UNIVERSIDAD VERACRUZANA

FACULTAD DE ESTADÍSTICA E INFORMÁTICA

CLASIFICACIÓN DE REQUISITOS DE SOFTWARE MEDIANTE UN ENFOQUE NEUROEVOLUTIVO

Modalidad:
TESIS

Como requisito parcial para obtener el grado de:
LIC. EN INGENIERIA DE SOFTWARE

Presenta:
DELMER ALEJANDRO LÓPEZ HERNÁNDEZ

Directores:
DR. JORGE OCTAVIO OCHARÁN HERNÁNDEZ
DR. EFREN MEZURA MONTES

ESTADOS UNIDOS MEXICANOS
XALAPA-ENRIQUEZ, VERACRUZ
2021

DIRECTORIO

Dr. Martín Gerardo Aguilar Sánchez

Rector

Dra. Elena Rustrián Portilla

Secretaria académica

Dr. Arturo Bocardo Valle

Dirección del Área Académica Económico-Administrativa

Dr. Luis Gerardo Montané Jimenez

Dirección de la Facultad de Estadística e Informática

Dr. Jorge Octavio Ocharán Hernández

Jefatura de carrera de la Licenciatura en Ingeniería de Software

A la persona que nos motiva a ser mejores en la familia, Dylan.

AGRADECIMIENTOS

Quiero agradecer a mis padres, Gloria Hernández Zapata y Delmer López Arias, porque su sacrificio me permitió crecer en mis años como estudiante. A mis maestros, pues gracias a su atención logré formarme. A mis compañeros de carrera, que confiaron en mi y que tendieron su mano en momentos difíciles.

También quiero agradecer a Erick Jair Morales Romero y Bruno Antonio López Lujan, porque siempre mostraron su apoyo dentro y fuera de clases; a Clemente Hernández, por su tiempo dedicado a enseñarme las bibliotecas de python usadas en la tesis y por sus opiniones acerca de los resultados que compartía; al Dr. Rafael Rivera López, por enseñarme acerca de la selección de características; a mis directores Jorge Octavio Ocharán Hernández y Efrén Mezura Montes y al sinodal Ángel Juan Sánchez García, por su gran apoyo en la elaboración de la tesis, en la publicación de los artículos presentados en congresos y su enorme paciencia, dedicación y empatía mostrados a lo largo de este año; y finalmente al Mtro. Rubén Darío Gutiérrez Campo, no solo por ser un amigo incondicional, sino por inspirarme a vivir el campo de la investigación.

RESUMEN

La clasificación de requisitos de software es una tarea realizada por el humano, por lo que la vuelve propensa a errores, además de ser costoso en cuestiones de tiempo y esfuerzo. Para apoyar al ingeniero de requisitos se han aplicado algoritmos de aprendizaje máquina y así reducir el esfuerzo del humano y los errores introducidos durante la fase de análisis de requisitos. Anteriormente, Pérez-Verdejo, Sánchez-García, Ocharán-Hernández, y Mezura-Montes (2021) construyeron un conjunto de datos de requisitos de software que clasificó utilizando Evolución Diferencial con un 67.9 % de precisión. En esta tesis, se toma dicho conjunto de datos para comparar el rendimiento de una red neuronal con la neuroevolución, logrando hasta un promedio de 70.9 % y 59 % de precisión respectivamente. También se experimenta aplicando los métodos de selección de características de CFS, FCBF, MRMR y Relief en los que se obtienen un máximo de 70 % de precisión para la red neuronal y 51 % de precisión para la neuroevolución. Finalmente se realizan experimentos reduciendo la cantidad de clases en el conjunto de datos para así balancear el conjunto de datos obteniendo así una precisión del 0.78 % para la red neuronal y 0.64 % de precisión para la neuroevolución.

Palabras clave: *Clasificación de Requisitos de Software, Aprendizaje Máquina, Neuroevolución, Selección de Características*

CLASIFICACIÓN DE REQUISITOS DE SOFTWARE MEDIANTE UN ENFOQUE NEUROEVOLUTIVO

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Antecedentes

Los Requisitos de Software (RS) son parte esencial de los proyectos de desarrollo de sistemas. Son una especificación de lo que debe estar implementado en el sistema y describe cómo este se comporta para así dar restricciones durante el desarrollo (Wiegers y Beatty, 2013). Estos poseen una clasificación que usualmente los autores dividen en: requisitos funcionales y requisitos no funcionales.

La ingeniería de requisitos posee cuatro subdisciplinas que ayudan a formar los requisitos que guían el desarrollo de software. La primera, la elicitación, que ayuda a obtener los datos que se analizarán para formar los requisitos. El segundo, el análisis, que se encarga de identificar y clasificar los requisitos. El tercero, la especificación, que se encarga de definir los requisitos en una sintaxis y presentarlos formalmente. Finalmente, la validación, en la cual los requisitos especificados son presentados a los interesados para asegurarse los requisitos satisfagan sus necesidades (Wiegers y Beatty, 2013) (Bourque, Fairley, y eds., 2014).

Para un proyecto de desarrollo de software es de suma importancia el análisis de los requisitos de software. Este proceso toma tiempo, pero es de gran ayuda para guiar los esfuerzos del equipo. Por tal, clasificar los requisitos de una forma automatizada nos permite ahorrar en gran medida tiempo y esfuerzo con la ventaja de acercarnos a una clasificación correcta y consistente de los requisitos. Dentro del análisis de los requisitos, se encuentra la clasificación de los requisitos. Esta puede ser utilizada para ordenar los requisitos de acuerdo a su prioridad, su volatilidad y el tipo de requisito, ya sea funcional o no funcional (Bourque y cols., 2014).

Anteriormente, diversos autores han clasificado RS automáticamente mediante otros enfoques, como Kurtanovic y Maalej (2017), quienes buscan clasificar Requisitos No Funcionales (RNF), aplicando aprendizaje máquina utilizando una base de datos de 625 elementos siguiendo un proceso de tratado de datos en la cual, apoyándose de herramientas de análisis lingüístico, logra, con un promedio del 92 %, clasificar RNF.

Otro trabajo de clasificación automática, pero de Requisitos Funcionales (RF), es el de Salman, Hammad, Seriai, y Al-Sbou (2018), quienes logran clasificarlos utilizando un método de clustering jerárquico aglomerativo con el objetivo de agruparlos según su tipo de RF. Salman logra agrupar RF de cuatro proyectos diferentes donde el mayor promedio fue de un 83 % de precisión, mientras

que el menor fue de un 72 %.

Winkler y Vogelsang (2017) emplean redes neuronales convolucionales, una variación de redes neuronales, para distinguir entre requisitos e información en un documento de clasificación de requisitos con una precisión entre el 73 % y el 89 %.

Uno de los trabajos más relacionados con la presente tesis es el de Baker, Deng, Chakraborty, y Dehlinger (2019), los cuales proponen dos formas de clasificar requisitos de software no funcionales, uno mediante redes neuronales artificiales y otro usando redes neuronales convolucionales Ambos métodos ofrecen resultados favorables para la clasificación, siendo la red neuronal convolucional la que mejor resultados arroja con un rango de precisión entre el 82 % y 94 % en contraste con el método de red neuronal artificial con un rango de precisión de entre el 82 % y 90 %.

Finalmente la investigación de Pérez-Verdejo y cols. (2021), de la cual este trabajo de tesis se desprende, realiza la reunión de los requisitos de diversas fuentes de datos para formar su propio conjunto de datos, los preprocesa y los clasifica utilizando evolución diferencial, un algoritmo evolutivo de método de búsqueda paralela directa (Storn y Price, 1997), obteniendo un 69 % de precisión.

Existen varias propuestas para clasificar RS por distintos métodos, pero ninguno ha intentado clasificarlos combinando algoritmos evolutivos y redes neuronales. En este trabajo de tesis abordaremos esa combinación de técnicas de inteligencia artificial para satisfacer una clasificación correcta de los RS.

1.2. Planteamiento del problema

La clasificación de requisitos realizada durante el análisis de los requisitos es una actividad realizada por el humano. La cantidad de requisitos a clasificar por tipo puede llegar a ser numerosa cuando se trata de un proyecto de mayor tamaño que involucra demasiados procesos e impacta en el tiempo necesario para asignarles una clasificación. Esta tarea es también susceptible a los errores humanos, pues en el caso de que los requisitos no sean identificados y corregidos a tiempo, pueden llegar a causar efectos negativos en los recursos asignados a un proyecto de software.

Para agilizar esta tarea, autores como Winkler y Vogelsang (2017), Baker y cols. (2019), entre otros, han realizado la clasificación de los requisitos utilizando redes neuronales. Sin embargo, el

diseñar un clasificador de requisitos con redes neuronales también necesita de tiempo para poder concretarse.

Resumiendo, tanto la clasificación de los requisitos como la creación de una red neuronal optimizada para esta tarea conllevan un costo de tiempo y esfuerzo humano.

1.3. Hipótesis

La hipótesis que se se aborda es, que a través de la neuroevolución, se pueden crear redes neuronales con mínima intervención humana que compita con una red neuronal diseñada totalmente por el humano en la clasificación de requisitos de software no funcionales.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar un esquema de neuroevolución que clasifique de manera competitiva requisitos de software no funcionales (atributos de calidad). La clasificación debe dar como resultado el agrupamiento de los requisitos a ser procesados en los grupos propuestos más recurrentes de atributos de calidad de software.

1.4.2. Objetivos específicos

Dentro de los objetivos específicos de la tesis se encuentran:

- Incluir una técnica de redes neuronales artificiales con una técnica evolutiva que funcionen en conjunto para la clasificación de requisitos (esquema neuroevolutivo).
- Realizar experimentos empleando el esquema neuroevolutivo.
- Analizar resultados para el proceso de neuroevolución.

1.5. Justificación

La principal razón de esta tesis es apoyar al proceso de clasificación de requisitos de software, aportando a los ingenieros de requisitos y personas dedicadas a analizar requisitos de software una propuesta neuroevolutiva para clasificar RNF. Los beneficios generados por la automatización de este proceso radica en reducir tiempo y esfuerzo durante las primeras etapas del desarrollo de software, reducir los errores manuales humanos y malentendidos encontrados para favorecer las etapas siguientes de desarrollo como la toma de decisiones de los administradores de proyectos y los arquitectos de software de modo que estos reciban los requisitos más claros, en un menor tiempo y se logre la calidad del software.

Como se ha mencionado antes, la literatura especializada muestra propuestas de clasificación de requisitos mediante redes neuronales, pero la aplicación de un enfoque neuroevolutivo, de acuerdo al mejor conocimiento del autor, no ha sido explorada. Los resultados de la clasificación de requisitos por otras técnicas de aprendizaje máquina como las redes neuronales convolucionales, redes neuronales artificiales, entre otras, son más abundantes y han sido exploradas anteriormente por diversos autores logrando resultados positivos en rendimiento y precisión. Entre estos otros enfoques, no todos se centran en la clasificación RNF, y si lo hacen, solo clasifican un grupo seleccionado de RNF.

Desde la perspectiva de la inteligencia artificial, es importante aportar un esquema neuroevolutivo para la clasificación de requisitos, debido a que la revisión de la literatura especializada no muestra resultados en propuestas similares en el campo de inteligencia artificial aplicada a la ingeniería de software. La mayoría de las propuestas existentes se enfocan en otros procesos de la ingeniería de software mediante diferentes métodos.

La clasificación automatizada de requisitos es un campo que cuenta con diversas propuestas de inteligencia artificial, pero aún no alcanza un grado de exploración suficiente. Es vital promover el uso de la inteligencia artificial para las tareas de la ingeniería de software para expandir su uso, en particular, en la clasificación de requisitos, creando beneficios como la agilización del desarrollo de software.

1.6. Alcances y limitaciones

Con el fin de definir un alcance, es necesario decir que los experimentos se centrarán en la clasificación de requisitos de calidad en las clases de disponibilidad, tolerancia a fallos, mantenibilidad, rendimiento, escalabilidad, seguridad y usabilidad. No se abordará la clasificación en otro tipo de clases. El estudio no desarrollará un modelo de preprocesamientos de datos, puesto que los datos que se utilizan ya han pasado por la etapa de preprocesamiento en el trabajo previo de Pérez-Verdejo y cols. (2021).

Para apoyar la clasificación de requisitos mediante un enfoque neuroevolutivo, se utilizarán las librerías de NEAT en Python y otras librerías que apoyen a la visualización de los datos. No se pretende incluir otros lenguajes de programación para desarrollar el clasificador de requisitos.

También dentro de nuestro alcance, se contempla la publicación de artículos derivados de este trabajo de tesis, siendo uno de estos una revisión sistemática de la literatura para resaltar la relevancia de la tesis.

1.7. Trabajo derivado

Durante la elaboración de la tesis y sus experimentos, se publicaron dos artículos. El primero, una revisión sistemática de la literatura que muestra el estado del arte de las redes neuronales y reúne otras técnicas utilizadas para la clasificación de requisitos y que se describe en el Capitulo 3. El segundo artículo es resultado del Capitulo 4, describiendo el proceso por el cual se construye la red neuronal diseñada por el humano para clasificar los requisitos.

Capítulo 2

Marco teórico

2.1. Introducción

Para comprender los términos que se usan en el desarrollo de la tesis, este capitulo describe los elementos usados. En primer lugar, se describen los requisitos de software, los cuales son el elemento fundamental de la tesis que lo liga a la ingeniería de software. Por otro lado, se presentan los elementos de inteligencia artificial usados para desarrollar los clasificadores de requisitos.

2.2. Requisitos e ingeniería de requisitos

El desarrollo de software es un proceso que consta de diversas etapas. Entre las primeras se encuentra la etapa de análisis de requisitos, que llega después de la elicitación de los requisitos. Los requisitos son esenciales, pues dirigen las decisiones de diseño en los sistemas (Lawrence, 1997), pues son estos los que especifican lo que debe estar implementado en un sistema (Sommerville y Sawyer, 1997). Existe un campo relacionado con la ingeniería de software que se dedica a trabajar con los requisitos, la ingeniería de requisitos. La ingeniería de requisitos establece métodos estructurados para desarrollar y administrar los requisitos (IIBA, 2015).

2.2.1. Niveles de requisitos

Wiegers y Beatty (2013), proponen tres niveles distintos de requisitos: los requisitos de negocio, los requisitos de usuario y los requisitos funcionales. Existen diferentes tipos de taxonomías de requisitos, como la de Sommervile (2005), que organiza la información en tres niveles: requisitos de usuario, requisitos de sistema y especificaciones de diseño de software. Otra diferente taxonomía sugiere que los niveles de requisitos pueden ser: requisitos funcionales, requisitos no funcionales, y requisitos de dominio (Laplante, 2009). Esta última taxonomía se enfoca en el tipo de requisito.

Aunque exista una variedad de taxonomías de requisitos, el desarrollo de esta tesis toma la taxonomía descrita por Wiegers y Beatty (2013), por la cobertura que realiza al describir los requisitos y porque es la taxonomía principal estudiada por la academia de ingeniería de software de la Facultad de Estadística e Informática de la Universidad Veracruzana.

2.2.2. Requisitos de negocio

Aunque exista una variedad de taxonomías de requisitos, el desarrollo de la tesis desarrolla la taxonomía descrita por Wiegers y Beatty (2013), por la cobertura que realiza al describir los requisitos y porque es la taxonomía principal estudiada por la academia de ingeniería de software de la faculta de estadística e informática de la Universidad Veracruzana.

2.2.3. Requisitos de usuario

Los requisitos de usuario describen las tareas o metas que el usuario puede lograr al usar el sistema y aportan valor al usuario. Las formas en las que se pueden representar y explorar los requisitos de usuario son mediante casos de uso de historias de usuario (Wiegers y Beatty, 2013).

2.2.3.1. Casos de uso

Los casos de uso describen una secuencia de acciones entre un actor externo, ya sea humano o sistema, que da como resultado que el actor externo cumpla su cometido con nuestro sistema (Wiegers y Beatty, 2013). Poseen la característica de ser nombrados empezando por un verbo seguido de un objeto, por ejemplo: “Recuperar usuario”, “Mostrar catalogo” o “Agregar inventario”. Dentro de su estructura, pueden trazarse flujos normales, que indican cómo el actor logra su cometido sin ningún problema, flujos alternos, que guían al actor a completar su tarea de una forma diferente al flujo normal o cancelar su proceso, y el flujo de excepción, que describen los pasos a seguir para recuperarse de excepciones que puedan ocurrir mientras el usuario utiliza el sistema.

2.2.4. Requisitos funcionales

Los requisitos funcionales detallan cómo el sistema debe comportarse bajo ciertas condiciones y también describen lo que los desarrolladores deben hacer para que el usuario cumpla sus objetivos con el sistema.

2.2.5. Requisitos no funcionales

2.2.5.1. Requisitos de calidad

Los requisitos de calidad describen las propiedades que el sistema construido debe cubrir (Wiegers y Beatty, 2013). Este tipo de requisitos está relacionado con los atributos de calidad, dentro de los cuales se puede encontrar el grupo de los *-ilities* (debido al sufijo característico en inglés), como disponibilidad, seguridad, usabilidad, entre otros, y que serán descritos posteriormente.

2.2.5.1.1 Atributos de calidad externa

Describen características del software que pueden ser apreciadas cuando está funcionando (Wiegers y Beatty, 2013). En este grupo de atributos de calidad se pueden encontrar disponibilidad, instalabilidad, rendimiento, confiabilidad, interoperabilidad, integridad, robustez, seguridad y usabilidad. Cada uno de los atributos de calidad abarca propiedades diferentes en el sistema. La disponibilidad es medida usando el tiempo que el servicio de un sistema debe estar completamente funcional, durante el cual los usuarios están usando el sistema. Un requisito de disponibilidad especifica en su sentencia el tiempo que el sistema debe cumplir en pleno funcionamiento.

Los requisitos de instalabilidad se refieren a la facilidad que tiene el software de ser instalado correctamente. La facilidad de instalación trae ventajas como el ahorro del tiempo y del esfuerzo de quienes instalan el sistema, además de disminuir el nivel de habilidades requerido para ser instalado (Wiegers y Beatty, 2013). La integridad trata de evitar la corrupción y la perdida de datos, trata de mantener la información en la forma correcta, usualmente, como se haya ingresado a un sistema. La información del software es vulnerable a perderse, situaciones como pérdida accidental de datos, corrupción, daños físicos o ataques pueden causar que la información sea comprometida (Wiegers y Beatty, 2013).

Cuando se dice que un sistema tiene interoperabilidad, se refiere a como un sistema puede intercambiar información o servicios con otros sistemas y que tan fácil puede integrar dispositivos de hardware externos (Wiegers y Beatty, 2013). Expresado de forma resumida, es la propiedad que poseen diferentes sistemas de trabajar en conjunto (Alexander y Beus-Dukic, 2009). Los requisitos de interoperabilidad pueden describir el formato en que los datos son intercambiados. Rendimiento es el atributo de calidad que expresa el grado de responsividad que un sistema debe poseer (Wiegers

y Beatty, 2013). También pueden describir que tan bien el sistema realiza los requisitos funcionales especificados (Young, 2004). La IIBA (2015), relaciona el rendimiento con la eficiencia.

La confiabilidad de un sistema es descrita como el tiempo que el software puede permanecer ejecutándose sin ningún fallo por un tiempo específico (Wiegers y Beatty, 2013). También puede expresar el total de fallos permitidos en un periodo de tiempo (Robertson y Robertson, 2013). Las posibles causas a los problemas de la confiabilidad son errores de código, fallos de hardware, entradas incorrectas o la ausencia de algún recurso en el sistema.

La robustez del software es el grado en el cual el sistema sigue funcionando cuando se presentan ciertas situaciones que podrían hacer que el sistema falle (Wiegers y Beatty, 2013). Asegura que el software es capaz de proveer servicios bajo situaciones que no deberían suceder (Robertson y Robertson, 2013). Los problemas que afronta la robustez son las entradas incorrectas, los fallos de conexiones derivados de la ausencia de algún componente de software o hardware y ataques externos (Wiegers y Beatty, 2013).

La seguridad se refiere a la propiedad del software de bloquear el acceso no autorizado a ciertas funciones del sistema o datos protegiéndolos de ataques y malwares (Wiegers y Beatty, 2013). También puede referirse al grado de protección que tiene el software de ser modificado o destruido por intentos malintencionados (Robertson y Robertson, 2013).

Cómo último atributo de calidad externa descrito, está la usabilidad, la cual está relacionada con la facilidad de uso (Wiegers y Beatty, 2013). Se puede decir que el software es usable cuando sus usuarios pueden aprender fácilmente su uso para lograr sus objetivos (Robertson y Robertson, 2013).

2.2.5.1.2 Atributos de calidad interna

Los atributos de calidad interna no pueden verse a al momento de ejecutar el software, sino que son propiedades que los desarrolladores perciben cuando evalúan los artefactos del software (Wiegers y Beatty, 2013). Este grupo de atributos de calidad abarca la eficiencia, la modificabilidad, portabilidad, reusabilidad, escalabilidad y verificabilidad.

Como se mencionó anteriormente, la eficiencia está relacionada con el rendimiento. La eficiencia mide que tan bien se aprovechan los recursos disponibles durante el uso del software (Wiegers y Beatty, 2013). Recordemos que los recursos pueden ser de software o hardware, como el ancho

de banda, el espacio en disco, el tiempo del procesador o la memoria.

La modificabilidad se refiere a la facilidad que tienen los artefactos del software, como el diseño o el código, en ser entendidos, modificados o extendidos (Wiegers y Beatty, 2013). Este atributo está relacionado con la verificabilidad. Es importante que los sistemas cuenten con un grado de modificabilidad para afrontar los cambios que puedan suceder en el desarrollo de un software, así incluso con cambios en el equipo, la curva de aprendizaje se aplanará.

Cuando se habla de portabilidad nos referimos a la facilidad que tiene el software de ejecutarse en ambientes distintos, como los navegadores o entre sistemas operativos. Otros pueden definir a la portabilidad con la internacionalización o la localización de un producto.

El aumento de los usuarios en un sistema es un reto que el software debe atender. Cuando este aumento sucede, es necesario plantear una estrategia que nos permita hacer que nuestro sistema pueda atender a todos sus usuarios sin sacrificar rendimiento. La escalabilidad es precisamente esto, el aumento de las capacidades del sistema ya sea de software o hardware para atender crecimiento y alcance que el sistema puede experimentar sin tener que disminuir otros aspectos, como el rendimiento o la robustez (Wiegers y Beatty, 2013).

Por último, se encuentra la verificabilidad, o la facilidad de ser probado, y se refiere al grado en que los componentes integrados en él pueden ser evaluados para demostrar que el sistema funciona de la forma esperada (Wiegers y Beatty, 2013).

2.2.5.2. Reglas de negocio

Los negocios deben conocer y definir las normas que rigen sus procesos. Existen dos tipos de perspectivas de las reglas de negocio, la del negocio y la de el sistema de información (Wiegers y Beatty, 2013). Desde la perspectiva del negocio, las reglas de negocio son una guía de que existe una obligación sobre conductas, acciones, prácticas o procedimientos en una actividad (Wiegers y Beatty, 2013). Desde la perspectiva de los sistemas de información, las reglas de negocio son declaraciones que definen o restringen algún aspecto del negocio y buscan controlar el comportamiento de una empresa (Wiegers y Beatty, 2013).

Las reglas de negocio son esenciales para crear requisitos funcionales y pueden presentarse en forma de políticas, condiciones y restricciones de las actividades que el negocio planea resolver con el sistema a desarrollar. También pueden encontrarse en decisiones en los procesos y pautas

bajo las cuales el negocio se rige (Young, 2004).

2.2.5.3. Restricciones

Las restricciones tienen la característica de limitar el alcance en ciertos aspectos del sistema, como en la implementación o permisos. Su tarea principal es restringir las acciones que el sistema o los usuarios pueden llegar a realizar (Wiegers y Beatty, 2013). A nivel de producto, las restricciones imponen limites en cómo construir los productos (Alexander y Beus-Dukic, 2009), y expresan las limitaciones de los proyectos o el diseño del producto (Robertson y Robertson, 2012).

A diferencia de los requisitos funcionales, las restricciones no añaden mas funcionalidades al sistema. En su lugar, controlan la forma en la que las funcionalidades se presentan en el producto (Dick, Hull, y Jackson, 2017).

Un requisito de restricción en la construcción del software puede verse como el siguiente ejemplo usado por Robertson y Robertson (2012):

- *El sistema debe operar tanto en iPad, iPhone, Android y aplicación de Blackberry.*

2.2.6. Otros tipos de requisitos

2.2.6.1. Requisitos de datos

Los requisitos de datos son útiles para conocer la información con la que el sistema debe trabajar, permitiendo anticiparse al manejo de estos. Este tipo de requisitos describen formato, tipos de datos, valores permitidos o valores por defecto de un elemento de un dato (Wiegers y Beatty, 2013). El siguiente es un ejemplo utilizado por Wiegers y Beatty (2013):

- *Una orden consiste en la identidad del cliente, la información de la compra, y uno o más productos, cada uno de ellos incluye el código del producto, número de unidades, precio unitario, y el precio total.*

2.2.7. Desarrollo de los requisitos

En la presente sección se describen las actividades en las que el desarrollo de requisitos se divide y se remarca la fase en la que la clasificación de requisitos tiene lugar.

Según Wiegers y Beatty (2013), quienes a su vez se apoyan de Abran, Moor, Bourque, y Dupuis (2004), los requisitos se desarrollan en cuatro partes esenciales: elicitación, análisis, especificación y validación (Ver Figura 2.1). Estas partes abarcan actividades para explorar, evaluar, documentar, y validar los requisitos (Wiegers y Beatty, 2013).

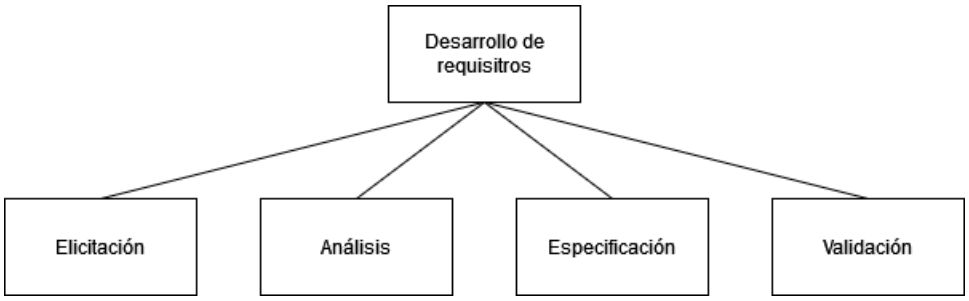


Figura 2.1: Partes del desarrollo de requisitos

2.2.7.1. Elicitación

El principal interés de la elicitación está relacionado con el origen de los requisitos de software y cómo estos pueden recolectarse. Es una actividad que depende en gran medida del humano y por tanto requiere que la comunicación con uno o varios stakeholders sea clara (Bourque y cols., 2014).

Los objetivos perseguidos durante la elicitación son la identificación de usuarios e interesados, entender las tareas y metas relacionadas al negocio, aprender acerca del dominio bajo el cual el producto se desarrolla y trabajar con los representantes de cada clase de usuario para aproximarse a sus expectativas acerca del producto (Wiegers y Beatty, 2013).

Las actividades realizadas durante la elicitación de requisitos son variadas: talleres, análisis de documentos, elaboración de prototipos, entre otros (Wiegers y Beatty, 2013).

2.2.7.2. Análisis

Esta es la etapa en la que la clasificación de requisitos tiene lugar. En esta, se busca formalizar los requisitos. Para ello se detectan y resuelven conflictos entre requisitos, se descubren los límites del producto y su comportamiento con los usuarios y el ambiente en el que opera, y se elaboran los requisitos del producto (Bourque y cols., 2014).

Entre las actividades realizadas durante el análisis de los requisitos se encuentran (Wiegers y Beatty, 2013):

- Analizar la información recibida de los usuarios e identificar requisitos funcionales, reglas de negocio, entre otros.
- Descomponer los requisitos de alto nivel a un nivel más apropiado.
- Derivar requisitos funcionales de otro tipos de requisitos e información.
- Clasificar los requisitos no funcionales o requisitos de calidad relacionados.
- Identificar brechas o requisitos innecesarios.
- Analizar la prioridad de los requisitos.

Otra de las actividades relacionadas con el análisis es la clasificación de los requisitos, que puede incluir clasificarlos por diferentes aspectos, tales como los que Bourque y cols. (2014), describen:

- Si el requisito es funcional o no funcional y el tipo de requisito no funcional o de calidad.
- El alcance de los requisitos, es decir, si el requisito debe cumplirse en varios componentes.
- La volatilidad, es decir, si el requisito es susceptible a cambiar durante el desarrollo del proyecto.

2.2.7.3. Especificación

Wiegers y Beatty (2013) describen la especificación de requisitos como la representación del conocimiento de los requisitos recolectados en un tipo de orden. Mientras que Bourque y cols. (2014) la definen como el proceso por el cual se elabora un documento que puede ser revisado, evaluado y aprobado por otros. El documento comúnmente realizado para esta etapa se llama Especificación de Requisitos de Software (*Software Requirements Specification*), el cual es evaluado antes de proceder al diseño del software y es utilizado para validar los requisitos allí descritos (Bourque y cols., 2014).

2.2.7.4. Validación

La validación de los requisitos confirma que el conjunto de requisitos especificados permitirá al equipo de desarrollo construir el producto de software deseado (Wiegers y Beatty, 2013). Algunas de los métodos descritos para validar los requisitos son (Bourque y cols., 2014): la revisión de los requisitos, el prototipado, la validación de modelos, y las pruebas de aceptación.

2.2.7.4.1 Revisión de requisitos

Esta es la técnica mayormente utilizada para validar requisitos en la que se inspecciona el documento de especificación de requisitos de software. La inspección es realizada por un grupo de revisores asignados para encontrar errores, supuestos incorrectos, falta de claridad y faltas a las buenas prácticas establecidas (Bourque y cols., 2014).

2.2.7.4.2 Prototipado

La elaboración de los prototipos sirve a los ingenieros de requisitos para validar la interpretación que se tiene sobre los requisitos redactados a la vez que para elicitar nuevos requisitos (Bourque y cols., 2014).

2.2.7.4.3 Validación de modelos

La validación de modelos busca evaluar la calidad de los modelos utilizados durante el análisis de los requisitos. Dentro de los modelos se busca que los objetos utilizados, el dominio y el flujo de datos sean consistentes y se encuentren dentro de un entendimiento mutuo (Bourque y cols., 2014).

2.2.7.4.4 Pruebas de aceptación

Cuando el sistema está por terminarse, el software es sometido a pruebas de aceptación. En ellas, el usuario objetivo prueba el sistema construido para validar que los requisitos fueron cumplidos satisfactoriamente (Bourque y cols., 2014).

2.3. Inteligencia artificial

La definición de inteligencia artificial tiene varios enfoques. Desde la informática, McCarthy (2007), define a la inteligencia artificial como la ciencia y la ingeniería de hacer programas de computo inteligentes. Esto nos lleva a un campo más delimitado, el computo inteligente, el cual es el estudio de mecanismos adaptativos para generar o facilitar el comportamiento inteligente en ambientes complejos, inciertos y cambiantes. Entre los diferentes paradigmas del computo inteligente se encuentran los algoritmos evolutivos y las redes neuronales.

2.3.1. Algoritmos evolutivos

Los algoritmos evolutivos se basan en la metáfora biológica de la evolución de las especies. Parten de la idea que, dentro de una población de individuos, la presión del ambiente donde habitan causará una selección natural, lo que llevará a una mejora de las cualidades de la población. Por esta razón, los algoritmos evolutivos son aplicados principalmente en problemas de optimización (Eiben y Smith, 2015).

Dicho de otra forma, los algoritmos evolutivos son aquellos que tienen la capacidad de evolucionar (mejorar) soluciones partiendo de una población inicial aleatoria (Yu y Gen, 2012). Este tipo de algoritmos poseen tres características principales: (1) inician con una población; (2) se orientan de acuerdo a aptitudes de las soluciones, con lo que se representa el rendimiento cada solución (individuo) dentro de la población, y (3) están dirigidos por variaciones en la cantidad de cambios que se realizan al modificar la genética (elementos) de cada individuo, necesarios para explorar espacios de búsqueda (Yu y Gen, 2012). Por otro lado, los algoritmos evolutivos son estocásticos, es decir, no obtendrán necesariamente la misma solución al ser ejecutado con los mismos parámetros de entrada pues su funcionamiento incluye valores aleatorios.

Para resolver problemas de búsqueda, el algoritmo evolutivo crea una población aleatoria de individuos (soluciones potenciales al problema), los cuales serán evaluados con una función de aptitud que representa el problema a resolver y que mide las cualidades de cada individuo. Una vez obtenidos los valores de aptitud, se eligen a los mejores individuos para reproducirse aplicando recombinación o mutación, es decir, crearán nuevos individuos. Finalmente, al crecer el tamaño

de la población con la presencia de los descendientes, se escogen a los que permanecerán para la siguiente iteración, llamada generación, y así mantener un tamaño fijo de la población. Este proceso se repite hasta que se cumpla una condición de paro (Eiben y Smith, 2015).

2.3.1.1. Componentes de los algoritmos evolutivos

Los algoritmos evolutivos poseen componentes en común con los cuales podemos identificarlos. Necesitan y parten de la representación de las soluciones para después generar la población aleatoria

El primer paso de un algoritmo evolutivo es representar al individuo que se encuentra dentro de nuestra población. Para representarlo debemos diferenciar entre fenotipo y el genotipo. El fenotipo es la manifestación del individuo en el contexto del problema, por ejemplo, la imagen de un unicornio. El genotipo es la representación usada por el algoritmo evolutivo sobre la cual se realiza la evolución, por ejemplo, la representación en bits de la imagen (Eiben y Smith, 2015).

2.3.1.1.1 Representación

Para definir un algoritmo evolutivo, es necesario saber cómo representar los objetos del mundo real. Para ello, debemos conocer los conceptos de fenotipo y genotipo. El primero, es la representación de la solución en su contexto, mientras que el segundo es la codificación del primero y con el cual el algoritmo funciona (Eiben y Smith, 2015). Con el fin entender la diferencia entre ambos, pensemos en el número 3 como el fenotipo que, para su representación, sabemos que está en un contexto de numeración decimal pero si codificamos dicho número a binario usando cuatro bits nos queda la cadena 0011 en su representación, su genotipo. Esto sirve para determinar el espacio en el que la evolución tendrá lugar (Vikhar, 2017).

2.3.1.1.2 Función de aptitud

Como sabemos, los algoritmos evolutivos trabajan con un conjunto de soluciones posibles, pero de entre todas ellas debemos saber cuáles son las mejores para cumplir con la emulación de la supervivencia del más apto. La función de aptitud es precisamente para eso, sirve para asignar una medida de calidad a cada solución en la población (Eiben y Smith, 2015). Gracias a esta medida, el

algoritmo puede identificar a las mejores soluciones y escogerlas durante el proceso de selección (Vikhar, 2017).

2.3.1.1.3 Población

La población es un componente muy importante del algoritmo evolutivo, puesto irá evolucionando hasta encontrar la solución que buscamos y que tenga un valor competitivo de la función de aptitud. Se define a la población como un conjunto de posibles soluciones y como la unidad de la evolución, debido a que es la población la que cambia constantemente gracias a la competencia y selección de sus individuos (Eiben y Smith, 2015).

2.3.1.1.4 Mecanismo de selección de padres

Para que la población evolucione, el algoritmo evolutivo distingue a las mejores individuos (soluciones) del resto para que se vuelvan padres de la siguiente generación. Gracias a este mecanismo, junto con la selección de sobrevivientes, se logra la evolución de la población y la supervivencia del mas apto (Eiben y Smith, 2015).

2.3.1.1.5 Operadores de variación

Una vez seleccionados los padres se crean los hijos mediante la recombinación o cruza. Después, los hijos resultantes pasan por un proceso en el que se decide si mutarán o no.

En la recombinación o cruza, la información de los padres se combina, usualmente de manera aleatoria, para crear nuevos individuos. Debido a que la selección de información a combinar es aleatoria, el operador de cruza es considerado como estocástico. Por otro lado, la mutación consiste en un cambio aleatorio a una solución (Vikhar, 2017) (Eiben y Smith, 2015). Esta operación también es estocástica (Eiben y Smith, 2015).

2.3.1.1.6 Mecanismo de reemplazo

El reemplazo o mecanismo de selección de sobrevivientes cumple la función de decidir cuáles de los individuos de la población original y sus descendientes son escogidos para pasar a la siguiente

generación. El criterio de selección puede estar basado en la aptitud de los individuos o en su edad (Eiben y Smith, 2015).

2.3.1.2. Tipos de algoritmos evolutivos

Existen cinco algoritmos evolutivos: programación evolutiva, estrategias evolutivas, programación genética, evolución diferencial y los algoritmos genéticos.

Los algoritmos genéticos son particularmente populares, usualmente usados en aprendizaje máquina, reconocimiento de patrones y problemas de optimización. Su operador principal es la recombinación o cruza y puede representar soluciones de maneras diversas, desde cadenas de bits hasta vectores de números reales (Vikhar, 2017).

La programación genética representa a las soluciones mediante árboles, por lo que puede ser aplicada en optimización simbólica como por ejemplo expresiones matemáticas, operaciones booleanas y funciones recursivas, entre otras (Vikhar, 2017).

La evolución diferencial representa a las soluciones mediante vectores de números reales y su operador principal se conoce como mutación diferencial, que es una resta de vectores. Usa cruza binomial o exponencial y el reemplazo se da únicamente entre padre e hijo.

Las estrategias evolutivas representan a las soluciones mediante vectores de números reales únicamente. Su operador principal es la mutación y la cruza es secundaria e incluso opcional. Sus aplicaciones son comunes en las redes computacionales, bioquímica, óptica y diseño industrial. Finalmente, la programación evolutiva emula la evolución natural a nivel de especies, por lo que no hay operador de cruza y funciona únicamente vía mutación. Se encuentran aplicaciones en juegos y control automático (Vikhar, 2017).

2.3.2. Redes neuronales artificiales

Al igual que los algoritmos evolutivos, las redes neuronales forman parte de los modelos bioinspirados debido a que están inspiradas en el cerebro biológico. Dentro de su estructura, una red neuronal artificial se encuentra compuesta de capas: la capa de entrada, que es la que recibe los datos, las capas ocultas, encargadas de transformar la información, y la capa de salida, que se encarga de hacer el último procesamiento de la red neuronal para dar un resultado. Cada capa está com-

puesta de neuronas, y la neurona es la unidad de procesamiento básica de las redes neuronales, es una abstracción de la neurona biológica, y es la encargada de recibir datos, procesarlos y transmitir su salida a otras neuronas (Gershenson, 2003).

Las aplicaciones de las redes neuronales es diversa. Se pueden usar para resolver problemas de clasificación, procesamiento de imágenes, regresiones, problemas de reconocimiento, entre otros.

2.3.2.1. Componentes de las redes neuronales

Las redes neuronales están compuestas de neuronas y estas neuronas forman capas. Las redes neuronales pueden tener capas completamente conectadas o no, y el número de conexiones depende de la cantidad de capas ocultas entra las capas de entrada y salida. Dependiendo del número de capas y de neuronas por capa se puede identificar si se trata de una red poco profunda o una red profunda.

Para resolver problemas, la red neuronal ejecuta el algoritmo de *feedforward* y *backpropagation* para generar un proceso de aprendizaje. Primero se realiza el *feedforward*, que consiste en presentar un patrón a la red partiendo de la capa de entrada, pasando por las capas ocultas y hacia a la capa de salida, con el propósito de que proporcione una respuesta. Después se lleva a cabo el *backpropagation*, que consiste en calcular el error de la respuesta dada e ir iterando por capa partiendo desde la capa de salida hasta la primera capa oculta y así sucesivamente, actualizando los pesos de las neuronas, que es donde se almacena el conocimiento adquirido.

2.3.2.1.1 La neurona

La neurona actúa como un nodo que recibe las salidas de otras neuronas, las cuales son ponderadas por los pesos de la neurona. Esos valores ya ponderados son usualmente acumulados mediante una sumatoria, es decir, se multiplican las salidas de las otras neuronas (que ahora son las entradas de la neurona en cuestión) por sus respectivos pesos, para después generar una señal de salida mediante una función de activación (Gershenson, 2003). En la Figura 2.2 se ilustra cómo se realiza la activación de una neurona, recibiendo las entradas y activando los resultados de la sumatoria de las entradas por los pesos para dar un resultado que pasa a la siguiente capa o a la salida de una red.

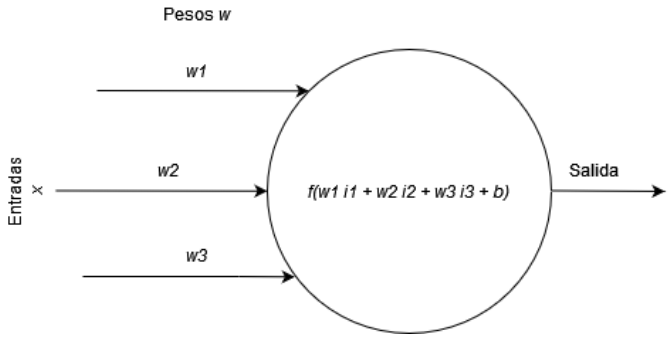


Figura 2.2: Flujo de información a través de una sola neurona

2.3.2.1.2 El algoritmo de *feedforward* y *backpropagation*

Para que la red neuronal pueda aprender, se realizan dos pasos principales: primero ejecutar el algoritmo *feedforward* que presenta un patrón a la red y realiza el recorrido completo por ella para dar un resultado; acto seguido, se ejecuta el algoritmo de *backpropagation*, que calcula el error del resultado de la red y actualiza los pesos de la red desde la última a capa hasta la primera.

Para calcular el error de la salida de la red neuronal se utiliza la Ecuación 2.1, seguida de la Ecuación 2.2 con la cual se actualizarán los pesos de acuerdo al error de la respuesta dada por la red con respecto a la respuesta esperada.

$$E(\overline{x}, \overline{w}, \overline{d}) = \sum_{j} (O_j(\overline{x}, \overline{w}) - d_j)^2 \tag{2.1}$$

$$\Delta W_{ji} = -2\eta(O_j - d_j)O_j(1 - O_j)x_i \tag{2.2}$$

Donde O_j es la activación sumatoria de las entradas $\overline{x}$ por los pesos $\overline{w}$, d_j la salida esperada, y x_i el valor actual de la salida de la neurona.

2.3.3. Selección de características

La selección de características forma parte del preprocesamiento de los datos que serán presentados a la red neuronal artificial u otro método de aprendizaje y consiste en identificar a los atributos más importantes y descartar los menos importantes de un conjunto de datos. La selección de características aporta los beneficios de mejorar el rendimiento de los algoritmos de aprendizaje

máquina y reducir la cantidad de datos (Sánchez-Maroño, Alonso-Betanzos, y Tombilla-Sanromán, 2007).

Los algoritmos de selección de características se dividen en tres grandes grupos: (1) los métodos de envoltura, que utilizan un algoritmo de aprendizaje para guiar el proceso, (2), los métodos de filtrado, que utilizan técnicas estadísticas para reducir la cantidad de atributos con un costo computacional menor al de los métodos de envoltura, pues no utilizan un algoritmo de aprendizaje, y (3) los métodos embebidos, que combinan el proceso de aprendizaje con el de selección de características (ejemplo, árboles de decisión).

En la presente tesis, dado el alto costo computacional de la neuroevolución, sólo se utilizan métodos de filtrado y son los siguientes:

- *Relief*: estima la calidad de los atributos de acuerdo a qué tan bien se puede distinguir entre instancias de datos.
- *Correlation Feature Selection* (CFS): ordena de mejor a peor las características de acuerdo a una medida de correlación.
- *Fast Correlated Based-filter* (FCBF): se basa en la incertidumbre simétrica, la cual es definida como el radio entre la información y la entropía de dos características.

Capítulo 3

Marco Referencial

3.1. Revisión de la literatura

Durante la segunda mitad del 2020, se realizó una revisión sistemática de la literatura acerca de las técnicas de aprendizaje máquina empleadas para la clasificación de los requisitos. Los objetivos principales de la revisión fueron:

- Conocer los tipos de redes neuronales empleadas para la clasificación de requisitos
- Conocer las técnicas de aprendizaje máquina utilizadas para la clasificación de requisitos.
- Conocer las clases utilizadas para la clasificación de requisitos.
- Conocer los resultados de los experimentos realizados para la clasificación de requisitos.

La revisión de la literatura sirve a esta tesis para apoyar la elección de la neuroevolución como método para desarrollar la clasificación de los requisitos. Como se mencionó en los objetivos anteriores, es de nuestro interés saber la posición actual de las redes neuronales en el estado del arte de la clasificación de requisitos, pues ella nos dará un indicador sobre la novedad del desarrollo de la clasificación de los requisitos de software mediante el enfoque neuroevolutivo.

Como resultado de la revisión, 14 estudios fueron seleccionados y analizados. Entre las técnicas más populares se encontraron: en primer lugar Naive Bayes con 7 aplicaciones, seguido de las técnicas de *Support Vector Machine* (SVM), y las redes neuronales artificiales con 6 aplicaciones.

De las redes neuronales analizadas, se encontraron tres variantes distintas, siendo la red neuronal convolucional la más aplicada con un total de 4 clasificadores que utilizaron esta técnica, mientras que solo se reporta una aplicación de una red neuronal no profunda y una red neuronal *Long Short-Term Memory* (LSTM), respectivamente.

Entre los estudios seleccionados, se utilizaron de dos a trece clases para la clasificación de los requisitos. Una gran parte de los estudios utilizaron la base de datos PROMISE como principal fuente de requisitos, mientras que otros utilizaron la bases de datos de DOORS, SecReq y *Quality Attributes*, estos últimos dos realizados por el Departamento de Seguridad Nacional de Estados Unidos. También se reportaron otras fuentes de bases de datos de proyectos privados.

Como parte final de la revisión, se hizo la recopilación de los resultados de las aplicaciones de las técnicas, dando un total de 34 aplicaciones y se analizaron las principales métricas utilizadas

para medir el desempeño de los clasificadores, los cuales fueron: la precisión (Ecuación 3.1), el *recall* (Ecuación 3.2), y el *f-score* (Ecuación 3.3).

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

$$F1 = 2\frac{precision(recall)}{precision + recall} \tag{3.3}$$

donde TP son los verdaderos positivos, FP los falsos negativos y FN falsos negativos.

Esta revisión de la literatura, además de ayudarnos a conocer el estado del arte de la clasificación de requisitos de software, justifica que el enfoque neuroevolutivo aplicado a la clasificación de requisitos es una propuesta novedosa cuando hablamos del papel de la neuroevolución en este dominio particular de aplicación.

3.2. Trabajo previo

La presente tesis toma parte del trabajo previo de Pérez-Verdejo, Sánchez-García, y Ocharán-Hernández (2020), específicamente el conjunto de datos producto del preprocesamiento de los requisitos. Toma los requisitos de software y pasa por un proceso que transforma de lenguaje natural a uno que el método de aprendizaje pueda procesar. A lo largo de esta sección se explicará este proceso.

3.2.1. Conjunto de datos

El conjunto de datos producto de la tesis de Pérez-Verdejo y cols. (2021), abarca siete tipos de requisitos de calidad: disponibilidad, tolerancia a fallos, mantenibilidad, rendimiento, escalabilidad, seguridad y usabilidad. Este conjunto de datos proviene principalmente de la base de datos PROMISE (Sayyad Shirabad y Menzies, 2005), la cual identificamos previamente en la revisión de la literatura. La distribución del conjunto de datos puede observarse en la Figura 3.1

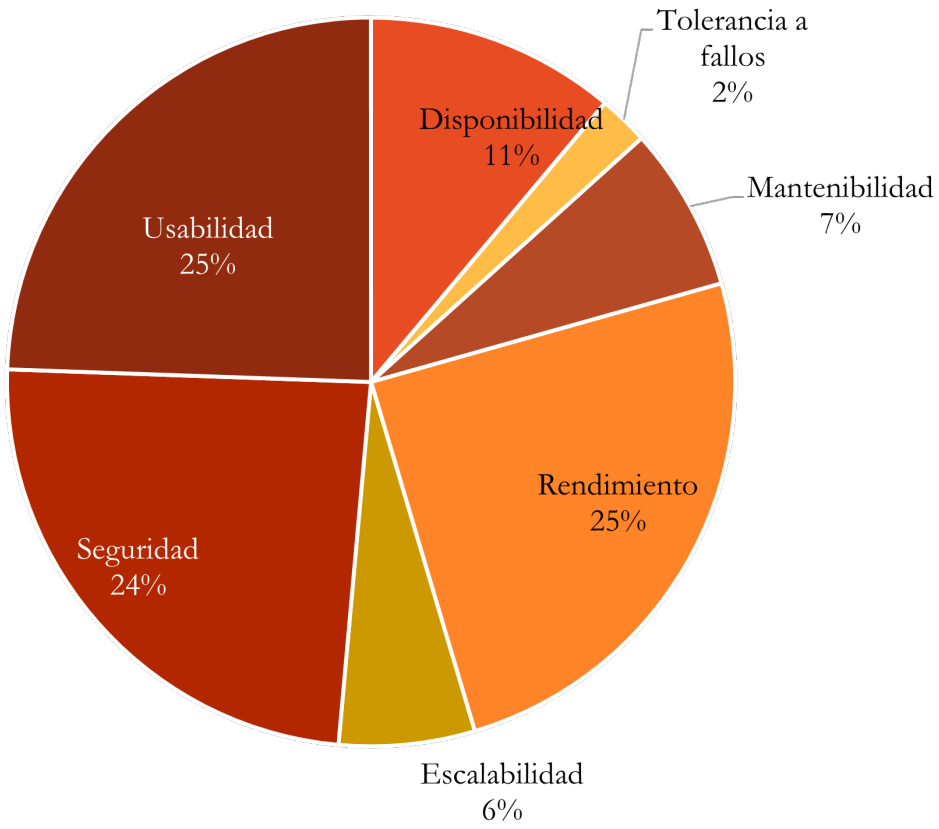


Figura 3.1: Distribución del conjunto de datos

3.2.2. Preprocesamiento

El preprocesamiento de los requisitos de software realizado por Pérez-Verdejo y cols. (2021) ha creado un conjunto de datos que permite usar los requisitos para una clasificación. Se componen de tres partes principales: el etiquetado, la sustitución y la remoción de vocabulario específico.

En el etiquetado se generan etiquetas a partir de palabras que expresan tiempo y a partir de ellas se generan reglas que determinan la sustitución de dichas expresiones de tiempo (Pérez-Verdejo y cols., 2021). Las reglas son las siguientes:

1.

2. Las expresiones etiquetadas con *Duration* incluyen las expresiones: *within*, *no longer than*, *in under*, *no more than*, entre otros.

3. Las expresiones que hacen referencia a las 24 horas del dia se sustituyen por *alltimes*.

4. Las expresiones *Duration* que incluyen un lapso de tiempo se sustituyen por *fast*.

5. Las expresiones que incluyen *timely* o *quick* se sustituyen por fast.

6. Las frases que indican un porcentaje mayor a 80 % se sustituyen por *alltimes*.

7. Los porcentajes iguales a 100 % se sustituyen por *all*, y los que estén entre 50 % y 100 % por *most*.

Después se pasa a un proceso llamado reconocimiento de nombres de entidades, el cual identifica nombres de entidades y generaliza los términos. Luego, se usa un enmascarado de palabras que representan nombres de usuarios o sistemas para generalizarlos como üsuarioƶ ”sistema”(Pérez-Verdejo y cols., 2021).

Finalmente, se aplica la representación de texto en la cual los requisitos pasan a ser vectorizados para que puedan tener un formato que la computadora pueda entender. Para lograrlo, se realizan los siguientes pasos (Pérez-Verdejo y cols., 2021):

1. Remoción de caracteres especiales, específicamente los saltos de línea y tabulaciones.

2. Conversión a minúsculas.

3. Remoción de signos de puntuación.

4. Lematización, en la que se reduce un término a su raíz más básica.

5. Remoción de palabras vacías.

6. Vectorización por *Term Frequency - Invert Document Frecuency*.

3.2.3. NeuroEvolution of Agumenting Topologies (NEAT)

En español, la neuroevolución es un algoritmo que pretende emplear aprendizaje máquina para apoyar a otra técnica de aprendizaje máquina, pues emplea un algoritmo genético para evolucionar topologías y propiedades de las redes neuronales Stanley y Miikkulainen (2002). Para esta tesis, el uso de la biblioteca de *neat-python* (McIntyre, Kallada, Miguel, y da Silva, 2015), nos ayuda a facilitar la aplicación de la neuroevolución para clasificar los requisitos.

3.2.4. Codificación genética

Cada genoma incluye una lista de conexiones de genes que representan pares de nodos conectados e indican el nodo de entrada y el nodo de salida, el peso de la conexión, si esta conexión está activa o no, y un número de innovación, el cual permite identificar cada gen cuando las redes neuronales son cruzadas y es incrementado cuando un nuevo gen aparece.

3.2.5. Mutación

NEAT puede mutar los pesos de las conexiones de las redes neuronales y la topología. La topología puede mutar de dos formas. La primera consiste en añadir conexiones entre dos nodos, creando un nuevo gen. La segunda añadiendo nuevos nodos que pueden ser insertados en una conexión existente, creando así dos nuevos genes y deshabilitando la antigua conexión, dando como resultado topologías más extensas. En la Figura 3.2, tomada del estudio de Stanley y Miikkulainen (2002), ilustra la forma de en que la topología muta cuando una conexión o un nodo es añadido a una red.

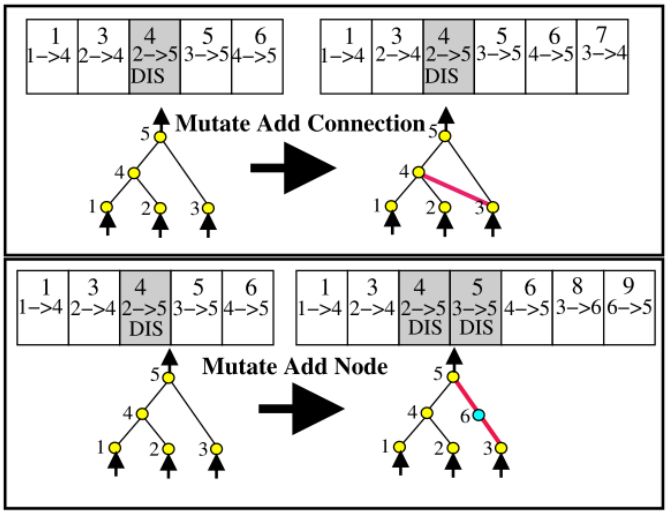


Figura 3.2: Mutación de topología en NEAT usada por Stanley y Miikkulainen (2002)

3.2.6. Seguimiento de genes a través de generaciones

NEAT identifica los genes que son parecidos entre individuos (redes neuronales) y a cuál ancestro pertenecen los individuos a través de generaciones gracias a el número de innovación. Este

número dado a los nuevos genes de una especie, indica el orden cronológico en el que se añaden nuevos genes (Stanley y Miikkulainen, 2002).

3.2.7. Cruza entre redes

Gracias a los números de innovación de los genes, NEAT puede realizar las cruzas de redes identificando cuales genes son distintos entre sí para luego cruzarlos, como ilustra la Figura 3.3, donde se aprecia como dos redes de distintas especies son cruzadas, identificando las diferencia entre genes por sus número de innovación.

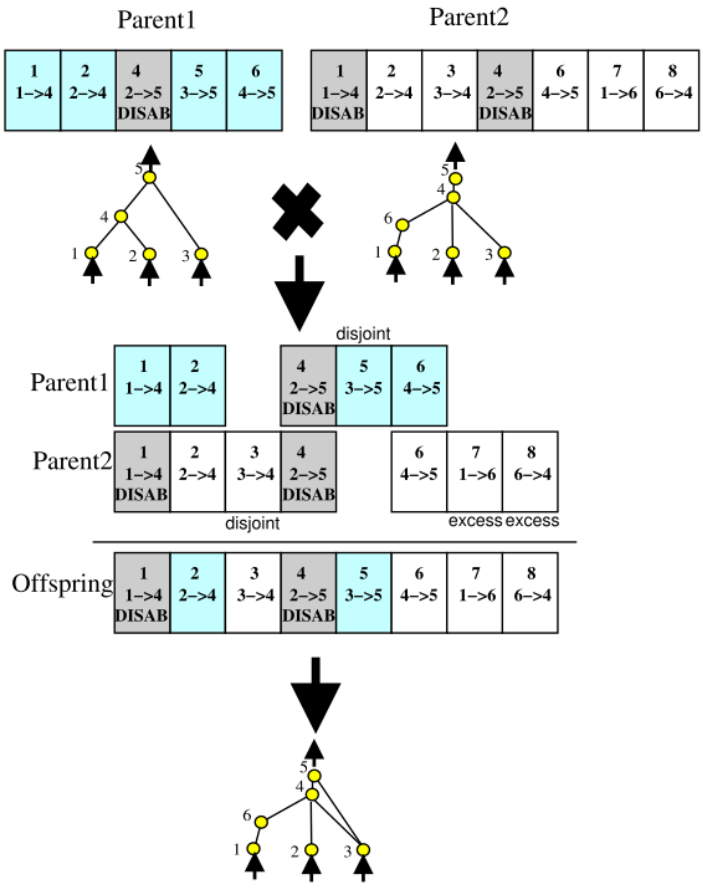


Figura 3.3: Cruza entre redes de diversas especies en NEAT usada por Stanley y Miikkulainen (2002)

3.2.8. Especies

Para proteger los números de innovación y aumentar la diversidad de individuos, NEAT separa la población en especies. Esto se define calculando la distancia que existe entre individuos con la Ecuación 3.4:

$$d = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3.\overline{W} \tag{3.4}$$

donde E representa el exceso, es decir, los genes que sobran con respecto al número máximo de número de innovación del otro gen, D los genes dispares, o sea, genes presentes en un individuo que el otro no posee dentro del menor valor de número de innovación, N el número de genes en el genoma más grande, y los coeficientes c_1, c_2 y c_3 indican la importancia de los tres factores.

Capítulo 4

Clasificadores de requisitos

4.1. Introducción

Los experimentos realizados para la clasificación de requisitos son explicados en este capítulo. En primer lugar, se muestra la obtención del conjunto de datos empleados. En segundo, se muestra el proceso de la realización de una red neuronal manualmente, con sus respectivos experimentos para cada parámetro. Finalmente, se presenta el proceso de elaboración del algoritmo con enfoque neuroevolutivo. El objetivo de realizar una red neuronal antes que un algoritmo neuroevolutivo es tener un referente con el cual comparar el proceso de neuroevolución. Así, se puede demostrar si existe ventaja al usar el enfoque neuroevolutivo en vez de una construcción manual de redes neuronales.

4.2. Conjunto de datos

El conjunto de datos empleado juega un papel importante para la realización de un clasificador. Gracias a este, se entrenan algoritmos para que puedan aprender a clasificar y, posteriormente, puedan ser probados.

Nuestro conjunto de datos es resultado del trabajo previo de Pérez-Verdejo y cols. (2020), el cual consta de 630 requisitos preprocesados y presentados con un formato de vectores.

4.3. Validación

Para validar los experimentos se emplea un método desarrollado por Refaeilzadeh, Tang, y Liu (2009), nombrado validación cruzada. En él, el conjunto de datos se divide en K partes o dobleces. De ellos, un conjunto se asigna para entrenar al modelo y otro segmento, usualmente más pequeño, se usa para probar el modelo entrenado. Al igual que el conjunto es dividido en K partes, también éste se repite K veces, logrando que cada parte en la que se divide la base de datos pruebe el modelo entrenado. El comportamiento de este método se puede observar en la Tabla 4.1

Tabla 4.1: Ejemplo de validación cruzada, donde 1 indica que el subconjunto es utilizado como prueba y 0 que se utiliza como entrenamiento

Iteración	**Doblez 1**	**Doblez 2**	**Doblez 3**	…	**Doblez N**
1	0	0	0	…	1
2	0	0	1	…	0
3	0	1	0	…	0
⋮	⋮	⋮	⋮	⋮	⋮
N	1	0	0	…	0

4.4. Experimentos con red neuronal diseñada por el humano

Cinco experimentos se consideran para esta red diseñada por el humano. El primero consiste en construir una red neuronal como base con la que más tarde se modificarán los parámetros en busca de mejores resultados. El segundo experimento modifica la cantidad de neuronas en la única capa oculta. El tercer experimento pone a prueba el rendimiento de diferentes funciones de activación. El cuarto experimento expande la topología de la red neuronal añadiendo una capa oculta adicional y probando la cantidad de neuronas en dicha capa. Finalmente, el quinto experimento analiza el *learning rate* de la red neuronal.

4.4.1. Primer experimento: Red neuronal base

Como primer experimento se construyó una red neuronal poco profunda y completamente conectada. Posteriormente, se modificaron sus parámetros para buscar una mejora en los resultados.

Para comenzar con este primer experimento, se usa una red neuronal con los parámetros mostrados en la Tabla 4.2. Esta red neuronal tiene una precisión del 68 %, lo cual la posiciona ligeramente por arriba del resultado reportado de Pérez-Verdejo y cols. (2020), el cual es de 67.9 %.

Tabla 4.2: Parámetros de la red neuronal inicial

Parámetro	**Valor**
Neuronas de entrada	
147 Radio de aprendizaje	0.001
Dobleces	10
Función de activación de capa oculta	Tangente hiperbólica
Función de activación de capa de salida	Tangente hiperbólica

4.4.2. Segundo experimento: Número de neuronas en capa escondida

El segundo experimento consiste en modificar el número de neuronas en la capa escondida. Este segundo experimento se divide en 3 partes, cada una es diferente de acuerdo con el rango e intervalo que se utilizó para experimentar. Se eligen los espacios de [97,197], [5, 55] y [250, 950] para observar si existe un cambio significativo en el desempeño del clasificador. Además, dichos espacios se consideran adecuados para no alargar el tiempo de entrenamiento de la red neuronal.

4.4.2.1. Primera parte

Esta parte varió, en intervalos de 10, el número de neuronas de la capa oculta de 97 a 197 neuronas, a excepción de 147 neuronas, puesto que es un parámetro que se definió en el primer experimento. Se toma la decisión iniciar con este rango e intervalo para saber si existe un cambio notable en los resultados.

Los resultados mostraron poca variabilidad. Sin embargo, el mejor resultado mostrado fue con 117 neuronas en la capa oculta. Los resultados de esta primera parte se muestran en la Tabla 4.3.

Tabla 4.3: Resultados de la primera parte del segundo experimento, variando el número de neuronas de la capa oculta. El mejor resultado se resalta en **negritas**

Número de neuronas	Precisión
197	0.696
187	0.693
177	0.695
167	0.696
157	0.696
137	0.688
127	0.692
117	**0.698**
107	0.695
97	0.695

4.4.2.2. Segunda parte

Para esta parte, se utiliza un intervalo con valores menores al de la primera, de 5 a 55 neuronas, pero conservando el intervalo de 10 entre los valores elegidos con la intención de explorar los resultados con una menor cantidad de neuronas.

Al igual que en la primera parte, los resultados se comportan de forma similar, a excepción que se reporta un resultado que supera ligeramente el 69.7 % de precisión reportado por Pérez-Verdejo y cols. (2020), con un parámetro de 25 neuronas, superando también al resultado mostrado usando 117 neuronas en la primera parte. Los resultados se resumen en la Tabla 4.4.

Tabla 4.4: Resultados de la segunda parte del segundo experimento, variando el número de neuronas de la capa oculta en un rango menor. El mejor resultado se resalta en **negritas**

Número de neuronas	Precisión
55	0.693
45	0.693
35	0.696
25	**0.701**
15	0.685
5	0.688

4.4.2.3. Tercera parte

En la última parte, se exploran los resultados de un conjunto con un mayor número de neuronas usando un intervalo de 250 a 950 neuronas, con intervalos de 100 neuronas.

El comportamiento presentado no varía demasiado respecto a los anteriores experimentos, pues se mantienen resultados cercanos entre sí, a excepción de dos donde los resultados superaron el 69.7 % de Pérez-Verdejo y cols. (2020). Sin embargo, no superaron el 70.1 % reportado en la segunda parte del experimento. Los resultados se pueden apreciar en la Tabla 4.5.

4.4.3. Tercer experimento: Funciones de activación

Las funciones de activación influyen en el resultado de las redes neuronales, puesto que estas permiten explorar comportamientos diferentes y que no son necesariamente lineales. En este ex-

Tabla 4.5: Resultados de la tercera parte del segundo experimento, variando el número de neuronas de la capa oculta en un rango mayor. El mejor resultado se resalta en **negritas**

Número de neuronas	Precisión
950	0.696
850	0.696
750	**0.7**
650	0.69
550	0.695
450	**0.698**
350	0.69
250	0.695

perimento se usaron tres de las funciones de activación más populares de la literatura: la tangente hiperbólica (Tanh), la unidad lineal rectificada (ReLU), y la sigmoidal.

La red neuronal de base emplea la función de activación de tangente hiperbólica para todas las capas (oculta y de salida). Este tercer experimento se resumirá en tres partes, cada una usando una función de activación diferente en la capa oculta variando la función de la capa de salida.

4.4.3.1. Primera parte: Tanh en capa oculta y variando la función de activación en la capa de salida

Durante la primera parte de este tercer experimento se varía la función de activación en la capa de salida usando las funciones de ReLU y sigmoidal y manteniendo la función Tanh en la capa oculta. Recordando, la red neuronal base utiliza Tanh en ambas capas (oculta y salida) y ya se conoce su resultado, el cual se añade como referencia en la Tabla 4.6.

El uso de la función Tanh con la función sigmoidal arroja un resultado debajo de lo esperado. Los valores de los resultados que se han estado presentado se aproximaban al 69.7 % de precisión reportado en el trabajo de Pérez-Verdejo y cols. (2020), pero no fue el caso para el uso de Tanh y la sigmoidal que presenta una precisión del 57.3 % (ver Tabla 4.6).

Los resultados de usar la función Tanh y ReLU tampoco muestran una mejora. Esta combinación tiene un rendimiento aún menor, con una precisión del 56.3 % (ver Tabla 4.6).

Tabla 4.6: Resultados de la primera parte del tercer experimento, variando la función de activación en la capa de salida. El mejor resultado se resalta en **negritas**

Función en capa oculta	Función en capa de salida	Precisión
Tanh	Sigmoide	0.573
Tanh	ReLU	0.563
Tanh	**Tanh**	**0.701**

4.4.3.2. Segunda parte: ReLU en capa oculta y variando la función de activación en la capa de salida

Los resultados de este segunda parte usan la función de activación ReLU en la capa oculta en conjunto con las funciones Tanh, sigmoide y ReLU en la capa de salida.

El mejor resultado reportado en esta parte, de acuerdo a lo reportado en la Tabla 4.7, es la configuración usando ReLU junto con la función Tanh, pero sigue sin superar el 70.1 % de precisión de la red neuronal que se viene construyendo. Mientras que el uso de las otras funciones de activación en la capa de salida muestran resultados inferiores al 60 % de precisión.

Tabla 4.7: Resultados de la segunda parte del tercer experimento, usando ReLU en la capa oculta y variando la función de activación en la capa de salida. El mejor resultado se resalta en **negritas**

Función en capa oculta	Función en capa de salida	Precisión
ReLU	**Tanh**	**0.7**
ReLU	Sigmoide	0.573
ReLU	ReLU	0.563

4.4.3.3. Tercera parte: Sigmoide en capa oculta y variando la función de activación en la capa de salida

Estos últimas combinaciones de las funciones de activación (Sigmoide en capa oculta combinada con Tanh, Sigmoide y ReLU en capa de salida) tienen resultados más bajos que los reportados en todo los experimentos con las funciones de activación (ver Tabla 4.8). De hecho, la combinación Sigmoide en capa oculta y ReLU ya no se realizó dado que en la Tabla 4.7 la combinación inversa (ReLU en capa oculta y Sigmoide en capa de salida) obtuvo resultados poco competitivos. En conclusión, el uso de la función Sigmoide en capa oculta o en la capa de salida parece afectar

el desempeño para la clasificación de requisitos propuesta en esta tesis.

Tabla 4.8: Resultados de la tercera parte del tercer experimento, usando Sigmoide en la capa oculta y variando la función de activación en la capa de salida. El mejor resultado se resalta en **negritas**

Función en capa oculta	Función en capa de salida	Precisión
Sigmoide	**Tanh**	**0.64**
Sigmoide	Sigmoide	0.33

4.4.3.4. Conclusión de tercer experimento

La red neuronal no cambia respecto a el estado que tenía cuando se inició el tercer experimento. El uso de la función de activación Tanh es la que mejores resultados obtiene para la red, tanto para la capa oculta como la capa de salida. Se demuestra que el uso de las funciones ReLU y sigmoide no son convenientes en este caso, puesto que los resultados no son competentes y no se aproximan al 69.7 % que reportan Pérez-Verdejo y cols. (2020).

4.4.4. Cuarto experimento: Capa oculta adicional

El objetivo de experimentar con una capa oculta adicional es demostrar si el número de capas y neuronas internas influye de forma significativa en el resultado de la clasificación de requisitos. Durante este cuarto experimento se exploraron de nuevo diferentes intervalos para la cantidad de neuronas empleadas. Por esta razón, el experimento se divide en cuatro partes.

4.4.4.1. Primera parte: 25 a 100 neuronas en la capa oculta adicional

En la primera parte del cuarto experimento, se usa un espacio de [25,100] con intervalos de 25 neuronas para comprobar si un numero bajo de neuronas en una capa oculta adicional puede mejorar el resultado de clasificación.

El mejor resultado reportado en esta primera parte del cuarto experimento logra igualar al máximo reportado hasta ahora de 0.701 (ver Tabla 4.9).

Tabla 4.9: Resultados de la primera parte del cuarto experimento, número bajo de neuronas en la capa oculta adicional. El mejor resultado se resalta en **negritas**

Número de neuronas en capa oculta adicional	Precisión
25	0.692
50	**0.701**
75	0.69
100	0.693

4.4.4.2. Segunda parte: 200 a 500 neuronas en la capa oculta adicional

Para observar el comportamiento de la red neuronal con un mayor número de neuronas en la capa oculta adicional, se propone experimentar en un espacio de [200, 500] con un intervalo de 100 neuronas.

El resultado de esta segunda parte no presenta mejoras en la exactitud de la clasificación de los requisitos (ver Tabla 4.10).

4.4.4.3. Tercera parte: 1000 a 2500 neuronas en la capa oculta adicional

En la tercera parte del experimento se aumenta el intervalo con el que se experimenta (500 neuronas), usando los valores de [1000, 2500] para experimentar con las neuronas en la capa oculta adicional. Los resultados de este experimento no muestran mejora en el resultado de la red neuronal (Tabla 4.11).

4.4.4.4. Cuarta parte: 5000 a 8000 neuronas en la capa oculta adicional

Con el fin de observar si la complejidad de la topología de la red ayuda a mejorar los resultados se elige un intervalo mayor (1000 neuronas) con valores entre [5000, 8000]. Este experimento en

Tabla 4.10: Resultados de la segunda parte del cuarto experimento, número medio de neuronas en la capa oculta adicional. El mejor resultado se resalta en **negritas**

Número de neuronas en capa oculta adicional	Precisión
200	**0.698**
300	0.688
400	0.693
500	0.693

Tabla 4.11: Resultados de la tercera parte del cuarto experimento, número alto de neuronas en la capa oculta adicional. El mejor resultado se resalta en **negritas**

Número de neuronas	Precisión
1000	0.695
1500	**0.698**
2000	0.69
2500	0.685

particular, es esencial para confirmar si la red tiende a clasificar mejor con una topología compleja o con una simple. Los resultados se muestran en la Tabla 4.12.

Tabla 4.12: Resultados de la cuarta parte del cuarto experimento, número muy alto de neuronas en la capa oculta adicional. Los mejores resultados se resaltan en **negritas**

Número de neuronas	Precisión
5000	**0.687**
6000	**0.687**
7000	0.686
8000	**0.687**

4.4.4.5. Conclusiones de cuarto experimento

Gracias este experimento comprobamos que el clasificador de requisitos no necesita una topología compleja. Esto se puede observar en la adición no solo de una capa oculta adicional, sino de un número significativo en la cantidad de neuronas en dicha capa. Lo que se pudo observar con esto es que al aumentar el número de neuronas cómo se hizo en la cuarta parte de este cuarto experimento, la precisión de la red neuronal decrece en comparación con un número más bajo de neuronas. Lo anterior es evidente si comparamos la Tablas 4.9 y 4.12.

Considerando que no hubo mejoras, la estructura de la red neuronal diseñada a mano se queda con una única capa oculta. Gracias a lo anterior, nuestra red neuronal mantiene una topología simple.

4.4.5. Quinto experimento: Rango de aprendizaje

Como último parámetro relevante para el diseño de la red neuronal, se experimenta con la tasa de aprendizaje, la cual controla la convergencia del algoritmo de aprendizaje usado en la red

neuronal propuesta.

Del valor inicial de una tasa de aprendizaje del 0.01, se experimentan valores mayores y menores. El experimento consta de tres partes en las que se prueban tres diferentes intervalos: el primero de [0.02, 0.0025], el segundo de [0.0006, 0.0009] y el tercero de [0.0002,0.0005]. Los resultados de cada parte se muestran en las Tablas 4.13, 4.14 y 4.15, respectivamente.

Tabla 4.13: Resultados de la primera parte del quinto experimento, Tasas de aprendizaje entre [0.02, 0.0025]. El mejor resultado se resalta en **negritas**

Tasa de aprendizaje	**Precisión**
0.02	0.677
0.0175	0.679
0.015	0.692
0.0125	0.695
0.01	**0.698**
0.0075	0.696
0.005	0.695
0.0025	0.693

Tabla 4.14: Resultados de la segunda parte del quinto experimento, Tasas de aprendizaje entre [0.0006, 0.0009]. El mejor resultado se resalta en **negritas**

Tasa de aprendizaje	**Precisión**
0.0006	**0.709**
0.0007	0.701
0.0008	0.698
0.0009	0.7

Tabla 4.15: Resultados de la tercera parte del quinto experimento, Tasas de aprendizaje entre [0.0002,0.0005]. El mejor resultado se resalta en **negritas**

Tasa de aprendizaje	**Precisión**
0.0002	0.668
0.0003	**0.701**
0.0004	0.701
0.0005	0.695

4.4.5.1. Conclusiones de quinto experimento

Después de experimentar con la tasa de aprendizaje resalta el comportamiento en la segunda y tercera parte del experimento (Tablas 4.14 y 4.15), en donde se muestra un aumento en la precisión de las redes neuronales cuando el valor de las tasas de aprendizaje están sobre el intervalo de [0.0002, 0.0009]. Dentro del mencionado intervalo, se encuentra el resultado sobresaliente a los demás con un valor de 0.0006 de tasa de aprendizaje y una precisión de 0.709 (Tabla 4.14), dando como resultado la red neuronal con mas precisión registrada en los experimentos. Por ende, valores bajos de la tasa de aprendizaje llevan a mejores resultados en esta investigación.

4.4.6. Conclusiones sobre la red neuronal diseñada por el humano

De la serie de experimentos realizados, se diseñó una red neuronal ajustando sus parámetros conforme se obtenían mejores resultados, iniciando con la experimentación de la topología y finalizando con la tasa de aprendizaje de la red neuronal.

De los parámetros experimentados, el más influyente ha sido la función de activación, pues su adecuada combinación en las capas oculta y de salida permitieron a la red neuronal alcanzar el 0.709 de precisión micro. El uso de la tangente hiperbólica sobrepasa a las funciones sigmoidal y ReLU con las cuáles se tenían expectativas de mejorar el resultado.

Los resultados detallados del entrenamiento de la red neuronal muestran hasta un 0.793 de exactitud y hasta una precisión de 0.742 de precisión macro en el noveno doblez. El diseño final de la red neuronal puede verse en la Tabla 4.16 y sus resultados en la Tabla 4.17.

Tabla 4.16: Parámetros finales de red neuronal diseñada por el humano

Parámetro	Valor
Número de capas ocultas	1
Número de neuronas en capa oculta	25
Épocas	2500
Tasa de aprendizaje	0.0006
Función de activación de capa oculta	Tanh
Función de activación de capa de salida	Tanh

Tabla 4.17: Resultados finales obtenidos por la red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F1-score
1	0.698	0.544	0.490	0.515
2	0.746	0.705	0.615	**0.657**
3	0.730	0.609	0.536	0.570
4	0.698	0.483	0.467	0.475
5	0.682	0.613	0.502	0.552
6	0.682	0.569	0.584	0.576
7	0.777	0.625	**0.625**	0.625
8	0.650	0.507	0.502	0.504
9	**0.793**	**0.742**	0.578	0.650
10	0.634	0.529	0.503	0.516
Average	0.709	0.593	0.566	0.564

4.5. Experimentos con neuroevolución

4.5.1. Primer experimento: parámetros iniciales

Para establecer los parámetros iniciales del esquema neuroevolutivo, se realizó un experimento en los que se fueron modificando ciertas características de NEAT para observar el comportamiento que mostraban los resultados. Como experimento base, se utilizaron los parámetros de los ejemplos de la documentación de NEAT Python (McIntyre y cols., 2015).

El mejor de los resultados fue el experimento 01a con una precisión de 0.59 y que tiene una topología más sencilla que las de los otros dos resultados con similar precisión, por lo que los parámetros de los siguientes experimentos llevaron la misma configuración 01a. Los resultados se muestran en la Tabla 4.18.

Tabla 4.18: Resultados del primer experimento con neuroevolución, parámetros generales de NEAT. El mejor resultado se resalta en **negritas**

Experimento	Población	Generaciones	Probabilidad de añadir y eliminar Nodos	Probabilidad de añadir o eliminar conexiones	Número de nodos escondidos	Probabilidad de mutación de pesos	Precisión micro	Nodos escondidos
01a	50	500	0.5, 0.5	0.5,0.5	0	0.8	**0.59**	0
01b	50	1000	0.5, 0.5	0.5,0.5	0	0.8	**0.59**	7
01c	50	500	0.5, 0.4	0.5,0.4	0	0.8	**0.59**	3
01d	50	500	0.6, 0.5	0.6,0.5	0	0.8	0.555	8
01e	50	500	0.5, 0.5	0.5,0.5	0	0.5	0.54	3
01f	50	500	0.5, 0.5	0.5,0.5	1	0.8	0.54	4

4.5.2. Segundo experimento: funciones de activación

Para el segundo experimento, se prueba el tipo de función de activación junto con algunos tipos de conexiones. Las funciones de activación elegidas son: tanh, sigmoide, ReLU.

Durante este experimento, se observa que el mejor resultado, al igual que en la red neuronal diseñada por el humano, se obtiene usando la tangente hiperbólica como función de activación principal y con una configuración inicial de todos los nodos conectados entre sí (experimento 02a). En cuanto a la topología, los resultados de las redes neuronales mostraban redes más compactas que la generada por el humano. Los resultados se muestran en la Tabla 4.19.

Tabla 4.19: Resultados del segundo experimento con neuroevolución, funciones de activación y conexiones. El mejor resultado se resalta en **negritas**

Experimento	Generaciones	Conexión incial	Activación	Precisión micro	Nodos escondidos
02a	1500	full_direct	tanh	**0.59**	0
02b	1500	none	tanh	0.41	1
02c	1000	full_direct	tanh, sigm	0.56	8
02d	1000	full_direct	tanh, sigm, ReLU	0.54	1
02e	1000	full_direct	tanh, sigm, ReLU	0.53	3
02f	2000	full_direct	tanh, sig, ReLU	0.56	6
02g	10000	none	tanh, sig, ReLU	0.44	2
02h	1000	none	tanh, sigm, ReLU	0.56	11
02i	1000	full_nodirect	tanh, sigm, ReLU	0.56	0
02j	10000	none	tanh, sigm, ReLU	0.47	20

4.5.3. Tercer experimento: nodos ocultos iniciales

Dado que las redes resultantes en los anteriores experimentos presentaban redes compactas, se decide experimentar con neuronas ocultas. La mayoría de los experimentos realizados se inician con 25 neuronas ocultas como en la red neuronal diseñada.

Al finalizar los experimentos, el mejor resultado sigue manteniendo una topología compacta (1 nodo escondido solamente) con una precisión de 0.55. Sin embargo, dicho resultado no mejora lo obtenido en los experimentos previos. Los resultados se muestran en la Tabla 4.20

4.5.4. Cuarto experimento: funciones de agregación

Por defecto la función de agregación de NEAT (McIntyre y cols., 2015) es la suma. Esta se realiza antes de activar las neuronas cuando suma todos los productos de los pesos con las entradas.

Tabla 4.20: Resultados del tercer experimento con neuroevolución, nodos ocultos. El mejor resultado se resalta en **negritas**

Experimento	Experimento relacionado	Nodos escondidos	Precisión
03a	02a	25	0.52
03b	02d	25	0.5
03c	02f	25	0.44
03d	02h	25	0.11
03e	02a	1	**0.55**
03f	20h	1	0.45

Otras funciones de agregación que NEAT soporta son el producto y el promedio. En este conjunto de experimentos se evolucionan las redes con estas funciones. Además al analizar previamente los pesos resultantes de las anteriores redes, se observó que todas mantienen un número bajo de pesos entre [3,-3], por lo que se va disminuyendo el intervalo de estos.

El mejor de los resultados sigue siendo la función de agregación de suma para las neuronas superando con gran diferencia a las demás redes con un 0.58 de precisión (Tabla 4.21).

Tabla 4.21: Resultados del cuarto experimento con neuroevolución, funciones de agregación. El mejor resultado se resalta en **negritas**

Experimento	Funciones de agregación	Adición y eliminación de nodos	Biases	Pesos	Resultado
04a	sum, product, mean	0.7, 0.3	[2,-2]	[2,-2]	0.407
04b	sum, product, mean	0.70.3	[3,-3]	[3,-3]	0.39
04c	sum, product, mean	0.7, 0.3	[3,-3]	[3,-3]	0.46
04d	sum	0.7, 0.3	[3,-3]	[3,-3]	**0.58**

4.5.5. Quinto experimento: población

En esta parte de los experimentos con neuroevolución, se prueba con distintos valores para la población, cuidando también que el costo computacional no sea alto. El intervalo [100, 250] se selecciona para probar el desempeño de la neuroevolución conservando los valores del experimento 01a.

La experimentación da como mejores resultados aquellos con poblaciones de 150 y 200 individuos. Sin embargo, no se mejoran los resultados del experimento 01a, donde la población es de 50 individuos, por lo que se puede concluir que el incrementar el tamaño de la población no mejora

Tabla 4.22: Resultados del quinto experimento con neuroevolución, tamaño de población. El mejor resultado se resalta en **negritas**

Experimento	**Población**	**Probabilidad añadir y eliminar nodos**	**Probabilidad añadir y eliminar conexiones**	**Precisión**
05a	100	0.5	0.5	0.54
05b	150	0.5	0.5	**0.56**
05c	200	0.5	0.5	**0.56**
05d	250	0.5	0.5	0.55

el desempeño de la red neuronal para clasificar requisitos. Los resultados se presentan en la Tabla 4.22.

4.6. Conclusiones

Durante la experimentación para comparar los enfoques de una red neuronal diseñada por el humano y el de una construida mediante neuroevolución encontramos un desempeño superior por parte de la red neuronal diseñada por el humano con un resultado de hasta 0.793 en precisión micro en contraste con un 0.59 de precisión micro en la neuroevolución.

Para la red neuronal diseñada por el humano, el parámetro más significativo fue la tangente hiperbólica como principal función de activación. También se mantuvo una topología de red simple con 25 neuronas en capa oculta, puesto que entre más neuronas, el resultado no mostraba mejora y aumentaba el costo computacional del entrenamiento.

Por otra parte, el enfoque neuroevolutivo también mostró una mejora significativa cuando se usaba la tangente hiperbólica como principal función de activación, al igual de mantener una topología todavía simple. Sin embargo, esta no alcanzó el desempeño de la red neuronal diseñada por el humano.

Capítulo 5

Experimentos con datos filtrados

5.1. Introducción

En vista de los problemas encontrados en el capítulo anterior en el que se mostraba una diferencia significativa en resultado de la clasificación de ambos enfoques, en el presente capitulo se detalla el proceso por el cual se somete el conjunto de datos a una selección de características y a una reducción de datos excluyendo aquellas clases que causan desbalance para observar el comportamiento de la neuroevolución, conocer las causas de su, hasta ahora, bajo desempeño y definir trabajo futuro.

En esta sección se detallan los resultados obtenidos de la clasificación de requisitos utilizando ambos enfoques, la red neuronal diseñada por el humano y la red resultado del proceso neuroevolutivo obtenidas de la experimentación realizada en el Capítulo 4 pero aplicando conjuntos de datos con preprocesamientos adicionales con el fin observar el desempeño de ambos clasificadores.

5.2. Técnicas de filtrado seleccionadas

Las técnicas de filtrado seleccionadas fueron: *Correlation-based Feature Selection* (CFS), *Fast Correlation Based Filter* (FCBF), *Maximum Relevance Minimum Redundancy* (MRMR) y *Relief* ya que, al ser métodos de filtrado, son más rápidos al momento de obtener un resultado. En la Tabla 5.1 se muestran las características seleccionadas por método de filtrado.

Tabla 5.1: Número de características seleccionadas por método de filtrado

Método de filtrado	**Número de características**
CFS	33
FCBF	19
MRMR	31
Relief	104

En cada experimento con el conjunto de datos filtrado por técnica, se presentan los resultados de los mejores dobleces reportados por cada clasificador para comparar de una forma visual el rendimiento por clase.

5.3. Resultados con CFS

Al utilizar los datos filtrados con CFS, ninguna de las redes muestra una mejora significativa en la clasificación de los requisitos (Tabla 5.2a y Tabla 5.2b). La red neuronal diseñada por el humano alcanzó un máximo de 0.74 de precisión macro con un promedio de precisión micro de 0.65, menor que la registrada en la Tabla 4.17. Por su parte, la neuroevolución alcanzó un máximo de 0.51 de precisión macro, con un promedio de precisión micro de 0.55. Este último no mejora el rendimiento del experimento 01a registrado en la Tabla 4.18.

Tabla 5.2: Resultados de clasificación usando el conjunto de datos sometido a CFS. Los mejores resultados se remarcan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.68	0.60	0.54	0.55
2	0.69	**0.74**	0.54	0.58
3	0.63	0.46	0.44	0.44
4	0.79	0.68	0.53	0.53
5	0.66	0.59	0.51	0.54
6	0.66	0.64	0.54	0.55
7	0.65	0.50	0.47	0.47
8	0.49	0.31	0.32	0.30
9	0.71	0.54	0.55	0.53
10	0.57	0.49	0.46	0.45
Promedio	0.65	0.56	0.49	0.50

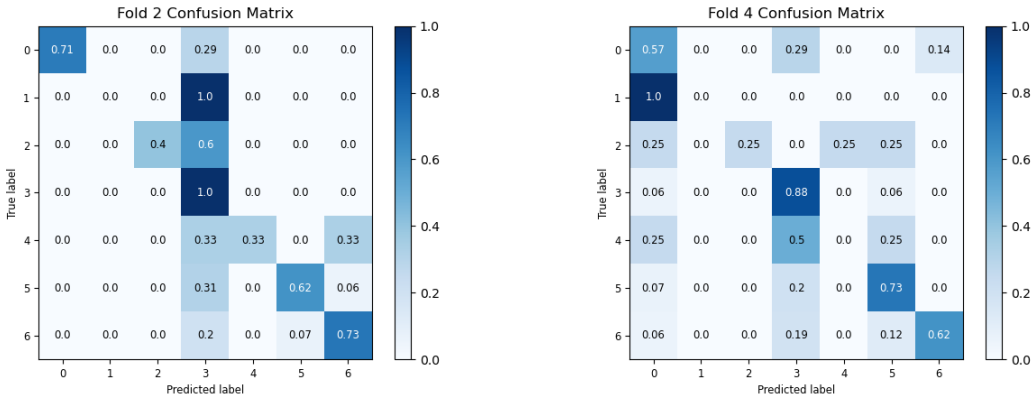
(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.58	0.26	0.34	0.29
2	0.50	0.36	0.33	0.31
3	0.57	0.35	0.38	0.36
4	0.63	**0.51**	0.43	0.43
5	0.60	0.37	0.39	0.37
6	0.58	0.36	0.40	0.37
7	0.55	0.33	0.36	0.33
8	0.52	0.32	0.33	0.30
9	0.57	0.39	0.41	0.38
10	0.41	0.18	0.24	0.20
Promedio	0.55	0.34	0.36	0.33

Tabla 5.3: Leyenda de matrices de confusión

Etiqueta	Clase	Número de registros
0	Disponibilidad	70
1	Tolerancia a fallos	14
2	Mantenibilidad	46
3	Rendimiento	156
4	Escalabilidad	38
5	Seguridad	152
6	Usabilidad	154

En un análisis comparativo de las matrices de confusión de los mejores dobleces de ambos clasificadores, se observa una clasificación más consistente del lado de la red neuronal diseñada por el humano (Figura 5.1a), cubriendo seis de las siete clases, siendo la clase de tolerancia a fallos la clase que no fue clasificada debido al número de registros que posee en el conjunto de datos



(a) Red neuronal diseñada por el humano (b) Neuroevolución

Figura 5.1: Matrices de confusión en mejor doblez de CFS

como se muestra en la Tabla 5.3. Por su parte, la neuroevolución no logró clasificar correctamente los requisitos de tolerancia a fallos y de disponibilidad (Figura 5.1b), dos de los requisitos con menos registros (Tabla 5.3). Ambos modelos lograron una clasificación satisfactoria con la clase de rendimiento, debido a que es el tipo de requisito con más registros.

5.4. Resultados con FCBF

El conjunto de datos resultante del filtrado usando FCBF presenta un resultado de menor calidad con respecto a la técnica de CFS. La mejor precisión macro registrada es de 0.53 para la red neuronal diseña por el humano (Tabla 5.4a) y de 0.41 para la neuroevolución (Tabla 5.4b). Sin embargo, los resultados de f-score, el cual es un promedio de la precisión macro y el recall, tienen valores más cercanos entre si en comparación también con el conjunto de datos de CFS.

En las matrices de confusión resultantes de los mejores dobleces, se observa que la cantidad de clases que no alcanzaron a ser clasificadas aumenta. Para la red neuronal diseñada por el humano no se clasifican dos de las siete clases, tolerancia a fallos y escalabilidad (Figura 5.2a). De parte de la neuroevolución, las clases no clasificadas son tres: tolerancia a fallos, mantenibilidad y escalabilidad (Figura 5.2b). Estas tres últimas son las clases con menos registros.

El aumento de clases no clasificadas en ambos clasificadores permite concluir que FCBF está removiendo características importantes para la clasificación, ya que en los resultados del conjunto

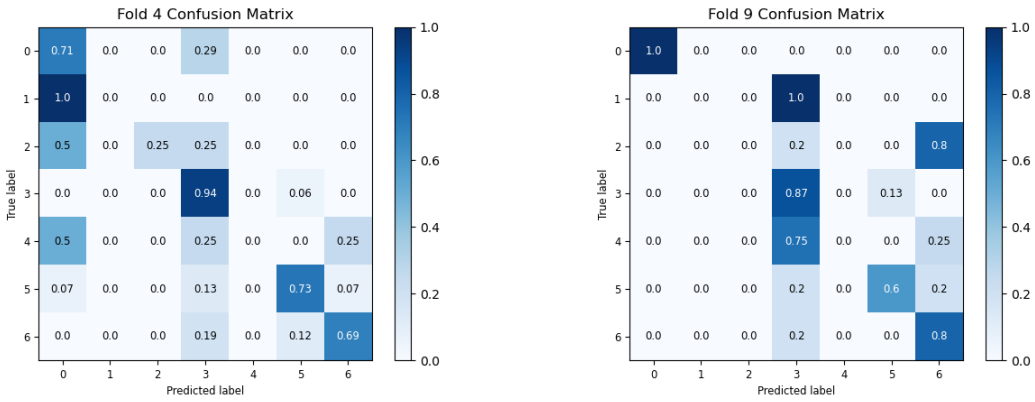
Tabla 5.4: Resultados de clasificación usando el conjunto de datos sometido a FCBF. Los mejores resultados se remarcan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.60	0.50	0.43	0.41
2	0.57	0.52	0.43	0.42
3	0.61	0.39	0.41	0.39
4	0.68	**0.53**	0.47	0.46
5	0.65	0.50	0.44	0.45
6	0.61	0.50	0.46	0.44
7	0.61	0.47	0.43	0.43
8	0.52	0.39	0.35	0.36
9	0.68	0.48	0.51	0.48
10	0.58	0.48	0.43	0.41
Promedio	0.61	0.58	0.43	0.42

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.60	0.37	0.40	0.38
2	0.38	0.22	0.23	0.21
3	0.50	0.37	0.31	0.30
4	0.68	0.39	0.43	0.41
5	0.57	0.35	0.37	0.35
6	0.49	0.21	0.28	0.24
7	0.58	0.36	0.39	0.36
8	0.49	0.27	0.31	0.28
9	0.65	**0.41**	0.46	0.43
10	0.47	0.22	0.28	0.24
Promedio	0.54	0.32	0.35	0.32



(a) Red neuronal diseñada (b) Neuroevolución

Figura 5.2: Matrices de confusión en mejor doblez de FCBF

de datos de CFS (Figura 5.1a), la única clase no clasificada es la de tolerancia a fallos.

5.5. Resultados de MRMR

Los resultados de la clasificación de MRMR son cercanos a los obtenidos con CFS. La red neuronal diseñada por el humano muestra una precisión macro hasta del 0.73 (Tabla 5.5a), cercano al máximo registrado con CFS del 0.74 (Tabla 5.2a). De forma similar, la neuroevolución presenta un resultado de hasta 0.52 (Tabla 5.5b) de precisión macro, cercano al 0.51 mostrado por el conjunto de datos sometido a CFS (Tabla 5.2b)

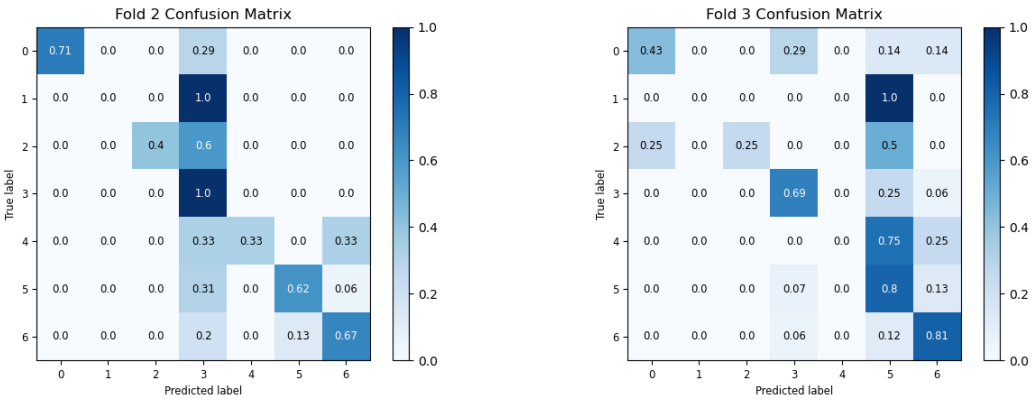
Tabla 5.5: Resultados de clasificación usando el conjunto de datos sometido a MRMR. Los mejores resultados se remarcan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.69	0.70	0.55	0.57
2	0.68	**0.73**	0.53	0.57
3	0.61	0.45	0.43	0.43
4	0.71	0.69	0.52	0.53
5	0.65	0.59	0.49	0.52
6	0.66	0.65	0.54	0.55
7	0.65	0.51	0.47	0.47
8	0.49	0.31	0.32	0.30
9	0.71	0.52	0.55	0.53
10	0.57	0.49	0.46	0.45
Promedio	0.64	0.56	0.49	0.49

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.58	0.34	0.38	0.35
2	0.52	0.44	0.35	0.34
3	0.63	**0.52**	0.42	0.43
4	0.58	0.34	0.38	0.35
5	0.53	0.24	0.31	0.26
6	0.50	0.24	0.29	0.25
7	0.55	0.36	0.38	0.35
8	0.49	0.28	0.30	0.26
9	0.55	0.26	0.33	0.28
10	0.50	0.34	0.37	0.33
Promedio	0.54	0.34	0.35	0.32



(a) Red neuronal diseñada (b) Neuroevolución

Figura 5.3: Matrices de confusión en mejor doblez de MRMR

Las matrices de confusión muestran el mismo comportamiento que las obtenidas con CFS. En la red neuronal, la clase no clasificada sigue siendo la de tolerancia a fallos (Figura 5.3a), mientras que en la neuroevolución se observa que las clases no clasificadas también se mantienen, tolerancia a fallos y escalabilidad (Figura 5.3b). Al ser resultados similares a lo visto con CFS y dados los resultados, el conjunto de datos filtrado con MRMR no alcanza a mejorar los resultados de la clasificación del conjunto de datos original.

Tabla 5.6: Resultados de clasificación usando el conjunto de datos sometido a Relief. Los mejores resultados se remarcan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.73	0.57	0.55	0.55
2	0.73	0.70	0.62	0.65
3	0.71	0.62	0.57	0.59
4	0.69	0.53	0.52	0.51
5	0.71	0.61	0.60	0.59
6	0.68	0.67	0.68	0.67
7	0.76	0.60	0.61	0.60
8	0.60	0.44	0.46	0.44
9	0.76	**0.85**	0.66	0.69
10	0.63	0.52	0.56	0.52
Promedio	0.70	0.61	0.58	0.58

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.53	0.30	0.33	0.30
2	0.49	0.21	0.28	0.24
3	0.42	0.20	0.24	0.21
4	0.55	0.30	0.35	0.32
5	0.55	0.23	0.31	0.27
6	0.53	**0.31**	0.36	0.32
7	0.53	0.25	0.31	0.27
8	0.52	0.23	0.30	0.26
9	0.61	0.26	0.37	0.30
10	0.38	0.17	0.22	0.27
Promedio	0.51	0.25	0.31	0.27

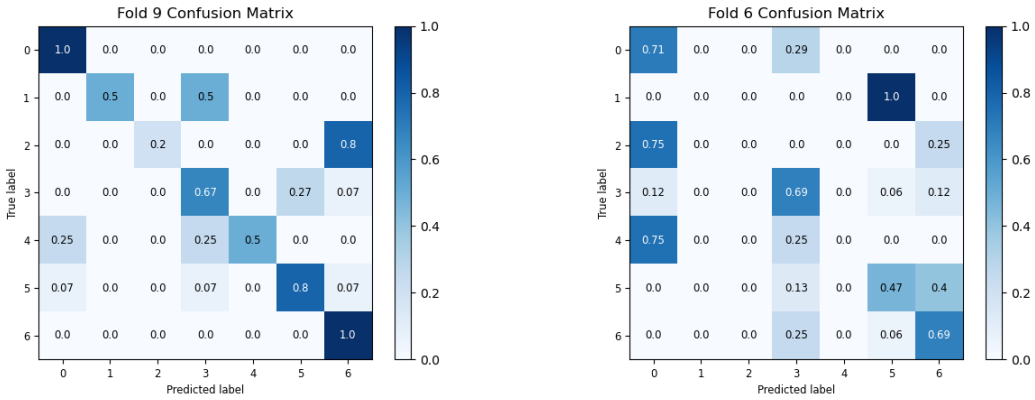
5.6. Resultados de Relief

El desempeño presentado por la clasificación del conjunto de datos filtrado por Relief muestra una ligera mejora con respecto a la del conjunto de datos original (Tabla 4.17), superando el promedio de precisión macro de 0.59 a 0.61 (Tabla 5.6a). Lo contrario sucede con la neuroevolución, puesto que no muestra una mejora en la precisión micro, de 0.51, en contraste con el mejor resultado reportado de 0.59 (Tabla 4.18 y Tabla 5.6b).

En la comparación de las matrices de confusión se observa que la red neuronal diseñada por el humano alcanza a clasificar todas las clases, siendo las clases de mantenibilidad, tolerancia a fallos y escalabilidad las que menos precisión muestran en ese orden (Figura 5.4a). Mientras, la neuroevolución clasifica cuatro de las siete clases, dejando fuera las clases antes mencionadas (Figura 5.4b).

5.7. Conclusión de clasificación con conjuntos de datos filtrado

Dados los resultados reportados en el Capítulo anterior en los que se construye una red neuronal diseñada por el humano y otra vía neuroevolución, se decidió hacer una selección de características del conjunto de datos. Las técnicas seleccionadas caen en la categoría de filtrado, siendo CFS, FCBF, MRMR y Relief los métodos seleccionados. De ellos, FCBF fue la que redujo más el conjunto de datos con un total de 19 características finales, mientras que el método que menos redujo



(a) Red neuronal diseñada (b) Neuroevolución

Figura 5.4: Matrices de confusión en mejor doblez de ReliefF

el conjunto de datos original due Relief con 104 características finales.

El mejor desempeño registrado fue el del conjunto de datos sometido al filtrado por Relief, el cual logra sobrepasar el rendimiento de la clasificación del conjunto de datos original con la red neuronal diseñada por el humano, pero empeorando la clasificación usando la neuroevolución. El desempeño de los clasificadores usando CFS y MRMR presentan un comportamiento similar dado que no realizan la clasificación de las mismas clases: tolerancia a fallos, para la red neuronal diseñada por el humano, y la adición de escalabilidad a los requisitos no clasificados por la red generada vía neuroevolución. Por otro lado, el peor desempeño registrado es el del conjunto de datos producto del filtrado mediante FCBF, en el que se aumentan las clases no clasificadas, tolerancia a fallos y escalabilidad para la red neuronal diseñada por el humano y estas últimas dos clases más la de mantenibilidad en la red generada vía neuroevolución.

En conclusión, la técnica de filtrado Relief mejora el desempeño de la red neuronal diseñada por el humano para la clasificación de requisitos, pero ningún método de filtrado ayuda a la neuroevolución a mejorar sus resultados. Como próximo paso, se experimenta removiendo las clases que causan desbalance en el conjunto de datos original para comprobar si el rendimiento de la neuroevolución mejora.

Capítulo 6

Experimentos con datos balanceados

6.1. Introducción

Los siguientes experimentos consisten en remover las clases que causan desbalance en el conjunto de datos de forma progresiva, es decir, se removerán dichas clases una por una empezando aquellas con menos registros para observar el desempeño de los clasificadores construidos en el Capítulo 4. Su análisis consta, al igual que en la experimentación con la base de datos filtrada, en la comparación del desempeño durante la validación cruzada y las matrices de confusión de los mejores dobleces de cada clasificador.

6.2. Resultados de clasificación usando 6 clases

Como primer experimento, se removieron los registros de la clase de tolerancia a fallos, dejando un total de seis clases y 616 registros. La clasificación realizada por la red neuronal diseñada por el humano (Tabla 6.1a) mejora los resultados registrados en la Tabla 4.17 pertenecientes al conjunto de datos original. Mientras que para la red generada vía neuroevolución (Tabla 6.1b) muestra un resultado todavía por debajo de la precisión micro del mejor resultado obtenido en el experimento 01a en la Tabla 4.18.

Tabla 6.1: Resultados de clasificación usando seis clases. Los mejores resultados se resaltan en **negritas**

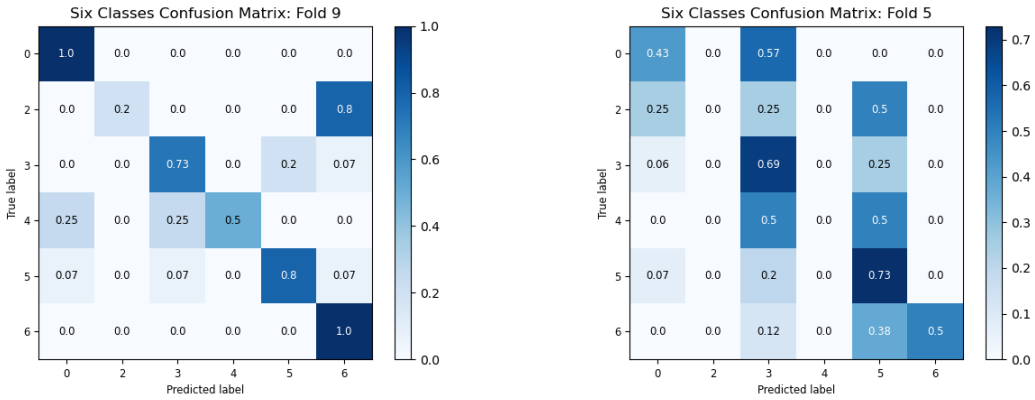
(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.75	0.73	0.68	0.69
2	0.72	0.75	0.69	0.71
3	0.72	0.66	0.60	0.62
4	0.70	0.73	0.65	0.65
5	0.67	0.69	0.62	0.64
6	0.69	0.67	0.69	0.68
7	0.81	0.79	0.75	0.76
8	0.72	0.67	0.70	0.68
9	0.78	**0.85**	0.70	0.71
10	0.62	0.54	0.56	0.54
Promedio	0.72	0.71	0.66	0.67

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.56	0.36	0.40	0.37
2	0.45	0.29	0.31	0.28
3	0.50	0.24	0.32	0.28
4	0.45	0.27	0.28	0.27
5	0.53	**0.40**	0.39	0.37
6	0.50	0.41	0.36	0.34
7	0.54	0.27	0.36	0.30
8	0.49	0.24	0.32	0.28
9	0.52	0.32	0.38	0.34
10	0.47	0.39	0.33	0.30
Promedio	0.50	0.32	0.34	0.31

En las matrices de confusión se observa que la red neuronal generada mediante neuroevolución sigue sin clasificar las clases con menos registros, dejando fuera la clases de mantenibilidad y



(a) Red neuronal diseñada (b) Neuroevolución

Figura 6.1: Matrices de confusión en mejor doblez usando seis clases

escalabilidad (Figura 6.1b). Por el contrario, la red neuronal diseñada por el humano sigue tomando en cuenta todas las clases del conjunto de datos, sin embargo las clasificación de las clases con menos registros siguen teniendo bajo desempeño (Figura 6.1a).

Concluyendo, la exclusión de la clase con menos registros, la tolerancia a fallos, que cuenta con solo 14, mejora la clasificación de la red neuronal diseñada por el humano, pero sigue sin favorecer a la red generada vía neuroevolución, puesto que se mantiene sin clasificar las clases menos balanceadas.

6.3. Resultados de clasificación usando 5 clases

La experimentación con cinco clases excluye a las llamadas tolerancia a fallos y escalabilidad, dejando a las clases de disponibilidad, mantenibilidad, rendimiento, escalabilidad, seguridad y usabilidad con un total de 578 registros.

En la Tabla 6.2a, la red neuronal diseñada por el humano muestra una mejora significativa en el promedio de las métricas utilizadas con respecto al conjunto de datos de seis clases, incluso se reporta un aumento en el máximo valor de la precisión macro en sus dobleces. La red generada mediante neuroevolución, por su parte, muestra una mejora en menor grado en su desempeño en las métricas usadas (Tabla 6.2b).

Las matriz de confusión de la red neuronal diseñada por el humano muestra una mejora en

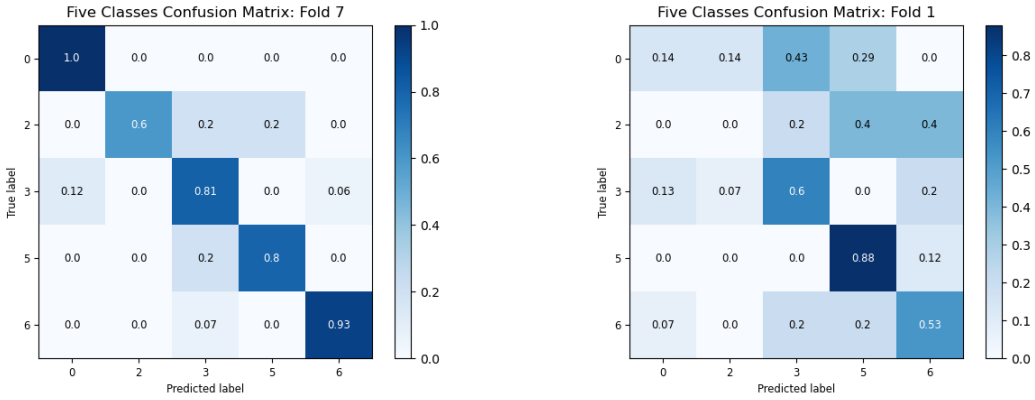
Tabla 6.2: Resultados de clasificación usando cinco clases. Los mejores resultados se resaltan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.75	0.74	0.72	0.72
2	0.75	0.79	0.74	0.76
3	0.81	0.84	0.79	0.81
4	0.77	0.83	0.76	0.76
5	0.68	0.71	0.64	0.66
6	0.77	0.80	0.79	0.79
7	0.84	**0.87**	0.82	0.83
8	0.70	0.65	0.63	0.63
9	0.80	0.87	0.74	0.75
10	0.68	0.71	0.69	0.69
Promedio	0.76	0.78	0.73	0.74

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.55	**0.40**	0.43	0.41
2	0.55	0.37	0.37	0.36
3	0.51	0.39	0.41	0.40
4	0.53	0.29	0.32	0.30
5	0.55	0.33	0.40	0.37
6	0.53	0.33	0.39	0.35
7	0.48	0.29	0.36	0.32
8	0.43	0.31	0.33	0.32
9	0.57	0.35	0.44	0.39
10	0.49	0.30	0.37	0.32
Promedio	0.52	0.34	0.38	0.35



(a) Red neuronal diseñada (b) Neuroevolución

Figura 6.2: Matrices de confusión en mejor doblez usando cinco clases

la clasificación de los requisitos, cubriendo todas la clases y con un buen desempeño por clase clasificada, donde la clase con menos puntuación es la clase de mantenibilidad con un valor de 0.6 y la de mayor puntuación es la de disponibilidad con un valor de 1.0 (Figura 6.1a). Por otro lado, la red generada con neuroevolución presenta una mejor cobertura en la cantidad de clases clasificadas, siendo la clase de mantenibilidad la que no logra clasificarse (Figura 6.2b). En cuanto a su desempeño por clase, la menos puntuado es la clase de disponibilidad con un 0.14, mientras que la mejor es la de seguridad con 0.88.

Tabla 6.3: Resultados de clasificación usando cuatro clases. Los mejores resultados se resaltan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.81	0.80	0.82	0.80
2	0.74	0.73	0.75	0.74
3	0.79	0.79	0.78	0.78
4	0.81	0.82	0.83	0.81
5	0.75	0.76	0.75	0.75
6	0.67	0.73	0.68	0.70
7	0.86	0.88	0.86	0.86
8	0.73	0.73	0.75	0.73
9	0.86	**0.89**	0.88	0.88
10	0.89	0.70	0.71	0.69
Promedio	0.77	0.78	0.78	0.77

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.66	0.42	0.45	0.42
2	0.51	0.38	0.37	0.37
3	0.52	0.64	0.47	0.47
4	0.56	0.45	0.48	0.46
5	0.52	0.53	0.47	0.47
6	0.54	0.44	0.47	0.44
7	0.54	0.41	0.47	0.43
8	0.52	0.53	0.57	0.54
9	0.71	**0.70**	0.71	0.70
10	0.56	0.41	0.48	0.44
Promedio	0.57	0.49	0.49	0.47

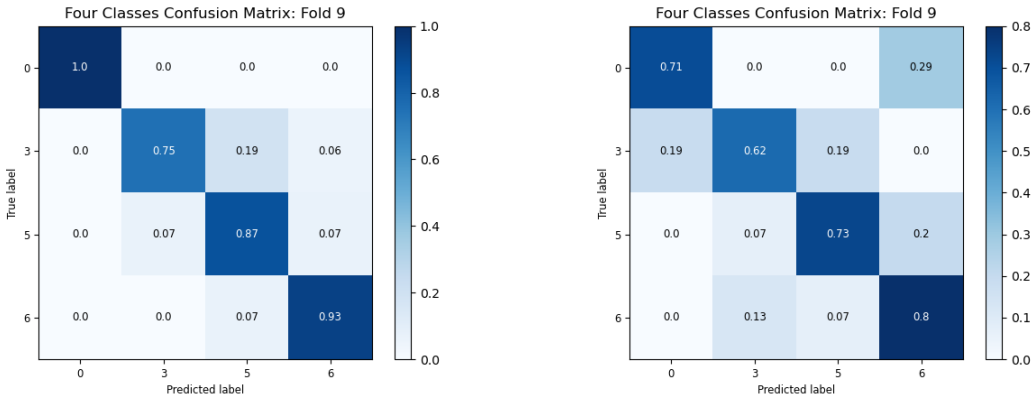
6.4. Resultados de clasificación usando 4 clases

En las pruebas realizadas al conjunto de datos con cuatro clases: disponibilidad, rendimiento, seguridad y usabilidad, se observa una mejora significativa en el noveno doblez para ambas redes neuronales (Tabla 6.3). La red neuronal diseñada por el humano obtuvo un máximo de 0.89 de precisión macro con un promedio de 0.78 (Tabla 6.3a). Mientras que la red generada por la neuroevolución muestra una precisión macro máxima de 0.70 con un promedio de 0.49 (Tabla 6.3b). Se observa que el intervalo de la precisión macro presentada en los dobleces de la red obtenida por la neuroevolución es amplia, pues el mínimo valor registrado es 0.38, mientras que el de la red neuronal generada por el humano es de 0.73.

Las matriz de confusión de la red obtenida vía neuroevolución muestra una mejora en su desempeño, dado que esta vez se clasifican las cuatro clases con un resultado mínimo de 0.62 al clasificar la clase de rendimiento (Figura 6.3b). Por su parte, la red neuronal diseñada por el humano mantiene un desempeño mayor con una mínima de 0.75 de precisión al clasificar la clase de rendimiento (Tabla 6.3a).

6.5. Resultados de clasificación usando 3 clases

El último de los experimentos consiste en la clasificación usando tres clases: rendimiento, seguridad y usabilidad. Los resultados reportados son superiores a los demás experimentos en los que



(a) Red neuronal diseñada (b) Neuroevolución

Figura 6.3: Matrices de confusión en mejor doblez usando cuatro clases

reducen las clases que causan un desbalance en el conjunto de datos.

La red neuronal diseñada por el humano alcanza una precisión macro de hasta 0.90 en el doblez siete y una mínima de 0.66 en el doblez seis con un promedio de 0.79 (Tabla 6.4a). Aunque se haya registrado un valor alto en la clasificación, el promedio no mejora demasiado cuando se compara con la prueba realizada con cuatro clases en la que se obtiene un 0.78 de promedio de precisión macro (Tabla 6.3a).

El desempeño de la red generada mediante neuroevolución muestra una mejora con respecto al experimento anterior, alcanzando un máximo de 0.82 de precisión macro en el primer doblez y una mínima puntuación de 0.46 en el décimo doblez con un promedio de 0.65 (Tabla 6.4b), superado significativamente a la clasificación de cuatro clases (Tabla 6.3b).

Para este experimento, las matrices de confusión cambian la etiqueta utilizada para cada clase: 1) rendimiento, 2) seguridad, 3) usabilidad. La red neuronal diseñada por el humao muestra una clasificación con mejor desempeño cuando las clases son balanceadas. En este caso, el mínimo es de 0.8 de precisión al clasificar la clase de seguridad y un máximo de 0.94 de precisión al clasificar los requisitos de rendimiento (Figura 6.4a). En cambio la red creada vía neuroevolución muestra un desempeño mínimo de 0.5 de precisión al clasificar requisitos de rendimiento, y un máximo de 1.0 al clasificar requisitos de seguridad (Figura 6.4b).

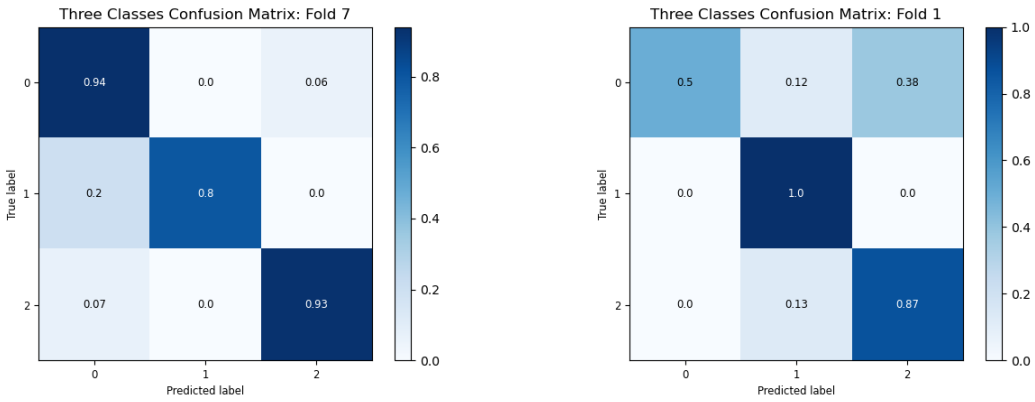
Tabla 6.4: Resultados de clasificación usando tres clases. Los mejores resultados se resaltan en **negritas**

(a) Red neuronal diseñada por el humano

Doblez	Precision micro	Precisión macro	Recall	F-score
1	0.82	0.83	0.82	0.82
2	0.78	0.79	0.78	0.78
3	0.82	0.83	0.82	0.82
4	0.80	0.82	0.80	0.79
5	0.78	0.78	0.78	0.78
6	0.65	0.66	0.65	0.65
7	0.89	**0.90**	0.89	0.89
8	0.76	0.76	0.76	0.74
9	0.84	0.84	0.85	0.84
10	0.65	0.64	0.64	0.62
Promedio	0.78	0.79	0.78	0.77

(b) Neuroevolución

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.78	**0.82**	0.78	0.77
2	0.68	0.68	0.68	0.68
3	0.71	0.72	0.72	0.71
4	0.65	0.65	0.65	0.65
5	0.67	0.67	0.67	0.67
6	0.65	0.69	0.65	0.63
7	0.56	0.56	0.56	0.56
8	0.63	0.63	0.63	0.62
9	0.63	0.63	0.63	0.63
10	0.45	0.46	0.45	0.45
Promedio	0.64	0.65	0.64	0.63



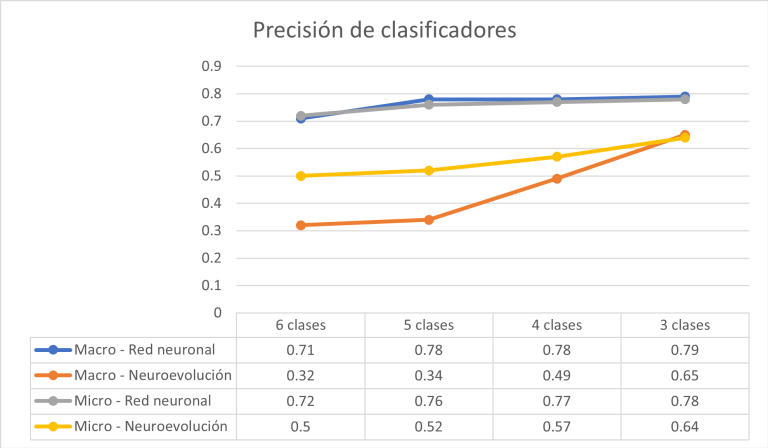
(a) Red neuronal diseñada (b) Neuroevolución

Figura 6.4: Matrices de confusión en mejor doblez usando tres clases

6.6. Conclusiones de experimentos con conjunto de datos balanceados

La experimentación con conjunto de datos balanceados en esta tesis consistió en remover las clases que causan un desbalance en los datos. Se removieron los requisitos de tolerancia a fallos, escalabilidad, mantenibilidad y disponibilidad en ese orden, ya que estos no cuentan con una cantidad como las que tienen los requisitos de rendimiento, seguridad y usabilidad (Tabla 5.3).

En el análisis de los resultados, se observó que, al extraer las clases no balanceadas, se obtienen mejores resultados en las métricas reportadas. La Figura 6.5 muestra que al disminuir el número de



	6 clases	5 clases	4 clases	3 clases
Macro - Red neuronal	0.71	0.78	0.78	0.79
Macro - Neuroevolución	0.32	0.34	0.49	0.65
Micro - Red neuronal	0.72	0.76	0.77	0.78
Micro - Neuroevolución	0.5	0.52	0.57	0.64

Figura 6.5: Precisiones micro y macro de la red neuronal y la neuroevolución

clases y dejar las clases más balanceadas, el resultado de la clasificación mejora.

Al centrarnos en los resultados de la red obtenida por la neuroevolución, se observa una mejora significativa en las precisiones promedio de la clasificación (Figura 6.5), a diferencia de la red neuronal diseñada por el humano, la cual se mantiene con un promedio constante en la precisión, la neuroevolución mejora notablemente si las clases están balanceadas.

Con estos experimentos se puede concluir que la la red obtenida mediante neuroevolución mejora su desempeño de forma significativa para clasificar el conjunto de datos de requisitos cuando está balanceado. Lo anterior sugiere una mejora en el preprocesameinto del conjunto de datos para obtener resultados de mayor calidad en la clasificación de requisitos, pues el conjunto actual presenta el problema de desbalance y está afectando a los clasificadores propuestos la tesis de Pérez-Verdejo y cols. (2021) y los presentados en esta tesis.

Dado que el procesamiento de los datos está fuera de los alcances de esta tesis, los siguientes pasos a realizar son el ajuste de los parámetros de la neuroevolución por conjunto de datos balanceados y el análisis de la topología de las redes resultantes del proceso de neuroevolución.

Capítulo 7

Afinación para conjuntos balanceados

7.1. Introducción

Con los resultados obtenidos de la experimentación con el conjunto de datos balanceado (Capítulo 6), en este Capítulo se afinan los parámetros de la neuroevolución de una forma similar a lo reportado en la Sección 4.5, tomando cada uno de ellos y tratando de mejorar los resultados de la clasificación.

Los parámetros elegidos en estos experimentos son: el intervalo de pesos, la probabilidad de mutación de los pesos y las conexiones iniciales. También se emplea la validación cruzada con diez dobleces y el uso de matrices de confusión para ilustrar el desempeño en la clasificación.

7.2. Método

Como primer experimento, se hace énfasis en modificar el intervalo de pesos en los cuales la neuroevolución tomará sus valores. En esta sección se realizan cuatro esquemas de neuroevolución, uno por cada base de datos balanceada, primero tomando como intervalos a probar los siguientes: [5, -5], [10, -10], [20, -20], [30, -30]. Como segunda parte, la probabilidad de mutación de los pesos tomará los valores de 0.2, 0.4, 0.6 y 0.8. Finalmente se experimenta con las conexiones iniciales

7.3. Afinación de parámetros con 6 clases

En esta sección, se detallan los experimentos realizados con los intervalos de pesos seleccionados aplicándolos a la neuroevolución usando seis clases.

7.3.1. Intervalos de pesos con 6 clases

7.3.1.1. Intervalo [5, -5]

El primer experimento realizado consiste en limitar el intervalo de pesos a [5, -5] del cual la neuroevolución tomará sus valores para el conjunto de datos que omite la clase de tolerancia a fallos. En la Tabla 7.1, se aprecia que el mejor de los resultados obtenidos fue en el séptimo doblez con una precisión macro de 0.45 y logrando superar el promedio de *recall* y *F-score* del primer experimento

realizado con el mismo conjunto de datos de 0.34 a 0.36 y 0.31 a 0.32 respectivamente.

Tabla 7.1: Resultados de pesos con 6 clases en [5, -5]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.58	0.30	0.39	0.33
2	0.51	0.36	0.41	0.38
3	0.56	0.45	0.38	0.36
4	0.51	0.29	0.34	0.30
5	0.54	0.39	0.40	0.37
6	0.50	0.25	0.33	0.28
7	0.59	**0.45**	0.40	0.37
8	0.49	0.29	0.33	0.31
9	0.47	0.23	0.32	0.26
10	0.44	0.23	0.30	0.25
Promedio	0.52	0.32	0.36	0.32

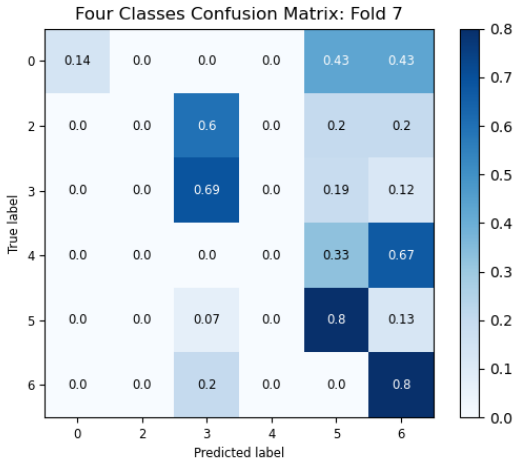


Figura 7.1: Matriz de confusión de experimento con 6 clases con pesos dentro de [5, -5]

Respecto al desempeño mostrado en la matriz de confusión (Figura 7.1), se observa que las clases de 2) mantenibilidad y 4) escalabilidad no están siendo clasificadas por la red neuronal ganadora en su mejor doblez.

7.3.1.2. Intervalo [10, -10]

Al aumentar el intervalo de pesos a [10, -10] (Tabla 7.2), se aprecia que el sexto doblez ha sido el que mejor desempeño ha tenido, con una precisión macro de 0.50, *recall* de 0.49 y *F-score* de 0.48, con un promedio en las tres métricas de 0.35, 0.40 y 0.37, respectivamente. Con estos resultados el

intervalo de pesos de [10, -10] se coloca como el parámetro de pesos con mejor desempeño en el proceso de neuroevolución.

Tabla 7.2: Resultados de pesos con 6 clases en [10, -10]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.59	0.29	0.40	0.33
2	0.43	0.26	0.29	0.27
3	0.56	0.36	0.39	0.37
4	0.58	0.38	0.44	0.41
5	0.59	**0.50**	**0.49**	**0.48**
6	0.58	0.41	0.43	0.41
7	0.59	0.30	0.39	0.34
8	0.54	0.28	0.35	0.31
9	0.62	0.35	0.41	0.38
10	0.52	0.38	0.43	0.40
Promedio	0.56	0.35	0.40	0.37

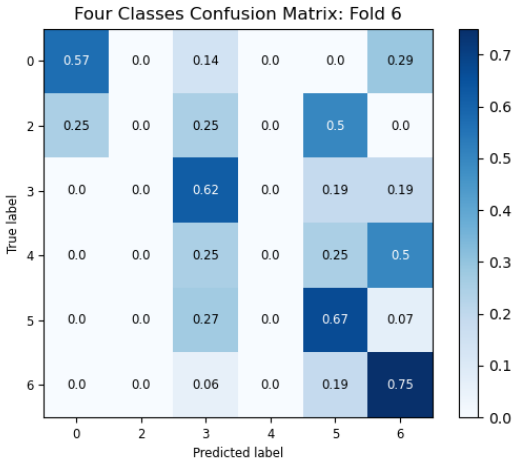


Figura 7.2: Matriz de confusión de experimento con 6 clases con pesos dentro de [10, -10]

Aunque el intervalo de [10, -10] haya sido el mejor reportado, las clases de 2) mantenibilidad y 4) escalabilidad siguen sin poder alcanzar una clasificación (Figura 7.2). De cualquier modo, se alcanzan precisiones superiores en comparación con la matriz de confusión del intervalo [5, -5] (Figura 7.1).

7.3.1.3. Intervalo [20, -20]

Al probar con el intervalo de [20, -20] para los valores de los pesos se muestra un resultado promedio de precisión macro de 0.35, *recall* de 0.37 y *F-score* de 0.35 (Tabla 7.3). El desempeño para este intervalo no muestra una mejora respecto a los valores de [10, -10] (Tabla 7.2). Finalmente, como antes se ha estado reportando, las clases de 2) mantenibilidad y 4) escalabilidad siguen sin clasificarse (Figura 7.3).

Tabla 7.3: Resultados de pesos con 6 clases en [20, -20]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.50	0.34	0.36	0.34
2	0.43	0.30	0.31	0.30
3	0.50	0.26	0.29	0.27
4	0.45	0.26	0.29	0.25
5	0.58	0.43	0.43	0.42
6	0.51	0.38	0.40	0.38
7	0.60	0.41	0.46	0.43
8	0.55	0.45	0.41	0.39
9	0.63	**0.46**	**0.48**	**0.45**
10	0.40	0.20	0.27	0.23
Promedio	0.51	0.35	0.37	0.35

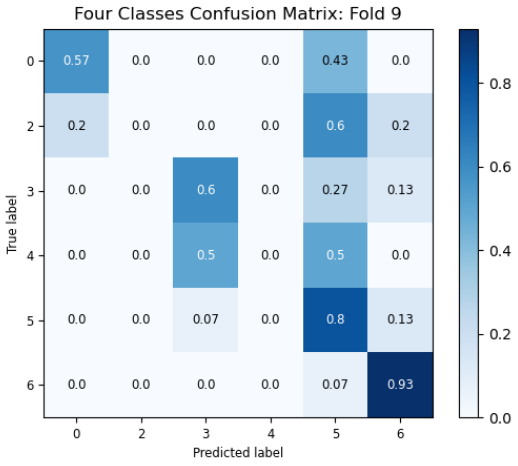


Figura 7.3: Matriz de confusión de experimento con 6 clases con pesos dentro de [20, -20]

7.3.1.4. Intervalo [30, -30]

El desempeño con los pesos en [30, -30] (Tabla 7.4), muestra una mejora con respecto al intervalo de [10, -10] (Tabla 7.2), con un promedio en precisión macro de 0.38, *recall* de 0.41 y *F-score* de 0.38. A su vez, la matriz de confusión sigue sin poder clasificar las clases de 2) mantenibilidad y 4) escalabilidad en su mejor doblez (Figura 7.4).

Tabla 7.4: Resultados de pesos con 6 clases en [30, -30]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	**0.59**	**0.43**	**0.46**	**0.43**
2	0.50	0.34	0.37	0.35
3	0.56	0.36	0.43	0.39
4	0.56	0.41	0.47	0.43
5	0.50	0.40	0.35	0.34
6	0.51	0.36	0.36	0.35
7	0.45	0.31	0.36	0.33
8	0.55	0.34	0.40	0.36
9	0.59	0.43	0.45	0.42
10	0.54	0.39	0.43	0.40
Promedio	0.54	0.38	0.41	0.38

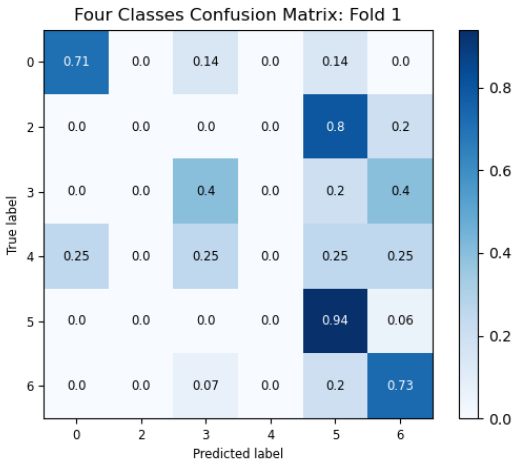


Figura 7.4: Matriz de confusión de experimento con 6 clases con pesos dentro de [30, -30]

7.3.1.5. Conclusión de experimentos con intervalos de pesos usando 6 clases

Con la finalización de estos experimentos, se ha seleccionado el intervalo de [30, -30] para los valores de pesos. Analizando los datos obtenidos, se tiene la hipótesis que este resultado es gracias al espacio que tiene la red neuronal para seleccionar sus pesos que podría estar ligada a la cantidad de clases que se manejan en el conjunto de datos. Se observa también que la clases de 5) seguridad y 6) usabilidad presentan mejores desempeños en su clasificación, debido a que son dos de las clases con más registros (Tabla 5.3).

7.3.2. Afinación de probabilidad de mutación de pesos con 6 clases

El segundo parámetro a experimentar es la probabilidad de mutación para los pesos en el conjunto de datos con seis clases.

7.3.2.1. Probabilidad de mutación de pesos de 0.2

El primer experimento con probabilidad de mutación de pesos de 0.2 utilizando seis clases muestra máximos valores en el tercer doblez con una precisión macro de 0.47, *recall* de 0.45 y *F-score* de 0.33 (Tabla 7.5). Por otro lado, la matriz de confusión muestra que las clases 2) mantenibilidad y 4) escalabilidad siguen sin ser clasificadas y que la clase mejor clasificada es 5) seguridad (Figura 7.5).

Tabla 7.5: Resultados de probabilidad de mutación de 0.2 con 6 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.58	0.29	0.39	0.33
2	0.45	0.23	0.31	0.26
3	0.61	**0.47**	**0.45**	**0.43**
4	0.48	0.24	0.32	0.27
5	0.58	0.41	0.44	0.42
6	0.45	0.23	0.30	0.26
7	0.51	0.25	0.33	0.29
8	0.56	0.44	0.38	0.35
9	0.54	0.36	0.38	0.35
10	0.51	0.26	0.34	0.30
Promedio	0.53	0.32	0.36	0.33

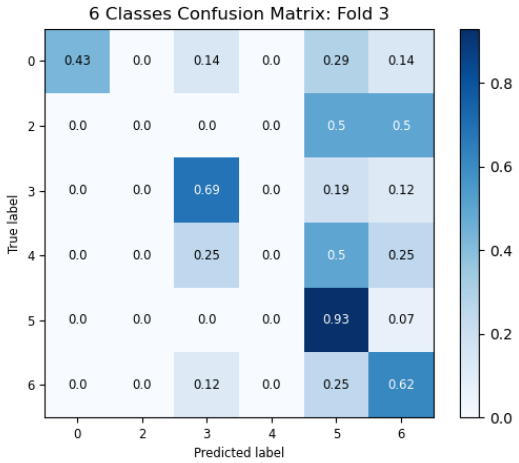


Figura 7.5: Matriz de confusión de experimento con 6 clases con mutación de pesos de 0.2

7.3.2.2. Probabilidad de mutación de pesos de 0.4

El desempeño de la neuroevolución utilizando una probabilidad de mutación de 0.4 (Tabla 7.6) no muestra una mejora con respecto a la probabilidad de mutación de 0.2 (Tabla 7.5), pues se reportan valores promedio de precisión macro, *recall* y *F-score* inferiores. Al observar la matriz de confusión (Figura 7.6), el comportamiento de no clasificar las clases 2) mantenibilidad y 4) escalabilidad persiste.

Tabla 7.6: Resultados de probabilidad de mutación de 0.4 con 6 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.52	0.26	0.35	0.29
2	0.42	0.29	0.29	0.27
3	0.58	**0.40**	**0.41**	**0.39**
4	0.52	0.27	0.34	0.30
5	0.55	0.28	0.36	0.31
6	0.53	0.28	0.35	0.31
7	0.43	0.21	0.28	0.24
8	0.52	0.35	0.40	0.37
9	0.56	0.29	0.38	0.32
10	0.48	0.32	0.39	0.35
Promedio	0.51	0.30	0.35	0.31

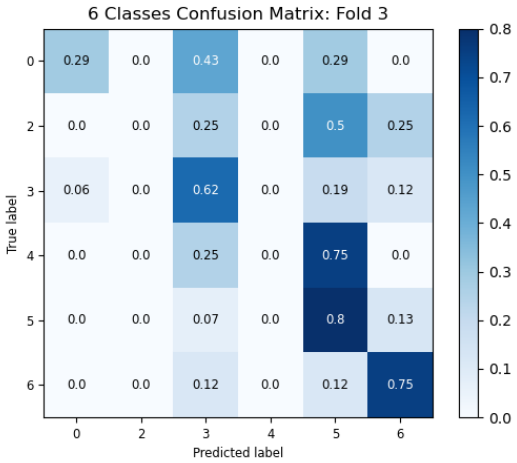


Figura 7.6: Matriz de confusión de experimento con 6 clases con mutación de pesos de 0.4

7.3.2.3. Probabilidad de mutación de pesos de 0.6

La probabilidad de mutación de pesos con valor de 0.6 sigue sin mostrar una mejora en el promedio de las métricas en contraste con la probabilidad de mutación de 0.2, pues tiene una precisión macro de 0.30, *recall* de 0.35 y *F-score* de 0.31 (Tabla 7.7), que no superan el 0.32, 0.36 y 0.33 obtenidos en el primer experimento (Tabla 7.5). En cuanto a la matriz de confusión, sigue son clasificar las clases 2) mantenibilidad y 4) escalabilidad (Figura 7.7).

Tabla 7.7: Resultados de probabilidad de mutación de 0.6 con 6 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.48	0.23	0.28	0.24
2	0.45	0.24	0.30	0.26
3	0.55	0.36	0.39	0.37
4	0.56	0.29	0.37	0.32
5	0.61	**0.47**	**0.43**	**0.41**
6	0.53	0.27	0.35	0.30
7	0.51	0.36	0.40	0.37
8	0.49	0.26	0.33	0.29
9	0.56	0.28	0.38	0.32
10	0.49	0.22	0.29	0.24
Promedio	0.52	0.30	0.35	0.31

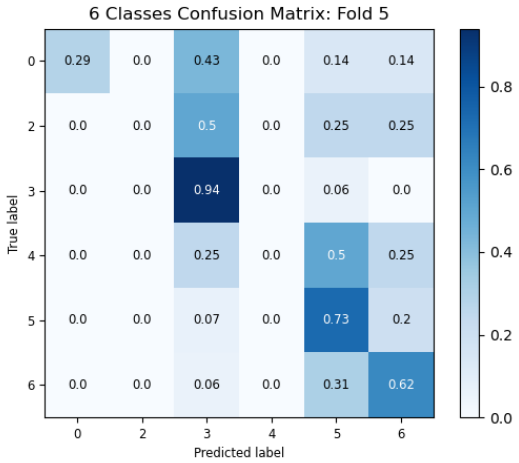


Figura 7.7: Matriz de confusión de experimento con 6 clases con mutación de pesos de 0.6

7.3.2.4. Probabilidad de mutación de pesos de 0.8

La probabilidad de mutación con valor de 0.8 muestra sus mejores resultados en el quinto doblez, con valores de precisión macro de 0.46, **recall** de 0.45 y **F-score** de 0.44 (Tabla 7.8) con un promedio de dichas métricas de 0.36, 0.37 y 0.35 siendo el que mejor resultados ha dado en la experimentación de este parámetro. Por su parte, la matriz de confusión no muestra un cambio en la clasificación de las clases 2) mantenibilidad y 4) escalabilidad (Figura 7.8).

Tabla 7.8: Resultados de probabilidad de mutación de 0.8 con 6 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.56	0.35	0.41	0.38
2	0.45	0.40	0.32	0.30
3	0.56	0.37	0.41	0.39
4	0.48	0.33	0.37	0.34
5	0.60	**0.46**	**0.45**	**0.44**
6	0.52	0.28	0.34	0.30
7	0.41	0.37	0.29	0.28
8	0.51	0.31	0.35	0.32
9	0.61	0.40	0.39	0.37
10	0.49	0.34	0.37	0.35
Promedio	0.52	0.36	0.37	0.35

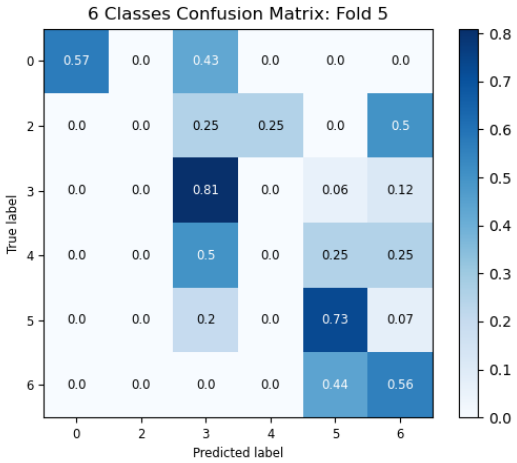


Figura 7.8: Matriz de confusión de experimento con 6 clases con mutación de pesos de 0.8

7.3.3. Afinación de conexión inicial con 6 clases

En anteriores experimentos realizados en el Capitulo 6, se utiliza la configuración obtenida de la construcción del esquema neuroevolutivo, el cual usa una conexión completa inicial para inicializar su población. En esta sección, se experimenta con el enfoque contrario, las conexiones en vez de inicializarse por completo, se deja al algoritmo crear las conexiones necesarias para las redes.

Los resultados de la neuroevolución sin conexiones iniciales presentan un desempeño inferior al de las conexiones iniciales completas. Los promedios de las métricas fueron de 0.29 para la precisión macro, 0.30 para el **recall** y 0.25 para **F-score** (Tabla 7.9), no logrando mostrar una mejora a los resultados de 0.32, 0.34 y 0.31 respectivamente mostrado en la Tabla 6.1b.

Por otro lado, se obtiene un resultado diferente en la matriz de confusión, pues a diferencia de todo el procedimiento para afinar los parámetros de la neuroevolución aplicada al conjunto de datos de 6 clases, en este experimento se logra clasificar la clase 3) escalabilidad, dejando solo a la clase 1) mantenibilidad sin ser clasificada (Figura 7.9). Este hecho puede deberse a que, al dejar al algoritmo evolucionar las conexiones desde cero, se realiza una selección de características, pues analizando las topologías resultantes se observa que se dejan nodos de entrada sin usar.

Tabla 7.9: Resultados de neuroevolución con 6 clases sin conexiones iniciales. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.44	0.29	0.29	0.24
2	0.47	0.26	0.31	0.26
3	0.44	0.24	0.29	0.25
4	0.39	0.30	0.29	0.25
5	0.39	**0.57**	**0.34**	**0.35**
6	0.47	0.37	0.33	0.30
7	0.43	0.22	0.28	0.24
8	0.41	0.15	0.27	0.19
9	0.41	0.15	0.28	0.19
10	0.43	0.30	0.29	0.24
Promedio	0.43	0.29	0.30	0.25

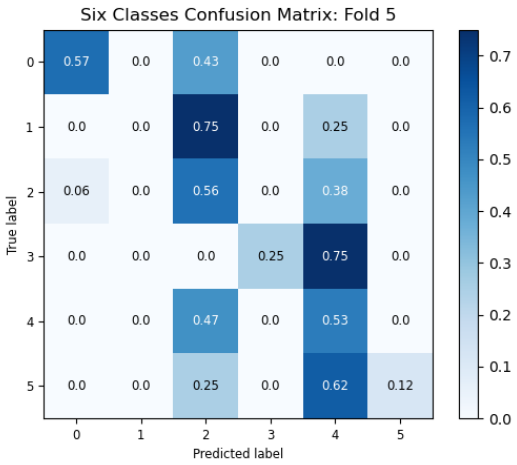


Figura 7.9: Matriz de confusión de neuroevolución con 6 clases sin conexiones iniciales

7.3.4. Conclusiones de afinación de parámetros con 6 clases

Los parámetros finales de la afinación de la neuroevolución usando seis clases se muestran en la tabla 7.10. El intervalo de pesos seleccionado para este esquema fue el de [20,-20], con una probabilidad de mutación de 0.8 y conexiones iniciales completas. En los experimentos realizados se muestra que las clases 2) mantenibilidad y 4) escalabilidad no son clasificados.

Tabla 7.10: Parámetros finales de afinación con 6 clases

Parámetro	Valor
Intervalo de pesos	[20, -20]
Probabilidad de mutación	0.8
Conexiones iniciales	Completa

7.4. Afinación de parámetros con 5 clases

En la siguiente sección se presentan los experimentos realizados para ajustar los parámetros de la neuroevolución aplicada al conjunto de datos con 5 clases.

7.4.1. Afinación de pesos con 5 clases

7.4.1.1. Intervalo [5, -5]

El primero de los intervalos a probar, [5, -5], muestra un máximo de precisión macro de 0.58, *recall* de 0.58 y *F-score* de 0.57 en el primer doblez con un promedio de 0.46, 0.58 y 0.46 respectivamente (Tabla 7.11). En una vista diferente del desempeño del clasificador, la matriz de confusión muestra que la clase 2) mantenibilidad sigue sin ser clasificada como sucede con los experimentos realizados en la Sección 7.3, pero las clases 3) rendimiento, 5) seguridad y 6) usabilidad siguen teniendo una mejor clasificación.

Tabla 7.11: Resultados de pesos con 5 clases en [5, -5]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	**0.68**	**0.58**	**0.58**	**0.57**
2	0.51	0.36	0.36	0.35
3	0.60	0.48	0.49	0.48
4	0.56	0.45	0.46	0.45
5	0.56	0.48	0.45	0.44
6	0.60	0.50	0.52	0.50
7	0.55	0.43	0.46	0.44
8	0.55	0.45	0.49	0.47
9	0.61	0.53	0.52	0.51
10	0.57	0.36	0.44	0.38
Promedio	0.58	0.46	0.48	0.46

Figura 7.10: Matriz de confusión de experimento con 5 clases con pesos dentro de [5, -5]

7.4.1.2. Intervalo [10, -10]

En el intervalo de [10, -10], el mejor desempeño de la neuroevolución se muestra en el cuarto doblez con una precisión macro de 0.47, *recall* de 0.47 y *F-score* de 0.46, con un promedio de 0.36, 0.39 y 0.36 respectivamente (Tabla 7.12). El resultado obtenido no superaron al del intervalo de [5. -5] (Tabla 7.11). Desde el punto de vista de la matriz de confusión, se muestra que la clase 2) mantenibilidad se mantiene sin poder ser clasificada.

Tabla 7.12: Resultados de pesos con 5 clases en [10, -10]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.51	0.41	0.41	0.40
2	0.46	0.29	0.35	0.31
3	0.56	0.49	0.46	0.46
4	0.70	**0.47**	**0.47**	**0.46**
5	0.43	0.39	0.28	0.27
6	0.46	0.29	0.34	0.31
7	0.51	0.31	0.39	0.34
8	0.50	0.39	0.39	0.37
9	0.57	0.34	0.44	0.38
10	0.43	0.27	0.33	0.28
Promedio	0.51	0.36	0.39	0.36

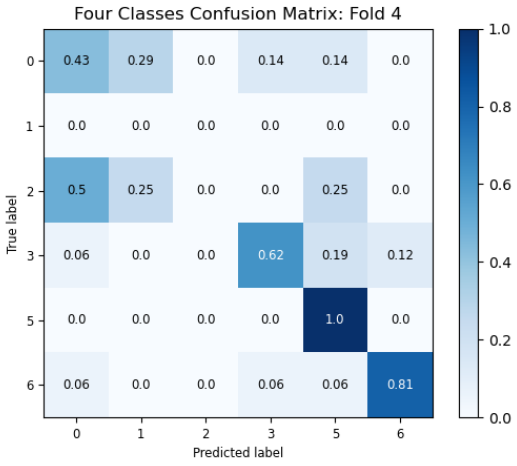


Figura 7.11: Matriz de confusión de experimento con 5 clases con pesos dentro de [10, -10]

7.4.1.3. Intervalo [20, -20]

El intervalo de [20, -20] presenta mejoras en el resultado dentro de los dobleces, alcanzando máximo de precisión macro de 0.78. *recall* de 0.67 y *F-score* de 0.70 en el quinto doblez, con un promedio en dichas métricas de 0.48, 0.49 y 0.47 (Tabla 7.13), superando así los resultados promedios del intervalo [5, -5] (Tabla 7.11). En cuanto a la clasificación de las clases, la matriz de confusión del quinto doblez muestra que todas las clases han sido clasificadas con un desempeño de 0.5 o más de precisión (Figura 7.12).

Tabla 7.13: Resultados de pesos con 5 clases en [20, -20]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.63	0.57	0.52	0.52
2	0.53	0.35	0.40	0.37
3	0.60	0.46	0.49	0.47
4	0.68	0.54	0.57	0.55
5	0.70	**0.78**	**0.67**	**0.70**
6	0.55	0.38	0.37	0.37
7	0.53	0.52	0.49	0.49
8	0.56	0.44	0.46	0.43
9	0.64	0.40	0.49	0.43
10	0.56	0.33	0.42	0.37
Promedio	0.60	0.48	0.49	0.47

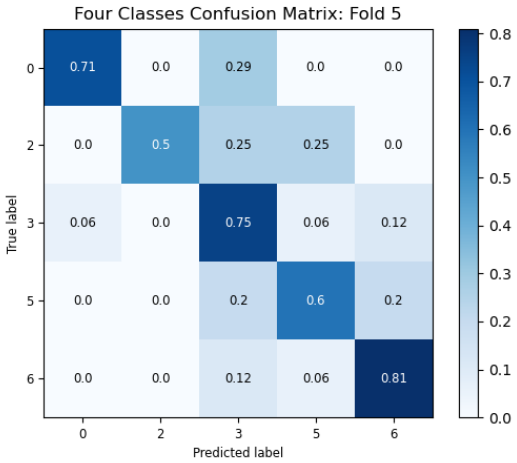


Figura 7.12: Matriz de confusión de experimento con 5 clases con pesos dentro de [20, -20]

7.4.1.4. Intervalo [30, -30]

El experimento realizado tomando valores en el intervalo [30, -30] no presenta una mejora para la clasificación de los requisitos usando las cinco clases, pues muestra un promedio de precisión macro de 0.39. *recall* de 0.42 y *F-score* de 0.39 (Tabla 7.14), quedando por debajo de los resultados del intervalo [20, -20] (Tabla 7.13). Además, la matriz de confusión muestra que la clase 2) mantenibilidad vuelve a no ser clasificada por la red neuronal creada por neuroevolución (Figura 7.13), en contraste a como sucede con el intervalo [20, -20] que clasifica todas las clases (Figura 7.12).

Tabla 7.14: Resultados de pesos con 5 clases en [30, -30]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.62	0.47	0.51	0.48
2	0.60	0.32	0.38	0.34
3	0.55	0.46	0.47	0.45
4	0.51	0.40	0.43	0.41
5	0.60	**0.49**	**0.52**	**0.49**
6	0.53	0.33	0.39	0.35
7	0.50	0.38	0.42	0.40
8	0.55	0.38	0.37	0.36
9	0.63	0.33	0.40	0.36
10	0.47	0.32	0.33	0.32
Promedio	0.55	0.39	0.42	0.39

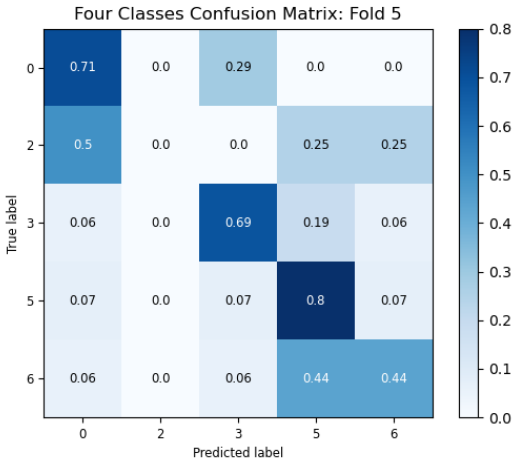


Figura 7.13: Matriz de confusión de experimento con 5 clases con pesos dentro de [30, -30]

7.4.1.5. Conclusiones de afinación de pesos usando 5 clases

Experimentando con el conjunto de cinco clases y los valores de los pesos se encontró que los valores dentro de [20, -20] han sido los que mejor han funcionado para la clasificación, logrando incluso que, en uno de los dobleces se logren clasificar todas las clases (Figura 7.12), fenómeno que no pasó en los demás intervalos.

7.4.2. Afinación de probabilidad de mutación de pesos con 5 clases

En esta sección se presentan los experimentos realizados con la probabilidad de mutación de los pesos en el esquema neuroevolutivo usando cinco clases. Como se menciona en anteriores secciones, se experimenta con probabilidades de 0.2, 0.4, 0.6 y 0.8.

7.4.2.1. Probabilidad de mutación de pesos de 0.2 usando 5 clases

Los resultados mostrados en la Tabla 7.15 reportan un máximo de precisión macro de 0.62, *recall* de 0.49 y *F-score* de 0.51 en el cuarto doblez, con un promedio de 0.40, 0.43, y 0.40 respectivamente. La matriz de confusión de la Figura 7.15 muestra que la clasificación abarca todas las clases, sin embargo el desempeño es menor a anteriores experimentos, como lo muestra la matriz de confusión de la Figura 7.12.

Tabla 7.15: Resultados de probabilidad de mutación de 0.2 con 5 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.59	0.35	0.44	0.39
2	0.60	0.37	0.46	0.40
3	0.60	0.31	0.37	0.34
4	0.57	**0.62**	**0.49**	**0.51**
5	0.67	0.41	0.50	0.45
6	0.59	0.36	0.43	0.39
7	0.52	0.42	0.44	0.42
8	0.53	0.35	0.40	0.37
9	0.54	0.55	0.43	0.40
10	0.47	0.30	0.36	0.31
Promedio	0.57	0.40	0.43	0.40

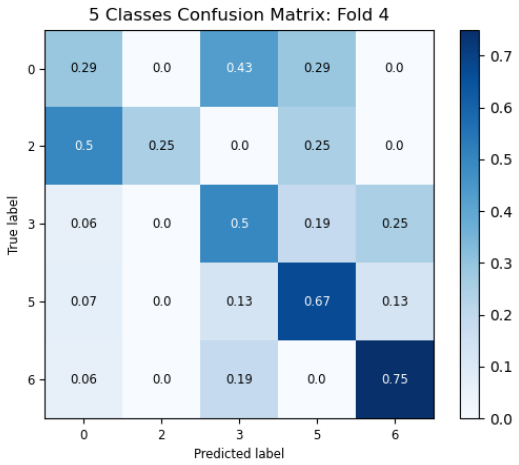


Figura 7.14: Matriz de confusión de experimento con 5 clases con mutación de pesos de 0.2

7.4.2.2. Probabilidad de mutación de pesos de 0.4 usando 5 clases

La probabilidad de mutación de los pesos en la neuroevolución con cinco clases obtiene promedios de precisión macro, *recall* y *F-score* similares a los resultados del anterior experimento en la Sección 7.4.2.1, pero registrando sus máximos en el séptimo doblez con 0.62, 0.49 y 0.51 (Tabla 7.16). En el análisis de la matriz de confusión, se muestra que la clase 2) mantenibilidad no es clasificada por el algoritmo (Figura 7.15).

Tabla 7.16: Resultados de probabilidad de mutación de 0.4 con 5 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.66	0.51	0.52	0.50
2	0.57	0.45	0.46	0.45
3	0.60	0.57	0.46	0.44
4	0.57	0.30	0.35	0.32
5	0.64	0.39	0.47	0.42
6	0.45	0.36	0.35	0.33
7	0.62	**0.60**	**0.49**	**0.47**
8	0.55	0.33	0.42	0.37
9	0.61	0.37	0.47	0.41
10	0.46	0.27	0.35	0.30
Promedio	0.57	0.41	0.43	0.40

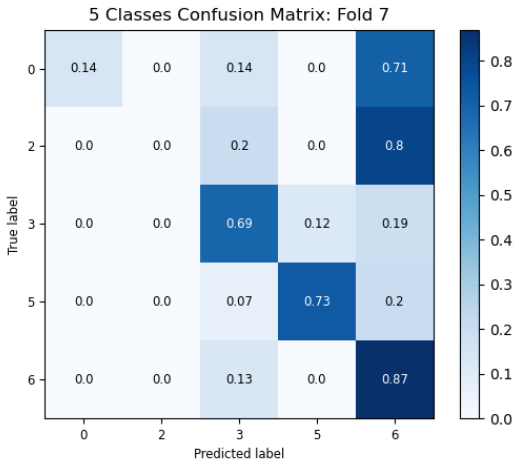


Figura 7.15: Matriz de confusión de experimento con 5 clases con mutación de pesos de 0.4

7.4.2.3. Probabilidad de mutación de pesos de 0.6 usando 5 clases

El experimento con probabilidad de mutación de 0.6 muestra un máximo desempeño en el quinto doblez con una precisión macro de 0.77, *recall* de 0.60 y *F-score* de 0.62, con un promedio de 0.43, 0.45 y 0.42 (Tabla 7.17), los cuales superan los resultados mostrados en la Tabla 6.2b pero quedando por debajo del desempeño mostrado en la Tabla 7.13. También, se puede observar en la matriz de confusión de la Figura 7.16 que todas las clases se están clasificando, sin embargo, la clase 2) mantenibilidad conserva un valor bajo de clasificación, similar al visto en la Figura 7.14.

Tabla 7.17: Resultados de probabilidad de mutación de 0.6 con 5 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.66	0.52	0.52	0.50
2	0.57	0.35	0.43	0.38
3	0.53	0.33	0.40	0.36
4	0.60	0.36	0.45	0.40
5	0.72	**0.77**	**0.60**	**0.62**
6	0.55	0.35	0.41	0.37
7	0.52	0.32	0.39	0.35
8	0.59	0.45	0.47	0.45
9	0.61	0.48	0.44	0.44
10	0.51	0.32	0.39	0.34
Promedio	0.59	0.43	0.45	0.42

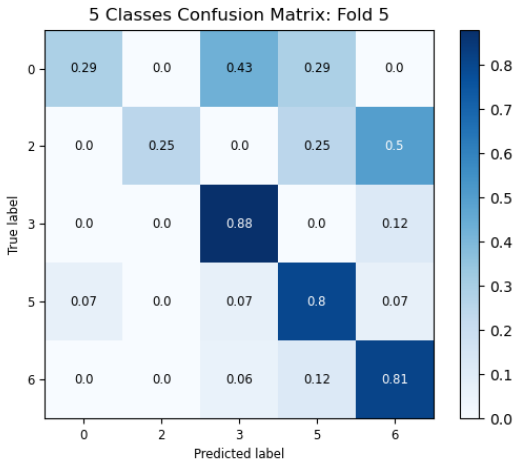


Figura 7.16: Matriz de confusión de experimento con 5 clases con mutación de pesos de 0.6

7.4.2.4. Probabilidad de mutación de pesos de 0.8 usando 5 clases

La probabilidad de mutación de 0.8 aplicada a la neuroevolución de cinco clases muestra un máximo de precisión macro de 0.59 en el quinto doblez, mientras que los máximos de *recall* y *F-score* se muestran en el primer doblez con valores de 0.59 y 0.57 (Tabla 7.18). El promedio de las métricas en este experimento supera al de los anteriores. Por otro lado, la clasificación de requisitos se muestra más robusta, a excepción de la clase 2) mantenibilidad, que en este caso no es clasificada (Figura 7.17). La alta precisión de las clases que sí fueron clasificadas compensa el desempeño de la que no ha sido clasificada, dando un resultado más alto en comparación a los de

las otras probabilidades de mutación.

Tabla 7.18: Resultados de probabilidad de mutación de 0.8 con 5 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.72	0.55	**0.59**	**0.57**
2	0.53	0.40	0.36	0.36
3	0.52	0.43	0.45	0.44
4	0.67	0.51	0.55	0.53
5	0.66	**0.59**	0.52	0.51
6	0.55	0.33	0.41	0.36
7	0.57	0.46	0.51	0.47
8	0.47	0.36	0.40	0.37
9	0.60	0.39	0.45	0.41
10	0.54	0.40	0.40	0.38
Promedio	0.58	0.44	0.46	0.44

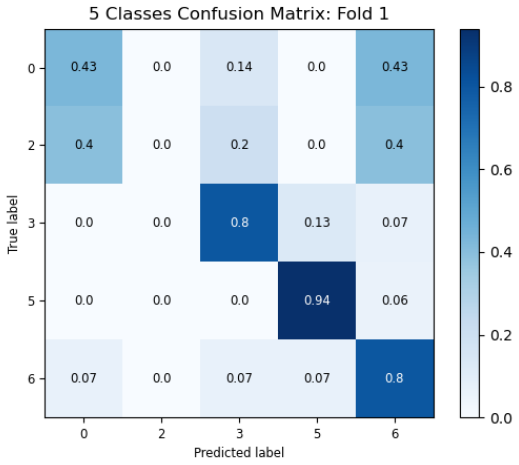


Figura 7.17: Matriz de confusión de experimento con 5 clases con mutación de pesos de 0.8

7.4.2.5. Conclusiones de afinación de probabilidad de mutación con 5 clases

Para nuestro caso, el valor de probabilidad de mutación que ha mostrado un mejor desempeño ha sido 0.8. Durante los experimentos, varias de las redes ganadoras resultantes muestran, a diferencia de los experimentos realizados con seis clases, una clasificación de todas las clases en el conjunto de datos.

7.4.3. Afinación de conexiones iniciales con 5 clases

En esta sección se presentan los resultados de el experimento realizado para probar el desempeño de un tipo de conexión diferente al realizado anteriormente, ya que la neuroevolución se ha estado llevando a cabo con conexiones completas iniciales. En el presente experimento, la neuroevolución inicia sin conexiones entre neuronas de distintas capas.

En la Tabla 7.19 la precisión macro, *recall* y *F-score* no muestran una mejora en la clasificación de los requisitos, aunque logra un máximo de precisión macro de 0.55 en el primer doblez, un *recall* de 0.46 y un *F-score* de 0.45 en el cuarto doblez. Este hecho descarta el uso de conexiones iniciales inexistentes para la neuroevolución usando cinco clases.

Además, se observa que la clase 2) mantenibilidad no se está clasificando, tal como se muestra en la matriz de confusión de la Figura 7.18, que también arroja un bajo desempeño respecto al mostrado en la Figura 7.17.

Tabla 7.19: Resultados de neuroevolución con 5 clases sin conexiones iniciales. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.50	**0.55**	0.43	0.41
2	0.43	0.41	0.35	0.33
3	0.36	0.27	0.31	0.26
4	0.57	0.53	**0.46**	**0.45**
5	0.50	0.43	0.41	0.38
6	0.47	0.51	0.36	0.32
7	0.47	0.33	0.43	0.36
8	0.38	0.16	0.28	0.20
9	0.51	0.34	0.39	0.35
10	0.40	0.26	0.31	0.24
Promedio	0.46	0.38	0.37	0.33

7.4.4. Conclusiones de afinación de parámetros con 5 clases

Los parámetros finales de la afinación de la neuroevolución usando cinco clases se muestran en la Tabla 7.20. Este esquema de neuroevolución posee uno de los intervalos más extensos con los que se ha experimentado y una probabilidad de mutación de pesos alta. En las conexiones, el principal hallazgo ha sido el hecho que, al dejar las conexiones completas, la neuroevolución llega

Figura 7.18: Matriz de confusión de neuroevolución con 5 clases sin conexiones iniciales

a valores más altos de clasificación a diferencia de dejar desarrollar dichas conexiones en cada generación del proceso evolutivo.

Tabla 7.20: Parámetros finales de afinación con 5 clases

Parámetro	Valor
Intervalo de pesos	[20, -20]
Probabilidad de mutación	0.8
Conexiones iniciales	Completa

7.5. Afinación de parámetros con 4 clases

Los experimentos para afinar los parámetros de la neuroevolución empleando el conjunto con 4 clases se incluyen en esta sección. Se inicia con el intervalo del que se pueden tomar los pesos. Después, se experimenta con la probabilidad de mutación de los pesos y, finalmente, con el tipo de conexión inicial.

7.5.1. Afinación de pesos con 4 clases

7.5.1.1. Intervalo [5, -5]

El primer intervalo probado para la clasificación de cuatro clases muestra un máximo de precisión macro en el tercer doblez con un valor de 0.71, un máximo de *recall* en el segundo doblez

con un valor de 0.58 y un máximo de *F-score* con un valor de 0.57, con un promedio en las tres métricas anteriores de 0.52, 0.49 y 0.47 (Tabla 7.21). La matriz de confusión (Figura 7.19) muestra una clasificación de todas las clases, siendo las clases 3) rendimiento, 5) seguridad y 6) usabilidad las que mejor precisión presentan dado que poseen más registros que la clase 0) disponibilidad.

Tabla 7.21: Resultados de pesos con 4 clases en [5, -5]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.61	0.59	**0.58**	**0.57**
2	0.56	0.43	0.48	0.45
3	0.60	**0.71**	0.54	0.54
4	0.60	0.71	0.54	0.53
5	0.55	0.55	0.49	0.48
6	0.55	0.34	0.38	0.35
7	0.51	0.49	0.49	0.48
8	0.49	0.36	0.43	0.39
9	0.55	0.67	0.51	0.53
10	0.49	0.36	0.42	0.38
Promedio	0.55	0.52	0.49	0.47

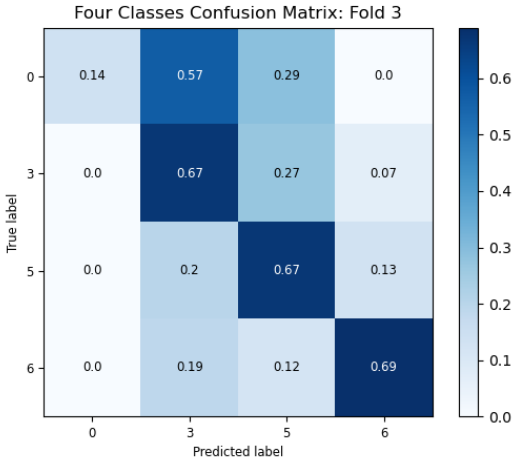


Figura 7.19: Matriz de confusión de experimento con 4 clases con pesos dentro de [5, -5]

7.5.1.2. Intervalo [10, -10]

El intervalo [10, -10] muestra mejora con respecto al intervalo [5, -5], pues presenta un promedio mayor de precisión macro con 0.56, *recall* de 0.52 y *F-score* de 0.52. En la matriz de confusión de la Figura 7.20 se puede observar que todas las clases están siendo clasificadas pero con mayor

precisión en la clase 0) disponibilidad con un valor de 0.43, que en el anterior intervalo (Figura 7.19), con un valor de 0.14.

Tabla 7.22: Resultados de pesos con 4 clases en [10, -10]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.61	0.48	0.52	0.49
2	0.65	0.40	0.45	0.42
3	0.64	0.74	0.57	0.57
4	0.68	0.67	**0.64**	0.64
5	0.60	0.52	0.45	0.46
6	0.66	0.56	0.55	0.55
7	0.57	0.56	0.42	0.44
8	0.57	0.50	0.51	0.50
9	0.66	**0.75**	0.63	**0.65**
10	0.55	0.41	0.47	0.43
Promedio	0.62	0.56	0.52	0.52

Figura 7.20: Matriz de confusión de experimento con 4 clases con pesos dentro de [10, -10]

7.5.1.3. Intervalo [20, -20]

En la prueba realizada con el intervalo [20, -20], se obtienen máximos de precisión macro de 0.75 en el octavo doblez y *recall* de 0.68 y *F-score* de 0.68 en el cuarto doblez, con un promedio de 0.54, 0.53 y 0.52, respectivamente. Este desempeño no muestra una mejora en la clasificación del conjunto de datos de cuatro clases. Sin embargo, en la Figura 7.21, la matriz de confusión del cuarto doblez muestra un mejor desempeño en la clasificación de los requisitos de 0) disponibilidad

que en la matriz de confusión del anterior experimento (Figura 7.20), de 0.45 a 0.57. También se observa una mejora en las demás clases. Para la clase 3) rendimiento, el resultado mejora de 0.56 a 0.60 y para la clase 6) usabilidad el resultado aumenta de 0.67 a 0.69.

Tabla 7.23: Resultados de pesos con 4 clases en [20, -20]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.65	0.54	0.49	0.50
2	0.61	0.41	0.42	0.40
3	0.62	0.59	0.58	0.58
4	0.70	0.69	**0.68**	**0.68**
5	0.64	0.71	0.63	0.65
6	0.60	0.37	0.42	0.39
7	0.66	0.48	0.46	0.45
8	0.66	**0.75**	0.63	0.65
9	0.66	0.50	0.57	0.53
10	0.49	0.38	0.42	0.39
Promedio	0.63	0.54	0.53	0.52

Figura 7.21: Matriz de confusión de experimento con 4 clases con pesos dentro de [20, -20]

7.5.1.4. Intervalo [30, -30]

Finalmente, el intervalo de [30, -30] muestra un máximo desempeño en el sexto doblez con una precisión macro de 0.70, *recall* de 0.60 y *F-score* de 0.62, con un promedio de 0.54, 0.53 y 0.52, respectivamente (Tabla 7.24). Sin embargo, los resultados promedios no superan a los presentados

por el intervalo [10, -10] (Tabla 7.22). Por su parte, la matriz de confusión de la Figura 7.22 muestra que todas las clases siguen siendo clasificadas. Sin embargo, no se muestra una mejora en la clasificación.

Tabla 7.24: Resultados de pesos con 4 clases en [30, -30]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.74	0.56	0.64	0.59
2	0.63	0.48	0.54	0.51
3	0.60	0.46	0.52	0.49
4	0.68	0.52	0.42	0.43
5	0.60	0.51	0.43	0.44
6	0.62	**0.70**	**0.60**	**0.62**
7	0.60	0.45	0.52	0.48
8	0.57	0.44	0.49	0.45
9	0.57	0.49	0.46	0.46
10	0.57	0.58	0.58	0.58
Promedio	0.63	0.54	0.53	0.52

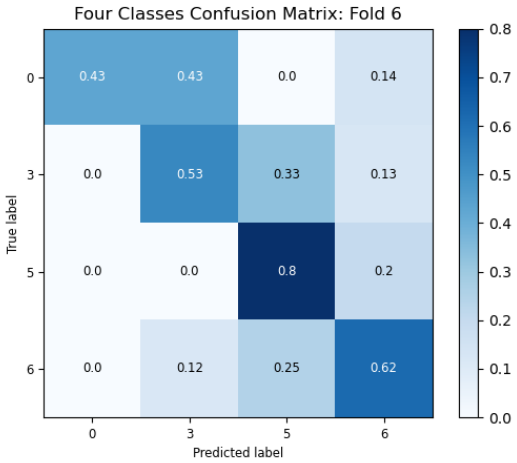


Figura 7.22: Matriz de confusión de experimento con 4 clases con pesos dentro de [30, -30]

7.5.1.5. Conclusiones de afinación de pesos con 4 clases

Al experimentar la clasificación de las clases de disponibilidad, rendimiento, seguridad y usabilidad con cuatro diferentes intervalos para los valores de sus pesos, se concluye que el intervalo más conveniente para la neuroevolución usando este conjunto de datos es [10, -10]. Se observa-

ron máximas precisiones macro que superan el 0.70 y una clasificación de todas las clases en los mejores dobleces debido a que las clases con más desbalance han sido removidas.

7.5.2. Afinación de probabilidad de mutación con 4 clases

7.5.2.1. Probabilidad de mutación de 0.2 usando 4 clases

El experimento para afinar probabilidad de mutación con valor de 0.2 muestra un máximo de precisión macro, *recall* y *F-score* en el séptimo doblez (Tabla 7.25). La matriz de confusión de la Figura 7.23 indica que todas las clases de este conjunto de datos están siendo clasificadas con mejor desempeño, dando a entender que el desbalance en los datos es un obstáculo para lograr mejores resultados en la neuroevolución.

Tabla 7.25: Resultados de probabilidad de mutación de 0.2 con 4 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.61	0.49	0.53	0.49
2	0.56	0.43	0.48	0.45
3	0.66	0.51	0.57	0.53
4	0.64	0.58	0.57	0.57
5	0.66	0.50	0.57	0.53
6	0.55	0.44	0.48	0.44
7	0.66	**0.71**	**0.65**	**0.67**
8	0.57	0.42	0.49	0.45
9	0.68	0.53	0.59	0.55
10	0.53	0.39	0.45	0.41
Promedio	0.61	0.50	0.54	0.51

7.5.2.2. Probabilidad de mutación de 0.4 usando 4 clases

La probabilidad de mutación de 0.4 presenta una mejora con respecto a la probabilidad e 0.2, con una precisión macro promedio de 0.57 y un *recall* promedio de 0.52 (Tabla 7.26). En los resultados detallados por doblez, se observan más dobleces que superan el 0.6 de precisión macro. También, se sigue presentando la clasificación de todas las clases del conjunto de datos en la matriz de confusión de la Figura 7.24.

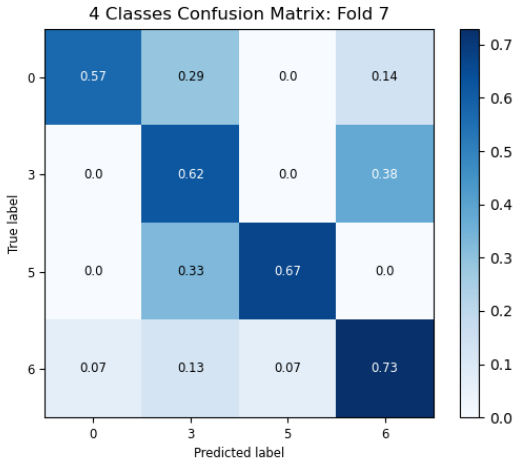


Figura 7.23: Matriz de confusión de experimento con 4 clases con mutación de pesos de 0.2

Tabla 7.26: Resultados de probabilidad de mutación de 0.4 con 4 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.61	0.70	0.57	0.58
2	0.63	0.71	0.58	0.59
3	0.64	0.67	0.61	0.62
4	0.58	0.64	0.56	0.56
5	0.51	0.38	0.44	0.40
6	0.57	0.43	0.49	0.45
7	0.64	0.41	0.44	0.42
8	0.62	0.70	0.60	0.61
9	0.70	**0.79**	**0.68**	**0.70**
10	0.47	0.27	0.32	0.28
Promedio	0.60	0.57	0.53	0.52

7.5.2.3. Probabilidad de mutación de 0.6 usando 4 clases

Los resultados para probabilidad de mutación de 0.6 muestran una mejora con respecto a los anteriores experimentos con un promedio de precisión macro de 0.57, *recall* de 0.54 y *F-score* de 0.54 y registrando valores máximos de 0.74, 0.68 y 0.69 en el cuarto doblez (Tabla 7.27). La matriz de confusión de la Figura 7.25 sigue presentando la clasificación de todas las clases con mejor desempeño por encima del 0.50.

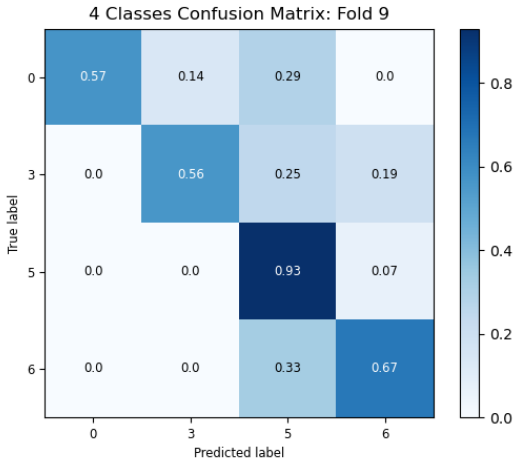


Figura 7.24: Matriz de confusión de experimento con 4 clases con mutación de pesos de 0.4

Tabla 7.27: Resultados de probabilidad de mutación de 0.6 con 4 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.59	0.47	0.51	0.49
2	0.50	0.39	0.43	0.41
3	0.58	0.56	0.44	0.45
4	0.68	**0.74**	**0.68**	**0.69**
5	0.64	0.70	0.65	0.67
6	0.64	0.61	0.58	0.56
7	0.62	0.52	0.46	0.45
8	0.66	0.65	0.65	0.65
9	0.74	0.61	0.59	0.59
10	0.60	0.42	0.41	0.41
Promedio	0.63	0.57	0.54	0.54

7.5.2.4. Probabilidad de mutación de 0.8 usando 4 clases

El desempeño obtenido al usar la probabilidad de mutación con un valor de 0.8 no logra superar al de 0.6, pues presenta promedios inferiores de precisión macro de 0.50 (Tabla 7.28), en contraste con un 0.57 (Tabla 7.27). Con respecto al análisis de la matriz de confusión de la Figura 7.26, no se encuentra novedad alguna, más que el comportamiento de clasificar todas las clases sigue presentándose.

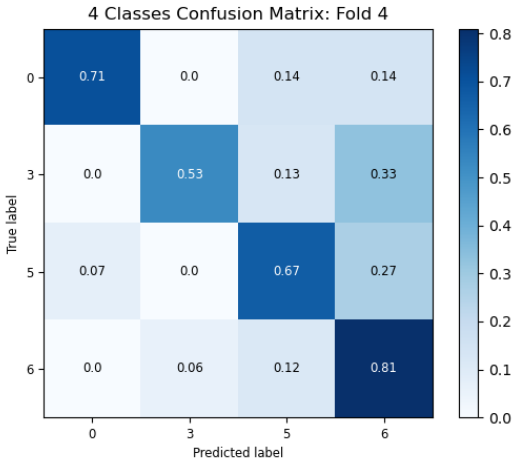


Figura 7.25: Matriz de confusión de experimento con 4 clases con mutación de pesos de 0.6

Tabla 7.28: Resultados de probabilidad de mutación de 0.8 con 4 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.61	0.47	0.52	0.49
2	0.50	0.33	0.35	0.32
3	0.60	0.37	0.42	0.39
4	0.70	0.53	0.60	0.56
5	0.66	0.51	0.57	0.53
6	0.68	**0.70**	**0.70**	**0.70**
7	0.53	0.55	0.47	0.46
8	0.66	0.43	0.46	0.44
9	0.62	0.48	0.54	0.50
10	0.58	0.64	0.54	0.52
Promedio	0.61	0.50	0.52	0.49

7.5.2.5. Conclusiones de afinación de probabilidad de mutación con 4 clases

En este experimento se probaron cuatro diferentes probabilidades de mutación para los pesos: 0.2, 0.4, 0.6 y 0.8, siendo el valor de 0.6 el que mejor resultados arroja en su ejecución. Dados estos resultados y comparándolo con el mismo experimento con los experimentos con 5 y 6 clases se observa el patrón que entre menos clases se tengan en el conjunto de datos, los valores de los parámetros tienden a ser menores, por ejemplo, para seis clases se utiliza un intervalo de [20, -20] pero para cuatro clases es suficiente con un intervalo de [10, -10].

Figura 7.26: Matriz de confusión de experimento con 4 clases con mutación de pesos de 0.8

7.5.3. Afinación de conexiones iniciales con 4 clases

Los valores para afinar este parámetro consta de conexiones completas iniciales o sin conexiones iniciales. Como todos los experimentos se han realizado con conexiones completas, en esta sección se experimenta el comportamiento de la neuroevolución cuando se inicia sin conexiones. El desempeño mostrado en la Tabla 7.29 es inferior a cuando se inicia con conexiones completas, aunque su mejor desempeño se presenta en el séptimo doblez. Por otra parte, la matriz de confusión de la Figura 7.27 indica una clasificación de todas las clases pero se observa que la clase 2) seguridad está confundiéndose con las demás clases.

Tabla 7.29: Resultados de neuroevolución con 4 clases sin conexiones iniciales. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.59	0.48	0.51	0.47
2	0.39	0.44	0.34	0.30
3	0.42	0.32	0.36	0.33
4	0.42	0.47	0.37	0.29
5	0.43	0.50	0.39	0.33
6	0.53	0.47	0.45	0.44
7	0.62	**0.79**	**0.56**	**0.56**
8	0.36	0.30	0.31	0.28
9	0.47	0.44	0.41	0.40
10	0.51	0.43	0.44	0.41
Promedio	0.47	0.46	0.41	0.38

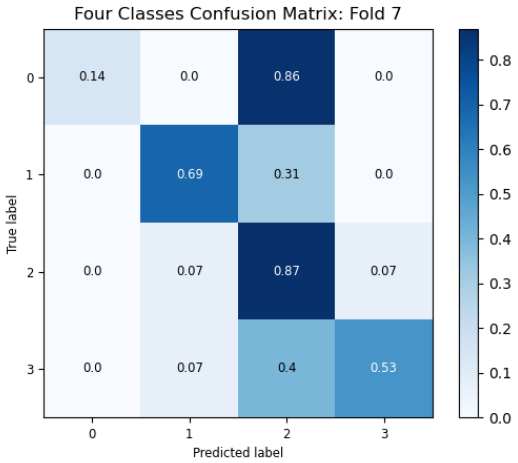


Figura 7.27: Matriz de confusión de neuroevolución con 4 clases sin conexiones iniciales

7.5.4. Conclusiones de afinación de parámetros con 4 clases

Los parámetros finales de la afinación del proceso de neuroevolución del conjunto de datos usando 4 clases tiene como parámetros finales los mostrados en la Tabla 7.30.

Tabla 7.30: Parámetros finales de afinación con 4 clases

Parámetro	Valor
Intervalo de pesos	[10, -10]
Probabilidad de mutación	0.60
Conexiones iniciales	Completa

7.6. Afinación de parámetros con 3 clases

En la presente sección se presentan los resultados realizados para afinar los parámetros de intervalo de pesos, probabilidad de mutación de pesos y el tipo de conexión inicial de la neuroevolución considerando 3 clases.

7.6.1. Afinación de intervalo de pesos con 3 clases

7.6.1.1. Intervalo [5,-5]

El desempeño del intervalo en [5,-5] muestra su máximo en el cuarto doblez y un promedio de precisión macro de 0.67, *recall* de 0.65 y *F-score* de 0.65 (Tabla 7.31). La exclusión de las clases que causan desbalance en nuestro conjunto de datos causa que la clasificación tenga un mejor desempeño. En la matriz de confusión de la Figura 7.28 se observa la clasificación de las tres clases con precisiones que superan al 0.6.

Tabla 7.31: Resultados de pesos con 3 clases en [5, -5]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.66	0.69	0.66	0.65
2	0.70	0.71	0.70	0.70
3	0.65	0.66	0.65	0.65
4	0.78	**0.78**	**0.79**	**0.78**
5	0.72	0.72	0.72	0.72
6	0.72	0.72	0.72	0.71
7	0.65	0.53	0.49	0.50
8	0.74	0.58	0.56	0.56
9	0.65	0.70	0.65	0.65
10	0.61	0.64	0.61	0.59
Promedio	0.69	0.67	0.65	0.65

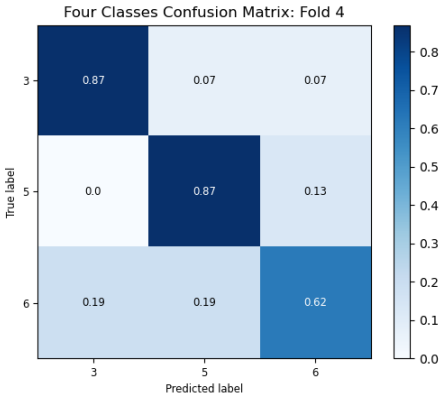


Figura 7.28: Matriz de confusión de experimento con 3 clases con pesos dentro de [5, -5]

7.6.1.2. Intervalo [10,-10]

El desempeño del intervalo [10, -10] muestra un máximo rendimiento en el primer doblez con valores de 0.84, 0.83 y 0.83 de precisión macro, *recall* y *F-score*, con un promedio de 0.69, 0.68 y 0.67 (Tabla 7.32) superando al promedio de mostrado en el anterior intervalo en la Tabla 7.31. A su vez, la matriz de confusión de la Figura 7.29 sigue mostrando la clasificación de todas las clases.

Tabla 7.32: Resultados de pesos con 3 clases en [10, -10]. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.83	**0.84**	**0.83**	**0.83**
2	0.57	0.44	0.43	0.43
3	0.72	0.57	0.54	0.55
4	0.80	0.82	0.80	0.81
5	0.78	0.79	0.78	0.78
6	0.63	0.64	0.64	0.62
7	0.76	0.80	0.76	0.76
8	0.70	0.57	0.52	0.53
9	0.80	0.82	0.81	0.80
10	0.65	0.65	0.65	0.64
Promedio	0.73	0.69	0.68	0.67

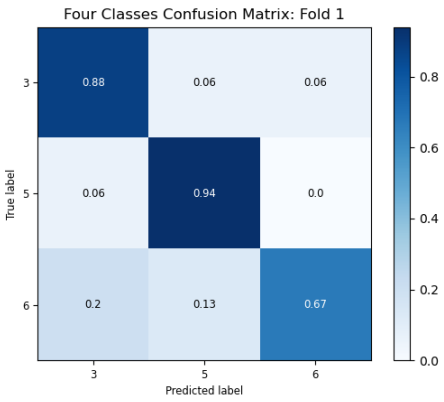


Figura 7.29: Matriz de confusión de experimento con 3 clases con pesos dentro de [10, -10]

7.6.1.3. Intervalo [20, -20]

El desempeño en el intervalo [20, -20] muestra máximos de 0.76, 0.74 y 0.74 de precisión macro, *recall* y *F-score* en el noveno doblez con promedios de 0.68, 0.67 y 0.67, respectivamente.

Sus resultados no mejoran los presentados en la Tabla 7.33. En cuanto a la matriz de confusión de la Figura 7.30, se mantiene la clasificación de todas las clases pero con un desempeño inferior al de la Figura 7.29.

Tabla 7.33: Resultados de pesos con 3 clases en [20, -20]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.70	0.73	0.70	0.69
2	0.62	0.61	0.62	0.61
3	0.67	0.67	0.67	0.67
4	0.72	0.73	0.72	0.72
5	0.72	0.73	0.72	0.72
6	0.70	0.69	0.70	0.69
7	0.72	0.56	0.54	0.54
8	0.74	0.74	0.74	0.74
9	0.74	**0.76**	**0.74**	**0.74**
10	0.57	0.57	0.56	0.54
Promedio	0.69	0.68	0.67	0.67

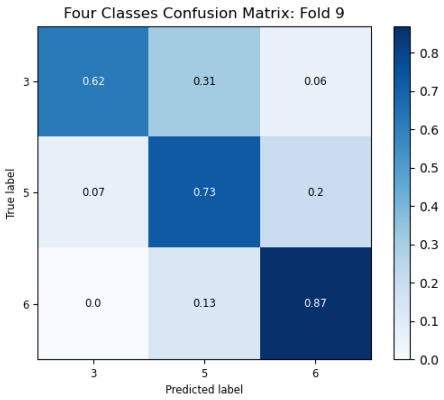


Figura 7.30: Matriz de confusión de experimento con 3 clases con pesos dentro de [20, -20]

7.6.1.4. Intervalo [30, -30]

En la Tabla 7.34 se muestra un máximo desempeño de 0.88, 0.87, 0.87 de precisión macro, *recall* y **F-score**, con un promedio de 0.63, 0.61 y 0.61, respectivamente. Por otro lado, en la matriz de confusión de la Figura 7.31 se sigue manteniendo la clasificación de las tres clases.

Tabla 7.34: Resultados de pesos con 3 clases en [30, -30]. El mejor resultado se remarca en **negritas**.

Doblez	**Precisión micro**	**Precisión macro**	**Recall**	**F-score**
1	0.87	**0.88**	**0.87**	**0.87**
2	0.64	0.50	0.48	0.48
3	0.70	0.54	0.52	0.53
4	0.67	0.52	0.51	0.51
5	0.72	0.77	0.72	0.71
6	0.57	0.36	0.34	0.35
7	0.72	0.77	0.72	0.71
8	0.72	0.72	0.72	0.71
9	0.83	0.83	0.83	0.83
10	0.50	0.40	0.37	0.38
Promedio	0.69	0.63	0.61	0.61

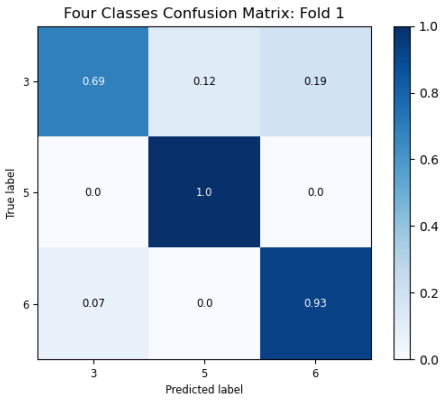


Figura 7.31: Matriz de confusión de experimento con 3 clases con pesos dentro de [30, -30]

7.6.1.5. Conclusiones afinación de pesos con 3 clases

La afinación de pesos con tres clases muestra un mejor desempeño al usar un intervalo de [10, -10]. Lo anterior debido a que el conjunto de datos contiene las clases más balanceadas del conjunto original, los requisitos de rendimiento, seguridad y usabilidad son clasificados de forma más consistente en comparación con otros experimentos con mayor número de clases, además de alcanzar una precisión promedio de 0.69.

7.6.2. Afinación de probabilidad de mutación con 3 clases

7.6.2.1. Probabilidad de mutación de 0.2 usando 3 clases

El desempeño usando un porcentaje de mutación de pesos de 0.2 es máximo en el tercer doblez con una precisión macro de 0.77, *recall* de 0.76 y *F-score* de 0.76, con promedios de 0.68, 0.66 y 0.66, respectivamente (Tabla 7.35). La matriz de confusión de la Figura 7.32 presenta la clasificación de todas las clases del conjunto de datos.

Tabla 7.35: Resultados de probabilidad de mutación de 0.2 con 3 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.70	0.72	0.70	0.69
2	0.68	0.69	0.68	0.68
3	0.76	**0.77**	**0.76**	**0.76**
4	0.70	0.69	0.70	0.69
5	0.65	0.51	0.49	0.49
6	0.65	0.68	0.65	0.65
7	0.67	0.69	0.67	0.66
8	0.67	0.70	0.68	0.67
9	0.67	0.72	0.68	0.67
10	0.63	0.62	0.63	0.62
Promedio	0.68	0.68	0.66	0.66

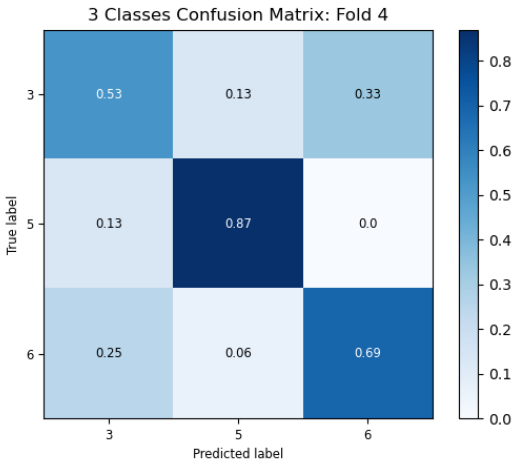


Figura 7.32: Matriz de confusión de experimento con 3 clases con mutación de pesos de 0.2

7.6.2.2. Probabilidad de mutación de 0.4 usando 3 clases

El desempeño usando una probabilidad de mutación de 0.4 muestra máximos de 0.87 en precisión macro, *recall* y *F-score* en el cuarto doblez, con promedio de 0.68 y 0.66 (Tabla 7.36). A su vez, la matriz de confusión reporta que se siguen clasificando las tres clases con la observación que la clase 5) seguridad es la mejor clasificada alcanzando un valor de 1.0 (Figura 7.36).

Tabla 7.36: Resultados de probabilidad de mutación de 0.4 con 3 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.79	0.61	0.59	0.59
2	0.72	0.56	0.54	0.55
3	0.74	0.74	0.74	0.74
4	0.87	**0.87**	**0.87**	**0.87**
5	0.78	0.79	0.79	0.78
6	0.70	0.73	0.70	0.70
7	0.67	0.74	0.67	0.67
8	0.63	0.63	0.63	0.63
9	0.80	0.81	0.81	0.80
10	0.61	0.60	0.61	0.60
Promedio	0.68	0.68	0.66	0.66

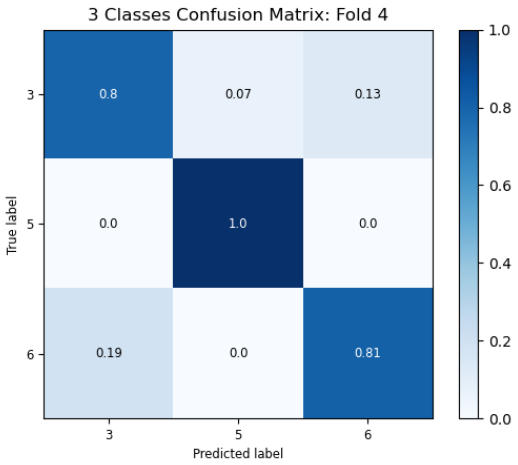


Figura 7.33: Matriz de confusión de experimento con 3 clases con mutación de pesos de 0.4

7.6.2.3. Probabilidad de mutación de 0.6 usando 3 clases

El desempeño de la clasificación usando una probabilidad de mutación de pesos de 0.6 tiene su máximo en el noveno doblez, con una precisión macro, *recall* y *F-score* de 0.83, 0.83 y 0.82, con promedios de 0.73, 0.72 y 0.72, respectivamente (Tabla 7.37). La matriz de confusión de la Figura 7.34 muestra que la clasificación de las tres clases se da con valores altos, destacando la clase 6) usabilidad con un valor de 1.0.

Tabla 7.37: Resultados de probabilidad de mutación de 0.6 con 3 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.79	0.81	0.79	0.79
2	0.74	0.74	0.74	0.73
3	0.72	0.72	0.72	0.72
4	0.78	0.80	0.79	0.78
5	0.65	0.66	0.65	0.65
6	0.74	0.76	0.74	0.74
7	0.78	0.60	0.59	0.59
8	0.76	0.77	0.76	0.75
9	0.83	**0.83**	**0.83**	**0.82**
10	0.61	0.64	0.61	0.60
Promedio	0.74	0.73	0.72	0.72

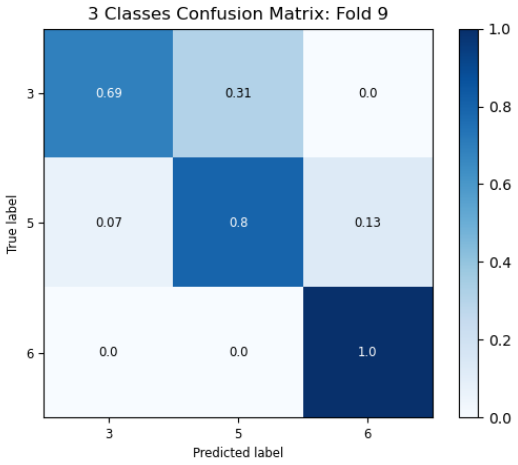


Figura 7.34: Matriz de confusión de experimento con 3 clases con mutación de pesos de 0.6

7.6.2.4. Probabilidad de mutación de 0.8 usando 3 clases

El desempeño usando una probabilidad de mutación de 0.8 tiene su máximo en el primer doblez con 0.78, 0.77 y 0.75 de precisión macro, *recall* y *F-score* (Tabla 7.38). Sin embargo, el promedio de las métricas de este experimento no logra superar a los que la probabilidad de 0.6 presenta (Tabla 7.37). La matriz de confusión de la Figura 7.35 muestra la clasificación de las tres clases, con la clase 5) seguridad siendo la mejor clasificada.

Tabla 7.38: Resultados de probabilidad de mutación de 0.8 con 3 clases. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.77	**0.78**	**0.77**	**0.75**
2	0.66	0.68	0.66	0.66
3	0.65	0.54	0.49	0.51
4	0.72	0.72	0.72	0.71
5	0.72	0.72	0.72	0.72
6	0.63	0.66	0.63	0.63
7	0.59	0.59	0.59	0.59
8	0.70	0.69	0.70	0.69
9	0.70	0.70	0.70	0.70
10	0.65	0.66	0.65	0.65
Promedio	0.68	0.67	0.66	0.66

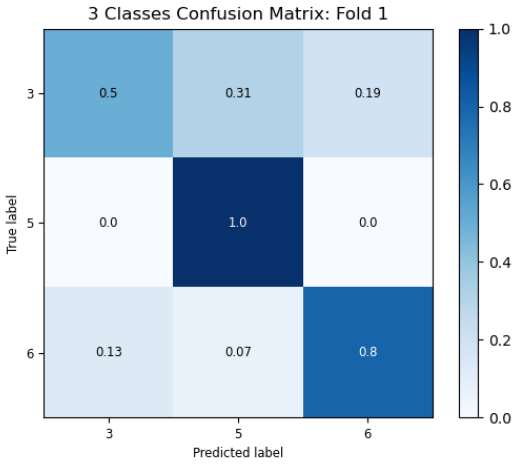


Figura 7.35: Matriz de confusión de experimento con 3 clases con mutación de pesos de 0.8

7.6.2.5. Conclusiones de afinación de probabilidad de mutación con 4 clases

En este experimento las probabilidades de mutación de 0.2, 0.4, 0.6 y 0.8 fueron puestas a prueba para comparar el desempeño de la neuroevolución en la clasificación de requisitos usando tres clases: rendimiento, seguridad y usabilidad. El mejor resultado reportado ha sido usando un valor de 0.6 con una precisión macro de 0.73, y *recall* y *F-score* de 0.72.

7.6.3. Afinación de conexiones iniciales con 3 clases

Dado que los experimentos realizados usando tres clases han iniciado con conexiones completas iniciales, en esta sección se presentan los resultados del experimento de evolucionar las conexiones de las redes neuronales partiendo de conexiones inexistentes, en otras palabras, dejar que la neuroevolución agregue las conexiones.

El máximo desempeño reportado utilizando la configuración de crear las conexiones conforme se evolucionan se muestra en el noveno doblez, con un promedio 0.67, 0.60 y 0.59 de precisión macro, *recall* y *F-score* (Tabla 7.39). Aunque el resultado no logra alcanzar el promedio reportado en la Tabla 7.37, sus valores son cercanos. A su vez, la matriz de confusión de la Figura 7.36, indica que la clasificación de los requisitos no excluye ninguna clase.

Tabla 7.39: Resultados de neuroevolución con 3 clases sin conexiones iniciales. El mejor resultado se remarca en **negritas**.

Doblez	Precisión micro	Precisión macro	Recall	F-score
1	0.68	0.72	0.68	0.66
2	0.43	0.49	0.42	0.37
3	0.70	0.72	0.70	0.70
4	0.54	0.59	0.55	0.54
5	0.52	0.57	0.52	0.52
6	0.65	0.76	0.66	0.65
7	0.63	0.73	0.63	0.61
8	0.57	0.71	0.57	0.54
9	0.74	**0.75**	**0.74**	**0.73**
10	0.57	0.63	0.57	0.56
Promedio	0.60	0.67	0.60	0.59

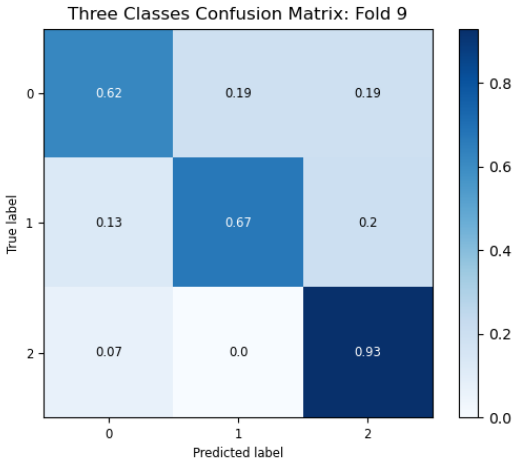


Figura 7.36: Matriz de confusión de neuroevolución con 3 clases sin conexiones iniciales

7.6.4. Conclusiones de afinación de parámetros con 3 clases

Los parámetros finales de la afinación del proceso de neuroevolución del conjunto de datos usando 3 clases tiene como parámetros finales los mostrados en la Tabla 7.40.

Tabla 7.40: Parámetros finales de afinación con 5 clases

Parámetro	Valor
Intervalo de pesos	[10, -10]
Probabilidad de mutación	0.60
Conexiones iniciales	Completa

Capítulo 8

Conclusiones y trabajo futuro

8.1. Reporte de resultados

Utilizando el trabajo previo de Pérez-Verdejo y cols. (2021), en este trabajo se toma el conjunto de datos utilizado para experimentar con un enfoque diferente, el de la neuroevolución. El objetivo es demostrar que la neuroevolución puede llegar a tener resultados competitivos con respecto a una red neuronal diseñada por el humano. A lo largo de los experimentos realizados, la red neuronal diseñada por el humano ha sido superior en los resultados mostrados, mientras que la neuroevolución ha estado por debajo de ella. En este capitulo se analizan y comparan los resultados obtenidos a lo largo de esta investigación.

8.1.1. Red neuronal diseñada y el esquema neuroevolutivo

En el capitulo 4, se realizaron tanto el diseño de la red neuronal por el humano y el esquema neuroevolutivo que ayuda a generar las redes neuronales usando un algoritmo evolutivo. En el primer caso se implementa el algoritmo descrito por Gershenson (2003), el cual consiste en ejecutar primero la red hacia adelante haciendo uso del algoritmo *feedforward* con el fin de llegar a un resultado, para después ejecutar el algoritmo de *backpropagation* y así medir qué tan lejos se está de los resultados esperados y actualizar los parámetros dentro de la red neuronal. En el segundo y principal enfoque de este trabajo, la neuroevolución, se genera una población de redes neuronales que con cada generación irá evolucionando para lograr crear una red neuronal que, en nuestro caso, clasifica los requisitos de calidad.

Como se mencionó anteriormente, la red neuronal diseñada por el humano superó a la obtenida vía neuroevolución al clasificar los requisitos de software utilizando las siete clases del conjunto de datos. La red neuronal diseñada por el humano alcanzó una precisión micro promedio de 0.7 mientras que la obtenida por neuroevolución un 0.59. Aunque se esperaba que la neuroevolución compitiera con el resultado de la red neuronal diseñada por el humano, esta última logra competir con el resultado usando evolución diferencial reportada por Pérez-Verdejo y cols. (2021), que reporta un valor de 0.69.

El parámetro más influyente en ambos enfoques fue la función de activación, ya que al utilizar ReLu y sigmoide, las redes neuronales llegaban a mostrar resultados inferiores a los de la tangente

hiperbólica. Seguido de eso, la cantidad de neuronas ocultas para la red neuronal diseñada por el humano debe ser un número bajo de neuronas y de capas.

8.1.2. Experimentación con datos filtrados

Después de los resultados a favor de la red neuronal diseñada a mano, se tomó la iniciativa de realizar una selección de características de los requisitos para verificar si los atributos de los datos estaban influyendo de forma negativa en la clasificación. Por ello, se seleccionaron cuatro técnicas de selección de características que entran en el grupo de las llamadas *filter-based* (basadas en filtro, en Español), las cuales fueron: CFS, FCBF, MRMR y Relief.

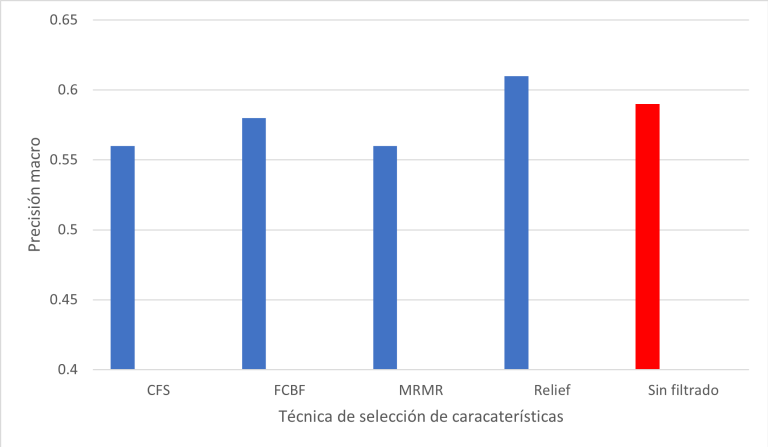


Figura 8.1: Precisión macro de uso de los conjuntos de datos filtrados en la red neuronal diseñada por el humano

La aplicación de dichas técnicas redujo el número de atributos de los datos en su mayoría, a excepción de Relief, el cual mantiene 104 de las 148 atributos de cada requisito. Como resultado, el mismo Relief obtuvo el mejor desempeño al emplear su conjunto de datos resultante en la red neuronal diseñada por el humano, incluso mejorando sus resultados originales (Figura 8.1). En contraste, para el caso de la red generada vía neuroevolución, los resultados fueron peores (Figura 8.2). Las demás técnicas de selección de características no mostraron una mejora al emplear sus datos en los clasificadores.

Gracias a estos experimentos, se observó la robustez de la red neuronal para clasificar los requisitos y que el conjunto de datos original es el más adecuado para la neuroevolución.

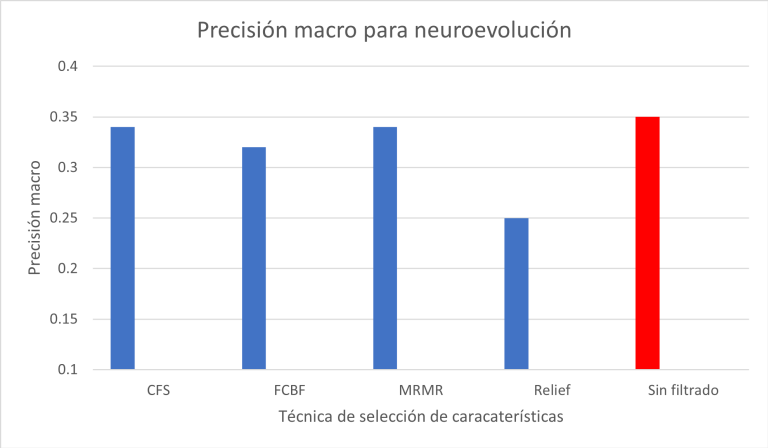


Figura 8.2: Precisión macro de uso de los conjuntos de datos filtrados en la red generada vía neuroevolución

8.1.3. Experimentación con datos balanceados

Considerando que la selección de características no benefició a la red neuronal obtenida por neuroevolución, se realizaron variantes del conjunto de datos en los que se excluyeron las clases que causan desbalance en los datos, dando como resultado cuatro conjuntos de datos que incluyen de seis a tres clases.

Con este experimento se logró observar cómo la neuroevolución mejora su resultado significativamente cuando los datos están balanceados, pues al llegar a usar solo las tres clases que están balanceadas dentro del conjunto de datos, se logra un aumento importante en su desempeño que se va aproximando a los resultados de la red neuronal diseñada por el humano. Por otro lado, la red neuronal diseñada por el humano mostró mayor robustez al conjunto desbalanceado de datos (Figura 6.5).

8.1.4. Afinación de parámetros de conjuntos de datos balanceados para neuroevolución

Una vez analizado el desempeño de las dos redes neuronales con los conjuntos de datos balanceados, se buscó mejorar el esquema neuroevolutivo, esta vez ajustando los parámetros por cada conjunto para evaluar la capacidad de la red obtenida con neuroevolución para obtener resultados similares a los de la red neuronal diseñada por el humano.

Se experimentó con tres principales parámetros: los pesos, la probabilidad de mutación de los pesos y el tipo de conexión con la cual la neuroevolución comienza a generar su población. La Figura 8.3 muestra que el desempeño de la afinación de parámetros por conjunto de datos mejora con respecto al esquema inicial conforme se excluyen las clases que causan desbalance, a tal punto que la precisión macro de la red evolucionada se aproxima a la registrada por la red neuronal diseñada por el humano.

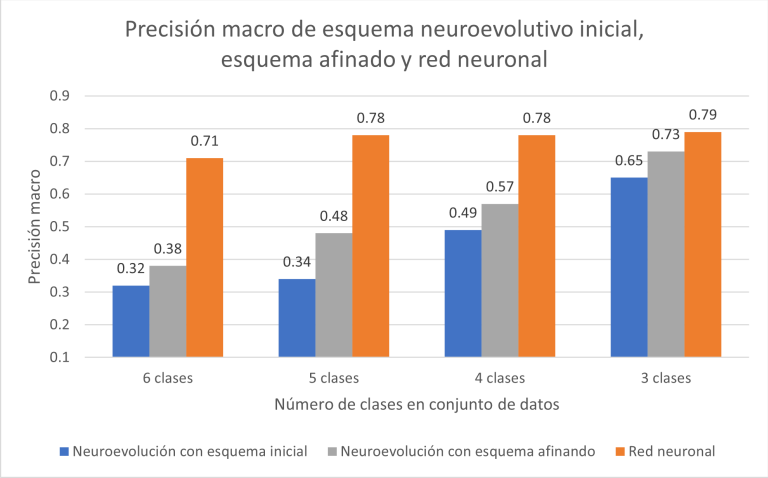


Figura 8.3: Precisión macro de uso de los conjuntos de datos filtrados en neuroevolución

8.1.5. El conjunto de datos

Con la realización de los experimentos con clases balanceadas, se concluye al observar el comportamiento de ambas redes neuronales que el conjunto de datos utilizado representa un obstáculo para la clasificación de los requisitos dejando a los clasificadores estancados por debajo del 0.8 de precisión macro promedio en la clasificación de tres o más clases.

8.2. Conclusiones finales

Nuestro objetivo de demostrar que una red neuronal artificial obtenida vía neuroevolución puede competir con una red neuronal diseñada por el humano y con clasificadores optimizados mediante evolución diferencial no se validó por ahora. Se detectan dos factores principales: el conjunto de datos y el tipo de aprendizaje que la red neuronal tiene. En el primero encontramos datos no balan-

ceados que van desde los 14 a los 150 registros por clases en una población de 630 registros, lo que genera un ambiente poco estable para la neuroevolución el cual no demostró una robustez al clasificar diversos tipos de requisitos de calidad, mientras que la red neuronal diseñada por el humano se adaptó mejor a la clasificación de varias clases. Con respecto al segundo factor, la red neuronal actualiza constantemente sus parámetros con base en el cálculo de su error, lo que le permite acercarse cada vez más a una clasificación adecuada, mientras que el enfoque de neuroevolución utilizado cambia los parámetros de las redes de forma aleatoria y combina los de mejor aptitud, por lo que, al cambiar sus propiedades al azar, es más costoso encontrar una solución competitiva cuando los recursos de cómputo y tiempo son limitados, llegando incluso a quedarse estancada por varias generaciones en el mismo resultado.

Como se ha mencionado, la clasificación de todas las clases por parte de la red obtenida mediante neuroevolución queda por debajo de la que obtuvo la red neuronal diseñada por el humano. Sin embargo, al balancear el conjunto de datos, la red evolucionada muestra un aumento significativo en su precisión como se muestra en la Figura 8.3, lo que nos permite decir que, en nuestro caso, el conjunto de datos debe estar balanceado para que la neuroevolución alcance el desempeño esperado.

8.3. Trabajo futuro

Los esfuerzos por lograr demostrar la hipótesis de que la neuroevolución puede generar redes neuronales que compitan con una red neuronal diseñada por el humano pueden continuar con las propuestas a futuro que se plantean a continuación.

Dentro de los limites de este trabajo se encuentra la no modificación de la vectorización del conjunto de datos, por lo que la propuesta de vectorizar los requisitos mediante otro método diferente a TF-IDF utilizado por Pérez-Verdejo y cols. (2021) es un camino de interés para investigar.

Otra de las actividades pendientes que no pudieron realizarse por cuestiones de tiempo, es la implementación del algoritmo de retropropagación a las redes neuronales dentro del proceso neuroevolutivo. Esta propuesta surge de comentarios que se tuvieron en presentaciones de reportes de avances realizados por alumnos de posgrado del grupo de investigación en redes neuronales y algoritmos evolutivos del Instituto de Investigaciones en Inteligencia Artificial de la Universidad

Veracruzana. Finalmente, se propone la generación de datos artificiales para balancear las clases en el conjunto de datos original que, aunque los datos no tendrán una trazabilidad, ayudará medir el desempeño de la neuroevolución en la clasificación de requisitos de software con datos desbalanceados.

Referencias

Abran, A., Moor, J. W., Bourque, P., y Dupuis, R. (2004). *Guide to the software engineering body of knowledge* (2004Version ed.). IEEE Computer Society Press.

Alexander, I. F., y Beus-Dukic, L. (2009). *Discovering Requirements: How to Specify Products and Services* (1st ed.). Wiley Publishing.

Baker, C., Deng, L., Chakraborty, S., y Dehlinger, J. (2019). Automatic multi-class non-functional software requirements classification using neural networks. *Proceedings - International Computer Software and Applications Conference*, *2*, 610–615. doi: 10.1109/COMPSAC.2019.10275

Bourque, P., Fairley, R., y eds. (2014). *Guide to the software engineering body of knowledge* (Version 3.0 ed.). IEEE Computer Society. Descargado de www.swebok.org

Dick, J., Hull, M. E. C., y Jackson, K. (2017). *Requirements Engineering, 4th Edition*. Springer.

Eiben, A. E., y Smith, J. E. (2015). *Introduction to Evolutionary Computing* (2nd ed.). Springer Publishing Company, Incorporated.

Gershenson, C. (2003). Artificial Neural Networks for Beginners. , 1–8. Descargado de http://arxiv.org/abs/cs/0308031

Kurtanovic, Z., y Maalej, W. (2017). Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*, 490–495. doi: 10.1109/RE.2017.82

McCarthy, J. (2007). What is artificial intelligence.

McIntyre, A., Kallada, M., Miguel, C. G., y da Silva, C. F. (2015). *neat-python*. Github. Descargado de https://github.com/CodeReclaimers/neat-python

Pérez-Verdejo, J. M., Sánchez-García, A. J., y Ocharán-Hernández, J. O. (2020). Generalización de requisitos de software de dominio específico para la clasificación de texto.

Pérez-Verdejo, J. M., Sánchez-García, A. J., Ocharán-Hernández, J. O., y Mezura-Montes, E. (2021). *Propuesta de aplicación de aprendizaje máquina y cómputo evolutivo en la clasifiación de requisitos de calidad* .

Refaeilzadeh, P., Tang, L., y Liu, H. (2009). Cross-Validation. En L. LIU y M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (pp. 532–538). Boston, MA: Springer US. Descargado de https://doi.org/10.1007/978-0-387-39940-9_565 doi: 10.1007/978-0-387-39940-9_565

Robertson, S., y Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right* (3rd ed.). Addison-Wesley Professional.

Salman, H. E., Hammad, M., Seriai, A. D., y Al-Sbou, A. (2018). Semantic clustering of functional requirements using agglomerative hierarchical clustering. *Information (Switzerland)*, *9*(9), 1–17. doi: 10.3390/info9090222

Sánchez-Maroño, N., Alonso-Betanzos, A., y Tombilla-Sanromán, M. (2007). Filter methods for feature selection - A comparative study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4881 LNCS*, 178–187. doi: 10.1007/978-3-540-77226-2_19

Sayyad Shirabad, J., y Menzies, T. (2005). *The PROMISE Repository of Software Engineering Databases*. School of Information Technology and Engineering, University of Ottawa, Canada. Descargado de http://promise.site.uottawa.ca/SERepository

Stanley, K. O., y Miikkulainen, R. (2002). Efficient evolution of neural network topologies. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, *2*(figure 1), 1757–1762. doi: 10.1109/CEC.2002.1004508

Storn, R., y Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341–359.

Vikhar, P. A. (2017). Evolutionary algorithms: A critical review and its future prospects. *Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC 2016*, 261–265. doi: 10.1109/ICGTSPICC.2016.7955308

Wiegers, K. E., y Beatty, J. (2013). *Software Requirements 3*. USA: Microsoft Press.

Winkler, J., y Vogelsang, A. (2017). Automatic classification of requirements based on convolutio-

nal neural networks. *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference Workshops, REW 2016*, 39–45. doi: 10.1109/REW.2016.16

Yu, X., y Gen, M. (2012). *Introduction to evolutionary algorithms*. Springer Publishing Company, Incorporated.