UNIDAD ACADÉMICA MULTIDISCIPLINARIA REYNOSA RODHE

Tesis:

"CONSTRUCCIÓN DE UN MODELO DE CLASIFICACIÓN DE TOXICIDAD EN EL DESARROLLO DE FÁRMACOS BASADO EN DEEP LEARNING"

Presenta:

Ing. Génesis Viridiana Varela Salinas

Como requisito parcial para obtener el grado de:
"Maestra en Ciencias y Tecnologías Computacionales"

Directores:

Dr. Adolfo Josué Rodríguez Rodríguez
Dr. Carlos Armando García Pérez (Externo)

Asesores:

Dr. Hugo Eduardo Camacho Cruz
Dr. José Lázaro Martínez Rodríguez
M. en C. Alfredo Juárez Saldívar (Externo)

Cd. Reynosa, Tamaulipas. Septiembre, 2020

UNIVERSIDAD AUTÓNOMA DE TAMAULIPAS

Unidad Académica Multidisciplinaria Reynosa – Rodhe

Unidad Académica Multidisciplinaria Reynosa-RODHE

Tesis:

"CONSTRUCCIÓN DE UN MODELO DE CLASIFICACIÓN DE TOXICIDAD EN EL DESARROLLO DE FÁRMACOS BASADO EN DEEP LEARNING"

Presenta:

Ing. Génesis Viridiana Varela Salinas

Como requisito parcial para obtener el grado de:
"Maestra en Ciencias y Tecnologías Computacionales"

Directores:

Dr. Adolfo Josué Rodríguez Rodríguez
Dr. Carlos Armando García Pérez (Externo)

Asesores:

Dr. Hugo Eduardo Camacho Cruz
Dr. José Lázaro Martínez Rodríguez
M. en C. Alfredo Juárez Saldivar (Externo)

Cd. Reynosa, Tamaulipas. Septiembre, 2020

RESUMEN

En este trabajo de tesis se evaluaron diferentes arquitecturas de redes neuronales profundas, configurándolas con diferentes hiperparámetros y características como son: las funciones de activación, cantidad de capas ocultas y nodos. Lo anterior, tiene como objetivo construcción de un modelo de clasificación de toxicidad en el desarrollo de fármacos basado en Deep learning, el cual lleva por nombre Modelo_binario. Para la evaluación de este modelo, así como las distintas arquitecturas mencionadas se crearon y utilizaron 4 Conjuntos de datos. Estos conjuntos de datos están conformados con los datos del archivo SMILES del ensayo AhR del programa Tox21 y agregando los descriptores para cada molécula que fueron calculados con la herramienta Pybel, la diferencia en cada conjunto de datos reside en los descriptores seleccionados para cada uno de ellos. En los resultados, el Modelo_binario alimentado con los 16 descriptores de Pybel, demostró una precisión entre 3 % y 5 % más alta y una pérdida del 2% al 5% más baja en comparación a las demás arquitecturas probadas. Por otra parte, también se encontró que al emplear la función Adam como optimizador se evitó el problema de desvanecimiento de la gradiente que presentó el optimizador SGD en las arquitecturas probadas. En conclusión y de acuerdo con los resultados obtenidos, los datos de entrada en una red neuronal son de suma importancia, ya que la cantidad o selección inadecuada de las características afecta el rendimiento del modelo. Por lo anterior, la selección de una gran cantidad de descriptores, así como características en una red neuronal; no está ligado a una buena predicción. En cuanto a la cantidad de nodos y funciones de optimización que se utilizarán en un modelo de Deep learning, estas deben ajustarse al trabajo y tipo de modelo ya que fácilmente se puede caer en un sobreajuste o subajuste.

ABSTRACT

In this thesis work, different deep neuronal networks architectures where evaluated using different hyperparameters and features, such as activation functions, the number of hidden layers and the number nodes. The mentioned above, has as an objective the construction of a toxicity prediction model for drug development based on Deep learning, which we refer to as Modelo_binario. For this model evaluation, and the different architectures mentioned before, 4 datasets where build and used. The datasets where build with the data from the SMILES file of the Tox21 program AhR assay, and the molecular descriptors calculated with the Pybel library, also different descriptors where selected for each dataset. On the results, Modelo_binario fed with the dataset containing the 16 Pybel descriptors, showed an accuracy between 3% and 5% higher and a loss 2% to 3% lower in comparison to the other proven architectures. On the other hand, it was found that using the Adam function as an optimizer avoided the gradient descent problem shown by the SGD optimizer. In conclusion and according to the obtained results, the input data of a neuronal network play a critical role, because the quantity or inefficient feature selection affects the model performance. Considering the mentioned above, the selection of a great number of features or in this case molecular descriptors; it's not linked to a good prediction. Otherwise, the number of nodes and the optimization functions that will be used in a Deep learning model, must be selected in order to fit the job and type of model, because it may easily lead to an overfitting or a underfitting.

ÍNDICE

CAPÍTULO I INTRODUCCIÓN

1.1. Antecedentes

Hoy en día con las distintas enfermedades existentes y el surgimiento de nuevas enfermedades, el desarrollo de nuevos fármacos fiables, es decir, que los fármacos sean fabricados conforme a estándares de alta calidad se ha convertido en una necesidad esencial. Por lo cual, para lograr estos estándares de calidad, se hace uso de la regulación.

La regulación en el desarrollo de fármacos garantiza, que sólo aquellos que demuestren ser lo suficientemente seguros, efectivos y de alta calidad, puedan ser comercializados. Sin embargo, actualmente no existen leyes globales para la regulación de los nuevos medicamentos.

En base a lo anterior, para el control de la venta y desarrollo de nuevos fármacos, cada país es el responsable de crear sus organismos reguladores. Las cuatro agencias u organismos reguladores más relevantes, que aprueban el uso de fármacos para uso humano, son: la Administración de Alimentos y Medicamentos de los Estados Unidos (FDA por sus siglas en inglés), la Agencia de Medicamentos Europea (EMA por sus siglas en inglés), el Ministerio de Salud en Japón y La Comisión Federal para la Protección contra Riesgos Sanitarios (Cofepris) en México (Guerrero-Villalobos & López, 2017).

El desarrollo de fármacos es llevado a cabo mediante un arduo proceso de investigación. Este proceso, inicia con la identificación de compuestos que se unen a un blanco, o que muestran actividad biológica en un ensayo de tamizaje; posteriormente, se identifican los compuestos que tengan propiedades farmacéuticas atractivas, tales como: baja toxicidad, solubilidad acuosa adecuada para administrarse vía oral, entre otras propiedades farmacocinéticas (Saldívar-González et al., 2017).

Por otra parte, el tiempo estimado para el desarrollo de un fármaco oscila entre 10 y 15 años, y su desarrollo requiere un estimado de 800 millones de dólares. Por otra parte, la cantidad de tiempo e inversión que conlleva el desarrollo de un fármaco, es debido a la cantidad de

moléculas que fallan en una o varias etapas, generalmente las que involucran pruebas clínicas en humanos (Juaristi & Rodriguez Jorge, 2016; Saldívar-González et al., 2017).

Con el propósito de facilitar el análisis que comprende desarrollar nuevos fármacos, los investigadores han trabajado en desarrollar y aplicar diferentes técnicas computacionales. Una de las técnicas computacionales mayormente empleada, es el aprendizaje automático (Machine learning traducido al idioma inglés), en conjunto a los últimos métodos de procesamiento, con el propósito de identificar fármacos eficaces y evitar fracasos (Fernandez, 2016).

Los métodos computacionales, han brindado ayuda a los científicos en la predicción de la combinación sinérgica de fármacos, para evitar su resistencia, incrementar la eficiencia del tratamiento y la reducción de la dosis del fármaco para evitar toxicidad (Wang et al., 2018). Dentro de estos métodos computacionales se encuentran la dinámica molecular (Sánchez Montero, 2016) y el acoplamiento molecular (García-Pérez et al., 2016).

Como se mencionó anteriormente, una de las principales herramientas para el estudio de biomoléculas, es la dinámica molecular. Sus simulaciones, permiten el estudio de procesos dinámicos complejos de los sistemas biológicos, tales como: estabilidad de proteínas, cambios conformacionales, reconocimiento molecular de biomoléculas y transporte de iones. Esta herramienta computacional, es clave en el desarrollo de nuevos medicamentos y en la determinación estructural de rayos X y resonancia magnética nuclear (RMN). Por otra lado, el acoplamiento molecular forma parte de los pasos decisivos en el diseño estructurado de fármacos, debido a que ayuda en la visualización de patrones de interacciones y de energía de unión entre los compuestos (Aguirre Valderrama, 2009; Wang et al., 2018).

Durante la última década, los avances computacionales en las diferentes técnicas del desarrollo de fármacos, como el cribado virtual de alto rendimiento (Saldívar-González et al., 2017), en conjunto con el incremento de la información biológica en las bibliotecas químicas, permiten a los investigadores la construcción de conjuntos de datos de gran volumen, utilizando unidades de procesamiento grafico (GPU por sus siglas en inglés) para su procesamiento. De esta forma se ha adentrado al desarrollo moderno de fármacos en la era del *big data* (Nikhil Buduma, 2016).

El resultado del surgimiento de este desarrollo computacional, ha llevado a la implementación de nuevas técnicas para favorecer al desarrollo de fármacos en el área de la inteligencia artificial. Entre las que se encuentra, la creación de modelos de redes neuronales con una gran cantidad de capas ocultas, este proceso es mejor conocido como *Aprendizaje Profundo* (Deep learning, por su traducción en inglés) (Zhang et al., 2017).

Una de las áreas de mayor interés dentro del diseño de fármacos, en la cual se emplean las técnicas de Deep learning, es la predicción de la toxicidad de las moléculas. Se entiende por toxicidad o acción tóxica, cuando una sustancia que entra en contacto con el organismo, produce efectos nocivos adversos en el organismo, tales como daños serios de las funciones del organismo a nivel celular, bioquímico o molecular, e incluso la muerte una vez que ha alcanzado un punto del organismo suceptible a su acción (Muster et al., 2008). Sin embargo, algunos compuestos químicos pueden presentar toxicidad en dosis altas, pero ser inofensivos e incluso beneficioso en dosis pequeñas. Por tal motivo, es que fallan en las últimas pruebas del desarrollo de fármacos, incluso aunque hayan obtenido resultados satisfactorios en las pruebas *in vivo* realizadas en animales (Saldívar-González et al., 2017; Verbist et al., 2015).

1.2. Planteamiento del problema

Dentro del diseño de fármacos, la evaluación de toxicidad una vez seleccionado el blanco farmacológico, es una etapa de suma importancia, debido a que es fundamental para que un fármaco sea aprobado para su consumo. En otras palabras, el desarrollo de un fármaco puede entenderse como la analogía de una llave que cierra una cerradura, dónde la llave es el fármaco y la cerradura el blanco farmacológico, por lo tanto, no basta con que el fármaco se acople perfectamente al blanco, sino además se debe evitar que el fármaco sea tóxico para el usuario final. Una de las posibilidades para desarrollar un fármaco, es tener las llaves y buscar qué cerraduras puede abrir o tener la cerradura y buscar la llave. En cualquier de los casos, es indispensable evitar aquellos fármacos que presenten toxicidad al consumidor. Es necesario remarcar que, aun cuando presenten toxicidad, pero tengan una fuerte unión al blanco, se podría modificar el fármaco y diseñar un análogo que suprima la toxicidad.

No obstante, los métodos que se realizan para evaluar la toxicidad son lentos, tediosos y costosos. Sin antes mencionar, que algunos plantean preocupaciones éticas, dado que involucran pruebas en animales o lo que se conoce como pruebas *in vivo* (Atkinson & Markey, 2007; Dearden C.J., 2003).

Sin embargo, a pesar de la inversión, tiempo y esfuerzo aplicado en las diferentes pruebas para el desarrollo de fármacos, se reporta que aproximadamente el mayor porcentaje de falla ocurre en la fase II del desarrollo de fármacos. La fase II, es la primera parte en que se realizan las pruebas de dosificación en humanos y estas dosificaciones son escaladas para alcanzar los niveles esperados para ser clínicamente activo, y solo el 11% de los fármacos acreditan todas las fases de la aplicación para aprobación de la FDA (Van Norman, 2016).

Debido a lo anterior, recurrir a la predicción de la toxicidad mediante técnicas computacionales, es conveniente para acelerar el proceso del desarrollo de fármacos, reduciendo las pruebas de laboratorio que puedan causar daños para la salud, asimismo el tiempo y costo que conllevan.

1.3. Justificación

Hoy en día, gracias al desarrollo y surgimiento de nuevas herramientas computacionales, es posible emplearlas para optimizar y disminuir los riesgos que conllevan las diferentes pruebas y métodos para el desarrollo de fármacos. Por lo tanto, con el fin de evaluar la toxicidad de los fármacos y reducir las pruebas físicas, se consideró el desarrollo de un modelo de clasificación binaria utilizando las herramientas *Tensorflow* y *Keras*, que apoyándose en algoritmos de Deep learning le sea posible detectar e identificar de manera temprana mediante un patrón si un fármaco cuenta con un nivel de toxicidad alto que pueda causar daños con el objetivo de minimizar el tiempo y riesgos, asimismo que permita aplicar el modelo a distintos fármacos.

1.4. Hipótesis

La evaluación de diferentes arquitecturas e hiperparámetros, implementando las técnicas de Deep learning y la creación de un conjunto de datos creado con los descriptores de la herramienta *Pybel*, permitirán el desarrollo de un modelo de clasificación binaria de Deep learning cuyo entrenamiento lo lleve a una precisión que le permita predecir adecuadamente si un fármaco es tóxico o no.

Tipo de hipótesis: Asociativa

Variables: 1. Componentes de la arquitectura, 2. Hiperparámetros.

Variable dependiente: Precisión en la predicción de toxicidad.

1.5. Objetivos

1.5.1. Objetivo general

Desarrollar un modelo de clasificación binaria para la predicción de toxicidad implementando las técnicas de Deep learning mediante el uso de las bibliotecas de *Tensorflow* y *Keras*.

1.5.2. Objetivos específicos

- Elaborar un conjunto de datos a partir del ensayo receptor de hidrocarburos de arilo (*AhR* por sus siglas en inglés) que se encuentra en formato *smiles* del programa *Tox21*, extrayendo el nombre de la molécula, cadena smiles y si es tóxico o no, además complementar el conjunto de datos con el cálculo de los descriptores que permite la herramienta *Pybel* para cada una de las 1870 moléculas que se encuentran en él.
- Configurar diferentes arquitecturas e hiperparámetros para la red neuronal del modelo de Deep learning con el objetivo de determinar cuáles brindan un mejor desempeño para la predicción de toxicidad.

- Evaluar los diferentes modelos resultantes de la configuración de distintos hiperparámetros y arquitecturas con diferentes conjuntos de datos, cuya diferencia entre los conjuntos será la cantidad y los descriptores seleccionados para cada uno de ellos.

1.6. Delimitación de la investigación

Se creará un modelo de Deep learning de clasificación binaria con redes neuronales profundas o también conocidas como “completamente conectadas” para su entrenamiento. Se creará un conjunto de datos utilizando los elementos del ensayo que identifica las moléculas que activan el *AhR* del proyecto *TOX 21* (Thomas et al., 2018), e incluyendo como características los descriptores de *Pybel*.

CAPÍTULO II MARCO TEÓRICO

2.1. Diseño de fármacos

Anteriormente, el surgimiento de nuevos fármacos era accidental. Debido a que accidentalmente se descubría, que una molécula provocaba una mejora en el tratamiento de ciertas enfermedades, o que una ya existente tenía otro efecto secundario no buscado. Sin embargo, actualmente el proceso del desarrollo de fármacos ha cambiado drásticamente (Marovac, 2001).

El desarrollo de fármacos, es el resultado de un proceso de investigación muy complejo, que debe ser realizado mediante una serie de etapas en específico (Marovac, 2001). En la Figura 1 se describen las etapas que se llevan a cabo durante el proceso.

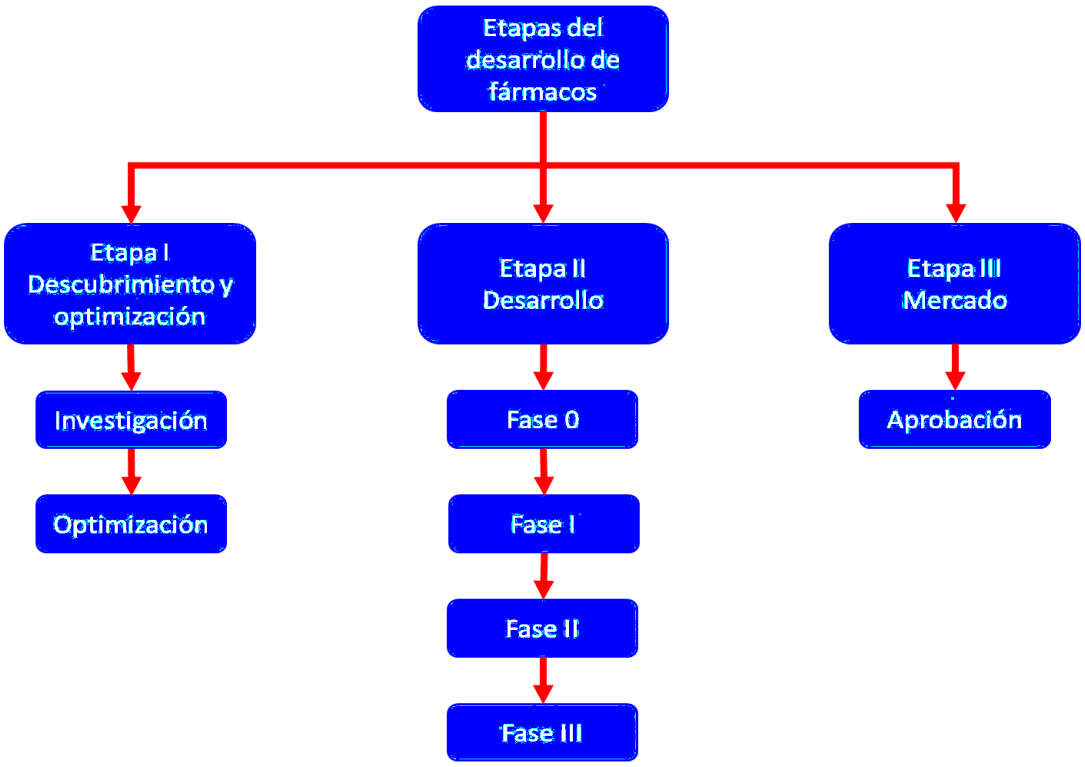

Figura 1 Etapas principales a seguir en el desarrollo de fármacos

A continuación, se detallarán los procedimientos realizados en las etapas del desarrollo de fármacos descritos en (Aguirre Valderrama, 2009; Saldívar-González et al., 2017; Van Norman, 2016):

Etapa 1 – Descubrimiento y optimización: En esta etapa, inicia la investigación acerca del padecimiento a tratar, sus causas, la identificación de los posibles blancos farmacológicos asociados a esta enfermedad, y la identificación de compuestos activos contra estos blancos. Posteriormente, su prioridad reside en la evaluación y optimización de la actividad biológica, la cual es evaluada en la siguiente etapa de desarrollo.

Etapa 2 – Desarrollo: En esta etapa se realizan las pruebas clínicas, es decir la aplicación en humanos. Esta etapa inicia con la prueba en animales o fase preclínica, las pruebas son realizadas mínimo con dos especies diferentes. Sin embargo, la fase 0 comprende el inicio de las pruebas en humanos. En estas pruebas, la cantidad de individuos con los que realizan las pruebas oscila entre los 10 y 15. En la siguiente fase, conocida como fase I, se aumenta el rango de los individuos entre 20 y 80. En esta fase los individuos son saludables, ya que solamente se desean identificar los posibles efectos secundarios e identificar la dosis correcta. Asimismo, se busca identificar el posible perfil tóxico que pueda presentar. Esto es realizado mediante la evaluación de los individuos con pequeñas dosis, que son derivadas de la cantidad sin efecto tóxico detectado de las pruebas en animales. A continuación, en la fase II, se comienza a evaluar la eficacia del fármaco, por lo que todos los individuos deben padecer la enfermedad. En esta fase, el rango de individuos a evaluar es de 100 a 300, con el fin de detectar posibles efectos no comunes. La fase III es la última de esta etapa, por lo que antes de continuar con esta fase, es necesario reportar los resultados obtenidos en las fases anteriores a la autoridad correspondiente, en especial aquellos sobre seguridad y toxicidad. Por otra parte, el propósito de esta fase es evaluar la efectividad del fármaco, es decir, confirmar que es seguro y eficaz, por lo cual las pruebas deben ser realizadas en un rango de 1000 a 3000 individuos. Los efectos observados son comparados contra otros tratamientos comúnmente utilizados para tratar la enfermedad. Los compuestos que culminan y aprueban todas las fases de esta etapa, son aprobados para la siguiente etapa por un agente regulatorio.

Etapa 3 – Mercado: En esta etapa, se debe realizar un documento, siguiendo los requerimientos de la entidad regulatoria, para que el fármaco sea aprobado y pueda ingresar al mercado. Este

documento, debe incluir toda la información referente al desarrollo del fármaco, como el proceso de manufactura y una descripción detallada del producto (formula, especificaciones, farmacodinámica aplicada, farmacocinética, indicadores, etiquetas y evaluación de riesgos propuesta). Un documento típico, contiene alrededor de 100,000 páginas.

2.1.1. Importancia de la toxicidad en el diseño de fármacos

El proceso del desarrollo de un fármaco tarda aproximadamente entre 10 y 15 años, y su costo ronda entre los 800 millones de dólares (Juaristi & Rodriguez Jorge, 2016). Sin embargo, a pesar de todas estas pruebas y el tiempo e inversión monetaria, solamente el 18% llega a la fase III, y el 11% de los fármacos culminan todas las etapas del proceso, y son aprobados. Esto se debe a que presentan efectos tóxicos, o que resultan ineficientes en el tratamiento de la enfermedad (Saldívar-González et al., 2017; Van Norman, 2016).

La toxicidad, se refiere al nivel de daño que un compuesto puede causar al organismo. Los efectos tóxicos de un fármaco dependen de la dosis y pueden afectar un órgano en específico, por ejemplo, el hígado o hasta un sistema completo como el sistema nervioso central. La toxicidad también está relacionada con daños psicológicos, como muerte neuronal o causar ciclos hormonales anormales. Además, la toxicidad generalmente ocurre en dosis que exceden la cantidad necesaria para la eficacia terapéutica del fármaco. No obstante, en algunas ocasiones los efectos tóxicos y terapéuticos pueden ocurrir simultáneamente (Riley & Kohut, 2010; Wang et al., 2018).

2.1.2. Diseño de fármacos asistido por computadora (DIFAC)

El desarrollo de fármacos involucra varias etapas que abarcan desde la identificación de blancos farmacológicos hasta las fases clínicas, la mayoría de las técnicas clásicas del DIFAC se centran en las primeras etapas. Uno de los métodos que se emplea con frecuencia para optimizar la detección de compuestos líder, es el acoplamiento molecular más conocido como “docking”. (Saldívar-González et al., 2017).

Este método conocido como *docking molecular*, consiste en optimizar la predicción de la orientación de dos moléculas: una pequeña (el ligando) y una macromolécula (receptor), de manera que formen un compuesto molecular energéticamente estable. Asimismo, encontrar la conformación óptima entre el receptor y el ligando, es decir, la que resulte con una mínima energía de acoplamiento (López-Camacho et al., 2015).

Otra técnica computacional empleada para el desarrollo de fármacos es el cribado virtual. Esta estrategia, es un filtrado computacional (*in silico*) de moléculas para seleccionar candidatos o hits computacionales, para evaluarlos mediante ensayos experimentales. Por lo cual, se reduce el número de compuestos a evaluar biológicamente. El cribado virtual, normalmente se realiza en forma de dos o más ciclos de refinamiento de los resultados. Además, utiliza diversos filtros los cuales pueden variar, dependiendo que tan robusta es la base de datos y con qué información experimental se trabajara (Medina-Franco et al., 2015; Saldívar-González et al., 2017).

2.1.3. Tox21

Con el fin de optimizar, y brindar apoyo al desarrollo de fármacos empleando herramientas computacionales que permitan el desarrollo de pruebas *in silico*, como el *cribado virtual de alto rendimiento o* el *docking molecular*, *surgió* Toxicología en el siglo 21 (*Tox21* por sus siglas en inglés). Tox21, es un programa de colaboración entre distintas agencias federales estadounidenses, entre las que se encuentran: Los Institutos Nacionales de la Salud (NIH), la Agencia de Protección Ambiental (EPA) y la Administración de Alimentos y Fármacos (FDA). Este programa de colaboración surgió en el año 2008, con el fin de brindar información de la reacción toxicológica de distintos compuestos moleculares, es decir, que si al interactuar o enlazarse el compuesto molecular con un receptor molecular (proteína) del cuerpo humano, dicha proteína se activará y generará una reacción tóxica en el cuerpo humano.

El programa *Tox21*, actualmente cuenta con un banco de datos con más de 8500 compuestos moleculares de libre acceso, cuya actividad biológica se obtuvo mediante cribado virtual de alto rendimiento. Esta información, está dividida en 12 ensayos divididos en dos categorías:

1. Panel de receptor nuclear: Esta categoría cuenta con 7 ensayos, receptor androgénico utilizando la línea celular MDA (AR), receptor de aril hidrocarburos (AhR), receptor androgénico (AR-LBD), receptor de estrógeno utilizando la línea celular BG1 (ER), receptor de estrógeno (ER-LBD), inhibidores de la aromatasa (aromatase) y el receptor de peroxisoma proliferador-activado gamma (PPAR-gamma). Las 7 proteínas con las que se probaron las moléculas son receptores de tipo nuclear, este tipo de receptores (NR's por sus siglas en inglés), constituyen una familia de factores que regulan la expresión de un gran número de genes de tipo celular.
2. Panel de Respuesta de estrés: Esta categoría cuenta con 5 ensayos, elemento de respuesta antioxidante (ARE), ATAD5, respuesta de choque térmico (HSE), potencial de membrana mitocondrial (MMP) y P53. Estas 5 proteínas son producidas debido a la respuesta de estrés de las células, y al reaccionar con algún compuesto molecular, pueden producir una toxicidad a nivel fisiológico (Riley & Kohut, 2010; Thomas et al., 2018).

2.2. Redes Neuronales

Antes de abordar el tema de las redes neuronales artificiales, es necesario conocer el funcionamiento de una red neuronal biológica. Una red neuronal biológica inicia con las células nerviosas biológicas, o mejor conocidas como neuronas, que, al recibir la señal de neuronas adyacentes mediante sus dendritas, procesan los pulsos eléctricos recibidos provenientes desde las células del cuerpo, y posteriormente, transmiten las señales de salida hacia los axones. Los axones, son una fibra nerviosa, larga y delgada que inicia en el cuerpo de la neurona, y termina en una ramificación que entra en contacto con las células del cuerpo, como las musculares, glandulares e incluso otras células nerviosas (Konar, 1999).

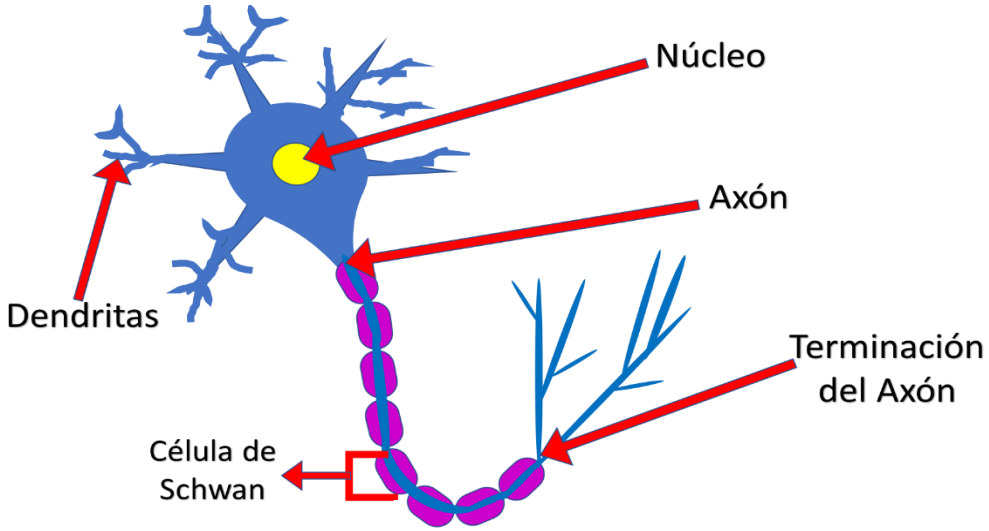

Figura 2 Estructura de una neurona

El surgimiento de las redes neuronales artificiales fue derivado de los estudios de la estructura del sistema nervioso central de los mamíferos, y gracias a ello surgió la estructura de la red neuronal artificial. Esta consiste, en que cada red está conformada por diferentes neuronas interconectadas entre sí, organizadas en capas, lo cual permite el intercambio de mensajes, y al igual que las redes neuronales biológicas, éstas se activan cuando surgen ciertas condiciones (Gulli & Pal, 2017).

Los estudios de las redes neuronales artificiales comenzaron a finales de los años 50, con la introducción del perceptrón. Más específicamente en 1957, cuando Frank Rosenblatt comenzó el desarrollo de una de las redes neuronales artificiales más antiguas: el perceptrón. Este modelo, es utilizado incluso hoy en día. Debido a que aun después de haber sido entrenado y aprender una serie de patrones, es capaz de reconocer otros similares, incluso aunque no se le hayan presentado antes. Debido a que estos procesos y modelos de las redes neuronales artificiales, han evolucionado hasta llegar a lo que conocemos hoy en día, han surgido diferentes aplicaciones de esta área, como la detección de imágenes, procesamiento de voz, video y un gran número de datos (Hilera & Martínez, 1995).

Actualmente, las redes neuronales realizan procesos y operaciones cada vez más complejos, esto ha sido posible gracias a las herramientas computacionales con las que se cuentan actualmente,

en conjunto a su programación para mejorar sus funciones y habilidades. Por ejemplo, la capacidad de aprendizaje adaptativo, esto se refiere a su capacidad de autoajustar sus parámetros constantemente, hasta alcanzar un estado de equilibrio y estabilización de los parámetros, optimizando los valores de pérdida y predicción (Arnold et al., 2011; Hilera & Martínez, 1995; Konar, 1999).

2.2.1. Tipos de redes neuronales

Entre los tipos de redes neuronales, se encuentra la red neuronal densa o completamente conectada. Esta es una de las más utilizadas, gracias a que permite una conexión completa, entre las neuronas, las capas de entrada, y las de salida. En este proceso, cada neurona recibe la información de cada una de las neuronas de la capa anterior, permitiendo que toda la información fluya, y que cada capa reciba el aprendizaje de la combinación de características de la anterior, y no se pierda información (Chen et al., 2018).

Existen diferentes tipos de redes neuronales que pueden ser empleadas en Deep learning. Dentro las cuales, se encuentran: las *Redes Neuronales Convolucionales* (CNN por sus siglas en inglés), tienen la capacidad de aprender características automáticamente a partir de los datos de entrada, este tipo de red neuronal es mayormente utilizada para la clasificación de imágenes, y las *Redes Neuronales Recurrentes* (RNN por sus siglas en inglés).

2.2.2. Funciones de pérdida y activación

Al igual que en las redes neuronales biológicas, las redes neuronales artificiales necesitan de una activación para transmitir la información de una neurona a otra. En este caso, las redes neuronales artificiales logran realizar este paso gracias a las funciones de activación.

Las funciones de activación son utilizadas para controlar las salidas de las redes neuronales, estas se dividen en dos tipos: lineares y no lineares, dependiendo la función que representen. La función debe ser seleccionada en base al problema que se desea resolver, por ejemplo, de

reconocimiento de objetos, regresión lineal o clasificación. Las funciones que se utilizan mayormente, son las no lineales, ya que permiten al modelo generalizar o adaptarse a una variedad de datos (Nwankpa et al., 2018).

Dentro de las funciones no lineales las más utilizadas son:

- Unidad lineal rectificada (ReLU por sus siglas en inglés): Hoy en día es la función más utilizada a nivel mundial, principalmente en las capas ocultas de la mayoría de las redes neuronales convolucionales. La función ReLU, realiza una operación para delimitar un umbral a cada valor de entrada, y rectificar cada valor menor a cero para igualarlo a cero, lo que permite eliminar el desvanecimiento de la gradiente. En consecuencia, llega alrededor de 6 veces más rápido al punto de la convergencia, en comparación con las funciones anteriormente mencionadas (Nwankpa et al., 2018; Pawan, 2019; Sharma, 2017).
- Sigmoidal: La función sigmoidal, toma los datos que pueden estar en el rango [-Inf; +Inf] y expresarlos a el rango (0; 1). Es empleada en las capas de salida, y muestra un buen resultado en problemas de clasificación binaria y predicción. Sin embargo, esta función debe ser evitada para inicializar una red neuronal con pequeños pesos.
- Tangente Hiperbólica: Esta función es conocida como tanh, a diferencia de la sigmoidal sus rangos van de {-1,1}, por lo que es utilizada en problemas de clasificación multiclase y en redes neuronales recurrentes. Una de sus ventajas, es que los valores negativos los mostrará en la gráfica con su valor negativo exacto, y los valores cero los graficará cerca del cero. Al igual que la función sigmoidal, tiene el problema del desvanecimiento de la gradiente, esto significa que la gradiente tiende a ir disminuyendo. Por lo que las neuronas que se encuentran en las primeras capas están aprendiendo de manera tardía con respecto de las últimas capas, lo que resulta en un descenso en el porcentaje de predicción, y entrenar el modelo toma mucho tiempo.

Para la creación de los modelos de redes neuronales, existen diferentes tipos de funciones, por ejemplo, las funciones de activación que se mencionaron anteriormente y las funciones de pérdida. Una función de pérdida es empleada para estimar la pérdida que tuvo el modelo, y le

sea posible ajustar los pesos en la red neuronal, para así reducir la pérdida en la siguiente evaluación. Por otra parte, es importante mencionar, que la elección de la función de pérdida debe ser en base al tipo de problema a resolver, ya sea de clasificación o de regresión (Arnold et al., 2011).

Para los modelos de clasificación binaria, se consideran apropiadas para modelar la predicción de los problemas de clasificación binaria tres funciones de pérdida:

- Binary Cross-Entropy

  Esta función utilizada habitualmente en problemas de clasificación binaria, donde los valores objetivo están establecidos como {0,1}. Asimismo, es también la más adecuada matemáticamente.

  El procedimiento que realiza es calcular un resultado resumiendo el promedio de la diferencia entre la probabilidad actual y la predicha. El valor perfecto y esperado de esta función es 0.
- Hinge Loss

  Una de las principales diferencias entre la función binary crossentropy y la hinge loss, está en que sus valores pueden encontrarse entre {-1,1}. Esta función, fue principalmente desarrollada para su implementación en los modelos de *Máquinas de Vectores de Soporte* (SVM por sus siglas en inglés).
- Squared Hinge Loss

  La función *hinge los*s, cuenta con diferentes adaptaciones, una de ellas es la *Squared hinged los*. Esta función calcula el cuadrado del resultado de la función *hinge loss*, como resultado, tiene el efecto de suavizar la superficie de la función de error, y hacerla numéricamente simple para trabajar (Brownlee, 2019; Gulli & Pal, 2017).

Durante el entrenamiento y proceso de aprendizaje, la red neuronal debe ir ajustando los pesos poco a poco, con la finalidad de reducir el error a la hora de la predicción. Por tal motivo, las funciones de activación, son empleadas para realizar progresivamente estos cambios a los pesos de entrada en las capas, y así reconocer las entradas que tengan más importancia para una mejor predicción (Sharma, 2017).

2.3. Fundamentos del Deep learning

2.3.1. Inteligencia Artificial

El campo de la inteligencia artificial cuenta con diferentes temas de investigación y aplicaciones prácticas. Lo que se busca lograr con la inteligencia artificial, no es el simple hecho de automatizar diferentes rutinas de trabajo, sino de que la máquina o software sea capaz de superar o imitar la capacidad de la mente humana. De igual manera, que desarrolle habilidades de comprensión de diálogos e imágenes, o de realizar diagnósticos en la medicina, y con esto dar sostén a la investigación científica. Uno de los grandes retos de la *Inteligencia Artificial*, es probar que puede realizar tareas que son fáciles de realizar para el humano, pero difícil de explicar. Por ejemplo, que resuelva intuitivamente problemas, como seguir indicaciones al escuchar unas palabras, reconocer a una persona entre muchas imágenes, o algo más complejo, como descifrar una jugada de ajedrez (Goodfellow et al., 2016; Konar, 1999).

2.3.2. Machine learning

El Aprendizaje automático (Machine learning traducido al idioma inglés), se refiere a la habilidad de las máquinas mediante su programación y hardware, de detectar patrones y aprender de ellos para realizar predicciones. El Machine learning, surge de la inteligencia artificial y de las bases y técnicas de la minería de datos. La minería de datos es un conjunto de diferentes áreas, como la base de datos, la estadística, la visualización de datos, la búsqueda y recuperación de la información, y de la computación de alta ejecución. En Machine learning, estos procesos son empleados para poder extraer y descubrir patrones en los datos, mediante el uso de algoritmos como: regresión lineal, regresión logística, de asociación, lógica difusa, árboles de decisión y redes neuronales entre otros (Goodfellow et al., 2016; Ian H. & Frank, 2005).

2.3.3. Deep learning

El *aprendizaje profundo* o mayormente conocido como Deep Learning (DL), es proveniente de la inteligencia artificial y es una subárea de las metodologías del Machine Learning (ML). El Deep learning, se define como un algoritmo automático estructurado o jerárquico que simula el aprendizaje humano, con la finalidad de adquirir ciertos conocimientos, mediante redes neuronales artificiales inspiradas en la estructura neuronal localizada en el cerebro humano. En Deep learning, la red neuronal configura los parámetros acerca de los datos de entrada, y entrena a la red neuronal para que aprenda por cuenta propia reconociendo patrones, es decir, aprende de forma automática alguna especie de patrón o conocimiento que no estaba en su programación, con el propósito de afrontar nuevas situaciones que pudiesen presentarse.

Un modelo de Deep learning, se divide principalmente en 3 capas:

Capa de entrada: La conforman las neuronas encargadas de procesar los datos de entrada, por ejemplo, una imagen o un conjunto de datos.

Capa oculta: Es la red que realiza el procesamiento de información y hace los cálculos intermedios.

Cada de Salida: Es la capa que toma la decisión o realiza alguna conclusión aportando datos de salida.

Dentro de las diferentes aplicaciones en que es utilizado el Deep learning, se encuentran: la detección de objetos, reconocimiento de voz, procesamiento de lenguaje, y otras diversas aplicaciones que las empresas están investigando e implementando hoy en día (Gulli & Pal, 2017; Najafabadi et al., 2015).

2.3.4. Bibliotecas para el desarrollo y evaluación de modelos de Deep learning

Python, al ser un lenguaje de programación con facilidad de aprender y con múltiples herramientas, se ha convertido en un lenguaje muy utilizado en la creación de modelos de Deep Learning. Debido a que permite la posibilidad de una programación orientada a objetos, cuenta con bibliotecas para la extracción y procesamiento, de datos y estructuras de datos. De igual

forma, cuenta con bibliotecas para la estructuración y análisis de estructuras moleculares como *Pybel* (O'Boyle et al., 2008), y bibliotecas que simplifican la creación de modelos de Deep learning, las cuales serán mostradas a continuación (Bogdanchikov et al., 2013).

- *TensorFlow*, es un sistema desarrollado por Google para la implementación de Machine learning y Deep Learning, opera a gran escala en diferentes ambientes. *TensorFlow* utiliza grafos para la representación computacional de las operaciones matemáticas. Una de sus grandes ventajas, es que puede ser empleado en diferentes dispositivos, como la unidad central de procesamiento (CPU, por sus siglas en inglés), la unidad de procesamiento gráfico (GPU por sus siglas en inglés), la unidad de procesamiento tensorial (TPU por sus siglas en inglés), e incluso en dispositivos móviles, como tabletas y teléfonos. Actualmente, con la versión *2.0*, *Tensorflow* se ha convertido en una de las bibliotecas más extensa, flexible y utilizada (Abadi et al., 2016).
- *Keras*, es una biblioteca de alto nivel para la creación de redes neuronales, soportando las redes neuronales convolucionales y recurrentes. Está escrito en *Python*, lo que le permite ser capaz de utilizarse en sistemas como *TensorFlow*, *CNTK*, o *Theano*. Fue desarrollado enfocándose en permitir una rápida experimentación con redes neuronales, y que sea posible utilizarlo en CPU's y GPU's (Chollet, 2015).
- *Scikit-learn* es una biblioteca para *Python*, integra algoritmos de Machine learning para la solución de problemas supervisados (a media escala) y no supervisados, su desarrollo se enfocó en ayudar a usuarios no especializados en Machine learning, para que sea sencillo, consistente y de uso libre (Varoquaux et al., 2015).
- *Pytorch*, es una biblioteca para Python que optimiza la creación de tensores en GPU's y CPU's, esto debido a que permite utilizer CUDA (Paszke et al., 2017).

2.4. Aprendizaje automático en el diseño de fármacos

En la actual era del *big data*, la cantidad de información biológica ha ido en aumento rápidamente. Esto permitiendo crear conjuntos de datos de gran tamaño, y provocando el desarrollo de innovaciones en el modelado y diseño de fármacos. No obstante, el procesamiento de una gran cantidad de información limita las técnicas computacionales convencionales. Sin

embargo, nuevas metodologías como el Machine o Deep learning, además del incremento en el número de GPU's, han ayudado a solventar estas dificultades.

En el diseño de fármacos, el Deep learning es capaz de aprender a reconocer, los descriptores moleculares que confieren la actividad o toxicidad a un tipo determinado de estructuras químicas. Hoy en día, la creación de diferentes modelos e innovaciones para la predicción de toxicidad, mediante la aplicación del aprendizaje automatizado en el área del desarrollo de fármacos, van en aumento día con día (Hong & Science, 2019; Muster et al., 2008; Riley & Kohut, 2010).

CAPÍTULO III METODOLOGÍA

3.1. Procedimiento para comprobar la hipótesis

En esta sección, se describirán cada uno de los pasos para desarrollar este trabajo. Los cuales permitieron cumplir con cada uno de los objetivos y lograr una comprobación de la hipótesis.

- Preparación del ambiente de pruebas: Uno de los aspectos importantes es la preparación de la máquina en la cual se realizará el entrenamiento del modelo de Deep learning. Este procedimiento se realizó de la siguiente manera:
    - Instalación del S.O. Linux Mint19.7 tessa. Esta versión fue seleccionada debido a su manejable interfaz, además de que facilita la portabilidad a python3.
    - Instalación de la tarjeta gráfica NVIDIA. El modelo de la tarjeta gráfica debe ser compatible con Tensorflow para GPU y habilitada para CUDA, los modelos compatibles pueden revisarse en la página oficial de Tensorflow.
    - Instalación del software y los controladores de la tarjeta gráfica. Estos fueron seleccionados y descargados desde la página oficial de NVIDIA, ingresando los datos del modelo de la tarjeta gráfica.
    - Para el funcionamiento de Tensorflow 2.0 para GPU, se debe realizar la instalación de CUDA 10.1.
    - Creación de un ambiente de Anaconda, debido a que facilita la instalación de los diferentes lenguajes y bibliotecas.
    - Instalación del lenguaje de programación Python. Para creación de los modelos de Deep learning empleando Tensorflow y Keras.
    - Instalación de Tensorflow (Abadi et al., 2016) y Keras (Chollet, 2015). Bibliotecas de uso libre para la creación de modelos de Deep learning.
    - Instalación de las bibliotecas: Pybel (O'Boyle et al., 2008) para el cálculo de los descriptores moleculares, Sklearn (Pedregosa et al., 2011) y Pandas (McKinney, 2010) fueron empleadas para el preprocesamiento de los datos y creación del conjunto de datos y Matplotlib (Hunter, 2007) para la graficación de los resultados.

- Creación del conjunto de datos: Para la creación del conjunto de datos, se extrajo de la base de datos del programa Tox21 el archivo SMILES llamado Receptor de Hidrocarburos de Arilo (AhR). El archivo SMILES, es un tipo de archivo utilizado en química, que contiene cadenas smiles que representan la estructura química de una molécula usando cadenas de caracteres alfanuméricos de tipo ASCII. En nuestro caso particular, el archivo contiene las estructuras químicas de las 8170 moléculas que se probaron para verificar su reacción con el AhR. El archivo final, contiene el nombre de la molécula, el estado de la actividad en formato binario, y la cadena de representación smiles. Posteriormente, se creó un archivo CSV utilizando Python y la biblioteca Pandas, además, se agregaron las columnas con los descriptores para cada molécula que fueron calculados con la herramienta Pybel, que permite calcular los siguientes 16 descriptores numéricos: Número de átomos (atoms), Número de enlaces (bonds), Número de donantes de enlace de hidrógeno (HBD), Número de receptores de enlace de hidrógeno 1 (HBA1), Número de receptores de enlace de hidrógeno 2 (HBA2), Número de átomos de flúor (nF), coeficiente de partición octanol / agua (logP), Filtro de peso molecular (MW), Número de enlaces triples (tbonds), refractividad molar (MR), Número de enlaces aromáticos (abonds), Número de enlaces simples (sbonds), Número de enlaces dobles (dbonds), área de la superficie polar topológica (TPSA), Filtro de enlaces rotativos (rotors) y Punto de fusión (MP).

- Normalización de los datos: Debido a la diferencia de rango de valores entre los descriptores numéricos, una buena práctica para el correcto funcionamiento de un modelo de Deep learning, es la normalización de los datos. Para este trabajo, se hizo uso de la función llamada Normalizer de la biblioteca Sklearn, debido a que normaliza el rango de datos entre 0 y 1 columna a columna, empleando la desviación estándar.
- Modelo de Deep learning: En este caso, debido a que el objetivo del modelo de Deep learning es la predicción de si un fármaco es tóxico o no tóxico, se llevaron a cabo las siguientes bases para la creación de un modelo de calificación binaria:
    a) Creación de una red neuronal con capas profundas o completamente conectadas
    b) La capa de salida será de 1 nodo con función de activación sigmoidal, debido a que se desea que arroje 0 ó 1 para predecir si el fármaco es tóxico o no tóxico.

c) Función de pérdida: entropía binaria cruzada cuya función es describir la pérdida entre las dos distribuciones de probabilidad.

La descripción a detalle de los modelos de hardware y versiones de Software se realizará en la sección de materiales del capítulo IV.

3.2. Nivel de estudio

En este trabajo de tesis, se implementaron dos tipos de nivel de estudio: exploratorio y explicativo. Exploratorio, debido a que se realizó un estudio del estado del arte para recopilar información y dar base a la investigación. Por otra parte, se considera explicativo, ya que se probaron distintos componentes de arquitectura e hiperparámetros, para examinar como afectan en el desempeño del modelo y lograr la creación de un modelo de Deep learning.

CAPÍTULO IV MATERIALES Y MÉTODOS

4.1. Materiales

Para el desarrollo de este trabajo se hizo uso de diferentes herramientas de hardware y software, que se describirán en la subsección 4.1.1.

4.1.1. Hardware & Software

La Tabla 1, muestra los elementos de hardware con los que cuenta la máquina que se utilizó, para la realización de las pruebas del modelo de Deep learning y las herramientas de software empleadas para la creación del modelo de Deep learning.

Hardware/Software	**Marca**	**Modelo/Versión**	**Proveedor/ Desarrollador**	**Observaciones**
Tarjeta gráfica	NVIDIA	GEFORCE GTX 1660 Ti	NVIDIA	Versión de controladores NVIDIA: 430.09 Empleando Cuda 10.1
Disco Duro	SSD 2.5" 1TB BLUE	WD	WD	SATA3 6GB/S 7MM
Computadora de escritorio	LENOVO	V530S-07ICB 10TY001ELS	LENOVO	Procesador: Intel Core i7 de 2 a 4.3 GHz
Sistema Operativo	Linux Mint	19.7 tessa	Linux	Edición: Cinnamon
Lenguaje de programación	Phyton	3.6		
Plataforma para la creación de modelos de Deep learning	Tensorflow	2.0		
Biblioteca para la creación de las capas en la red neuronal	Keras	2.2		
Biblioteca para el preprocesamiento (normalización) de los datos	Scikit-learn	0.23		
Biblioteca para creación del conjunto de datos en formato csv	Pandas	1.0.5		
Biblioteca para creación y visualización de gráficos	Matplotlib	3.2		

Tabla 1. Descripción de los elementos de hardware y software

4.2. Métodos

En esta sección, se abordará la descripción de los datos empleados para alimentar el modelo de Deep learning y las métricas para la evaluación del rendimiento del mismo.

4.2.1. Población y muestra

- Población

Tox21, es un programa que ha logrado desarrollar un conjunto de datos de 10,000 compuestos, entre los cuales se encuentran diferentes químicos ambientales y fármacos aprobados. Este conjunto está dividido en 12 ensayos, en los cuales se da prioridad a la evaluación toxicológica de los fármacos. Con el fin de permitir el desarrollo de modelos de predicción toxicológica, que permitan reducir tiempo, esfuerzo y costos asociados a las pruebas del desarrollo de fármacos, que conlleva a reducir y remplazar las pruebas *in vivo* realizadas en animales.

- Muestra

  Para la muestra se seleccionó el ensayo llamado AhR en archivo SMILES. El ensayo cuenta con 8169 moléculas, y cuenta con 3 columnas: nombre del compuesto, si activa o no el AhR y su cadena smiles.

4.2.2. Métodos utilizados

Para la evaluación del rendimiento y efectividad del modelo de Deep learning, se seleccionaron tres métricas distintas, las cuales se mencionarán a continuación:

1. Valor de Pérdida: Existen dos valores de pérdida: la pérdida en el entrenamiento (*loss*) y la pérdida en evaluación (*validation loss* o *val_loss*). El *loss,* nos indica el error de predicción del conjunto de datos de entrenamiento. *El val_loss,* indica el porcentaje de error de predicción, que presenta el modelo con los datos no conocidos del conjunto de datos de entrenamiento, además es en el que más se basa el análisis de los resultados del modelo. Debido a que si el val_loss es más alto que el *loss,* indica que existe un sobre-entrenamiento.

2. Valor de exactitud: La exactitud (accuracy), mide el porcentaje de casos que el modelo ha acertado. Al igual que en la pérdida, existen dos valores de exactitud, accuracy que mide el porcentaje de exactitud en el set de entrenamiento y val_accuracy que utiliza el conjunto de datos de validación con datos que el modelo no conoce.
3. AUC: El AUC es el área bajo la curva Característica Operativa del Receptor (ROC por sus siglas en inglés), esta métrica es aplicada en problemas de clasificación. La curva ROC (Hanley & McNeil, 1982), indica qué tan preciso es el modelo para distinguir entre dos clasificaciones. Por ejemplo, si un paciente tiene cáncer o no, se dice que un modelo es óptimo cuando es capaz de distinguir con precisión entre las dos clasificaciones. Por otra parte, el AUC calcula el área bidimensional ubicada debajo de la curva ROC, es decir, mide la eficacia de clasificación de las predicciones, en lugar de sus valores absolutos, y la calidad de las predicciones del modelo.

CAPÍTULO V DISEÑO Y EXPERIMENTACIÓN

El objetivo principal de esta investigación se centra en el desarrollo de un modelo de clasificación binaria para la predicción de toxicidad, empleando las técnicas de Deep learning. Sin embargo, los diferentes componentes e hiperparámetros que permite emplear el Deep learning, dan lugar a una gran cantidad de combinaciones posibles para la creación de un modelo de clasificación binaria. Por lo cual, se realizó un estudio del estado del arte, para extraer los componentes de diferentes arquitecturas de modelos empleados para la predicción de toxicidad.

5.1. Modelos

En base a la investigación del estado del arte, se seleccionaron 3 artículos (Abdul et al., 2019; Karim et al., 2019; Mayr et al., 2016), y se extrajeron los componentes de las arquitecturas empleadas en los modelos. Con los componentes de las diferentes arquitecturas se crearon 3 modelos que abreviaremos como: Prueba_1 (Mayr et al., 2016), Prueba_2 (Karim et al., 2019) y Prueba_3 (Abdul et al., 2019). Los cuales al observar y detectar sus componentes e hiperparámetros más significativos, llevaron a la creación de la arquitectura del modelo propuesto y creado para este trabajo de tesis, al que nos referiremos como Modelo_binario.

A continuación, en la Tabla 2 se describe a detalle cada uno de los componentes de la arquitectura de los cuatro modelos y el nombre de cada uno de los artículos de los que se extrajeron las arquitecturas de los modelos Prueba_1, Prueba_2 y Prueba_3.

Modelo	Conjunto de datos	Normalización	Capas	Optimización	Validación
Prueba_1 Artículo: Toxicity Prediction using Deep Learning	2500 descriptores biblioteca Chemopy	Normalización de (descriptores) con la desviación estándar	1ª: RELU – entrada 2ª: RELU 1024 nodos 3ª: RELU 4096 nodos 4ª: sigmoidal – salida Dropout: 20% en la capa de entrada. 50% en la capa oculta.	Gradiente descendente estocástica (SGD)	10 splits empleando la validación cruzada K-fold
Prueba_2 Artículo: Efficient Toxicity Prediction via Simple Features Using Shallow Neural Networks and Decision Trees	270 descriptores biblioteca Padel		1ª: RELU – entrada. 2ª: RELU 10 nodos. 3ª: sigmoidal – salida. Dropout: 50% en la capa oculta.	Estimación de Momento de Adaptativa a (Adam)	
Prueba_3 Artículo: Toxicity Prediction by Multimodal Deep Learning	1422 descriptores de Padel		1ª: sigmoidal - entrada 2ª: RELU 100 nodos 3ª: RELU 100 nodos 4ª: sigmoidal– salida Dropout: 10% en la capa oculta.		
Modelo_binario	8 descriptores de Pybel		1ª: RELU - entrada 2ª: RELU 6 nodos 3ª: sigmoidal– salida		

Tabla 2. Descripción de las características del modelo Prueba_1

5.1.1. Arquitectura del Modelo_binario

Para la creación del Modelo_binario, se observó el comportamiento de los distintos hiperparámetros y características de las arquitecturas de los modelos Prueba_1, Prueba_2 y Prueba_3. Esto, con la finalidad de optimizar la selección de los elementos que conforman la arquitectura del Modelo_binario, como: el número de capas ocultas, funciones de activación de las capas y función de optimización.

El modelo binario está conformado por 3 capas:

- Capa de entrada: la capa de entrada configurada con la función de activación RELU.
- Capa oculta: al igual que la capa de entrada, emplea la función de activación RELU y cuenta con 6 nodos. La función RELU, fue seleccionada ya que demuestra evitar el desvanecimiento de la gradiente y saturación, esto gracias a que es una función rectificada, esto significa que rectifica entre 0 ó 1 los valores de entrada ya sean positivos o negativos.
- Capa de salida: emplea la función de activación sigmoidal y cuenta con 1 nodo debido a que el modelo es de clasificación binaria y predecirá con 0 ó 1.

Para el Modelo_binario se seleccionó la función ADAM como optimizador y la entropía binaria cruzada como función de pérdida.

5.2. Conjuntos de datos

En cada uno de los artículos de donde se extrajeron los componentes de la arquitectura de los modelos: Prueba_1, Prueba_2 y Prueba_3, se utilizaron una cantidad de bibliotecas y cantidad de descriptores diferentes. Por lo tanto, para probar las arquitecturas de Prueba_1, Prueba_2, Prueba_3 y Modelo_binario, se crearon 4 conjuntos de datos diferentes. La diferencia en los 4 conjuntos de datos reside en la cantidad de descriptores, esta cantidad se seleccionó empleando un porcentaje en proporción a la cantidad utilizada en los artículos, distribuyendo los 16 descriptores numéricos que permite calcular la herramienta Pybel (O'Boyle et al., 2008).

- Conjunto de datos 1: En este conjunto de datos se calcularon los 16 descriptores de Pybel: número de átomos (atoms), número de enlaces (bonds), número de donantes de enlace de hidrógeno (HBD), número de receptores de enlace de hidrógeno 1 (HBA1), número de receptores de enlace de hidrógeno 2 (HBA2), número de átomos de flúor (nF), coeficiente de partición octanol / agua (logP), filtro de peso molecular (MW), número

de enlaces triples (tbonds), refractividad molar (MR), número de enlaces aromáticos (abonds), número de enlaces simples (sbonds), número de enlaces dobles (dbonds), área de la superficie polar topológica (TPSA), filtro de enlaces rotativos (rotors) y punto de fusión (MP).

- Conjunto de datos_2: Este conjunto de datos está conformado por 8 de los descriptores de Pybel: abonds, atoms, bonds, HBA1, HBA2, HBD, logP y TPSA.
- Conjunto de datos_3: Este conjunto de datos está conformado por 8 de los descriptores que no fueron seleccionados para el conjunto de datos_3: dbonds, HBA2, MP, MR, MW, nF, rotors y sbonds.
- Conjunto de datos_4: Este conjunto de datos está conformado por 4 de los descriptores numéricos de Pybel: HBA1, HBD, logP y TPSA.

5.3. Pruebas

Para observar el comportamiento de los diferentes componentes en la arquitectura y parámetros de los modelos: Prueba_1, Prueba_2, Prueba_3 y Modelo_binario, con diferentes cantidades y selección de descriptores, se realizó un entrenamiento de cada uno de los modelos alimentándolos con cada uno de los conjuntos de datos. Asimismo, para una mayor comprensión de los resultados de los modelos, y validar si muestra una generalización adecuada, es decir, que al presentarle datos que no conoce los clasificará adecuadamente, se empleó una validación cruzada con 10 splits.

CAPÍTULO VI ANÁLISIS DE RESULTADOS

En los capítulos anteriores, se describieron los objetivos, procedimientos y características para desarrollar un modelo de clasificación binaria para la predicción de toxicidad empleando técnicas de Deep learning. Por lo tanto, en esta sección se analizarán los resultados obtenidos de este trabajo.

6.1. Resultados

Uno de los principales puntos a destacar en las diferentes pruebas, es el tiempo de ejecución en cada época. Dado que el modelo Prueba_1, fue el único que, en cada una de las pruebas con los 4 conjuntos de datos diferentes, demoraba 5 segundos en cada época alcanzando una precisión promedio del 89%. Por lo que se decidió realizar una prueba extra con este modelo.

En dicha prueba, se redujo el número de nodos de las capas ocultas al 25% de su total de nodos, es decir, en la primera capa oculta se redujeron los nodos de 1024 a 256 y en la segunda de 4096 a 1024, y se utilizó el Conjunto de datos_1. Como resultado de esta prueba, el tiempo de cada época se redujo a 1 segundo y la precisión promedio fue del 88.6%, mostrando una reducción del 0.4%. Esta misma configuración en los nodos, se utilizó para realizar una prueba con el Conjunto de datos_2, la cual mostró un incremento en la precisión, alcanzando un promedio del 90%.

A continuación, en la Tabla 3 se muestran los resultados del porcentaje de pérdida y exactitud obtenidos de aplicar la validación cruzada, empleando 10 splits en el entrenamiento de las pruebas realizadas a los cuatro modelos con cada uno de los conjuntos de datos.

Modelo	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	Pérdida	Exactitud
Prueba_1	x	x	x	x	x	X	x	x	X	x	x	x	x	x	x	x	**28.58%**	**88.53%**
	x	x	x	x	x		x				x			x			**29.30%**	**88.37%**
						X		x	X	x		x			x	x	**29.91%**	**87.45%**
			x	x			x							x			**34.31%**	**88.37%**
Prueba_2	x	x	x	x	x	X	x	x	x	x	x	x	x	x	x	x	**27.63%**	**83.38%**
	x	x	x	x	x		x				x			x			**28.94%**	**81.74%**
						X		x	x	x		x			x	x	**27.76%**	**77.26%**
			x	x			x							x			**33.60%**	**66.01%**
Prueba_3	x	x	x	x	x	X	x	x	x	x	x	x	x	x	x	x	**25.31%**	**84.52%**
	x	x	x	x	x		x				x			x			**26.49%**	**83.71%**
						X		x	x	x		x			x	x	**29.28%**	**79.92%**
			x	x			x							x			**32.91%**	**70.80%**
modelo_binario	x	x	x	x	x	X	x	x	x	x	x	x	x	x	x	x	**26.79%**	**89.84%**
	x	x	x	x	x		x				x			x			**27.69%**	**89.62%**
						X		x	x	x		x			x	x	**30.09%**	**88.16%**
			x	x			x							x			**34.91%**	**88.37%**

Tabla 3. Resultados de pérdida y optimización obtenidos al realizar las pruebas con los diferentes modelos y conjuntos de datos. Modelo es el nombre del modelo evaluado, las columnas 2 a 17 son los descriptores numéricos de Pybel y se marcaron con una x los descriptores utilizados en esa prueba, d1= atoms, d2= bonds, d3= HBD, d4= HBA1, d5=HBA2, d6=nF, d7=logP, d8=MW, d9= tbonds, d10=MR, d11=Abonds, d12= sbonds, d13=dbonds, d14=TPSA, d15=rotors y d16=MP

Al observar la Tabla 3, es posible apreciar, que en los cuatro modelos influye de manera significativa la cantidad de descriptores y la selección de estos para cada conjunto de datos. Debido a que, a menor cantidad de descriptores, la precisión o exactitud disminuyó y la pérdida

aumento. Asimismo, al emplear el Conjunto de datos_2 y Conjunto de datos_3 (aunque contienen la misma cantidad de descriptores) los 8 descriptores seleccionados para el Conjunto de datos_2, mostraron un mejor rendimiento en comparación al Conjunto de datos_3, ya que, al emplear este conjunto de datos para el entrenamiento de los modelos, la exactitud disminuyó entre el 2% y 4%, y la pérdida aumento de 4% a 6%.

A pesar de que al observar la Tabla 3, el modelo Prueba_1 se encuentra dentro de los primeros dos modelos que obtuvieron los mejores resultados, la Figura 3 en donde se observa el entrenamiento de Prueba_1 con 20 épocas, muestra que al emplear el optimizador SGD y una gran cantidad de nodos en las capas ocultas, provocó un desvanecimiento de la gradiente, en otras palabras, los pesos y parámetros de la red neuronal no se fueron ajustando en cada época, y crearon un ciclo en el cual la exactitud no aumentó, provocando un sobreajuste debido a la gran cantidad de capas ocultas.

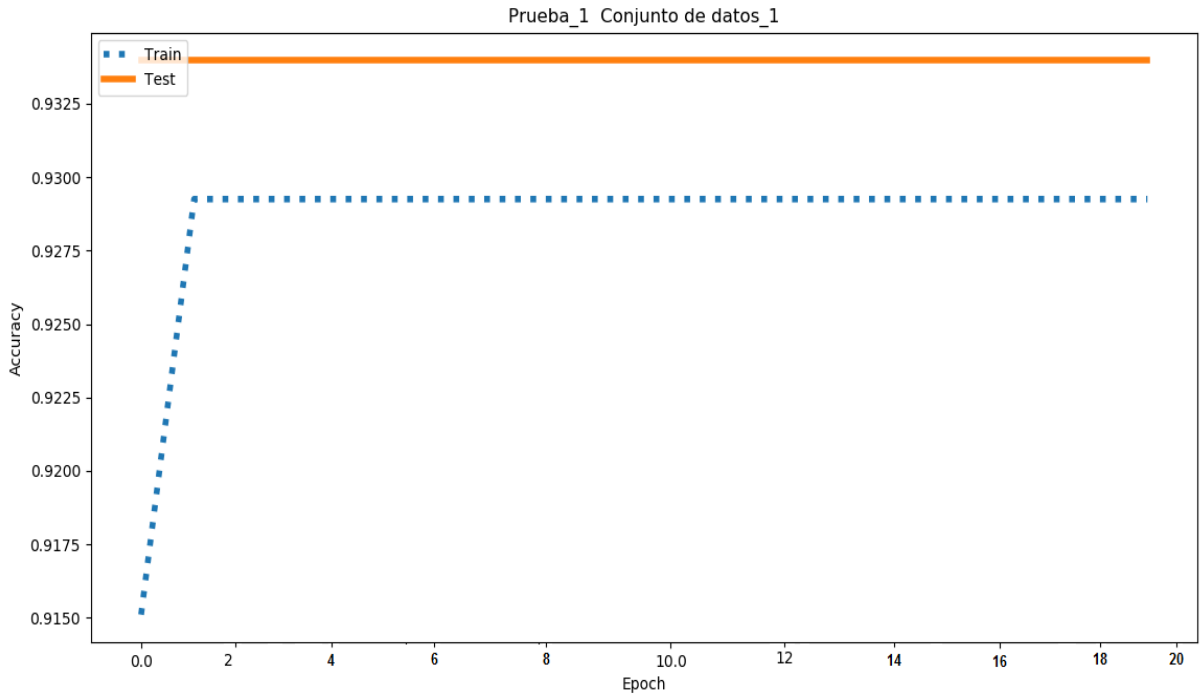

Figura 3 Resultados del entrenamiento de Prueba_1 con el Conjunto de datos_1

En la Figura 4, se muestra el resultado del entrenamiento de Prueba_1 y Modelo_binario, ambos alimentados con el Conjunto de datos_1, empleando 20 épocas y una validación cruzada de 10 splits. En esta gráfica, es posible notar la mejoría en la pérdida y exactitud del Modelo_binario, que emplea el optimizador Adam y una capa oculta con 6 nodos, en comparación con Prueba_1.

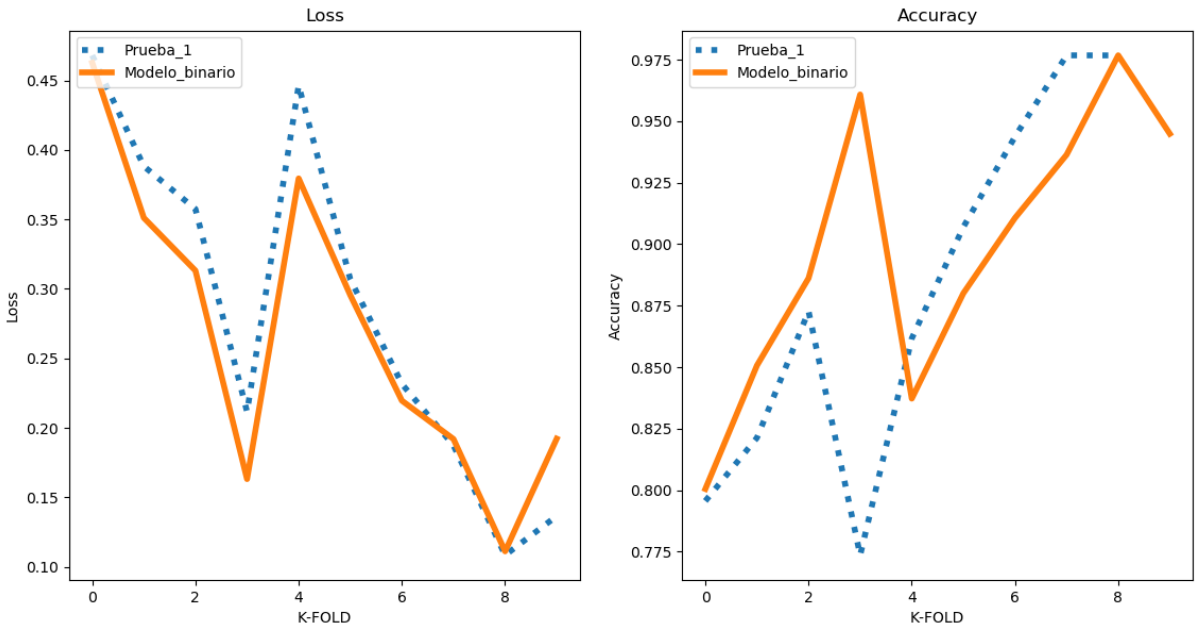

Figura 4. Entrenamiento de Prueba_1 y modelo binario con el Conjunto de datos_1

Para obtener un análisis más detallado de los resultados obtenidos de Prueba_2, en el cual se produjo un bajo rendimiento en los resultados de perdida y exactitud, se realizó una matriz de confusión. En ella, se muestra la cantidad de compuestos que se predijeron correctamente como tóxicos y no tóxicos, además de la cantidad de veces que la predicción de los compuestos tóxicos o no tóxicos se realizó incorrectamente.

La Figura 5, muestra la matriz de confusión del entrenamiento realizado con Prueba_2 utilizando el Conjunto de datos_4. No obstante, sin importar el conjunto de datos con que se entrenara Prueba_2, obtuvo los mismos resultados en la matriz de confusión, cada una de las moléculas las clasificaba como no tóxicas.

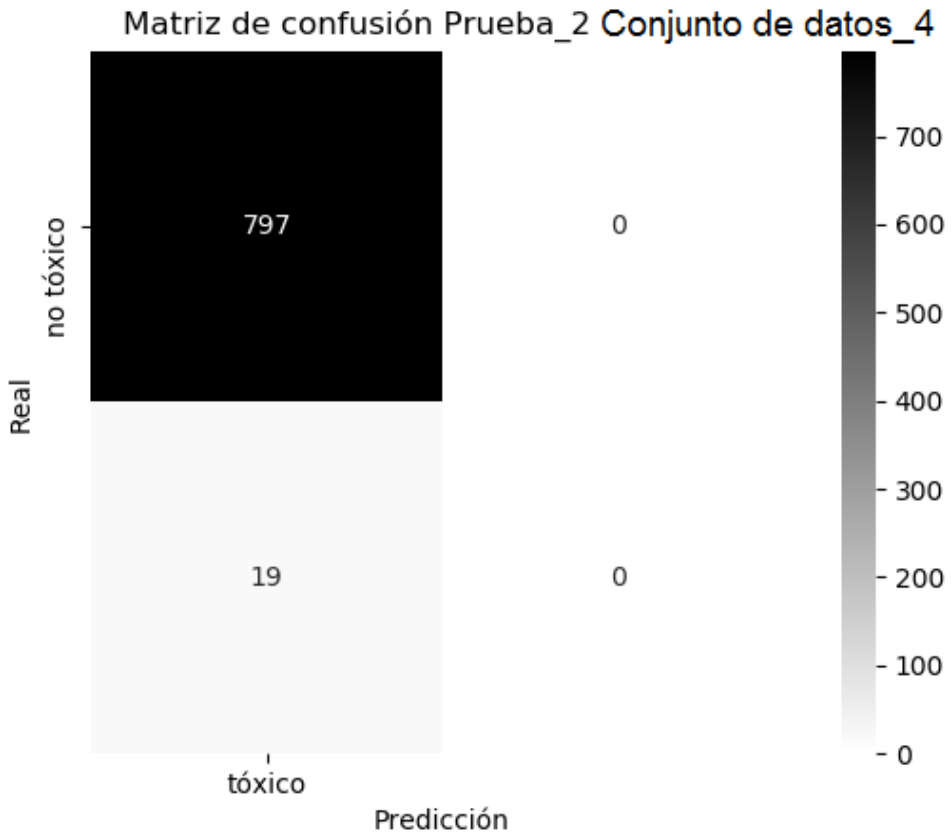

Figura 5 matriz de confusión del entrenamiento realizado con Prueba_2

Al implementar este tipo de análisis más a fondo, fue posible descubrir que Prueba_3, alimentado con el Conjunto de datos_1, fue el que mejor clasificó los compuestos como tóxicos y presento el valor de pérdida más bajo. Esto fue posible encontrarlo gracias a su matriz de confusión, que se muestra en la Figura 6 y al reporte de clasificación en la Tabla 4.

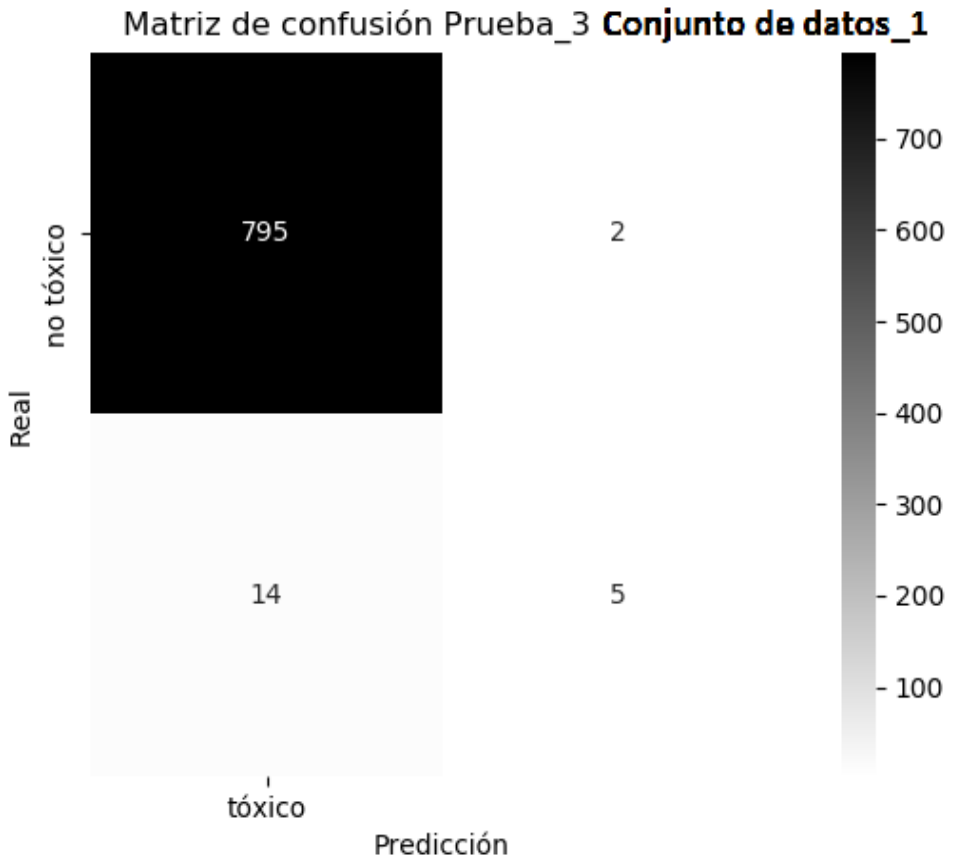

Figura 6. matriz de confusión del entrenamiento realizado con Prueba_3 con el Conjunto de datos_1

	Precision	**recall**	**f1-score**	**Support**
0	0.98	1.00	0.99	797
1	0.71	0.26	0.38	19
Accuracy			0.98	816
macro avg	0.85	0.63	0.69	816
weighted avg	0.98	0.98	0.98	816

Tabla 4. Reporte de clasificación obtenido del entrenamiento realizado con Prueba_3 con el Conjunto de datos_1

Por otra parte, el Modelo_binario empleando el Conjunto de datos_1, obtuvo unos resultados muy similares en la matriz de confusión que se muestra en la Figura 7 y el reporte de clasificación en la Tabla 5, a pesar de emplear solamente una capa oculta con 6 nodos. Además, en la tabla comparativa que se muestra en la Tabla 3 demostró un 5% más de exactitud que Prueba_3, cuando se empleó con el Conjunto de datos_1.

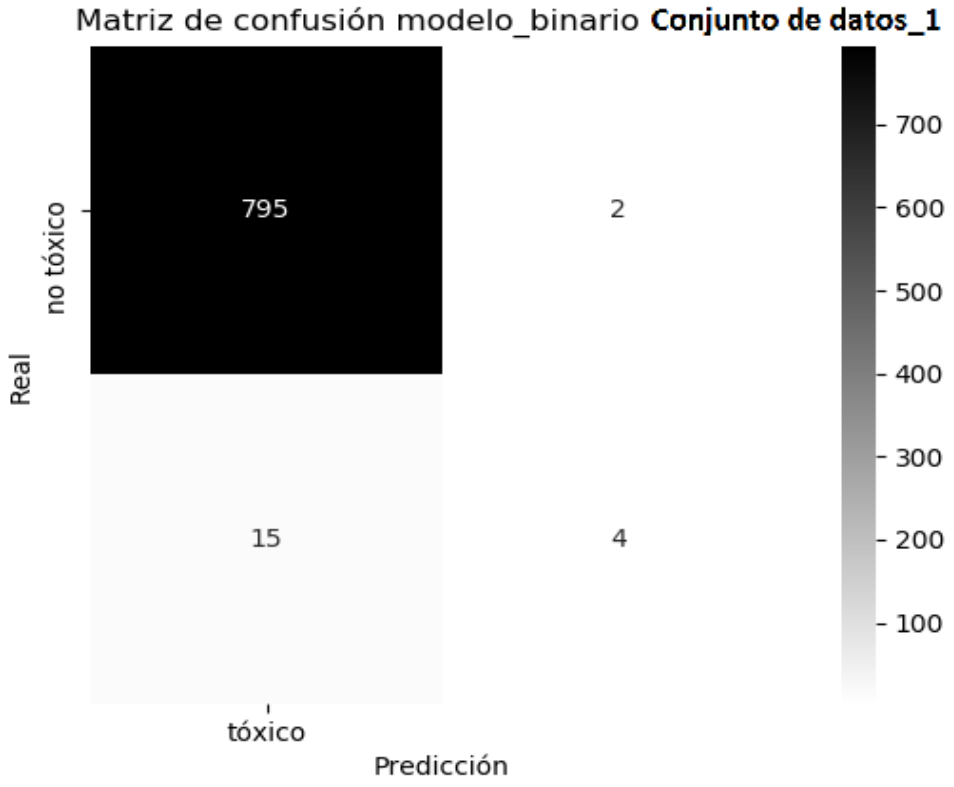

Figura 7. matriz de confusión del entrenamiento realizado con modelo_binario con Conjunto de datos_1

	Precision	**recall**	**f1-score**	**Support**
0	0.98	1.00	0.99	797
1	0.67	0.21	0.32	19
Accuracy			0.98	816
macro avg	0.82	0.60	0.65	816
weighted avg	0.97	0.98	0.97	816

Tabla 5. Reporte de clasificación del entrenamiento de modelo_binario con Conjunto de datos_1

Por otra parte, la cantidad de capas ocultas y optimizadores jugaron un papel importante en el rendimiento de los diferentes modelos. En este caso, Prueba_1 y el Modelo_binario, mostraron los mejores resultados de los 4 modelos con cada uno de los conjuntos de datos, sin embargo, ambos modelos obtuvieron los mejores resultados al emplear el Conjunto de datos_1. No obstante, el Modelo_binario mostró una menor pérdida y una mayor precisión al utilizar el optimizador Adam y una capa oculta.

6.2. Discusión

En la prueba realizada al modelo Prueba_1, en la cual se redujo la cantidad de nodos en las capas ocultas, se produjo una notable reducción en el tiempo de ejecución y fue mínima la cantidad que se redujo la exactitud. No obstante, al reducir simultáneamente el conjunto de datos y seleccionar 8 de los 16 descriptores y reducir el número de nodos en las capas ocultas, la precisión aumento 1%. Por su parte, el Modelo_binario, el cual contiene la menor cantidad de nodos y una capa oculta mostró los mejores resultados. Además, pese a que Prueba_2 cuenta con características similares a la del Modelo_binario, éste fue el que obtuvo los resultados más bajos en precisión y más altos en la pérdida, por lo que el Dropout empleado en la capa oculta de Prueba_2 no resultó ser beneficioso.

En los 3 artículos, de los cuales se extrajeron los componentes de la arquitectura de sus modelos originales, se empleó como métrica el AUC. Debido a que el AUC demuestra si el modelo predice correctamente, por lo cual, si obtiene un AUC del 50%, se dice que el modelo no tiene la habilidad para distinguir las clases. El modelo de (Mayr et al., 2016) con 1024 nodos en la 1er capa oculta y 4096 en la 2a capa oculta, y con un conjunto de datos de 2500 descriptores obtuvo un AUC de 84.87; el modelo de (Karim et al., 2019) empleando una capa oculta con 10 nodos y 270 descriptores en el conjunto de datos mostró un AUC de 83.6%; el modelo de (Abdul et al., 2019) con dos capas ocultas de 100 nodos cada una y un conjunto de datos de 1422 descriptores mostró un AUC del 82%. No obstante, a pesar de no emplear una capa oculta con cientos o miles de nodos y el Conjunto de datos_1 que contiene 16, el Modelo_binario arrojó un AUC del 81.91%.

CAPÍTULO VII CONCLUSIONES

Tras observar y analizar los resultados del empleó de los diferentes parámetros para el desarrollo de un modelo de clasificación binaria, que clasifique anticipadamente si un fármaco será toxico o no. Es posible decir, que los datos con que se alimenta la red neuronal son de suma importancia, ya que la cantidad o selección inadecuada de las características afecta el rendimiento del modelo.

Al realizar un modelo de clasificación binaria, es importante el uso de diferentes métricas en la evaluación de los modelos de prueba, así como comprender el funcionamiento de las diferentes características y parámetros, como son: la cantidad de capas, nodos, funciones de pérdida y funciones de optimización. Para así desarrollar un modelo con las características y los parámetros óptimos, para realizar una correcta predicción.

Otro aspecto para considerar es la selección de la cantidad de nodos y las funciones de optimización que se utilizarán en un modelo de Deep learning. Debido a que estos deben de ajustarse al trabajo y tipo de modelo que se desee realizar, ya que fácilmente se puede caer en un sobreajuste o subajuste, lo que puede terminar en una mala predicción por una incorrecta selección de características o cantidad de nodos.

Las propiedades fisicoquímicas moleculares o descriptores moleculares pueden ser calculados mediante diferentes herramientas computacionales. Asimismo, la cantidad de descriptores disponibles, para ser calculados, va a depender del software con el cual serán calculados, y puede ir desde una decena a miles de descriptores. Por otra parte, la selección de una gran cantidad de descriptores como características en una red neuronal, no está ligado a que el modelo de Deep learning realice una buena predicción. Por lo cual, la selección de descriptores que serán utilizados debe realizarse adecuadamente, y en este caso, los 16 descriptores numéricos que permite calcular la biblioteca Pybel, demostraron un buen comportamiento para predicción de toxicidad.

Referencias

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning This paper is included in the Proceedings of the TensorFlow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16).*, *12*.

Abdul, K., Singh, J., Avinash, M., Abdollah, D., & M. A., Hakim Newton Abdul, S. (2019). Toxicity Prediction by Multimodal Deep Learning. *Knowledge Management and Acquisition for Intelligent Systems, 16th Pacif*(Lecture Notes in Artificial Intelligence), 142–152. https://doi.org/https://doi.org/10.1007/978-3-030-30639-7

Aguirre Valderrama, A. (2009). *Grupo de Modelización y Diseño Molecular Departamento de Química Orgánica Facultad de Ciencias*.

Al-rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-lewandowski, N., Bouthillier, X., Br, A. De, Breuleux, O., … Warde-farley, D. (2016). *Theano: A Python framework for fast computation of mathematical expressions*. 1–19.

Arnold, L., Rebecchi, S., & Chevallier, S. (2011). An Introduction to Deep Learning. *European Symposium on Artificial Neural Networks (ESANN), January*.

Atkinson, A. J., & Markey, S. P. (2007). Biochemical Mechanisms of Drug Toxicity. *Principles of Clinical Pharmacology, Figure I*, 249–271. https://doi.org/10.1016/B978-012369417-1/50056-0

Bogdanchikov, A., Zhaparov, M., & Suliyev, R. (2013). Python to learn programming. *Journal of Physics: Conference Series*, *423*(1), 12027. https://doi.org/10.1088/1742-6596/423/1/012027

Brownlee, J. (2019). *How to Choose Loss Functions When Training Deep Learning Neural Networks*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-

choose-loss-functions-when-training-deep-learning-neural-networks/

Chollet, F. (2015). *Keras*. GitHub Repository. https://github.com/fchollet/keras

Dearden C.J. (2003). In silico prediction of drug toxicity. *Journal of Computer Aided Molecular Design*, *17*(5), 119–127.

Díaz-Narváez V.P., V. P., & Calzadilla-Núñez A., A. (2016). Artículos científicos, tipos de investigación y productividad científica en las Ciencias de la Salud. *Ciencias de La Salud*, *14*(1), 115–121. https://doi.org/10.12804/revsalud14.01.2016.10

Fernandez. (2016). *Las nuevas tecnologías aceleran el desarrollo de fármacos*. Elsevier. https://www.elsevier.com/es-es/connect/ciencia/las-nuevas-tecnologias-aceleran-el-desarrollo-de-farmacos

García-Pérez, C., Peláez, R., Therón, R., & López-Pérez, J. L. (2016). JADOPPT: Java based AutoDock Preparing and Processing Tool. *Bioinformatics*, *14*, btw677. https://doi.org/10.1093/bioinformatics/btw677

García Velázquez, M. del R., & Hernández Gracia, T. J. (2013). Metodología de la investigación. *Boletín Científico de Las Ciencias Económico Administrativas Del ICEA*, *2*(3). https://doi.org/10.29057/icea.v2i3.61

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT PRESS. http://www.deeplearningbook.org/

Guerrero-Villalobos, L. R., & López, F. O. (2017). Rational Design, Synthesis and Characterization of Hybrid Molecules With Pyrazoline, Pyrimidine and Thiazolidine Nuclei As Potential Antibacterial Agents. *CBU International Conference Proceedings*, *5*, 1096–1103. https://doi.org/10.12955/cbup.v5.1077

Gulli, A., & Pal, S. (2017). *Deep Learning with Keras Implement neural networks with Keras on Theano and TensorFlow*.

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29–36. https://doi.org/10.1148/radiology.143.1.7063747

Hilera, J., & Martínez, V. (1995). Redes neuronales artificiales: fundamentos, modelos y

aplicaciones. In *Madrid: Ra-ma* (1st ed., Issue January). RAMA. http://en.scientificcommons.org/7007722

Hong, H., & Science, R. (2019). *Advances in Computational Toxicology* (Vol. 30). https://doi.org/10.1007/978-3-030-16443-0

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Ian H., W., & Frank, E. (2005). DATA MINING Practical Machine Learning Tools and Techniques. In *Journal of Chemical Information and Modeling* (2nd ed., Vol. 53, Issue 9). Elsevier.

Juaristi, E., & Rodriguez Jorge, L. F. (2016). *Importancia de la computacion en la ciencia y en nuestra vida diaria* (primera). El Colegio Nacional.

Karim, A., Mishra, A., Newton, M. A. H., & Sattar, A. (2019). Efficient Toxicity Prediction via Simple Features Using Shallow Neural Networks and Decision Trees [Research-article]. *ACS Omega*, *4*(1), 1874–1888. https://doi.org/10.1021/acsomega.8b03173

Konar, A. (1999). *Artificial Intelligence and Soft Computing Artificial Intelligence and Soft Computing Behavioral and Cognitive Modeling* (Primera).

López-Camacho, E., García Godoy, M. J., García-Nieto, J., Nebro, A. J., & Aldana-Montes, J. F. (2015). Docking Inter/Intra-Molecular Mediante Metaheurísticas Multi-objetivo. *LCC - Conferencias Cientificas*. http://dspace.uma.es/xmlui/handle/10630/8808

Marovac, J. (2001). Investigación y desarrollo de nuevos medicamentos: de la molécula al fármaco. *Revista Médica de Chile*, *129*(1), 99–106. https://doi.org/10.4067/S0034-98872001000100015

Mayr, A., Klambauer, G., Unterthiner, T., & Hochreiter, S. (2016). DeepTox: Toxicity prediction using deep learning. *Frontiers in Environmental Science*, *3*(FEB). https://doi.org/10.3389/fenvs.2015.00080

McKinney, W. and others. (2010). PANDAS: Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56. http://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf

Medina-Franco, J. L., Fernández-de Gortari, E., Naveja, J. J., Medina-Franco, J. L., Fernández-de Gortari, E., & Naveja, J. J. (2015). Avances en el diseño de fármacos asistido por computadora. *Educación Química*, *26*(3), 180–186. https://doi.org/10.1016/j.eq.2015.05.002

Muster, W., Breidenbach, A., Fischer, H., Kirchner, S., Müller, L., & Pähler, A. (2008). Computational toxicology in drug development. *Drug Discovery Today*, *13*(7–8), 303–310. https://doi.org/10.1016/j.drudis.2007.12.007

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1–21. https://doi.org/10.1186/s40537-014-0007-7

Nikhil Buduma. (2016). Fundamentals of Deep Learning_ Designing Next-Generation Machine Intelligence Algorithms. In *Nature*. https://doi.org/10.1016/j.compchemeng.2004.08.021

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 1–20. http://arxiv.org/abs/1811.03378

O'Boyle, N. M., Morley, C., & Hutchison, G. R. (2008). Pybel: A Python wrapper for the OpenBabel cheminformatics toolkit. *Chemistry Central Journal*, *2*(1), 1–7. https://doi.org/10.1186/1752-153X-2-5

Pawan, J. (2019). *Compute Guide to Activation Functions*. Towards Data Science.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Riley, A. L., & Kohut, S. (2010). Drug Toxicity. In I. P. Stolerman (Ed.), *Encyclopedia of Psychopharmacology* (p. 441). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-68706-1_1131

Saldívar-González, F., Prieto-Martínez, F. D., & Medina-Franco, J. L. (2017). Descubrimiento y desarrollo de fármacos: un enfoque computacional. *Educacion Quimica*, *28*(1), 51–58. https://doi.org/10.1016/j.eq.2016.06.002

Sánchez Montero, J. M. (2016). Molecular modeling methodologies in the design, synthesis and rational explanation of results. *ANALES DE LA REAL ACADEMIA NACIONAL DE FARMACIA*, *82*(2), 168–184. http://www.analesranf.com/index.php/aranf/article/view/1708

Sharma, S. (2017). *Activation functions in neural networks*. Towards Data Science.

Thomas, R. S., Paules, R. S., Simeonov, A., Fitzpatrick, S. C., Crofton, K. M., Casey, W. M., & Mendrick, D. L. (2018). The US Federal Tox21 Program: A strategic and operational plan for continued leadership. *Altex*, *35*(2), 163–168. https://doi.org/10.14573/altex.1803011

Tinta, S., & Daen, M. (2011). TIPOS DE INVESTIGACIÓN CIENTÍFICA. *Revista de Actualizacion Clínica*, *9*, 621–624.

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, *13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

Van Norman, G. A. (2016). Drugs, Devices, and the FDA: Part 1: An Overview of Approval Processes for Drugs. *JACC: Basic to Translational Science*, *1*(3), 170–179. https://doi.org/10.1016/j.jacbts.2016.03.002

Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., & Mueller, A. (2015). Scikit-learn. *GetMobile: Mobile Computing and Communications*, *19*(1), 29–33. https://doi.org/10.1145/2786984.2786995

Verbist, B., Klambauer, G., Vervoort, L., Talloen, W., Shkedy, Z., Thas, O., Bender, A., Göhlmann, H. W. H., & Hochreiter, S. (2015). Using transcriptomics to guide lead optimization in drug discovery projects: Lessons learned from the QSTAR project. *Drug Discovery Today*, *20*(5), 505–513. https://doi.org/10.1016/j.drudis.2014.12.014

Wang, X., Song, K., Li, L., & Chen, L. (2018). Structure-Based Drug Design Strategies and

Challenges. *Current Topics in Medicinal Chemistry*, *18*(12), 998–1006. https://doi.org/10.2174/1568026618666180813152921

Zhang, L., Tan, J., Han, D., & Zhu, H. (2017). From machine learning to deep learning: progress in machine intelligence for rational drug discovery. *Drug Discovery Today*, *22*(11), 1680–1685. https://doi.org/10.1016/j.drudis.2017.08.010