.51E-05	15.40	3.77E-05	10.40	7.21E-05
	σ	4.00E-04	15.50	2.14E-05	8.59	3.04E-05	6.91	6.91E-07
	Error estándar de la media	2.00E-04	7.75	1.07E-05	4.29	1.52E-05	3.46	3.46E-07
	μ-M	2.20E-04	-1.16	-3.92E-05	6.80	2.62E-05	-0.09	4.52E-07
	Estadístico de prueba	1.10	-0.15	-3.67	1.58	1.72	-0.03	1.31
	$t_{0.05,4}$	2.35	2.35	2.35	2.35	2.35	2.35	2.35

Dichas pruebas se realizan para confirmar que el entrenamiento sigue una tendencia normal en los datos, además de que, la prueba t nos indicará si existe alguna diferencia significativa a la hora de realizar los muestreos en los datos de entrada, y por ende, existan diferencias en los resultados finales del modelo. Dichos resultados se discutirán a profundidad en la sección 6, de análisis de resultados.

5.2 Elucidación de constantes de acoplamiento en la tamapina

Con el propósito de demostrar la eficacia del software previamente desarrollado, se utilizarán las “trazas” del espectro de RMN, de la proteína conocida comúnmente como tamapina. Cada una de estas trazas representa un aminoácido y su posición en la cadena peptídica. Dicha proteína y sus derivados, han demostrado tener aplicaciones importantes en el combate contra el cáncer, y dado que su estructura ha sido previamente reportada por otros métodos, se pueden comparar los resultados y tenerlos como punto de referencia, para saber si J Doubling ya validado, funciona de manera adecuada.

Se analizaron 5 espectros experimentales diferentes, a los que se les determinó su ancho de señal *W* y relación señal/ruido *SNR,* tal y como se puede apreciar en la Figura 29. Posteriormente, estas señales fueron procesadas por J Doubling, obteniendo la constante de acoplamiento J_{det} y el primer subarmónico *Sh*, cuyos datos se reportan en la Tabla 12.

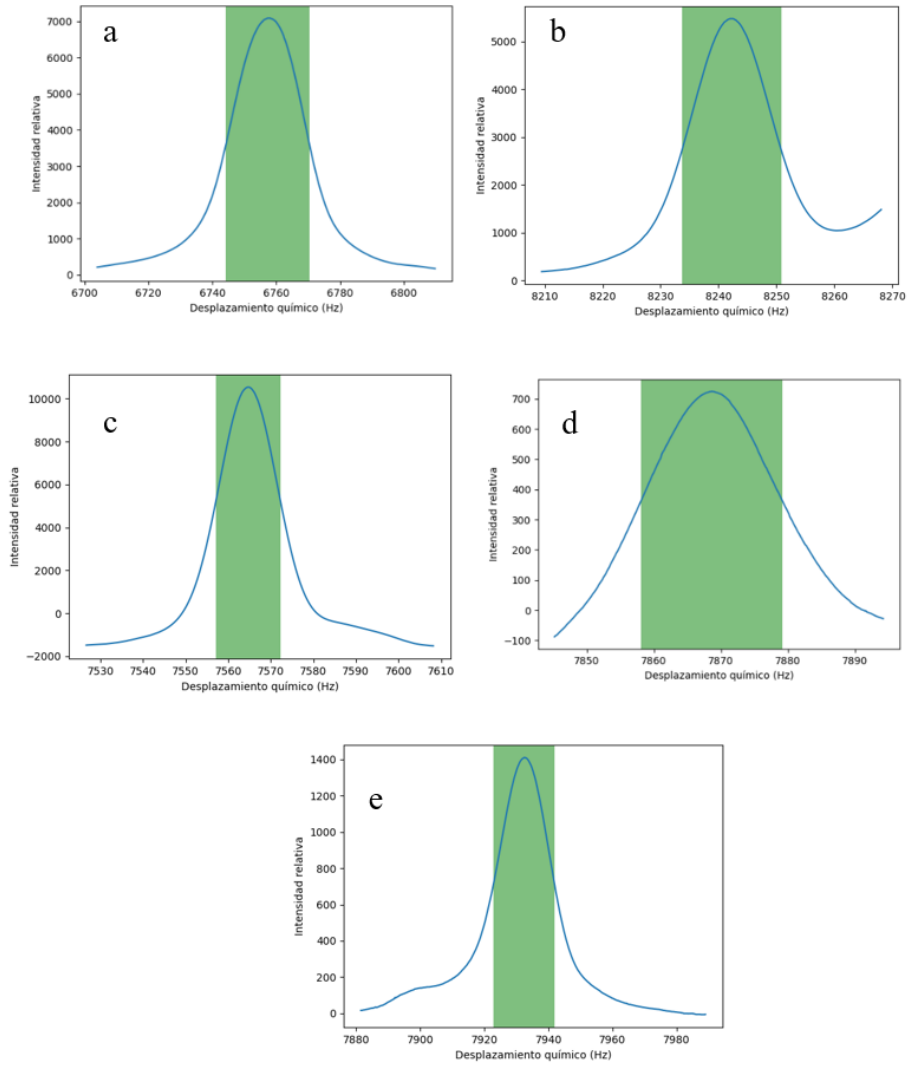


Figura 29. Trazas a analizar de la tamapina. Se tratan de espectros de RMN ^{1}H de primer orden. Orden de aparición: a=6, b=8, c=15, d=17 y e=19 respectivamente. Cada sombreado color verde representa el ancho del pico a media altura en Hz.

Estos datos de entrada fueron procesados por los tres algoritmos de validación, obteniendo el error en la determinación (Hz), si cumple o no el LOQ con respecto a los datos de referencia y, su incertidumbre asociada tipo A (Hz) con factor de cobertura del 95%, también reportados en la Tabla 12.

Tabla 12. Resultados de la determinación de las constantes de acoplamiento y sus respectivos parámetros de validación para 5 trazas de la Tamapina. Resolución digital: 0.1738 Hz

Aminoácido	W (Hz)	SNR	1Subh (Hz)	Jdet (Hz)	Error (Hz)	LOQ	A ± (Hz)	J corregida (Hz)
6	26.44	71.08	3.82	10.78	-1.32	0.00	9.60E-07	12.09
8	17.06	151.23	2.09	7.47	-0.40	0.00	5.26E-07	7.88
15	15.08	10540.00	2.26	6.78	-0.31	0.00	8.14E-01	7.09
17	21.07	68.16	3.13	8.86	1.23	0.00	2.94E-07	7.63
19	19.17	12.50	2.09	7.13	1.24	-1.00	1.50E-07	5.89

6. ANÁLISIS DE RESULTADOS

6.1 PLS o mínimos cuadrados parciales

La manera correcta para determinar el número de componentes a utilizar, se basa en la validación cruzada, en donde el conjunto de datos de trabajo, se divide en n agrupaciones aproximadamente iguales. Posteriormente, se separan los datos en un conjunto formado por n-1 agrupaciones, a las cuales se va a ajustar el modelo, mientras que las restantes servirán para comprobar la bondad del análisis. Este proceso se repetirá n veces (10 en este caso), de forma que, cada uno de los conjuntos sirva una sola vez de comprobación, reduciendo el sesgo. (Márquez Ruiz, 2017)

Un paso crucial y anterior al entrenamiento de los algoritmos principales, es el preprocesamiento y comprensión de los datos de entrada. Existen diferentes metodologías para lograr este objetivo, siendo PLS la elegida, ya que transforma la matriz de datos de entrada X, con ayuda del vector de respuestas o predicciones Y, en una matriz de componentes o variables no correlacionadas. Esto contrasta con algoritmos como PCA o análisis de componentes principales, ya que estos son obtenidos usando solo la matriz de datos de entrada X. (Vega–Vilca & Guzmán, 2011)

La figura 24 analiza la influencia del número de componentes en función de dos parámetros, MSE y el coeficiente de determinación r2, que evalúan el desempeño del ajuste, y su reducción de dimensionalidad. Este algoritmo nos dice que con 4 componentes (cruz color rojo en la Figura 24), se tiene un desempeño prácticamente perfecto con respecto a ambas métricas de desempeño.

Derivado de lo anterior, y apoyándose en los datos de la Tabla 5, se tomó la decisión de utilizar solo tres componentes, fundamentada en dos argumentos. Esto es debido en primer lugar, a que un coeficiente de determinación perfecto (r2=1.0000), lleva innegablemente al sobre ajuste en los datos, tal y como se ha comentado en la sección 2.5.5, haciendo que el algoritmo se vuelva torpe ante nuevas entradas. El segundo argumento, está basado en la contribución de la cuarta componente (señalada en rojo, parte derecha, Figura 25) y la tendencia vertical que se observa, la cual indica que no es función de la J determinada, siendo así, razón suficiente para descartarla del entrenamiento.

6.2 Red neuronal

Supervisar el entrenamiento es fundamental si lo que se desea obtener son buenas predicciones, para ello se realizan diferentes pruebas. La primera consiste en observar a tiempo real, como va disminuyendo la métrica de desempeño conforme avanzan los epoch o épocas en el adiestramiento. MAE y MSE son las más comúnmente utilizadas para redes de regresión.

Un paso fundamental, fue la implementación del entrenamiento tipo "*EarlyStopping",* ya que se encarga de decidir de manera automática, cuando dejar de entrenar el algoritmo, para así, evitar un sobreajuste en los datos, en

función de si una métrica deja de mejorar o no. Dichos comportamientos se observan en la Figura 26, en donde se ve claramente que, con un poco más de 120 vueltas, es suficiente para tener una red neuronal perfectamente funcional.

Sin embargo, esto no es un indicativo suficiente para concluir si el entrenamiento es satisfactorio o no. Para ello, se procede a recuperar las métricas del conjunto de entrenamiento y pruebas (este último, es completamente independiente del entrenamiento, y sirve para validar si las predicciones son correctas con respecto al valor real ya conocido) reportadas en la Tabla 6. La diferencia entre ambos lotes de datos y la MAE y MSE, es de -0.0126 Hz y -0.0192 Hz respectivamente. De esto se pueden concluir dos cosas:
1. La diferencia radica esencialmente en las centésimas de frecuencia (menor a la resolución utilizada en los espectros).
2. No hay cambios importantes en la métrica de acuerdo a la resolución espectral empleada.

Para la segunda afirmación, se recurrirá a observar el comportamiento real de los datos vs. las predicciones, representadas en la Figura 27. Es evidente que se ha realizado un buen ajuste hacia los datos, ya que se mantiene la tendencia lineal esperada desde un principio, sin embargo, no se recupera toda la dispersión, ya que solo realiza un ajuste sobre ellos. Es por eso que la red neuronal se complementa con un proceso gaussiano, el cuál es elegido específicamente para realizar un análisis estadístico completo, y del que se habla con más detalle en la sección 6.3.

6.3 Máquina de vector soporte

La elección del kernel es fundamental, ya que dependiendo de el, se podrá mejorar o empeorar el desempeño del algoritmo. Por esta razón, y realizando una inspección visual del comportamiento de los datos de entrada (J_{det} en función de SNR), se decidió que sería poco adecuado utilizar uno lineal, ya que se puede observar claramente que el error aumenta a valores bajos de J.

Debido a esto, y para ahorrar tiempo de ejecución, se optó por un kernel radial, cuya función se describe como:

$$K(x_1, x_2) = exp\,(-\gamma \sum_{j=1}^{p} (x_{1ij} - x_{2ij})^2)$$

γ es el parámetro que controla la varianza y suavidad del límite de decisión del modelo, en este caso 0.5, siendo el más comúnmente utilizado, se introduce de manera manual y propicia poca varianza.

Una vez optimizado el modelo, se obtiene la ecuación resultante o de decisión, dada por el kernel elegido. Se puede observar en la Figura 28, una línea color negro que divide a ambas clases. Su interpretación es crucial, ya que, para cada constante de acoplamiento determinada, se asocia un valor límite de SNR, que se interpreta como límite de cuantificación.

Esta función de decisión será de gran importancia, ya que clasificará si J doubling realiza una buena determinación de la constante de acoplamiento, en función de la cantidad de ruido de nuestra señal experimental.

Para poder evaluar el buen funcionamiento del algoritmo, existe un hiperparámetro conocido como C, el cual controla el número y severidad de los casos que estarán fuera del hiperplano de separación. Esto es importante, ya que, cuanto más tiende a infinito, se permitirán menos casos fuera del hiperplano, en cambio, entre más tienda a cero, el modelo será más laxo y se permitirán muchos errores o casos fuera del hiperplano. De acuerdo a la Tabla *7,* C es de 545559.47, siendo este valor lo suficientemente severo, evitando que se cometan demasiados errores, confirmado en la Figura 25.

Por último, la matriz de confusión reportada en la Tabla *8*, da como resultado la posibilidad de evaluar el desempeño del modelo, clasificando datos nunca vistos y comparándolos con su contraparte real (proveniente del conjunto de datos de pruebas). De dicho experimento se obtienen cuatro métricas diferentes, las cuales son:

Sensibilidad: 58.%
Especificidad: 91.3%
Precisión: 84.9%
Exactitud: 72.1%

Empezando a interpretarlos en orden de aparición, podemos decir que el modelo es poco sensible, ya que el algoritmo errará en un 41.40 % de las veces, casos positivos (o sea, se tendrá cincuenta y ocho por ciento de casos verdaderos positivos). Esto es debido claramente a la alta cantidad de ruido, tal y como se observa en la Figura 28.

Por otro lado, la especificidad es especialmente buena, y por ende, nuestro algoritmo tendrá como resultado, muy pocos casos de falsos positivos. Por último, se puede concluir que el algoritmo es más preciso que exacto. Si bien, la diferencia es menor al 13%, la precisión en modelos de clasificación, se refiere a la cantidad de casos bien clasificados con respecto al valor original de positivos, mientras que la exactitud, nos habla de la cantidad total de veces acertadas frente al valor original, de casos tanto positivos, como negativos. Dada esta premisa, se puede afirmar que en el 72.12 % de las veces, podremos estar seguros de que nuestro algoritmo se desempeña de la manera correcta.

6.4 Proceso gaussiano

En el caso de los procesos gaussianos, se suele evaluar la varianza para cada uno de los kernel establecidos. Tal y como se mencionó en redes neuronales, existen dos problemáticas derivadas de un mal entrenamiento, el sobreajuste de los datos por exceso de entrenamiento y, errores inherentes al conjunto de datos.

Por este último motivo, se decidió realizar un muestreo aleatorio simple, a partir de cuatro semillas generadoras de datos diferentes, con lo cual, se pretende detectar si la repartición/elección de los datos de entrenamiento, es igual en

todos los casos o no. Para ello, se deben realizar dos pruebas. La primera, consiste en verificar que el comportamiento de nuestras métricas de desempeño sea normal, a través de una prueba de Shapiro Wilk, la segunda es un análisis t de Student para una muestra independiente, que nos dirá si existe una diferencia significativa entre la media de las cuatro varianzas a trabajar.

Dado que son dos métricas diferentes para cada kernel más el ruido, da en total siete resultados independientes, a los cuáles se debe de verificar si son normales o no (Tabla *10*). La hipótesis nula nos dice que $H_0 : Xi \approx N(\mu,\sigma^2)$, y se rechazará solo si el estadístico de prueba es menor que el valor critico seleccionado, para un nivel de significancia dado.

En la Tabla 11, sección de Shapiro Wilk, se observa que en todos los casos, el resultado es mayor a una $SW_{0.05,4}=0.785$, además de que, el p-valor en cada uno de ellos es mayor al error que estamos dispuestos a aceptar (significancia 0.05/5%), por lo que, la probabilidad de que la hipótesis nula sea cierta es bastante alta. Por lo tanto, podemos concluir con un 95% de confianza, que nuestros datos son normales para así, proseguir con la siguiente prueba.

Luego de verificar que nuestros datos a analizar tienen un comportamiento normal, se procede a realizar un análisis t de Student para una muestra independiente, donde verificaremos que la media de la muestra (los 4 entrenamientos con diferentes semillas), es igual a la métrica de desempeño del primer entrenamiento.

Los resultados se reportan en la sección t de Student de la tabla 11, donde la hipótesis nula es $H_0 : \mu$ = métrica del primer entrenamiento. Esta se aceptará mientras el estadístico de prueba caiga dentro del intervalo $t_{0.05,4} = \pm 2.353$, la cual es válida para todos los casos, excepto la varianza del segundo kernel.

Derivado de esta prueba se pueden concluir dos cosas, la primera es que, se afirma con un 95% de confianza que el algoritmo ha entrenado correctamente y, no se presentan anomalías dependiendo de los datos que se elijan. La segunda es, el rechazo en la hipótesis nula del kernel 2-varianza, ya que uno esperaría concluir que, dado que existe un rechazo de hipótesis, no se podría afirmar que el algoritmo ha sido bien entrenado, sin embargo, esta métrica solo corresponde a una séptima parte de todas las variables que dictaminan el desempeño, por lo que, en este caso, se considerará irrelevante dado el conjunto.

6.5 Elucidación de tamapina

Tal y como se puede apreciar en la Figura 29, las señales de RMN de proteínas son sumamente anchas y complejas de analizar por métodos directos. Es por esta razón, que cinco de ellas fueron elegidas para trabajarlas en J Doubling y posteriormente, validarlas por los tres algoritmos discutidos previamente.

De acuerdo a la tabla 12, se puede observar que todas las señales tienen una W en el orden de los (15 a 25) Hz, mientras que sus constantes de acoplamiento abarcan un orden no mayor de (6 a 12) Hz aproximadamente. Es por esta razón que, una de las principales virtudes de J Doubling, es su alta capacidad para

poder analizar señales sumamente anchas, para que ahora que ha sido validado el método, se pueda dilucidar la capacidad real de éste.

Las correcciones proporcionadas por la red neuronal, serán las encargadas de determinar hasta qué punto se desempeñó adecuadamente J Doubling, y en igual proporción, devolver un resultado basado en referencias tomadas como verdaderas. Estas correcciones varían desde (0.4 a 1.3) Hz, siendo casi del 1.81% del valor total del mensurando en promedio. Se puede deber a varios factores, pero el principal está ligado a la falta de corrección en la línea base, ya que como se puede observar en la Figura 26, dichas señales no contienen una cantidad considerable de ruido.

El postulado anterior se refuerza todavía más, una vez se observan los resultados en la clasificación de la SVM, ya que como la cantidad de ruido está directamente ligada a los valores numéricos de la intensidad de la señal (y al no realizar dichas correcciones), se obtiene una clasificación no aprobatoria para el aminoácido 19, la cual coincide con el espectro donde incluso, tiene valores negativos de intensidad. Sin embargo, la determinación del LOD para cada una de las señales, fue perfectamente funcional de acuerdo a los datos que alimentaron al algoritmo.

El hecho de no haber realizado los ajustes de línea base a dichos espectros, se debe a que no fue una variable en el entrenamiento de los algoritmos, y de haberlo realizado, se estarían introduciendo valores de entrada distintos a las condiciones con los que fueron entrenados inicialmente, creando falsos positivos. Se decidió no realizar simulaciones con corrección de línea base, debido al alto coste computacional que derivaba, ya que se tendría que realizar un nuevo análisis del modelo, implicando un tiempo de ejecución mucho mayor y una gran cantidad de espacio en memoria.

De igual manera, existe un caso similar, pero ahora con la incertidumbre tipo A derivada del proceso gaussiano, ya que en la traza 15 (incertidumbre = 0.8144 Hz), se tiene un valor de SNR sumamente alto (10540.0). Esto se debe a que, la base de datos con la que fue entrenado el algoritmo, no abarcaba valores tan grandes de SNR, y por ende, su incertidumbre para ese punto será muchísimo mayor en comparación a las demás, teniendo un orden de magnitud de 10^{-7} Hz, que es donde el algoritmo si tuvo suficientes puntos para alimentarse. Es por esto, y derivado de que se demostró numéricamente que los tres algoritmos se desempeñan de manera adecuada, que se asume cada una de estas determinaciones como correctas.

7. CONCLUSIONES

Fue posible evaluar mediante datos simulados el desempeño del método modificado de duplicación de J, así como realizar futuras predicciones de los parámetros de validación LOQ, incertidumbre y cuantificación del error del modelo, cumpliéndose la hipótesis de que, un método se puede validar a través de algoritmos de aprendizaje automático.

Se puede decir que J Doubling es un método robusto, ya que los diferentes algoritmos lo confirman, a través de cada una de las métricas de desempeño evaluadas a lo largo del presente proyecto.

Por último, cabe resaltar que se cumplieron los objetivos, ya que además de generar la base de datos completa a partir de simulaciones, se generó un programa el cual está al alcance de todos los usuarios, por medio de plataformas como GitHub, haciendo de este, un programa abierto para toda la comunidad científica.

8. PERSPECTIVAS

Las limitaciones de este trabajo, radican en la falta de estudio de variables como la corrección de línea base, derivado del alto coste computacional que implicaba. Sin embargo, se pretende estudiar a fondo este parámetro, en el subsecuente trabajo de investigación que se pretende realizar.

Este trabajo pretende a largo plazo, determinar la estructura terciaria de una proteína solamente con el espectro de RMN ^{1}H de primer orden, abriendo así, las puertas de un campo que hasta hace algunos años era poco explorado para los químicos, como lo es la inteligencia artificial y el aprendizaje automático.

9. ANEXO I. PALABRAS CLAVE

- Algoritmo: Se puede definir como un conjunto ordenado de operaciones sistemáticas, en donde se tiene una entrada y una salida. Dichas operaciones permiten realizar un cálculo, para hallar la solución a un tipo de problema. Las características básicas que debe tener es que sea correcto, es decir, que produzca el resultado deseado en tiempo finito, además de eficiente. (Duch, 2007)
- Datos de entrada y salida: El funcionamiento básico de un ordenador o un algoritmo es, un sistema que tiene como entradas datos o instrucciones, para producir en función de estos, resultados o salidas. (Vallejo López, 2011)
- Datos de entrenamiento: los datos que ingresan a la IA, se separaran en dos conjuntos, entrenamiento y prueba. Los de entrenamiento serán aquellos con los que, el algoritmo reconocerá y aprenderá las tendencias entre ellos. (Torres, 2020)
- Datos de prueba: Los datos de prueba se encargan de corroborar que, las predicciones hechas por el algoritmo después del entrenamiento, se parezcan a los datos reales. (Torres, 2020)
- Datos de validación: Son un subconjunto de los datos de entrenamiento, y se utilizan cuando se está llevando a cabo el entrenamiento. Con ellos se comparará la métrica de desempeño, hasta encontrar el valor optimo de éste. (Torres, 2020)
- Entrenamiento: Se refiere al proceso donde se trabaja en ajustar los mejores pesos y sesgos a un algoritmo, para así, minimizar la función de perdida sobre el rango de predicción deseado. (Weedmark, 2021)
- Función de activación: Se encarga de, comparar el valor de entrada con el valor base de dicha función. Si el valor de entrada es mayor que el base, la neurona se activará, si no, la información no se transmitirá a la capa siguiente. (Parthiban, 2022)
- Función de perdida: Son aquellas funciones que definen como optimizar los algoritmos de aprendizaje automático. También indica que tan lejos esta la predicción, del valor correcto. (Weedmark, 2021)
- Multiplete: Es aquella señal de RMN que se divide, siguiendo un patrón especifico de intensidades. (Balci & Metin, 2005)
- Bias: Se define como la constante que se suma al producto de los pesos de la red neuronal. Sirve para compensar el sesgo del modelo, cambiando las funciones de activación hacia el lado positivo o negativo. (Vardi, 2021)
- Peso: Es aquel parámetro dentro de una red neuronal que, transforma las entradas de la información a través de las capas de procesamiento. (Lee, 2010)
- Subarmónico: En este trabajo se referirá a, los múltiplos nones de la constante de acoplamiento.

10. REFERENCIAS

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning.* Springer.

Andris, P., & Frollo, I. (2016). Interference in measured NMR images. *Measurement, 77*, 29-33.

Balci, & Metin. (2005). *Basic 1H- and 13C-NMR spectroscopy.* Amsterdam: Elsevier.

Bentz, C., Baudzuz, L., & Krummrich, P. (2014). Signal to Noise Ratio (SNR) Bentz, C., Baudzus, L., & Krummrich, P. (2014). Signal to Noise Ratio (SNR) Enhancement Comparison of Impulse-, Coding- and Novel Linear-Frequency-Chirp-Based Optical Time Domain Reflectometry (OTDR) for Passive Optical Network. *Photonics*, 33-46.

Bloch, F. (1946). Nuclear Induction. *American Physical Society, 70*.

Bonaccorso, G. (2017). *Machine Learning Algorithms.* Birmingham: Packt.

Borceguí Rubio, J., Chávez, M. I., & del Rio Portilla, F. (2001). Aplicaciones de la Medición Precisa de Constantes de Acoplamiento en Resonancia Magnética nuclear. *Revista de la Sociedad Química de México, 45*, 200-205.

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. . *5th Annual ACM Workshop on COLT* (págs. 144-152). Pittsburgh: ACM Press.

Britannica, E. V. (1991). *Encyclopedia Britannica.* Londres, Londres.

Castells, M. (1999). *La Revolución de la tecnología de la Información .* La era de la revolución: economía, sociedad y cultura .

Cun, L., Vengio, & Hinton, G. (2015). Deep Learning. *Nature*, 436-444.

Davidovits, P. (2018). *Physics in biology and medicine.* Academic Press.

Davidovits, P. (s.f.). *Physics in biology and medicine.*

De Graaf, R. A. (2019). *In vivo NMR spectroscopy: principles and techniques* (Tercera ed.). Hoboken, Nueva Jersey: Wiley & Sons.

Del Rio Portilla, F., Blechta, V., & Freeman, R. (1994). Measurement if Poorly Resolved Splittings by J Doubling in the Frequency Domain. *Journal of Magnetic Resonance, 111*, 132-135.

Duch, A. (2007). *Análisis de Algoritmos.* Barcelona: Universidad Politécnica de Barcelona.

Erb, R. J. (1993). Introduction to Backpropagation Neural Network Computation. *Pharmaceutical Research*, 165-170.

España, E., P.P, & colaboradores, M. y. (2016). *Guía Eurachem: La adecuación al uso de los métodos analíticos – Una Guía de laboratorio para la validación de métodos y temas relacionados.*

Flores Tapia, C. E. (2021). Pruebas para comprobar la normalidad de datos en procesos productivos: Anderson-Darling, Ryan-Joiner, Shapiro-Wilk y Kolmogórov-Smirnov. . *Societas*, 83-106.

García, L., & Scherer, C. (1997). *De la maquina de Vapor al Cero Absoluto (Calor y Entropía). Segunda Edición.* Ciudad de México: Fondo de Cultura Económica .

Garza García, A., Ponzanelli Velázquez, G., & del Rio Portilla, F. (2001). Deconvolution and Measurement of Spin-Spin Splittings by Modified J Doubling in the Frecuency Domain. *Journal of Magnetic Resonance*, 214-219.

Giro, L. (2020). Máquinas térmicas desde la Antigüedad al siglo XVII: análisis histórico desde la Filosofía de la Técnica. *Revista de la Sociedad Española de Historia de las Ciencias y de las Técnicas*, 29-43.
Goh, A. T. (1995). Backpropagation neural network for modeling complex systems. *Artificial Intelligence in Engineering*, 143-151.
Goodfellow, I., Vengio, Y., & Courville, A. (2016). *Deep Learning (Adaptive Computation and Machine Learning series).* MIT Press.
GPflow. (2020). *Basic (Gaussian likelihood) GP regression model*. Obtenido de https://gpflow.readthedocs.io/en/master/notebooks/basics/regression.html
Gunther, H. (2013). *NMR Spectroscopy: Basic Principles, Concepts and Applications in Chemistry .* Wiley-VCH.
H. Levvit, M. (2008). *Spin Dynamics Basics of Nuclear Magntic Resonance* (Segunda ed.). Wiley.
H.C, T. (1956). Bloch equations with diffusion terms. *Physical Reviews*, 563-565.
He, K., Zhang, X., Ren, S., & Sun., J. (2016). Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition,*, (págs. 770-778).
Hecht Nielsen, R. (1989). Theory of the Backpropagation Neural Network. *Proceedings of the International Joint Conference on Neural Networks*, (págs. 593-611). San Diego.
Hilzgrade, U., Wawer, I., & Diehl, B. (2008). *NMR Spectroscopy in Pharmaceutical Analysis.* Elsevier Science.
Jayawardana, R., & Sameera Bandaranayake, T. (2021). NALYSIS OF OPTIMIZING NEURAL NETWORKS AND ARTIFICIAL INTELLIGENT MODELS FOR GUIDANCE, CONTROL, AND NAVIGATION SYSTEMS. *International research Journal of Modernization in Engineering Technology and Science* , 743-759.
John, R. D. (1959). *Nuclear Magnetic Resonance. Applications to Organic Chemistry.* New York: McGraw-Hill.
Kristjanson Duvenaud, D. (2014). Automatic Model Construction with Gaussian Processes. *(Doctoral Thesis).* University of Cambridge, Pembroke College.
Kubat, M. (2017). *An Introduction to Machine learning.* springer International Publishing .
Lawrence, S., Giles, C. L., & Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. *Proceedings of the Fourteenth National Conference on*, (págs. 540-545). California.
Lee, B. K. (2010). Improving propensity score weighting using machine learning. *Statistics in medicine*, 337-346.
M., R., & Neal. (1998). Regression and Classification Using Gaussian Process Priors. *Bayesian Statics*, 0-0.
Maier, Rupenyan, Bobst, & Wegener. (2020). Self-optimizing grinding machines using Gaussian process modelsand constrained Bayesian optimization. *The International Journal of Advanced Manufacturing Technology*, 539-552.
Márquez Ruiz, C. (2017). Modelo de regresión PLS. *Universidad de Sevilla*.
McCarthy, J., Minsky, M. L., & Shannon, C. E. (2006). McCarthy, J., Minsky, M. L., Rochester, N., & ShannonA proposal for the dartmouth summer

research project on artificial intelligence, august 31, 1955. *AI Magazine*, 12-12.
McConnell, H. (1958). Reaction rates by nuclear magnetic resonance. *Journal of Chemical Physics, 28*, 430-431.
Noble, & S, W. (2006). What is a Support Vector Machine? *Nature Biotechnology*, 1565-1567.
Parthiban, M. (2022). *How Activation Functions Work in Deep Learning.* Guiding Tech Media.
Pinilla, J. E. (2011). Análisis de sensibilidad de la prueba de Student para una muestra. *Comunicaciones en Estadística*, 121-129.
Pisner, D. A., & Schnyer, D. (2020). *Machine Learning. Methods and Applictions to Brain Disorders.* Academic Press.
Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 761-767.
Qi, J. D., Siniscalchi, S. M., Ma, X., & Lee, C.-H. (2020). On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression. *IEEE Signal Processing Letters*, 1485-1489.
Quinonero-Candela, J., & Rasmussen, C. (2005). A Unifying View of Sparse Approximate Gaussian Process Regression . *Journal of Machine Learning Research*, 1939-1959.
Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian Processes for Machine Learning.* London: MIT Press.
Reed, R., & MarksII, R. (1999). *Neural Smithing.* MIT Press.
Rich, E. (1983). *Artificial Intelligence.* McGraw-Hill.
Rogers, M. S. (1998). *A Complete Introduction to Modern NMR Spectroscopy.* USA: Wiley & Sons .
Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization. *Psychological Review*.
Rouaud, M. (2017). *Probability, Statics and Estimation. Propagation of Uncertainties in Experimental Measurement .* Francia: Creative Commons.
Rudraksha, D. M. (2015). A nuclear magnetic resonance spectroscopic investigation of the molecular structure and agregation behavior of asphaltenes. *}tesis*.
Salgueiro, A. P., Rodríguez, J. U., Rodríguez, L. G., & Mateo, I. D. (2020). Laboratorios para la enseñanza de los algoritmos de Aprendizaje Reforzado y Aprendizaje Profundo. *Taller Internacional de Cibernetica aplicada.* La Habana.
Sametz, G. M. (2021). *nmrsim*. Obtenido de nmrsim: https://nmrsim.readthedocs.io/en/latest/jupyter/1-Introduction-to-API.html#Scenario:-User-wants-to-simulate-individual-first-order-multiplets
SciPy, C. (2021). *scipy.org*. Obtenido de https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html
Sibi, P. J., & Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. . *ournal of theoretical and applied information technology*, 1264-1268.
Taylor, M. (2014). *Introduction to JavaScript Object Notation: a to-the-point guide to JSON.* CreateSpace Independent Publishing Platform.

Torres, J. (2020). *Python deep learning (Vol. 1).* Barcelona: Macombo.
Vallejo López, F. (2011). INTRODUCCIÓN A LA INFORMÁTICA: UNIDADES DE ENTRADA Y SALIDA EN UN ORDENADOR. *REVISTA DIGITAL ENFOQUES EDUCATIVOS. No 73*, 130-144.
Vapnik, V., & Lerner, A. (1963). Pattern recognition using generalized portrait method. *Autom. Remote Control*, 774-780.
Vardi, G. Y. (2021). Learning a single neuron with bias using gradient descent. *Advances in Neural Information Processing Systems*, 28690-28700.
Vega–Vilca, J. C., & Guzmán, J. (2011). Regresion PLS y PCA Como Solución al Problema de Multicolinealidad en Regresion Multiple. *Revista de Matemática Teoría y Aplicaciones* , 9-20.
Visa, S., Ramsay, B., Ralescu, A. L., & Van Der Knaap, E. (2011). Confusion matrix-based feature selection. . *MAICS*, 120-127.
W., K. R., & K.R, W. (1989-1990). "The Fourier Transform in Chemistry". a four-part series. *Journal of Chemical Education*.
Weedmark, D. (2021). *Machine Learning Model Training: What It Is and Why It's Important.* Dominio.
White, L. (1974). *Medieval Technology and Social Change.* Londres: Oxford University Press.
Wolfgang, E. (2017). *Introduction To Artificial Intelligence.* Weingarten: Springer.
Zahangir, M. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, 292.
Zurada, J. M. (2006). Introduction to Artificial Neural Systems. *Third International Symposium on Neural Networks* (págs. 32-36). Chengdu, China: Springer Link.