Implementación de una interfaz software para establecer grupos de riesgo de pacientes sospechosos de Tuberculosis Pulmonar

Lenny Tatiana Silva Soche

Facultad de Ingeniería Mecánica, Electrónica y Biomédica
Programa de Ingeniería Biomédica
Universidad Antonio Nariño
Bogotá, Colombia
2022

Implementación de una interfaz software para establecer grupos de riesgo de pacientes sospechosos de Tuberculosis Pulmonar

Lenny Tatiana Silva Soche

Trabajo Integral de Grado presentado como requisito para optar al título de:
Ingeniero Biomédico

Director:
Andrés Leonardo Jutinico Alarcón Ph.D
Codirector:
Álvaro David Orjuela Cañon Ph.D

Línea de Investigación:
Algoritmos de aprendizaje automático
Programación - Interfaz software
Grupo de investigación en Bioinstrumentación, Control, Inteligencia Computacional y Energías Alternativas (GIBIO)

Universidad Antonio Nariño
Facultad de Ingeniería Mecánica, Electrónica y Biomédica
Programa de Ingeniería Biomédica
Bogotá, Colombia
2022

Nota de Aceptación

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado 1

_____

Firma del Jurado 2

Bogotá D.C. _____

Agradezco a Dios por darme salud y fortaleza para afrontar todos los obstaculos que se me presentaron en el camino, a mis padres y a mi novio que con su amor me ayudaron a lograr esta gran meta de mi vida.

Agradecimientos

Agradezco a Dios porque me dio la fuerza y el potencial para poder lograr todo reto y dificultad con salud y vida. A mis padres Hugo Orlando Silva y Luz Marina Soche que creyeron en mi y que con su infinito amor, paciencia y apoyo esto no hubiera sido posible. A mi novio Ciro Rafael que siempre me alentó y nunca me permitió rendirme y a mi familia por estar siempre presente. Gracias a ustedes pude secar mis lágrimas y al otro día continuar con la frente en alto. En mi formación como ingeniera biomédica estuve acompañada de grandes personas como lo fueron mis maestros ingenieros y mis compañeros con los cuales compartí el mismo sueño y enfrentamos muchas adversidades juntos. Gracias comunidad UAN y facultad FIMEB que hiceron parte de este proceso de formación donde cada uno aportó un granito de arena no solo en mi vida profesional sino también en mi vida personal.

Resumen

La tuberculosis pulmonar (TB) es una enfermedad infecciosa que suele afectar a los pulmones, esta enfermedad es curable y prevenible, sin embargo, el retraso diagnóstico y de tratamiento puede causar la muerte. Se estima que en 2019 enfermaron de tuberculosis 10 millones de personas en todo el mundo y un total de 1.4 millones de personas murieron según la Organización Mundial de la Salud (OMS) en 2020 [1]. En Colombia, en el 2020 se enfermaron 11390 personas, de las cuales 10632 fueron casos nuevos y en promedio durante los últimos 5 años, se han presentado 1077 fallecidos por año a causa de tuberculosis, según el Sistema Nacional de Vigilancia en Salud Pública (SIVIGILA) en 2021 [2]. A nivel nacional la alta demanda de pacientes, dificulta la prioridad y calidad de atención por las Instituciones Prestadoras de Servicio de Salud (IPS). Estos problemas se originan debido a la mala gestión de recursos y procedimientos generando como consecuencia el avance de la enfermedad o letalidad por TB, en particular en regiones aisladas. El presente proyecto contribuye en la toma de decisiones de los especialistas de la salud bajo condiciones precarias a partir de un sistema automatizado, el cual consiste en una interfaz software que permite establecer el grupo de riesgo alto o bajo al que pertenecen los pacientes sospechosos de tuberculosis al ingresar una serie de características sociales, demográficas y del estado de salud del paciente. Se trabajaron tres algoritmos de aprendizaje automático previamente entrenados y validados Fuzzy C-means, K-means y Perceptron Multicapa, de los tres algoritmos se eligió la red Perceptron Multicapa para realizar la predicción del riesgo por tener la sensibilidad más alta con un valor del 95 %.

Palabras Claves: Interfaz software, Tuberculosis pulmonar, algoritmos de aprendizaje automático, Matlab.

Abstract

Pulmonary tuberculosis (TB) is an infectious disease that usually affects the lungs, this disease is curable and preventable, however, delayed diagnosis and treatment can cause death. In 2019 an estimated 10 million people worldwide fell ill with tuberculosis and a total of 1.4 million people died according to the World Health Organization (WHO) in 2020 [1]. In Colombia, in 2020 11390 people became ill, of which 10632 were new cases, and on average during the last 5 years, there have been 1077 deaths per year due to tuberculosis, according to the National Public Health Surveillance System (SIVIGILA) in 2021 [2]. At the national level the high demand for patients hinders the priority and quality of care by the Health Service Provider Institutions (IPS). These problems originate due to the poor management of resources and procedures, generating as a consequence the progression of the disease or lethality due to TB, particularly in isolated regions. This project contributes to the decision-making of health specialists under precarious conditions from an automated system, which consists of a software interface that allows establishing the high or low risk group to which suspected tuberculosis patients belong by entering a series of social, demographic and health status characteristics of the patient. Three previously trained and validated automatic learning algorithms Fuzzy C-means, K-means and Perceptron Multilayer were worked on, of the three algorithms the Perceptron Multilayer network was chosen to perform the risk prediction because it has the highest sensitivity with a value of 95 %.

Keywords: Software interface, Pulmonary tuberculosis, machine learning algorithms, Matlab.

Tabla de contenidos

Lista de Figuras

Lista de Tablas

1 Aspectos generales del proyecto

1.1. Introducción y antecedentes

La tuberculosis pulmonar (TB) es una enfermedad que se disemina por medio de partículas aerolizadas que contienen una bacteria llamada Mycobacterium tuberculosis, liberadas cuando una persona contagiada tose, estornuda o habla. Esta bacteria por lo general ataca a los pulmones (tuberculosis pulmonar) pero también puede atacar a otras partes del cuerpo como los riñones, columna vertebral y el cerebro (tuberculosis extrapulmonar)[12]. Cuando las bacterias vencen el sistema inmunitario y comienzan a multiplicarse destruyen los tejidos, creando orificios en ellos.Las personas que padecen TB pueden presentar expectoración (expulsión mediante la tos de flemas) alrededor de 21 días y esto es considerado como como síntoma de alerta, pero los síntomas que generan mayor certeza del contagio son: disminución de peso, pérdida de apetito, sudores nocturnos, fiebre, cansancio y escalofríos [13] [14].

En todo el mundo, la tuberculosis es una de las 10 principales causas de muerte y la principal causa por un único agente infeccioso (por encima del VIH/sida), se estima que en 2019 enfermaron de tuberculosis 10 millones de personas en todo el mundo: 5.6 millones de hombres, 3.2 millones de mujeres y 1.2 millones de niños según la OMS [1].En el país la tasa de incidencia de tuberculosis de todas las formas en 2020 es de 20.88 por 100 000 habitantes según el SIVIGILA [2].

Esta enfermedad tiende a presentarse y propagarse más fácilmente en las personas que viven en condiciones de pobreza, con hacinamiento, desnutrición, mala higiene personal, también cuando presentan malos hábitos como el consumo de alcohol y cigarrillo, relaciones promiscuas y drogadicción. Por último, las comorbilidades relacionadas son diabetes, cáncer, virus de inmunodeficiencia humana (VIH) y enfermedad pulmonar obstructiva crónica (EPOC) [12] [13] [15] [16].

A pesar de los avances en el manejo de la TB, la enfermedad puede terminar siendo mortal para los pacientes que tengan retraso diagnóstico y para los pacientes que ingresan a recibir tratamiento en un estadio avanzado de la enfermedad. Lo anterior, demuestra necesario el apoyo al personal médico para agilizar la tamización y el diagnóstico en un paciente de TB a través de una herramienta de cómputo aplicando inteligencia computacional [15] [17] [18][19][20].

Haciendo una revisión bibliográfica acerca de sistemas automatizados utilizados para el diagnóstico médico en diferentes partes del mundo, se encontraron proyectos que utilizan Inteligencia Computacional (IC) en el desarrollo del sistema. Dos de ellos surgieron por la necesidad de tener un sistema experto capaz de predecir las enfermedades transmisibles de manera sencilla y además que le permitiera al médico elegir un tratamiento adecuado para el paciente a través de una interfaz de usuario adecuada [21] [22]. El tercer trabajo, desarrollado en el departamento de economía de la universidad de Zimbabwe en África abordó un proyecto en el cual se predice la presencia de TB utilizando un modelo de redes neuronales artificiales (ANN), herramienta que está presente en el GWERU PROVINCIAL HOSPITAL. En este caso, este estudio intenta modelar y pronosticar los casos de tuberculosis en el país y de acuerdo a los resultados trabajar en las políticas de salud pública, por ejemplo, adquisición de medicamentos y programas de educación sanitaria sobre TB [23].

Por último, el cuarto proyecto "A Comparative Study of Tuberculosis Detection Using Deep Convolutional Neural Network" desarrollado en 2020 en Dhaka, Bangladesh, a diferencia de los anteriores proyectos trabaja con imágenes de rayos x de TB pulmonar de dos diferentes hospitales y no con síntomas para la entrada del sistema. Este proyecto consistió en probar cuatro modelos previamente entrenados de redes neuronales, DenseNet-169, MobileNet, Xception, e Inception-V3, donde compararon el rendimiento y la precisión de validación entre las cuatros para elegir el mejor [24].

1.2. Planteamiento del problema

La TB es una enfermedad infecciosa y una de las 10 principales causas de muerte en todo el mundo. En 2019, alrededor de 10 millones de personas desarrollaron tuberculosis y 1,4 millones murieron debido a su capacidad de propagación rápida cuando no es tratada oportunamente por las entidades de salud [1]. Según el Sistema Nacional de Vigilancia en Salud Pública (SIVIGILA) en 2021 se notificaron 11390 casos nuevos de Tuberculosis en el país. Dentro de los departamentos con mayor índice de casos se encuentran: Antioquia (20,4 %), Bogotá (8,6 %), Cali (7,7 %), Santander (5,0 %), Barranquilla (4,6 %) y Valle del Cauca (4,5 %). En promedio en los últimos 5 años, se han presentado 1077 fallecidos por año a causa de tuberculosis en Colombia. Siendo estos aproximadamente 85 % de los casos pertenecientes con TB pulmonar y el restante a TB extra pulmonar [2].

Por causa de este alto número de contagios, las personas que presentan síntomas respiratorios sospechosos o la enfermedad latente de TB son de gran magnitud, así mismo, la demanda de los servicios de diagnóstico y tratamiento. El problema consiste en que debido a que en los centros de salud no se atiende e identifica el nivel de prioridad del paciente a tiempo, las personas que inicialmente ingresan al sistema con síntomas, desarrollan la enfermedad

de manera avanzada. Así mismo, debido a la inadecuada priorización de los pacientes, las personas que presentan la enfermedad latente podrían recibir el servicio cuando ya no hay mucho que hacer por su recuperación [25]. Lo anterior implica el contagio a un porcentaje mayor de personas, que el número de casos fatales aumente y que los hospitales deban invertir más en hospitalización de tercer nivel, en particular, en casos de pacientes que pudieron haberse recuperado con servicios de primer nivel o en casa [18].

No obstante, cabe señalar que Colombia actualmente sigue un protocolo para el diagnóstico de TB activa, en el cual se realizan pruebas de laboratorio entre las más comunes: prueba de tuberculina (PPD), prueba de interferón gamma (IGRA), baciloscopia, rayos X de tórax, cultivo, y expert [26]. Sin embargo estos procedimientos requieren una gran inversión en infraestructura para el uso de laboratorios adecuados, herramientas e insumos. Además, es necesario contar con el personal capacitado para cada procedimiento, por esta razón muchos de los territorios de bajos recursos no tienen acceso a estos servicios de calidad. En Colombia se evidencia insuficiencia en la infraestructura hospitalaria, inclusive en las ciudades que están un poco más dotadas, lo que refleja la negligencia por parte del estado. Adicionalmente, en la literatura consultada, se resaltan casos en los que han ocurrido falsos positivos en los resultados de laboratorio por falta de la correcta operación humana o gestión de operaciones. Lo anterior conlleva a que el porcentaje final de diagnosticados sea muy pequeño comparado con el número de personas que pueden tener la infección latente [15] [18] [25].

Como se mencionó anteriormente, existen herramientas tecnológicas que apoyan la toma de decisiones del personal de la salud en diferentes campos de la medicina. Sin embargo, no existen suficientes desarrollos tecnológicos para la detección y el análisis de TB pulmonar en Colombia, en especial que pueda ser utilizada en regiones alejadas de las grandes ciudades. Lo anterior constituye un reto para solucionar este problema de salud pública, que puede ser apoyado desde la ingeniería biomédica, electrónica y de software, entre otras.

Por lo tanto en este proyecto se ha desarrollado una interfaz software, que permite al usuario (personal de la salud) determinar si el paciente se encuentra en un riesgo bajo o alto de tener TB y así determinar la prioridad y la urgencia del tratamiento. Es importante resaltar que esta herramienta se ha desarrollado con base en la información de pacientes sospechosos de tuberculosis pulmonar suministrada por la subred integrada de servicios de salud centro oriente (Subred CO). Lo anterior dado que este proyecto se ha desarrollado en el marco del proyecto de investigación “Generación de modelos alternativos basados en inteligencia computacional para tamización y diagnóstico de Tuberculosis pulmonar”, el cual es un proyecto liderado por la UAN con colaboración de la Subred CO y la universidad del Rosario, y financiado por MINCIENCIAS.

1.3. Justificación

La población contagiada por TB en Colombia está en aumento, de manera que desencadena una serie de eventos perjudiciales directamente sobre las personas que la padecen, su entorno social y las entidades prestadoras de servicios de salud más cercanas. Enfatizando en lo anterior, en las IPS la lista de posibles personas contagiadas en espera de un diagnóstico y tratamiento inmediato es larga, debido a que en el país no hay muchos recursos para mejorar la gestión en estos aspectos, la enfermedad se transmite o pasa a un estado avanzado.

Por consiguiente con la interfaz gráfica software, que permite establecer el grupo de riesgo de TB en un paciente, el personal de la salud podrá concebir una idea del estado de salud del paciente de manera más rápida y su nivel de prioridad en atención. Por consiguiente el paciente disminuiría el nivel de inquietud, el servicio de atención al paciente en la IPS aumentaría su calidad y los gastos en pruebas innecesarias para ambas partes disminuirían.

En respuesta a estas dificultades, es importante que la ingeniería le apueste a técnicas de inteligencia computacional en el área de salud ya que a nivel nacional la IC no se ha trabajado hasta el presente año de manera exhaustiva, puesto que en la búsqueda bibliográfica realizada se encontraron pocos trabajos desarrollados en Colombia. Por otro lado, si se encontraron entidades que desarrollan proyectos relacionados en otros países: el hospital Distrito de Gweru (África), el Departamento de Ciencias de la Computación en la Universidad de Haripur (Pakistán), la Universidad King Fahd de Petróleo y Minerales en (Arabia) y la Universidad Politécnica de Duhok en Iraq. Todos los mencionados, utilizan técnicas de IC e interfaces software para la predicción temprana de enfermedades y mejora en los diagnósticos médicos [23] [24] [25] [27].

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar una interfaz software que permita establecer el grupo de riesgo al que pertenecen los pacientes sospechosos de tuberculosis pulmonar de la Subred CO.

1.4.2. Objetivos específicos

- Determinar las variables de entrada de la interfaz a partir de información proporcionada por profesionales en la salud.
- Diseñar la interfaz, centrada en las necesidades del personal de la salud.

- Implementar la interfaz a partir del uso de algoritmos validados para la determinación de los grupos de riesgo y el empleo del software Matlab.
- Evaluar y validar la interfaz desarrollada a partir de reuniones con profesionales en salud.

2 Marco Teórico

2.1. El sistema respiratorio.

La principal función del sistema respiratorio consiste en llevar oxígeno a todas las células del cuerpo y recoger el dióxido de carbono para ser eliminado afuera del cuerpo con ayuda del sistema cardiovascular. Cuando el aire inhalado es llevado hasta los alveolos a través de las vías aéreas, se realiza el intercambio de gases entre los alvéolos y los capilares sanguíneos pulmonares por medio de una membrana (alveolo-capilar)[28].

El sistema respiratorio está conformado por las vías aéreas y los pulmones. Las vías aéreas se clasifican en la vía aérea superior e inferior. En la primera vía se encuentran la nariz, la faringe y la laringe, en la segunda vía se encuentran la tráquea, los bronquios, los bronquiolos y los pulmones donde se realiza el intercambio de gases a través de los alveolos como se puede ver en la Figura **2-1** [28].

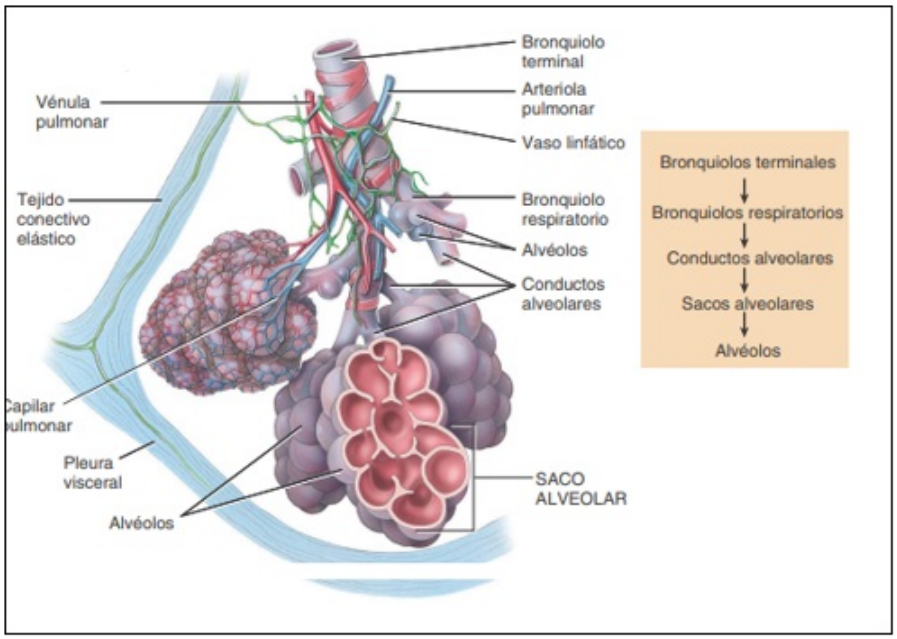


Figura 2-1: Bronquios y alveolos.Tomado de [3]

2.2. El sistema inmunitario.

Este sistema es el encargado de la defensa del cuerpo en contra de patógenos es decir microorganismos causantes de enfermedades como virus y bacterias presentes en el medio interno y externo del cuerpo. La inmunidad o resistencia es la capacidad de protegerse de las lesiones o de las enfermedades por medio de las propias defensas, mientras que la vulnerabilidad o la falta de resistencia se denomina susceptibilidad.La inminudad se clasifica en inmunidad innata e inmunidad inespecifica [3].

2.2.1. Inmunidad innata (inespecífica).

Es la primera respuesta del cuerpo cuando hay presencia de un patógeno no especifico, la defensa del cuerpo actúa para impedir el alojamiento del patógeno en las células del cuerpo y eliminar a los que si lo lograron. Los componentes de la inmunidad se clasifican enlos de la primera línea y los de la segunda línea. Entre los componentes de primera línea de defensa se encuentran las mucosas, los pelos, los cilios, el aparato lagrimal, la saliva, la orina, el vomito y la defecación.Entre los componentes de segunda línea se encuentran las células natural killer (NK), los fagocitos, la fiebre e inflamación [3].

2.2.2. Inmunidad especifica (adaptativa).

Cuando un microorganismo sobrevive a las barreras de la inmunidad innata, el cuerpo activa las defensas en contra de un microorganismo en específico. El sistema encargado de este tipo de inmunidad es el linfático, el cual se compone de una red de órganos, la linfa, los capilares, los vasos linfáticos y los ganglios; la linfa es un líquido compuesto por lípidos, glóbulos blancos y proteínas que es transportada a través de los capilares linfáticos a todo el cuerpo, que luego se agruparan en vasos linfáticos y desembocaran en lo ganglios, los cuales contienen linfocitos para combatir cualquier tipo infección[29].

2.3. Tuberculosis pulmonar.

La tuberculosis en una enfermedad infecciosa potencialmente grave que afecta principalmente a los pulmones. La bacteria responsable de este padecimiento recibe el nombre de Mycobacterium tuberculosis, la cual se transfiere cuando una persona infectada emite gotas de saliva al aire por medio de un estornudo o tos y son inhaladas por una persona susceptible.

2.3.1. Fisiopatología

- **Mycobacterium tuberculosis:**

  La Mycobacterium tuberculosis es una bacteria con aspecto de bacilo recto y alargado que mide alrededor de 0.4 x 5 μm como se puede ver en la Figura **2-2**. La bacteria se asienta en áreas con alto contenido de oxígeno y flujo sanguíneo elevado ya que es de tipo aerobio obligado. Su replicación se realiza durante 12 horas o más dentro de los fagosomas de los macrófagos alveolares. Su pared celular se compone de un alto contenido de lípidos, proteínas y polisacáridos. Además, es altamente hidrofóbica lo que la hace altamente permeable y resistente a la mayoría de antibióticos, ácidos y alcoholes, por el contrario el punto vulnerable de la bacteria es el calor (mayor a 65°C), los rayos ultravioletas y los rayos del sol [30] [6] [31].

Figura 2-2: Aspecto de la bacteria Mycobacterium tuberculosis.Tomado de [4].

- **Tuberculosis primaria:**

  Cuando las gotas de saliva infectadas ingresan a un cuerpo a través de la inhalación, en primer lugar, llegan a las vías aéreas superiores donde actúa la inmunidad innata evitando que la bacteria ingrese a las vías áreas bajas a través de un barrido ciliar producido por las células de la mucosa, pero por lo general el menor al 10 % de ellas logran llegar hasta los alveolos ya que es la zona con más alto contenido de oxígeno para vivir.Cuando el cuerpo reconoce el patógeno envía otros mecanismos de defensa como lo son la fiebre, la inflamación de la zona y la acción de los macrófagos. Pero como la Mycobacterium tuberculosis muere solo si la temperatura es mayor a los 65°C y se reproduce dentro de los macrófagos no es posible detenerla. Entonces los macrófagos piden ayuda a los mecanismos de la inmunidad especifica e inmediatamente entran a actuar los linfocitos T y los ganglios linfáticos rodeando la zona del pulmón infectada por la bacteria y volviendo de ella una zona anaeróbica. Lamentablemente la forma de atacar los linfocitos T a la bacteria es destruyendo el tejido, pudiendo llegar hasta una necrosis por calcificación tomando por nombre granuloma o tubérculo y para el caso

de los ganglios, complejo de Gohn [31].

- **Tuberculosis latente:**

  Una vez la bacteria de tuberculosis fue controlada por el sistema inmune del cuerpo después de haber ingresado por primera vez, esta podrá permanecer dormida en el cuerpo sin presentar síntomas molestias o contagio, sólo si el sistema inmune de la persona se encuentra fortalecido.

- **Tuberculosis activa o reactivación:**

  Si la persona no tiene un sistema inmune fuerte posiblemente por malas condiciones de vida, hábitos o padece de enfermedades crónicas su sistema no va a ser capaz de detener la acción de la tuberculosis, empezará a multiplicarse y a generar efectos en el cuerpo, es decir a presentar síntomas de la enfermedad.Algunos de los casos más frecuentes en los que se puede presentar esta reactivación son: En hospedadores infectados con VIH, individuos con inmunodepresión, leucemia, receptores de trasplante o que reciben fármacos como etanercept, infliximab o corticoesteroides y aquellos con enfermedades crónicas selectas como diabetes e insuficiencia renal crónica [6].

2.3.2. Manifestaciones clínicas

- **Signos y síntomas**[32] [33] [34]:

  Tos: Expulsión brusca y ruidosa del aire contenido en los pulmones producida por la irritación de las vías respiratorias o para mantener el aire de los pulmones limpio de sustancias extrañas.

  Expectoración: Expulsión mediante la tos de las flemas u otras secreciones formadas en las vías respiratorias.

  Hemoptisis: Expectoración de sangre proveniente de los pulmones o los bronquios causada por alguna lesión de las vías respiratorias.

  Fiebre: aumento temporal en la temperatura del cuerpo.

  Escalofrió: Sensación de frío intensa y repentina acompañada de un ligero temblor del cuerpo, generalmente producida por un cambio brusco de temperatura.

Diaforesis: Sudoración abundante.

Pérdida de peso: pérdida de peso regularmente no intencional.

Antecedentes de enfermedades crónicas [6]:

Diabetes: La diabetes consiste en un problema de metabolismo, en el cual la glucosa que se encuentra en la sangre no puede ser aprovechada por las células debido a la baja producción de insulina en el páncreas. Entonces la debilidad y susceptibilidad podrían aparecer cuando las células no obtienen suficiente glucosa.

Cáncer: El cáncer inicia cuando ocurre un alteración biológica y genética de las células que conforman los tejidos y estas empiezan a multiplicarse sin control hasta superar el número de células sanas, devastando los tejidos cercanos o trasladándose a través de los vasos sanguíneos o el sistema linfático a otras partes del organismo.

La susceptibilidad en este caso puede ser causa por la enfermedad en si misma y por los tratamientos como los son las quimioterapias, inmunoterapia, trasplantes de medula ósea entre otros en los cuales se destruyen gran cantidad de glóbulos blancos durante su proceso [35].

VIH (Virus de inmunodeficiencia humana): Daña el sistema inmunitario al destruir un tipo de glóbulo blanco que ayuda al cuerpo a combatir las infecciones.

Enfermedad Renal Crónica: El cuerpo disminuye su capacidad para eliminar los desechos y excesos de agua a causa de un daño que han sufrido los riñones y su capacidad para filtrar la sangre. La persona en este caso es propensa a adquirir infecciones con facilidad ya que al padecer la enfermedad renal no puede convertir suficiente vitamina D a su forma activa (calcitriol), por lo que los niveles de hormona paratiroidea pueden aumentar implicando una disminución en la densidad de los huesos y generando complicaciones en la producción de linfocitos[36].

Antecedentes de uso de corticoides (prednisolona, dexametasona, prednisona) Los corticoides son hormonas similares a las que producen las glándulas suprarrenales las cuales son usadas para combatir el estrés relacionado con enfermedades y traumatismos. La principal función de este tipo de hormonas es suprimir la inflamación

y el sistema inmunitario para enfermedades en las cuales el sistema inmunológico ataca a las células sanas [37].

- **Datos de laboratorio** [32] [6] [31]:

PPD (prueba cutánea de proteína purificada derivativa): Esta prueba también es conocida con el nombre de Test de Mantoux, consiste en aplicar de manera intradérmica 0.1 ml de derivado proteínico de Tuberculosis en el antebrazo. Si el cuerpo inyectado ha estado expuesto al Mycobacterium tuberculosis anteriormente, se generará una reacción en la piel aproximadamente 48 a 72 horas después como respuesta al reconocimiento de la bacteria por parte del sistema inmune.La reacción de la piel se presenta como una induración, es decir, endurecimiento de la piel la cual se medirá su diámetro con una regla en mm o unidades de tuberculina, la prueba se puede encontrar positiva a partir de los 5mm según las condiciones del sujeto.

Test de liberación de Interferón Gamma (IGRA): Prueba en sangre que mide la producción de interferón gamma (citocina) por los linfocitos T como respuesta a la estimulación con antígenos específicos al complejo Mycobacterium tuberculosis.

Microscopia de frotis de esputo y cultivo de esputo: Mediante esta técnica se debe recoger en ayunas una prueba de esputo (flemas generadas en el pulmón) en un recipiente para ser llevada al laboratorio. En el laboratorio se realiza en primer lugar una tinción de Gram mediante la cual se puede observar a través de un microscopio que la muestra no se encuentre contaminada con saliva u otro tipo de material procedente de la vía respiratoria, de no ser así se procede a realizar la tinción de Ziehl-Neelsen sobre la muestra permitiendo detectar las bacterias de M. tuberculosis al tomar un color morado sobre un azul de fondo por tener la propiedad de ser acido alcoholes resistentes.Una vez identificada la bacteria objetivo, se cultiva en condiciones ambientales apropiadas para su desarrollo durante varias semanas y se analiza sus reacciones a ciertos medicamentos u otros aspectos. (ver Figura **2-3**).

Radiografía torácica: La radiografía se utiliza para identificar la enfermedad en personas con síntomas pulmonares o en personas con prueba PPD positiva. En las imágenes diagnosticas se pueden observar opacidades parenquimatosas (tejido pulmonar opaco), cavitaciones (zona definida por perdida del tejido pulmonar y ocupada por otro elemento que no sea aire) como se puede ver en la Figura **2-4**, granulomas o nódulos de Ghon en caso de ser positivo a la infección por Mycobacterium Tuberculosis [33].

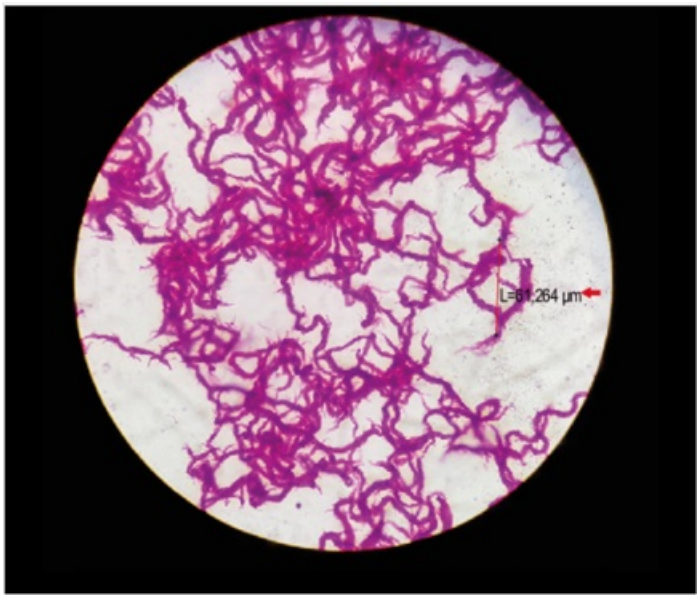


Figura 2-3: Identificación de la Mycobacterium tuberculosis por medio del cultivo de esputo.Tomado de [5]

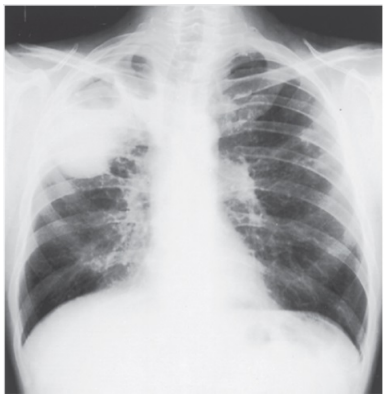


Figura 2-4: Tuberculosis cavitaria del lóbulo superior derecho.Tomado de [6].

Ensayo molecular (GeneXpert): El ensayo GeneXpert es una técnica moderna y costosa en la cual se identifica el Complejo Mycobacterium tuberculosis y detecta las mutaciones más frecuentes en el gen rpo asociadas a resistencia a rifampicina (RIF), directo de muestras de pacientes con síntomas de tuberculosis, en menos de dos horas. Esta técnica puede emplearse para diagnosticar TB en diversas muestras de origen pulmonar y extrapulmonar por ejemplo liquido cefalorraquídeo o liquido pleural.La plataforma consiste en un sistema cerrado de biología molecular que utiliza cartuchos, los cuales funcionan como un mini laboratorio de biología molecular, en cuyo interior se realizan todos los pasos para realizar la PCR en tiempo real: la liberación del ADN, la combinación con los reactivos, la amplificación y la detección a través de fluores-

cencia liberada por sondas específicas. Para ello, el cuerpo del cartucho se encuentra segmentado en distintas cámaras, las que contienen todas las soluciones y reactivos necesarios para la realización de la técnica. Como se puede ver en la figura **2-5** [5].

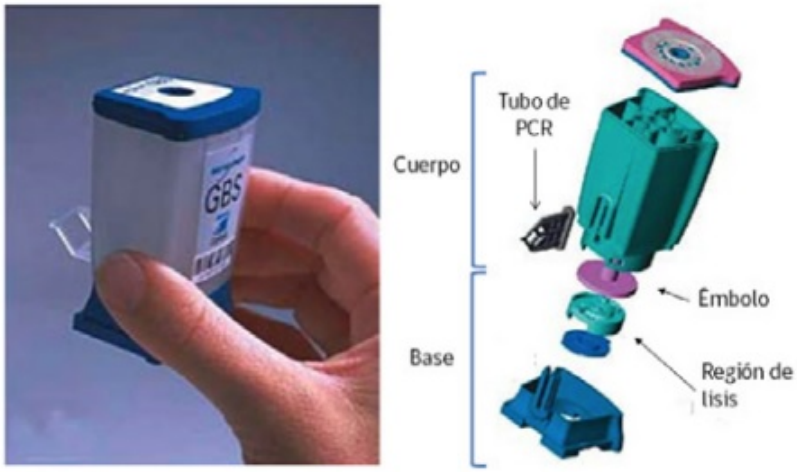


Figura 2-5: Ensayo molecular (GeneXpert).Tomado de [5]

2.3.3. Tratamiento

Cuando un sujeto no ha pasado por pruebas de laboratorio o imágenes diagnósticas y solo existe la sospecha de tuberculosis se administran antibióticos con base empírica para neumonía, se hospitaliza al paciente en una cama de aislamiento y se inician los análisis de laboratorio. Una vez se ha descubierto la presencia de tuberculosis en el sujeto, se procede a tratar la enfermedad con la ingesta de múltiples fármacos durante un tiempo prolongado. Se ha presentado el caso en el cual la persona infectada es resistente a los diferentes fármacos asignados en primera instancia por los médicos, por ello deben tomar otro tipo de tratamiento con medicamentos de segunda línea [6] [33]. Entre los medicamentos para tratar la Tuberculosis están:

Fármacos de primera línea: Isoniazida, Rifampicina, Rifapentina, Etambutol, Pirazinamida, Rifabutina.
Fármacos de segunda línea: Cicloserina, Etionamida, Fluoroquinolonas, Estreptomicina, Amikacina, Capreomicina.
Los pacientes se clasifican según su resistencia farmacológica en:

Tuberculosis farmacorresistente: Resistentes a la isoniazida o rifampicida.
Tuberculosis resistente a múltiples fármacos: Resistentes a la isoniazida y rifampicida.

Tuberculosis con resistencia farmacológica extensa: Resistentes isoniazida, rifampicida, fluoroquinolonas, aminoglucósidos o capreomicina.

Tratamiento para tuberculosis latente: Dentro de este grupo están todos aquellos que dieron positivo a la prueba PPD, pero son asintomáticos. Los adultos, niños y pacientes VIH, deben ingerir isoniazida por 9 meses.

Tratamiento para tuberculosis activa: Ingerir Isoniazida, Rifampicina, Rifapentina, Etambutol y Pirazinamida en las primeras 8 semanas y se continua el tratamiento con la ingesta de solo dos de los medicamentos de primera línea por un tiempo de 18 a 31 semanas según la recuperación de cada paciente. Los sujetos con resistencia farmacológica deben seguir otro tipo de tratamiento según su afinidad con los medicamentos de segunda línea [34].

2.4. Interfaces de usuario.

Una interfaz de usuario es el medio por el cual podemos comunicarnos con un dispositivo electrónico. Este concepto abarca arquitectura de información, patrones y diferentes elementos visuales ya que sin ello lo único que se veria al encender el dispositivo electrónico sería un poco de líneas de programación sin saber que hacer ni a donde dirigirse como se puede ver en la Figura **2-6** [38].

Figura 2-6: Líneas de comando ocultas gracias a una interfaz. Tomado de [7].

2.4.1. Tipos de interfaz de usuario .

- **Interfaz de hardware:** Son todos aquellos dispositivos que permiten el intercambio de datos con la maquinas al ingresarlos por medio de teclas, manivelas, perillas o leyéndolos por medio de pantallas, diales, marcadores etc.

- **Interfaz de software:** Son programas que permiten manifestar las órdenes a la computadora o visualizar su respuesta.

2.4.2. Características de una interfaz de usuario .

Una interfaz de usuario debe ser clara y concisa con la información que presenta, debe ser uniforme y debe procurar emitir mensaje que causen mínima sorpresa. Además, presentar un atractivo visual y siempre guiar al usuario [38].

2.4.3. Diseño de interfaces de usuario .

Para diseñar una interfaz de usuario es necesario evaluar las habilidades, el entorno de trabajo y las expectativas de sus usuarios previstos para que los usuarios no se enfrenten con interfaces pocos atractivas e inapropiadas, con dificultad para acceder a algunas características del sistema o aparición de constantes errores, se debe cumplir con una serie de pasos previos a su implementación como se puede ver en la Figura **2-7** [8]:

Figura 2-7: Diseño de interfaces de usuario. Tomado de [8].

1. **Análisis del usuario:** Se evalúan aspectos como capacidades físicas y mentales, si es

un usuario casual o potencial, si el usuario al momento de hacer uso de la interfaz se va a encontrar en situaciones bajo presión, edad, cargo a desempeñar entre otros.

2. **Prototipado del sistema:** El prototipo es un paso importante previo a la implementación de la interfaz, el cual inicialmente debe ser plasmado sobre una hoja de papel. Esto permite que el ingeniero plasme sus ideas de diseño como número de pestañas, ventanas, colores etc y proyecte su idea a los usuarios finales para que den su aporte e indiquen las posibles mejorías a realizar desde su posición.Durante el prototipado el usuario se enfrenta ante las siguientes preguntas: ¿Cómo debe interactuar el usuario con el sistema informático? y ¿Cómo se debe presentar la información del sistema informático al usuario? En el libro Ingenieria del software se nombran algunas posibles herramientas para [8]:

   Interacción del usuario: La información puede ser ingresada mediante manipulación directa, selección de menús, rellenado de formularios y lenguajes de comando.

   Presentación de la información: Mediante presentación numérica, presentación textual y se sugiere no usar más de 4 o 5 colores para una ventana y no más de 7 en toda la interfaz. También sugiere utilizar un solo color para una tarea en especifico por ejemplo el rojo únicamente para emitir mensajes de error.

3. **Evaluación de la interfaz :** Una vez implementada la interfaz se analiza la forma en que el usuario de desenvuelve con el sistema, si se siente a fin y si no se le dificulta su uso y por supuesto si alcanza el objetivo inicial. Esa evaluación se puede realizar mediante encuestas, entrevistas de usuarios y observaciones o comúnmente una mezcla de todas ellas y los resultados se pueden analizar a través de tareas y estudios etnográficos.

2.5. Aprendizaje automático.

El aprendizaje automático o también conocido como aprendizaje de máquinas, es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, en el que los modelos aprenden de ejemplos de datos, aprovechando la idea de ajustar parámetros en tareas de clasificación o regresión [39].

- **Tipos de aprendizaje automático.** De acuerdo al tipo de datos que se tengan, existen diferentes enfoques con los que se puede trabajar los cuales son [39]:

- Aprendizaje supervisado: Los datos de entrada se encuentran etiquetados por lo cual se conocen de antemano el grupo al que pertenecen en la salida.

- Aprendizaje semi supervisado: Algunos datos se encuentra etiquetados pero la mayoría no.

- Aprendizaje no supervisado: Los datos no tienen etiquetas por lo cual el algoritmo debe encontrar el patrón de agrupación.

- **Etapas para llevar a cabo un proyecto de aprendizaje automático:**

  Todos los proyectos deben seguir un riguroso plan para obtener al final excelentes resultados y no suponer su fracaso [40] [41], a continuación se presentan los pasos a seguir (ver Figura **2-8**):

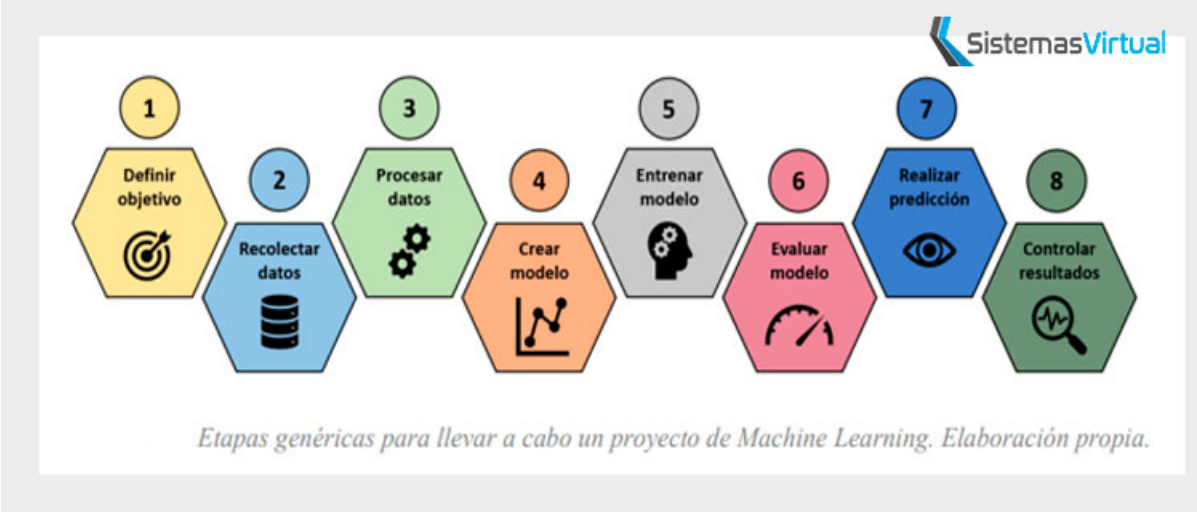


Figura 2-8: Etapas para llevar a cabo un proyecto de aprendizaje automático.Tomado de [40]

- Definición del objetivo: En esta etapa es importante identificar el problema que necesita una solución a mediano y largo plazo de acuerdo con las necesidades de la empresa y las capacidades de acuerdo con los datos disponibles.También se clasifica si el problema es supervisado o no supervisado, incluso se selecciona el tipo de modelo entrenado (regresión, clasificación, agrupamiento).
- Recopilación y preparación de datos: En esta etapa se define la cantidad y el tipo de datos necesarios al mismo tiempo que se visualiza y analiza cuáles son las variables que representan mejor aquello que queremos. Además se preparan los

datos en un formato determinado para poder ser procesados por la máquina de la manera más sencilla posible.

- Elección del modelo: Se define perfectamente el modelo que mejor se ajusta al problema: regresión lineal, árbol de decisión, red neuronal, vecino más cercano, etc. Este paso es importante pero generalmente no complicado porque los algoritmos están en bibliotecas predefinidas.

- Entrenamiento del modelo: Esta etapa consiste en suministrar la información que permita que el algoritmo de Machine Learning haga su aprendizaje inicial.

  Durante esta etapa, los datos deben coincidir por completo y contener las respuestas correctas, también conocido como el atributo de destino. De esta forma, el algoritmo de aprendizaje puede sugerir correlaciones en los datos de entrenamiento que se han especificado en los atributos de entrada, y se proporciona un modelo para almacenar esas correlaciones.

- Evaluar modelo: En siguiente paso consiste en verificar la precisión del modelo ingresando los datos de prueba, que son desconocidos para la máquina.Una precisión de 50 % no es suficiente para validar un modelo, ya que indica que la mitad de las veces fallará.

- Realizar predicción: En esta etapa se pone a prueba el modelo con el mundo real, esto puede ser a través de una simulación.Es importante ver los tipos de errores que se están presentando para modelar y cambiar esos aspectos y mejorar el rendimiento los cuales pueden ser de la siguiente manera:

  Cambiar de modelo. Puede que en un principio se haya usado un modelo de clasificación binaria y en realidad el proyecto requería un modelo de regresión.

  Aumentar los datos para nutrir el modelo.

  Diagnosticar los fallos en la definición del objetivo y entender cuáles son las posibles mejoras.

- Controlar resultados: Se verifica y asegura que el modelo cumpla con los objetivos planteados .

2.5.1. Algoritmos de agrupamiento.

El agrupamiento es una tarea de aprendizaje automático no supervisado que consiste en como su nombre lo indica, agrupar las muestras en función de sus similitudes de características[42], donde los datos dentro de un mismo grupo son muy similares o los datos en diferentes grupos son bastantes distintos.

Los algoritmos son capaces de agrupar los datos de acuerdo a un criterio en específico como lo pueden ser: la densidad, la distribución, los centroides o jerarquías. Se hará énfasis en la agrupación basada en centroides ya que bajo esta técnica operan los algoritmos k-means y fuzzy c-means que se trabajaran en el desarrollo del proyecto.

- Agrupación dura y blanda.

  Existen dos formas de agrupar datos, en la primera las características de un dato se pueden identificar únicamente con uno de los grupos, la cual se nombra agrupación dura, para el caso de la segunda agrupación el dato se puede identificar con varios grupos expresando el sentido de pertenencia en un valor numérico comprendido entre el rango [0 1] la cual se nombra agrupación blanda o Fuzzy Clustering como se puede ver en la Figura **2-9** [9] .

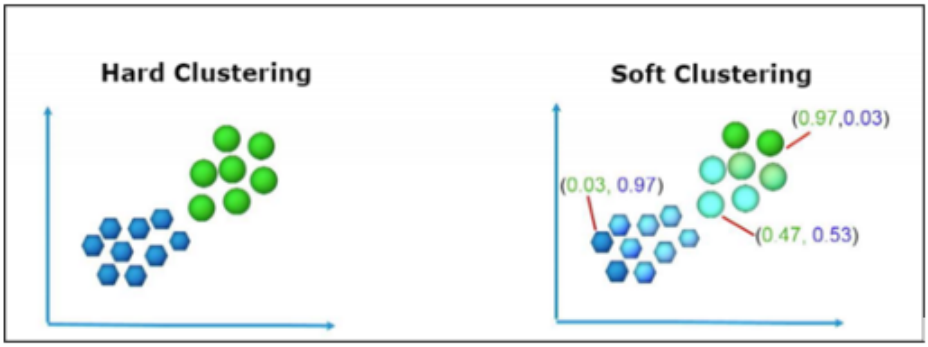


Figura 2-9: Agrupación dura y blanda. Tomado de [9].

- Técnica de agrupación basada en centroides.

  Esta técnica es utilizada por los algoritmos k-means y fuzzy c-means, la cual consiste en asignar un punto centroide en representación de cada grupo y a partir de él agrupar todos los datos. k-means y fuzzy c-means tienen la particularidad de requerir el número de grupos que se desean encontrar a la salida. El centroide se halla al promediar las distancias de características similares. El algoritmo selecciona cada dato de entrada y mide la distancia euclidiana que existe entre el dato y los n centroides, de tal forma, la

menor distancia con el grupo n indica a que grupo pertenece el dato. En la figura **2-10** se puede observar un ejemplo donde hay 3 grupos, 3 centroides y el dato elegido para la agrupación es llamado x_i, además se puede observar que la menor distancia desde el dato x_i y los tres puntos centroides es d3 por ello x_i pertenecería a este grupo [10].

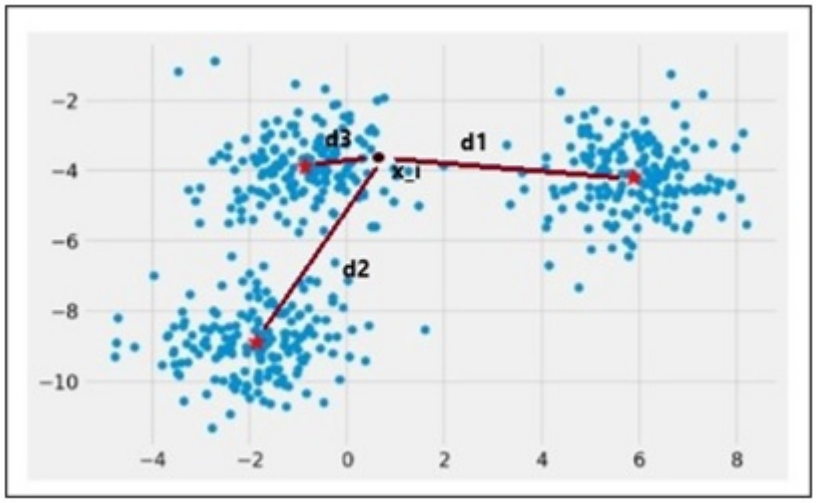


Figura 2-10: Técnica de agrupación basada en centroides. Tomado de [10].

- K-Means.

  Es un método de agrupamiento, que tiene como objetivo la partición de un conjunto de n muestras en k grupos en el que cada muestra pertenece al grupo cuyo valor medio es más cercano. El algoritmo K- means trabaja con la agrupación dura y bajo la técnica de centroides [43].

  Para agrupar las muestras en función de sus características se sigue los siguientes cuatro pasos:

  1. Elija aleatoriamente k centroides de los puntos de muestra como centros de conglomerados iniciales.
  2. Asigne cada muestra al centroide más cercano.
  3. Mover los centroides al centro de las muestras que le fueron asignadas.
  4. Repita los pasos 2 y 3 hasta que las asignaciones de grupos no cambien o se alcance una tolerancia definida por el usuario o el número máximo de iteraciones.

  Para medir la similitud entre datos se usa la distancia euclidiana al cuadrado entre dos puntos x e y en un espacio m-dimensional, teniendo en cuenta que la similitud es

inversamente proporcional a la distancia. Ver la ecuación 2-1.

$$d(x,y)^2 = \sum_{j=1}^{m}(x_j - y_j)^2 \tag{2-1}$$

- Fuzzy C-Means.

  El método de este algoritmo consiste en asignar una probabilidad para cada punto que pertenece a un grupo. Expresando la pertenencia a un grupo de una muestra x con un vector disperso de valores binarios. Esto se logra representando la similitud entre un elemento y un grupo por una función, llamada función de pertenencia, que toma valores entre cero y uno. Los valores cercanos a uno indican una mayor similitud, mientras que los cercanos a cero indican una menor similitud ver la Ecuación 2-2 [43].

$$\begin{bmatrix} \mu^{(1)} = & 0 \\ \mu^{(2)} = & 1 \\ \mu^{(3)} = & 0 \end{bmatrix} \tag{2-2}$$

Aquí, cada valor cae en el rango [0, 1] y representa una probabilidad de pertenencia al respectivo centroide del clúster. La suma de las membresías para una muestra dada es igual a 1 ver la Ecuación 2-3.

$$\begin{bmatrix} \mu^{(1)} = & 0.10 \\ \mu^{(2)} = & 0.85 \\ \mu^{(3)} = & 0.05 \end{bmatrix} \tag{2-3}$$

Podemos resumir el algoritmo FCM (Fuzzy c-means) en cuatro pasos clave:

1. Especifique el número de k centroides y asigne aleatoriamente las pertenencias a grupos para cada punto.
2. Calcule los centroides del conglomerado $\mu\hat{}$(j), j ε [1,..,k].
3. Actualice las membresías del clúster para cada punto.
4. Repita los pasos 2 y 3 hasta que los coeficientes de membresía no cambien, o se alcance la tolerancia definida por el usuario o el número máximo de iteraciones.

2.5.2. Redes neuronales artificiales.

Una red neuronal es un modelo basado en imitar el procesamiento de información de un cerebro humano. Para entrenar una red se debe trabajar con pesos y umbrales, los cuales son propiedades de las neuronas que se van ajustando hasta lograr sintonizar la red para que sus salidas estén cerca de los valores deseados. De acuerdo a la estructura de una red neuronal se puede dividir en tres capas. La capa de entrada: responsable de recibir datos del entorno externo. La capa oculta: compuesta por neuronas que se encargan de extraer patrones asociados al sistema que se está analizando. La capa de salida: compuesta por neuronas y por lo tanto es responsable de producir y presentar las salidas finales de la red [44].

- Red neuronal artificial de una sola capa.

  Esta red neuronal artificial tiene solo una capa de entrada y una sola capa de salida. En la Figura **2-11** se muestra una red de retroalimentación de capa simple compuesta de n entradas y m salidas. Dentro de este grupo se encuentran las redes Perceptron y las ADALINE, cuyos algoritmos de aprendizaje utilizados en sus procesos de entrenamiento se basan respectivamente en la regla de Hebb y la regla Delta [44].

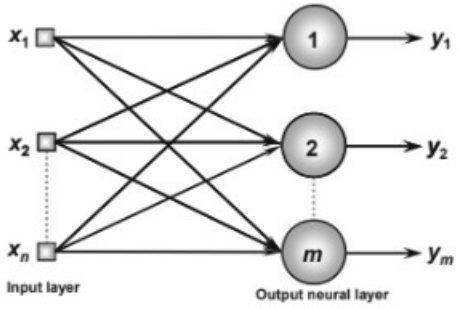


Figura 2-11: Arquitectura de una red neuronal artificial de una sola capa. Tomado de [44]

- Red neuronal artificial multicapa.

  Las redes multicapas se componen de una o más capas neuronales ocultas. En la Figura **2-12** se muestra una red con una capa de entrada con n señales de muestra, dos capas neuronales ocultas que consisten en n1 y n2 neuronas respectivamente y, finalmente, una capa neuronal de salida compuesta por m neuronas. Dentro de este grupo se encuentran las redes Perceptrón Multicapa (MLP) y la red de Función de Base Radial (RBF), cuyos algoritmos de aprendizaje utilizados en sus procesos de entrenamiento se basan respectivamente en el regla delta y la regla competitiva/delta [44].

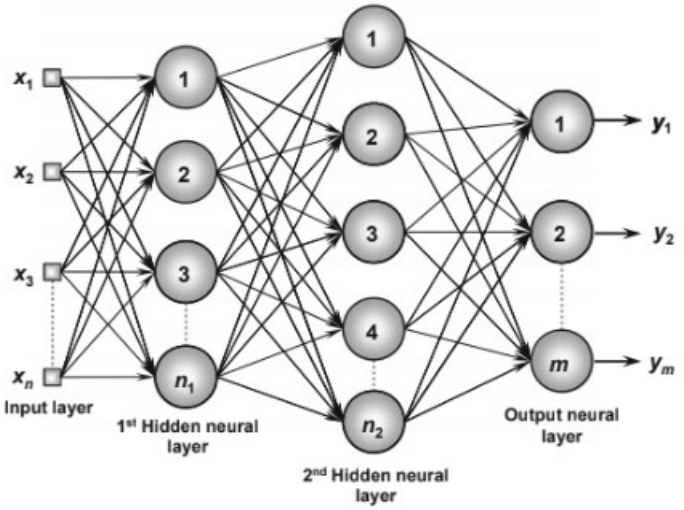


Figura 2-12: Arquitectura de una red neuronal artificial multicapa. Tomado de [44]

3 Desarrollo metodológico.

3.1. Selección de las variables de entrada de la interfaz

La interfaz utiliza datos de pacientes sospechosos de TB del hospital Santa Clara, perteneciente a la Subred CO. La información suministrada es anonimizada y corresponde a una serie de variables que describen las características sociales, demográficas y del estado de la salud de los pacientes. Los datos utilizados son seleccionados, pensando en que puedan ser adquiridos en consulta por el médico o el personal de la salud correspondiente. En particular, son datos de la etapa inicial del protocolo de TB, en la cual se solicita información de la enfermedad actual. Es importante resaltar, que la selección de las variables se ha realizado mediante la sugerencia de los expertos, médicos del hospital Santa clara, pertenecientes al personal científico del proyecto de investigación "Generación de modelos alternativos basados en inteligencia computacional para tamización y diagnóstico de Tuberculosis pulmonar". Para lo anterior, se realizaron diversas reuniones con profesores y estudiantes de Maestría que pertenecen al proyecto. Las variables seleccionadas son: Sexo, Edad, Grupo poblacional, Localidad de Bogotá en la que habita, condición VIH y si actualmente está recibiendo tratamiento antirretroviral (TAR). Teniendo como opción de respuesta para sexo, femenino o masculino; para la edad un valor numérico de 0-100; para el Grupo poblacional, habitante de calle, desplazado, migrante, población carcelaria, víctima de violencia armada, indígena, u otro; para ubicación dentro de la ciudad, las 19 localidades de Bogotá y la opción fuera de la ciudad y desconocido en caso de que no apliquen a ninguna localidad; y finalmente para las condiciones VIH y TAR las opciones de respuesta son positivo, negativo y sin dato, como se puede ver en la Tabla **3-1**.

N°	VARIABLES	VALORES
1	Sexo	Femenino, Masculino.
2	Edad	0 - 100
3	Grupo poblacional	Habitante de calle, Desplazado, Migrante,Población carcelaria, Víctima de violencia armada,Indigena, otros.
4	Localidad	Antonio Nariño, Barrios Unidos, Bosa, Chapinero, Ciudad Bolívar, Engativá, Fontibón, Kennedy, La Candelaria, Los Mártires, Puente Aranda, Rafael Uribe Uribe, San Cristóbal, Santa Fe, Suba, Teusaquillo, Tunjuelito, Usaquén, Usme, No aplica, Sin dato.
5	VIH	Positivo, Negativo, Sin dato.
6	TAR	Positivo, Negativo, Sin dato.

Tabla 3-1: Variables seleccionadas para la entrada de la interfaz.Elaboración propia.

3.2. Diseño de la interfaz centrado en las necesidades de los profesionales de la salud

El Hospital Santa Clara de Bogotá es una importante institución de carácter público asociada a la Subred CO, esta institución atiende a poblaciones con bajos recursos económicos, poblaciones desplazadas, víctimas de la violencia armada y a personas que debido a sus malos hábitos de vida son más propensos a contraer enfermedades de transmisión sexual o adicciones que comprometan su sistema inmunológico. Este hospital como muchas otras instituciones debe seguir un programa de Tuberculosis según la normatividad vigente, en el cual se debe prevenir la mortalidad de los pacientes por Tuberculosis Pulmonar con prácticas de diagnóstico y tratamiento oportuno. El plan actual que sigue la Subred CO es el siguiente: Todo aquel sujeto que se acerque al hospital con síntomas de una enfermedad respiratoria debe registrarse con su información básica y ser valorado por el personal médico, posteriormente a ello, Medicina interna realiza un examen más detallado. Si después de las anteriores valoraciones el paciente persiste con los mismos síntomas, se realizan las pruebas de laboratorio correspondientes a la microscopia de frotis de esputo, cultivo de esputo y el ensayo molecular (GenXpert) para diagnosticar al paciente y poder iniciar un tratamiento. Sin embargo los pacientes pueden iniciar su terapia antituberculosa sólo si las 3 pruebas de laboratorio son positivas, es decir, tomar aislamiento e iniciar a tomar los medicamentos respectivos para combatir la enfermedad. No obstante, la realización de estas pruebas requiere de tiempo el cual puede ocasionar que la enfermedad avance o se contagien otras personas.

De acuerdo a lo anterior la interfaz se diseño de la siguiente manera: El médico, enfermera o profesional de la salud ingresa a la interfaz las variables seleccionadas y obtenidas dentro del protocolo de TB, posteriormente al presionar un botón el podrá ver si el paciente tiene alto o bajo riesgo de tener TB, a través de una señal visual. En la figura **3-1** se muestra el diagrama de lenguaje unificado de modelado (UML) empleado para el desarrollo de la interfaz.

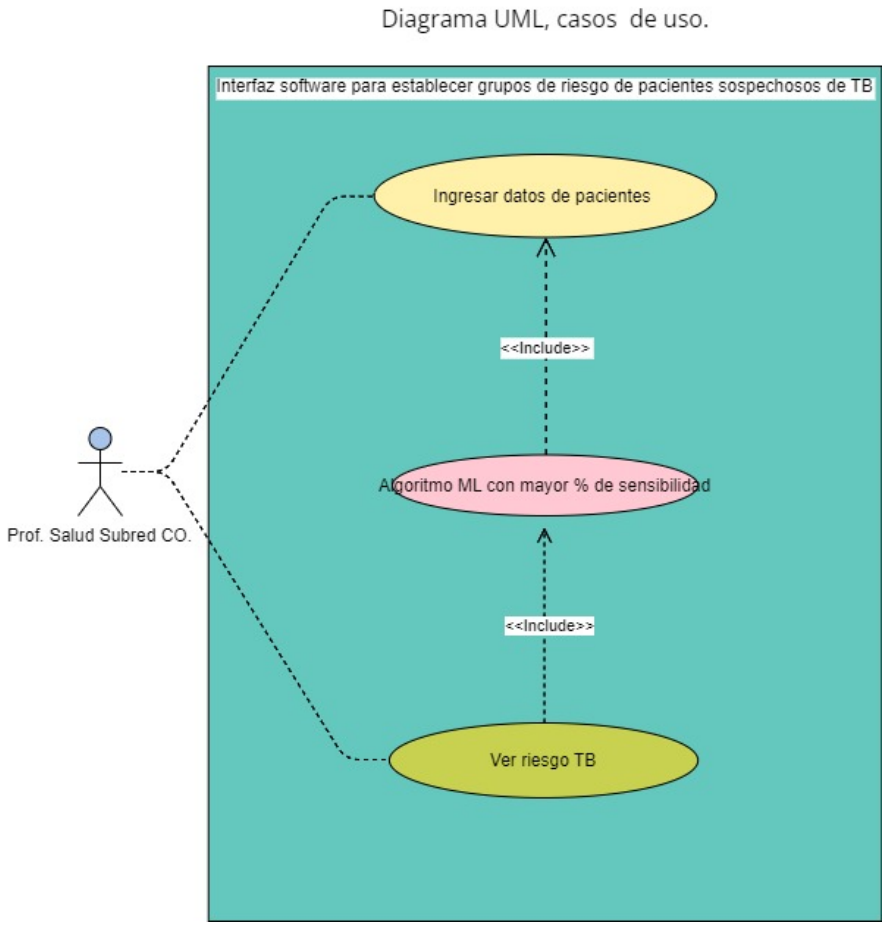


Figura 3-1: Diagrama UML, caso de uso para un profesional de la salud. Elaboración propia.

Adicionalmente, el profesional de la salud puede ver en la ventana el porcentaje de sensibilidad y especificidad de la prueba. Teniendo en cuenta que la sensibilidad es la probabilidad de que la prueba identifique como enfermo a aquél que efectivamente lo está y la especificidad es la probabilidad de que la prueba identifique como no enfermo a aquél que efectivamente no lo está.

3.3. Exploración de los posibles algoritmos de aprendizaje automático a implementar en la interfaz

3.3.1. Base de datos usada para el entrenamiento y validación de los algoritmos

Para llegar a la etapa de predicción a través de un algoritmo por aprendizaje automático, anteriormente el ingeniero Andrés Jutinico entrenó los algoritmos Fuzzy C-means, K-means y una red de perceptrón multicapa con una base de datos entregada por la subred CO de manera retrospectiva y anónima. La información fue recopilada dentro del protocolo de Tuberculosis durante un periodo de tiempo comprendido entre enero de 2017 a diciembre de 2019, en el cual 43 pacientes se acercaron al hospital en el año 2017, 74 en el año 2018 y 65 en el año 2019 (Ver Tabla**3-2**).

GRUPO	AÑO	PACIENTES SOSPECHOSOS TB
1	2017	43
2	2018	74
3	2019	65
Total		182

Tabla 3-2: Número de pacientes por año de recopilación de la base de datos usada para el entrenamiento de los algoritmos. Elaboración propia.

Para tener una mayor noción del contenido dentro de esta base de datos se obtuvieron diagramas de barras clasificadas por tipo de dato y según el año de recopilación. Como se puede ver en las Figuras **3-2** ,**3-3**, **3-4**, **3-5**, **3-6**, **3-7** y de acuerdo a ellas podemos sacar las siguientes conclusiones:

A partir de las Figuras **3-2** y **3-3** se puede observar que la población que más se acerca al hospital por tener síntomas respiratorios son adultos de 27 a 59 años de edad, siendo la mayoría hombres. La figura **3-4** muestra que la población desplazada, la población víctima de la violencia armada, migrantes y población carcelaria es casi nula durante la toma de esta base de datos, de la Figura **3-5** se puede contemplar que el lugar en el que habitan las personas que se acercan a la Subred CO es muy distribuido ya que no hay una alta frecuencia de una localidad de Bogotá en específico y que también llegan al hospital personas de las afueras y por ultimo de las Figuras **3-6** y **3-7** se puede observar que si existe la condición VIH en un alto porcentaje de la población y que muchos de ellos no inician su terapia

antirretroviral, condición que los vuelve mas vulnerables de contraer TB. Es importante resaltar que estas estadísticas nos permitieron idear el tipo de población especifico.

Figura 3-2: Sexo de la población que hace parte de la base de datos usada por los algoritmos, según el año de recopilación. Elaboración propia.

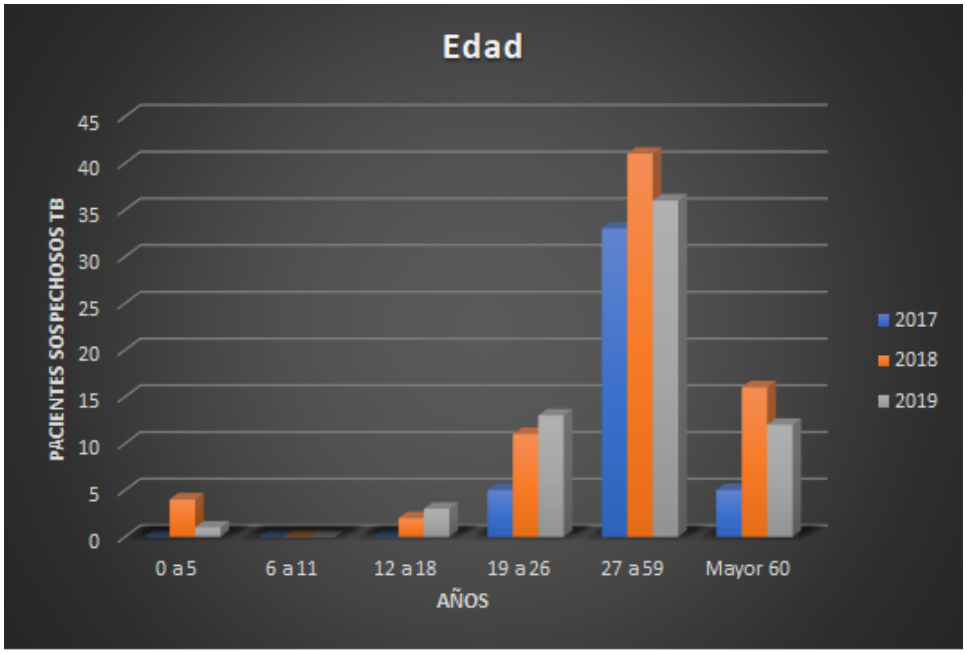


Figura 3-3: Edad de la población que hace parte de la base de datos usada por algoritmos, según el año de recopilación. Elaboración propia.

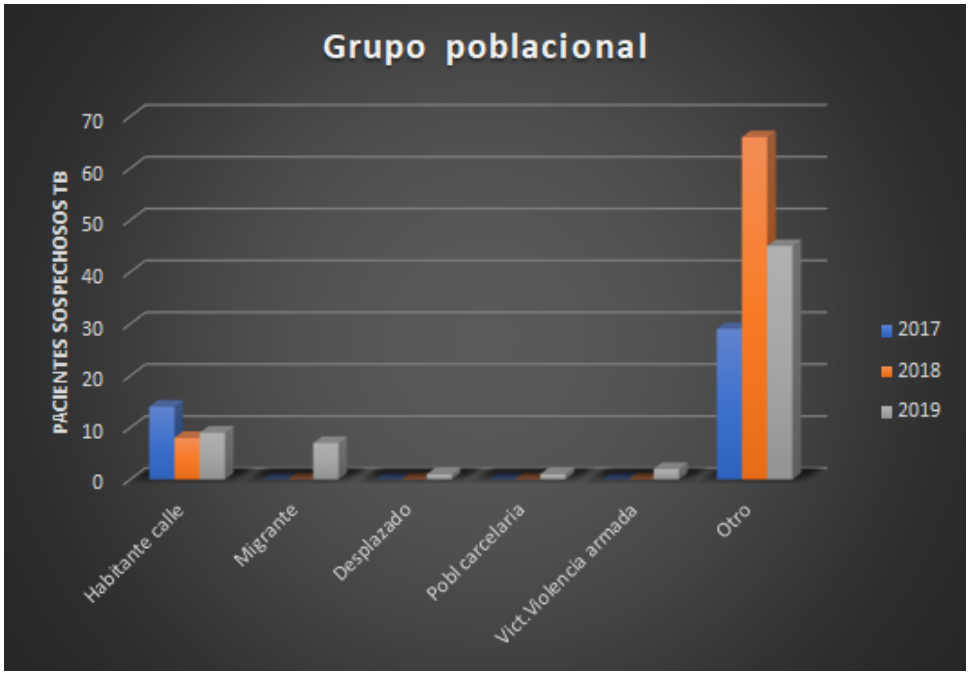


Figura 3-4: Grupos poblacionales al que pertenecen las personas de la base de datos usada por los algoritmos, según el año de recopilación. Elaboración propia.

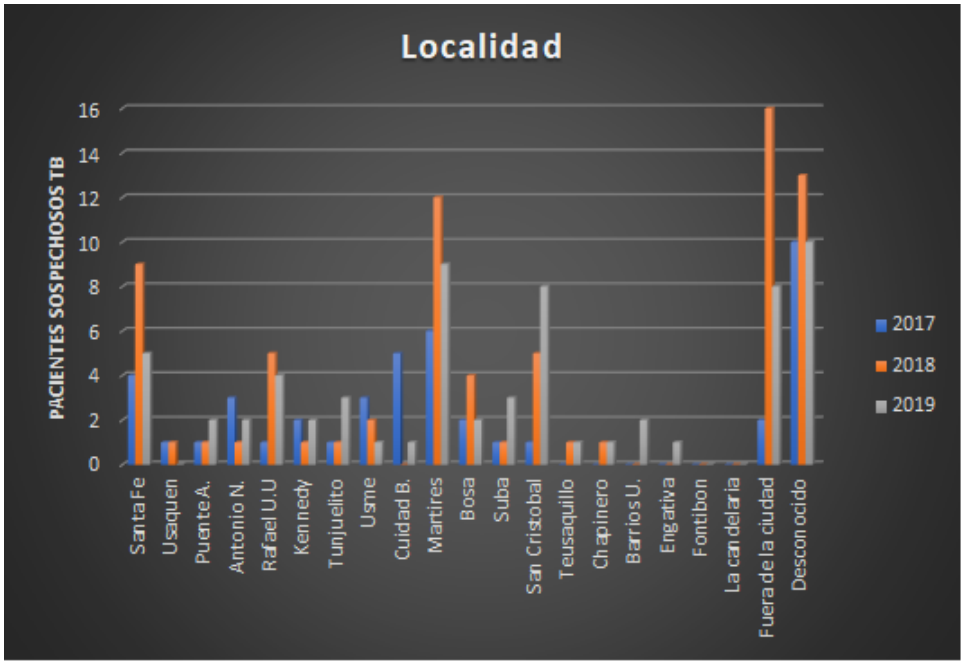


Figura 3-5: Localidades en las que habitan la población que hace parte de la base de datos usada por los algoritmos, según el año de recopilación. Elaboración propia.

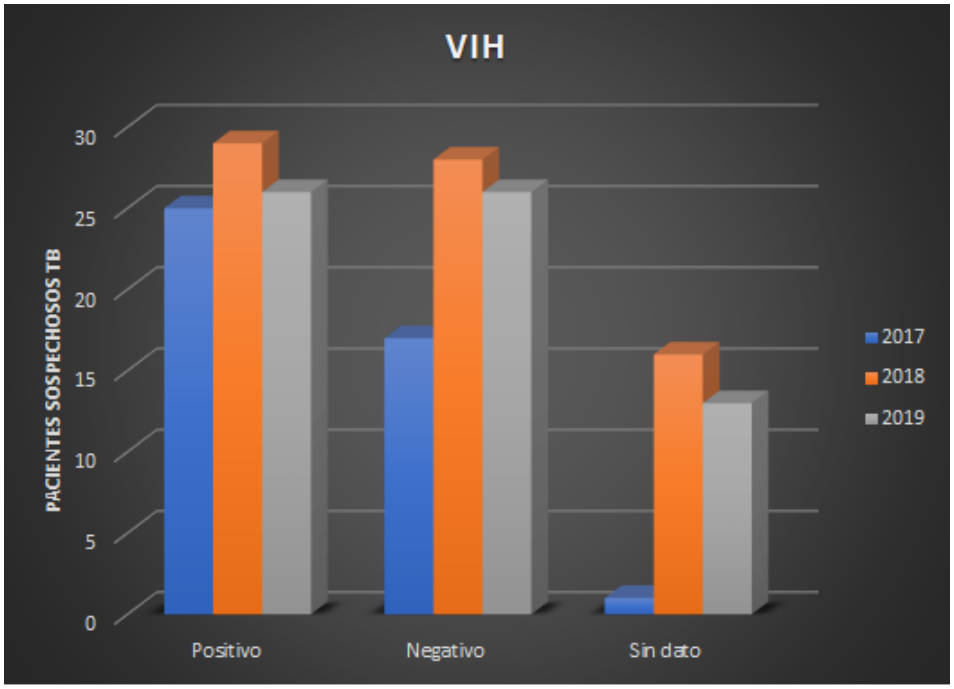


Figura 3-6: Condición VIH de la población que hace parte de la base de datos usada por los algoritmos, según el año de recopilación. Elaboración propia.

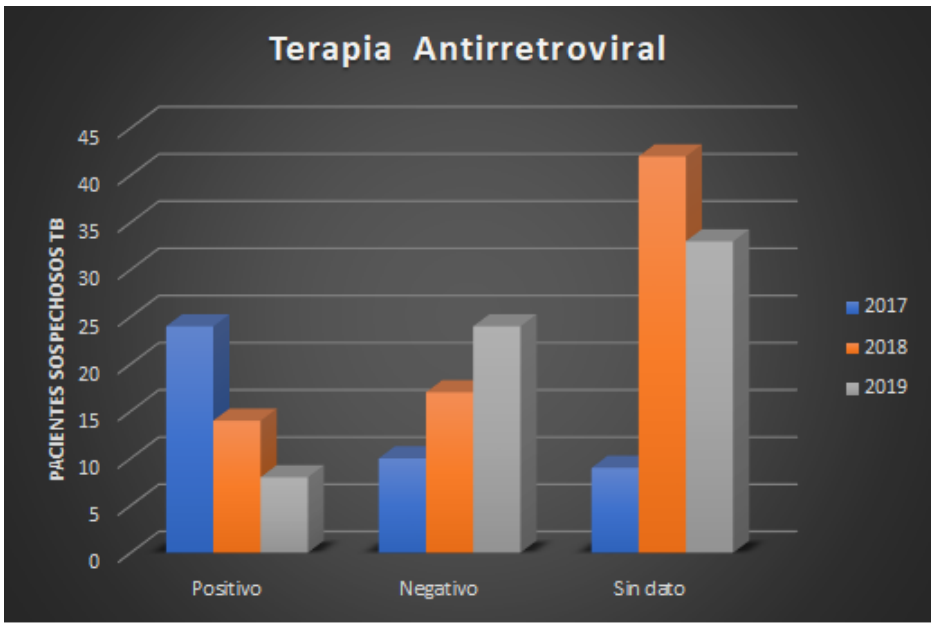


Figura 3-7: Condición TAR de la población que hace parte de la base de datos usada por los algoritmos, según el año de recopilación. Elaboración propia.

3.3.2. Selección del mejor algoritmo para realizar la predicción del riesgo de TB

En el presente proyecto se utilizaron dos tipos de algoritmos de aprendizaje automático: aprendizaje no supervisado y aprendizaje supervisado. Dentro del primer grupo se encuentran Fuzzy C-means y K-means los cuales manejan la técnica de clasificación por agrupamiento y dentro del segundo grupo se encuentra una red neuronal llamada perceptrón multicapa (Ver Tabla **3-3**).

TIPO DE ALGORITMO	ALGORITMO	TÉCNICA DE CLASIFICACIÓN	PARAMETRO A MODIFICAR
Aprendizaje no supervisado	Fuzzy C-means	Agrupamiento	N° Clúster
Aprendizaje no supervisado	K-means	Agrupamiento	N° Clúster
Aprendizaje supervisado	Perceptrón multicapa	Red neuronal artificial	N° Neuronas de la capa oculta

Tabla 3-3: Algoritmos implementados en el desarrollo del proyecto. Elaboración propia.

Para la elección del mejor algoritmo fue necesario evaluar el desempeño de cada uno, a través de las siguientes métricas: sensibilidad, especificidad y la tasa de error, las cuales permiten medir el rendimiento del algoritmo en la predicción.
Para llevar a cabo esta evaluación, se obtuvo a través de una herramienta llamada matriz de confusión la cantidad de datos que fueron correctamente clasificados y los que no lo fueron para cada algoritmo. En la Figura **3-8** se observa como se interpreta la matriz, donde cada columna representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 3-8: Matriz de Confusión.Tomado de [11].

Donde, **VP** es la cantidad de datos positivos que fueron clasificados correctamente como

positivos por el modelo. **VN** es la cantidad de datos negativos que fueron clasificados correctamente como negativos por el modelo. **FN** es la cantidad de datos positivos que fueron clasificados incorrectamente como negativos. **FP** es la cantidad de datos negativos que fueron clasificados incorrectamente como positivos. Una vez se conocen los valores de los VP, VN, FN y FP, se reemplazan en las siguientes ecuaciones:

$$Sensibilidad\,\% = \frac{VP}{FN + VP} \times 100 \tag{3-1}$$

$$Especificidad\,\% = \frac{VN}{VN + FP} \times 100 \tag{3-2}$$

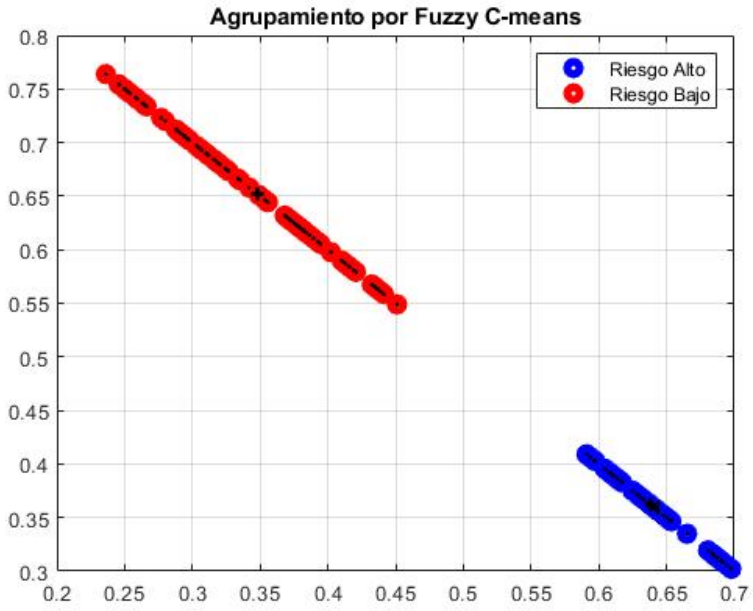
$$Error\,\% = \frac{FP + FN}{Total} \times 100 \tag{3-3}$$

En segunda instancia, se procedió a la variación de los parámetros de numero de clúster y el numero de neuronas ya que estas variaciones cambiaban significativamente las métricas a evaluar. En los algoritmos de agrupamiento se evaluó con 2 y 3 clúster, siendo 2 clúster la mejor opción.Teniendo en cuenta que 3 clúster indican riesgo alto, riesgo medio y riesgo bajo y 2 clúster indican riesgo alto y riesgo bajo. Posteriormente, según el análisis hecho a los algoritmos de agrupamiento y basados en sus resultados, para la red neuronal se usó únicamente riesgo alto y riesgo bajo, procediendo a variar únicamente el número de neuronas de la capa oculta de la red en un rango de 1 a 20, siendo el mejor resultado 9 neuronas.(Ver Tabla **3-4**).

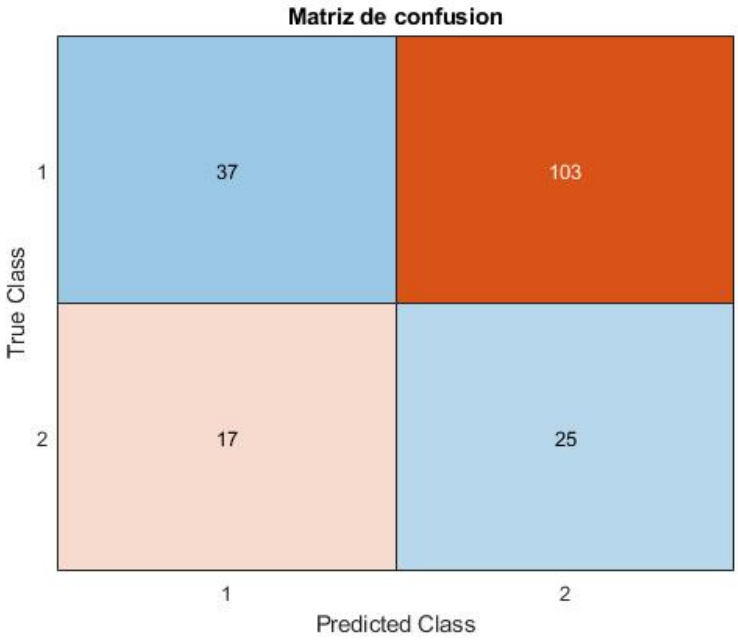
EVALUACIÓN DE LOS ALGORITMOS			
ALGORITMO	SENSIBILIDAD (%)	ESPECIFICIDAD (%)	ERROR PREDICCIÓN (%)
Fuzzy C-means 2 Clúster	26.42	59.52	34.1
Fuzzy C-means 3 Clúster	39.6	69.8	60.43
K-means 2 Clúster	26.42	60	34.1
K-means 3 Clúster	45.1	72.52	53.3
Perceptrón multicapa con 9 neuronas	95	10	17

Tabla 3-4: Evaluación del desempeño de los algoritmos. Elaboración propia.

Una vez obtenidos los resultados de los algoritmos y tal como se muestra en la tabla **3-4**, se eligió la red neuronal para predecir el riesgo de TB y usarla en la sesión de profesional de la salud de la interfaz; basados en que el 95 % de sensibilidad fue el valor más alto y de acuerdo a ello el médico podrá tener una alta confianza de que la interfaz clasificará como enfermo a quien realmente lo esta. Además, se puede observar que la tasa de error de este algoritmo es baja, razón que también nos inclino a su elección. A continuación se muestra gráficamente el desempeño de cada uno de los algoritmos, en la predicción del riesgo por TB de acuerdo al numero de clúster y el numero de neuronas a través de un plano en el espacio y la matriz de confusión. En la Figuras **3-9** y **3-11** se realiza el agrupamiento con dos clúster, en las cuales los datos que pertenecen al grupo 1 tienen riesgo alto y los datos del grupo 2 tienen riesgo bajo. En la Figuras **3-10** y **3-12** se realiza el agrupamiento con tres clúster, las cuales los datos que pertenecen al grupo 1 tienen riesgo alto, los datos que pertenecen al grupo 2 riesgo medio y los datos del grupo 3 tienen riesgo bajo. En las Figuras **3-13** y **3-14** los datos que dieron -1 en la predicción pertenecen a riesgo bajo y datos que dieron 1 en la predicción pertenecen a riesgo alto.En las Figura **3-13** se muestra los resultados de la predicción de la red neuronal en su entrenamiento, realizado con el 70 % de los datos y en la Figura **3-14** el resultado de la predicción de la red, al evaluarse con el restante 30 % de los datos.

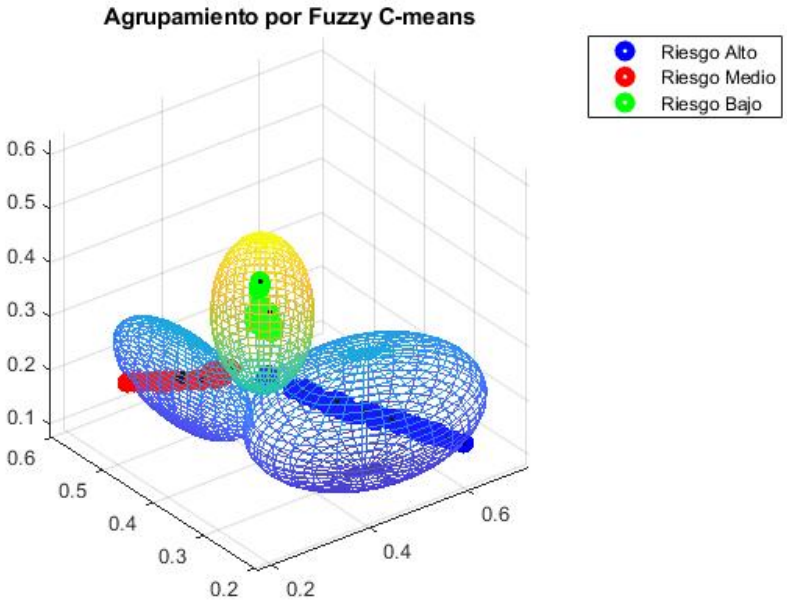


(a) Desempeño del algoritmo Fuzzy C-means con 2 clúster en un plano 2D .

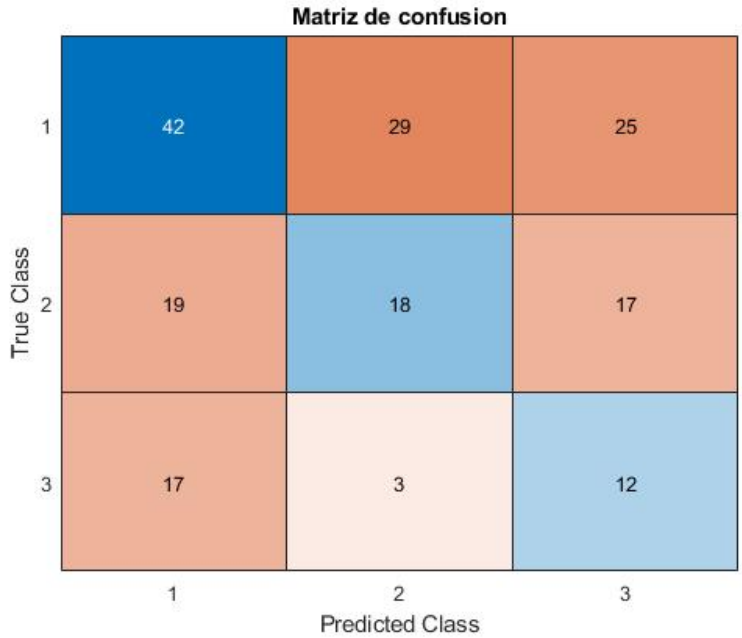


(b) Desempeño del algoritmo Fuzzy C-means con 2 clúster representado en la matriz de confusión.

Figura 3-9: Evaluación gráfica del algoritmo Fuzzy C-means con 2 clúster. Elaboración propia.

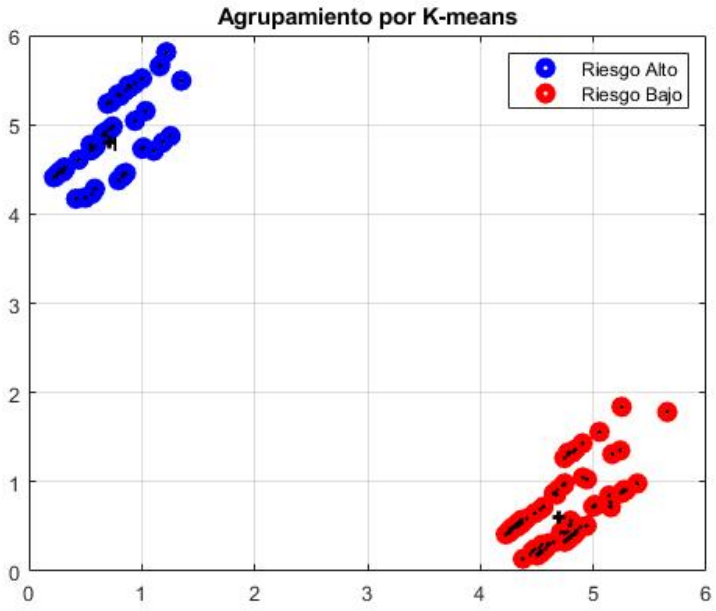


(a) Desempeño del algoritmo Fuzzy C-means con 3 clúster representado en un plano 3D

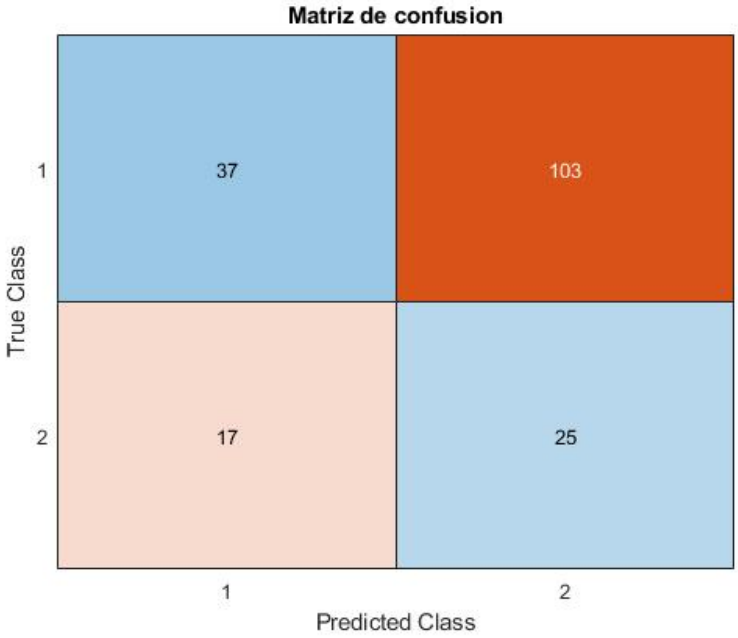


(b) Desempeño del algoritmo Fuzzy C-means con 3 clúster representado en la matriz de confusión.

Figura 3-10: Evaluación gráfica del algoritmo Fuzzy C-means con 3 clúster. Elaboración propia.

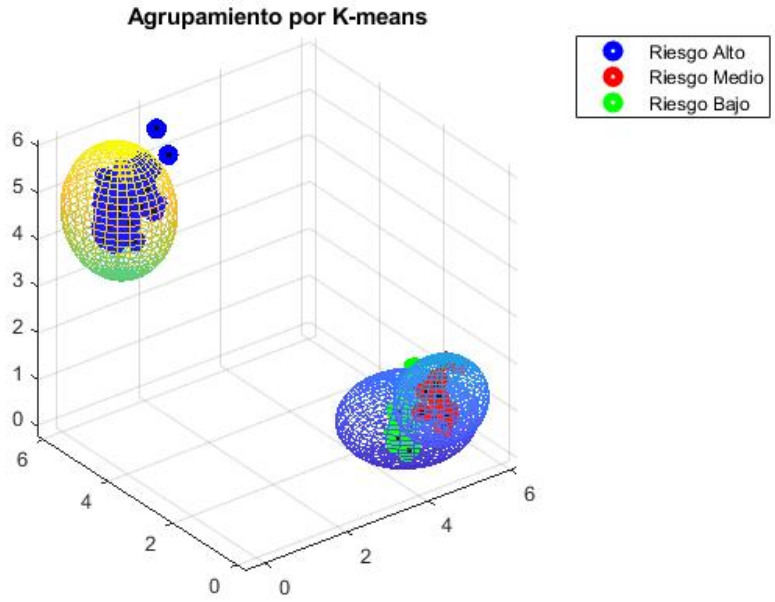


(a) Desempeño del algoritmo K-means con 2 clúster representado en un plano 2D.

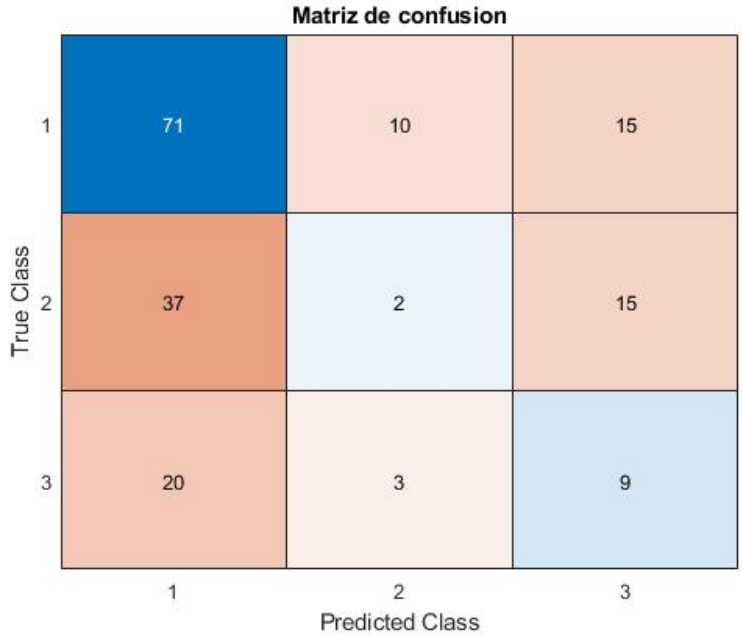


(b) Desempeño del algoritmo K-means con 2 clúster representado en la matriz de confusión.

Figura 3-11: Evaluación gráfica del algoritmo K-means con 2 clúster. Elaboración propia.



(a) Desempeño del algoritmo K-means con 3 clúster representado en un plano 3D



(b) Desempeño del algoritmo K-means con 3 clúster representado en la matriz de confusión.

Figura 3-12: Evaluación gráfica del algoritmo K-means con 3 clúster. Elaboración propia.

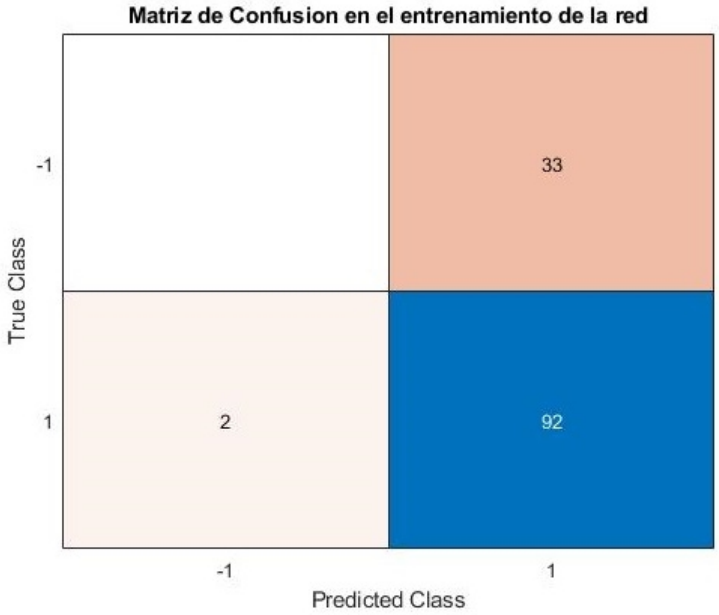


Figura 3-13: Predicción obtenida por la red neuronal, al ser entrenada con el 70 % de la base de datos. Elaboración propia.

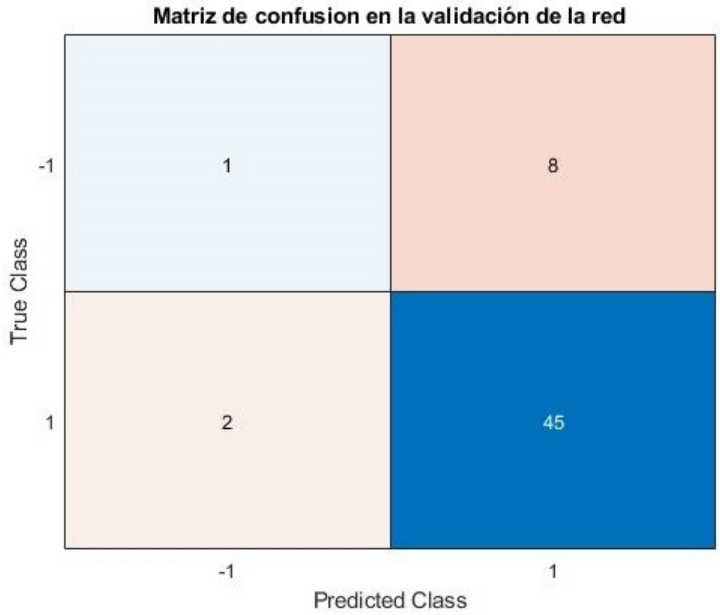


Figura 3-14: Predicción obtenida por la red neuronal, al ser validada con el 30 % de la base de datos. Elaboración propia.

3.3.3. Opción administrador anexada con fin educativo

Con fin de poder explicar el proceso desarrollado durante este proyecto de manera más lúdica y amigable a la comunidad educativa y el personal científico del proyecto, se creo una opción dentro de la interfaz llamada administrador. En la cual se puede importar la base de datos de la Subred CO y observarse de manera organizada en una tabla. Una vez cargada la base de datos el usuario podrá seleccionar el numero de cluster o el numero de neuronas según como lo desee y con solo presionar un botón el puede observar el comportamiento de los algoritmos Fuzzy C-means, K-means y de la red neuronal a través de las herramientas de evaluación en plano 3D, plano 2D y su respectiva matriz de confusión. En la Figura **3-15** se puede ver los dos casos de uso elaborados para la interfaz .

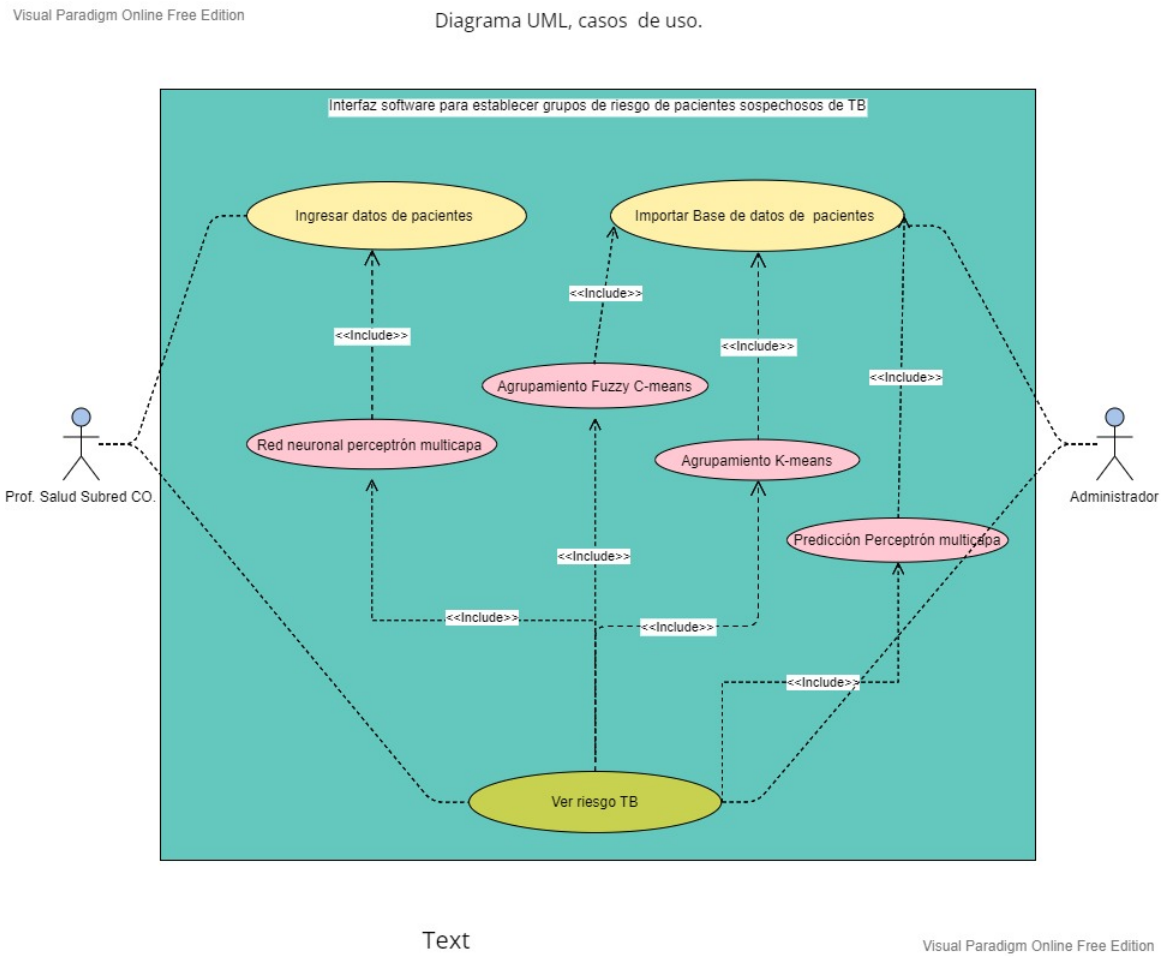


Figura 3-15: Diagrama UML, caso de uso para un profesional de la salud y caso de uso para el administrador. Elaboración propia.

3.4. Diseño final de la interfaz a través de un mapa de decisión.

En la secciones anteriores se detallaron las variables de entrada de la interfaz, las necesidades de los profesionales de la salud, los algoritmos de aprendizaje automático y la opción administrador anexada. En esta sección se reunieron cada uno de esos aspectos y se creó a través de un diagrama de decisión el diseño final de la interfaz, en el cual se plantean todos los posibles escenarios que el usuario puede explorar, como se muestra en la Figura **3-16**. El análisis se realiza de la siguiente manera: los rectángulos redondos de color amarillo indican las ventanas. Los rombos de color rosado representan las condiciones. Los rectángulos de color verde indican la ejecución de los algoritmos de aprendizaje automático. Los símbolos de documento de color morado indican los valores, diagramas y cuadros que se muestran en pantalla. Y los óvalos de color azul indican el inicio y el final de la interfaz.

Teniendo en cuenta lo anterior, la interfaz recibe al usuario en la ventana inicio, allí se colocan dos opciones de ingreso: en la primera opción el usuario podrá ingresar si es un profesional de la salud y en la segunda opción llamada administrador podrá ingresar toda persona que desee conocer el entorno educativo del proyecto. Si el usuario decide la primera opción, deberá digitar un usuario y contraseña o de lo contrario no podrá acceder a la ventana de trabajo del profesional de la salud, en la cual se encontrará el valor de la sensibilidad y especificidad del sistema automatizado, el cual realizará la predicción del riesgo de TB al ingresar el sexo, la edad, grupo poblacional, localidad, diagnostico VIH y TAR a la red neuronal perceptrón multicapa. Si de lo contrario el usuario desea ingresar al entorno educativo, también deberá digitar un usuario y contraseña, si es digitada correctamente lo dirigirá a la ventana administrador en la cual deberá importar la base de datos de la subred CO en una tabla. Una vez importados los datos el usuario podrá seleccionar si desea ver el comportamiento de los cuatro algoritmos de agrupamiento Fuzzy c-means con 2 y 3 cluster, k-means con 2 y 3 clúster al desplegar para cada caso una ventana, en las cuales se podrá observar: un plano 2D en caso de ser agrupamiento por dos clúster, un plano 3D en caso de ser agrupamiento por tres clúster, una matriz de confusión, el error de predicción, la sensibilidad y la especificidad de cada algoritmo. Dentro del mismo administrador se podrá ver el comportamiento de la red neuronal perceptron multicapa al ser entrenada con diferentes números de neuronas en la capa oculta, para este caso el usuario podrá observar la matriz de confusión, el error, la sensibilidad y especificidad de la red.

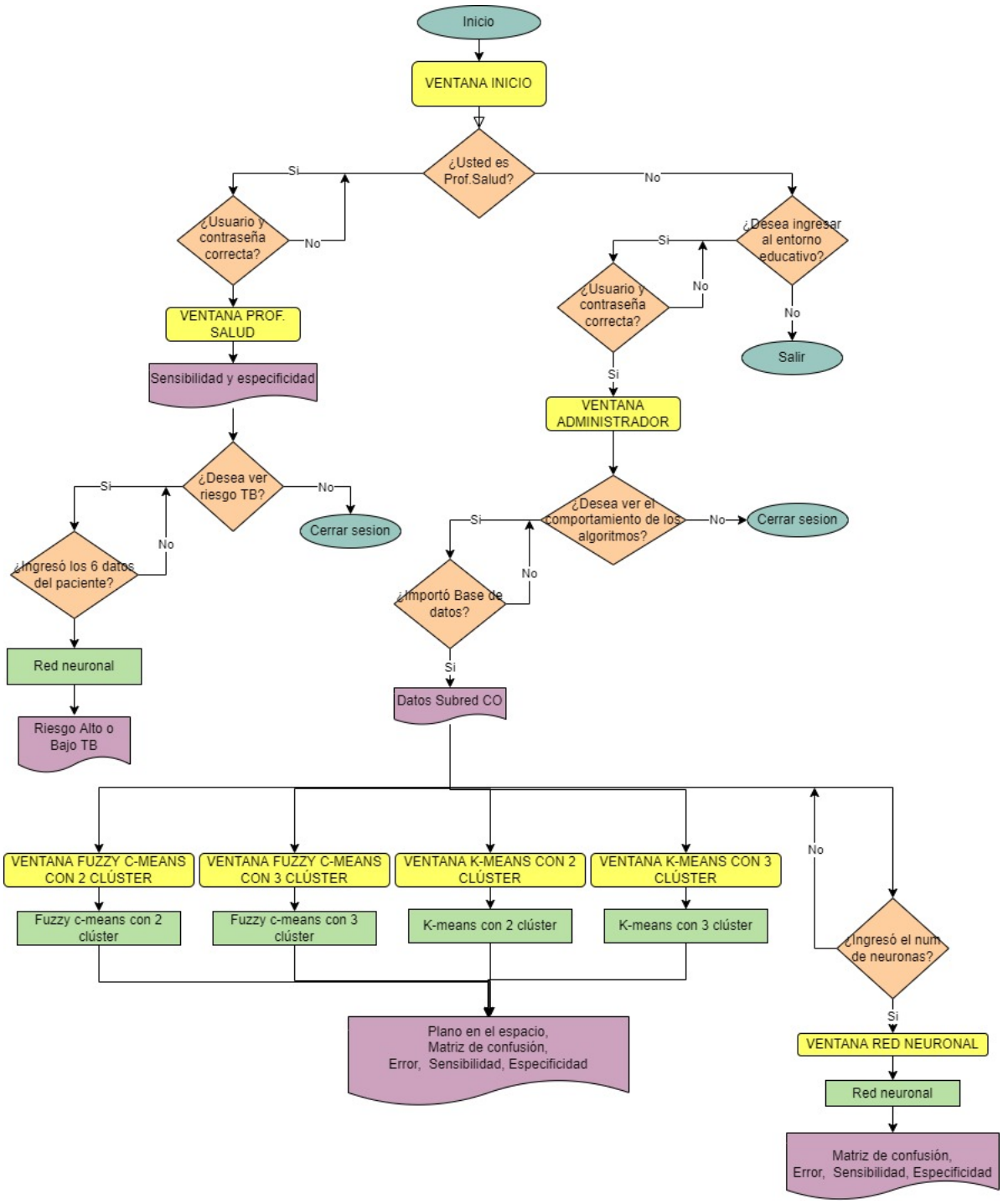


Figura 3-16: Mapa de decisión que representa las funciones de la interfaz. Elaboración propia.

3.5. Evaluación y validación de la interfaz desarrollada a partir de reuniones con profesionales en salud

La evaluación y validación de la interfaz se realizó a través de dos reuniones por Google Meet. La primera reunión se llevo a cabo el día 17 de Mayo del 2022 y estuvieron presentes los miembros del personal científico del proyecto de investigación "Generación de modelos alternativos basados en inteligencia computacional para tamización y diagnóstico de Tuberculosis pulmonar": la médico internista María Angélica Palencia y el médico neumólogo Carlos Awad de la Subred CO, el Dr. Andrés L .Jutinico de la Universidad Antonio Nariño, el Dr. Alvaro Orjuela de la Universidad del Rosario, y la estudiante de ingeniería biomédica Lenny Tatiana Silva. La segunda reunión se llevo a cabo el día 24 de Mayo del 2022 y estuvieron presentes tres personas: El investigador principal del proyecto, el médico neumólogo y mi persona.
El propósito de la primera reunión consistió en evaluar la interfaz y en la segunda reunión el objetivo fue mostrar la interfaz con las propuestas de mejora ya implementadas, para ser validada por los médicos. El orden de cada una de las reuniones fue el siguiente: se presentó la encuesta web diseñada para la evaluación a todos los participantes, para que tuvieran pleno conocimiento de los aspectos a evaluar (la encuesta se detallará mas adelante), posteriormente se mostró la ejecución de la interfaz a través de Matlab, donde se explico, cada una de las ventanas y sus opciones. El orden considerado fue: ventana de inicio, ventana de profesional de la salud y ventanas de administrador. Finalmente se escucharon las propuestas de los médicos y se aclararon todas las dudas generadas.

3.5.1. Diseño de la encuesta.

El método de evaluación elegido para este proyecto fue una encuesta web debido a que las reuniones fueron programadas de forma virtual. La herramienta utilizada fue Google Formularios ya que esta arroja estadísticamente los resultados facilitando su análisis, además que permite formular preguntas de rápida respuesta. ¿En que nos servía que las preguntas fuesen de rápida respuesta?. Este método se eligió dado que los médicos por cada sesión disponían de una hora para la presentación de la interfaz, así como, para la evaluación y solución de la encuesta. Por esta razón la mayoría de las preguntas fueron de selección múltiple.
En la encuesta se reunieron los aspectos a evaluar que se deben tener en cuenta para una interfaz, los cuales fueron basados en el libro "Ingeniería del software"[8]. Los aspectos considerados fueron de diseño y de usabilidad. Para el diseño se evalúa el numero y tamaño de las ventanas, el tamaño de la letra, los colores usados, los logos y botones entre otros. Para la usabilidad se evalua la comodidad del usuario objetivo con la interfaz, es decir la simplicidad, la claridad, la coherencia, la familiaridad y la velocidad durante la interacción. En la Tabla **3-5** se mencionan cada una de la preguntas planteadas, sin embargo las personas que

contestaron la encuesta no solo respondieron preguntas como se puede ver en los items del 1 al 6, sino que también indicaron su nivel de acuerdo con algunas premisas como se puede ver en los items del 7 al 10 y además valoraron la facilidad de uso y velocidad del sistema.

PREGUNTAS DE EVALUACIÓN		
ITEM	PREGUNTA O PREMISA	OPCIONES DE RESPUESTA
1	¿El número de ventanas es adecuado?	1 = muy pocas, 2 = adecuadas, 3 = muchas.
2	¿El tamaño de las ventanas es adecuado?	1 = muy pequeñas, 2 = adecuadas, 3 = muy grandes.
3	¿El tamaño de la letra es adecuado?	1 = muy pequeña, 2 = adecuada, 3 = muy grande.
4	¿Los colores son adecuados para el entorno médico?	1 = pocos colores, 2 = adecuados, 3 = muchos colores.
5	¿La ubicación de los logos es adecuada?	Abierta.
6	¿Cuál es su opinión sobre la organización de la información en pantalla ?	Muy confuso, un poco confuso, un poco claro o muy claro.
7	El lenguaje usado es apropiado y entendible	Muy en desacuerdo, en desacuerdo, de acuerdo, muy de acuerdo.
8	Las solicitudes de entrada son claras	Muy en desacuerdo, en desacuerdo, de acuerdo, muy de acuerdo.
9	La interfaz software cumple las expectativas y es una buena herramienta a implementar en el campo de la salud.	Muy en desacuerdo, en desacuerdo, de acuerdo, muy de acuerdo.
10	El usuario administrador ha sido un valor agregado significativo para la comunidad educativa.	Muy en desacuerdo, en desacuerdo, de acuerdo, muy de acuerdo.
11	Facilidad de uso	Valoración de 1 a 5 donde 1 = Difícil y 5 = Fácil
12	Velocidad del sistema	Valoración de 1 a 5 donde 1 = Lento y 5 = Rápido
13	Tiene algún comentario o sugerencia que pueda ayudarnos a mejorar la interfaz de usuario.	Abierta.

Tabla 3-5: Preguntas de evaluación de la interfaz con sus respectivas opciones de respuesta. Elaboración propia.

3.5.2. Análisis de los resultados de la encuesta.

Las preguntas se contestaron una sola vez por persona. Las cuatro personas que respondieron la encuesta fueron: el investigador principal del proyecto el codirector del proyecto, la médico internista y el médico neumólogo. Las tres primeras personas respondieron en la primera reunión y el médico neunologo en la segunda reunión, de acuerdo a esto podemos sacar las siguientes conclusiones: De manera general la interfaz tuvo una buena valoración y fueron pocas los detalles que se tuvieron que mejorar. Los cuatro evaluadores indicaron que el numero de ventanas y su tamaño son adecuados como se puede ver en las Figuras **3-17** y **3-18**. Para el tamaño de la letra tres evaluadores indicaron que es apropiada y solo uno indico que es muy pequeña como se puede ver en la figura **3-19**. Respecto a los colores (blanco y verde) a tres evaluadores les gusto y uno indico que eran muy poquitos como se puede ver en la Figura **3-20**. El voto respecto a la organización de la información en pantalla a nivel general, fue tres para información muy clara y un voto por información un poco clara como se puede ver en la Figura **3-21**.

Figura 3-17: Respuesta a la pregunta ¿El número de ventanas es adecuado?. Elaboración propia.

Figura 3-18: Respuesta a la pregunta ¿El tamaño de las ventanas es adecuado?. Elaboración propia.

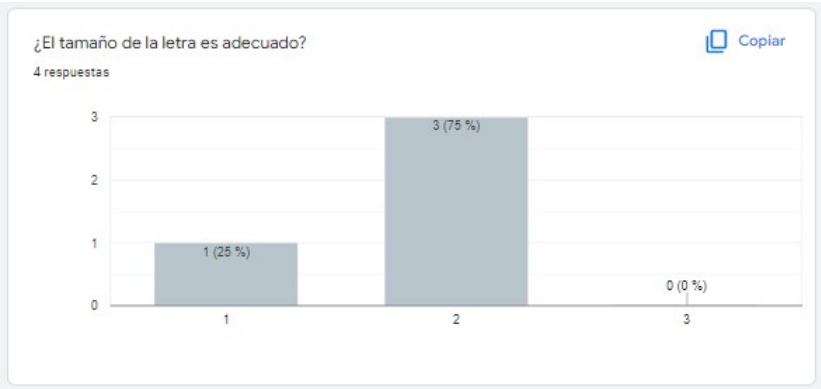


Figura 3-19: Respuesta a la pregunta ¿El tamaño de la letra es adecuado?. Elaboración propia.

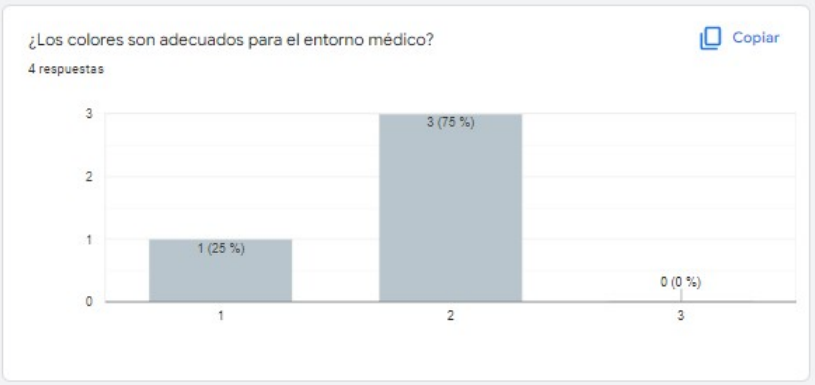


Figura 3-20: Respuesta a la pregunta ¿Los colores son adecuados para el entorno médico?. Elaboración propia.

Figura 3-21: Respuesta a la pregunta ¿Cuál es su opinión sobre la organización de la información en pantalla?. Elaboración propia.

Ahora analizando las premisas, de la Figura **3-23** podemos observar que solo un evaluador estuvo muy de acuerdo con que el lenguaje es apropiado y entendible y tres evaluadores

estuvieron únicamente de acuerdo. En la evaluación de si las solicitudes de entrada son claras hubo un voto negativo es decir, un evaluador que se encontró en desacuerdo, un voto de acuerdo y un voto muy de acuerdo, para el caso del voto negativo el evaluador comunicó la razón de su respuesta de manera oral y fue tenida en cuenta para las respectivas correcciones que se mostraran más adelante. En la figura **3-24** podemos observar a un solo evaluador que se encuentra muy de acuerdo con que la interfaz es una buena herramienta a implementar en el campo de la salud y tres evaluadores que indicaron estar de acuerdo. En la Figura **3-25** se evaluó el usuario administrador en el cual dos evaluadores indicaron estar de acuerdo y dos evaluadores estar muy de acuerdo en que ha sido un valor agregado significativo para la comunidad educativa.

Figura 3-22: Nivel de acuerdo con la premisa: El lenguaje usado es apropiado y entendible. Elaboración propia.

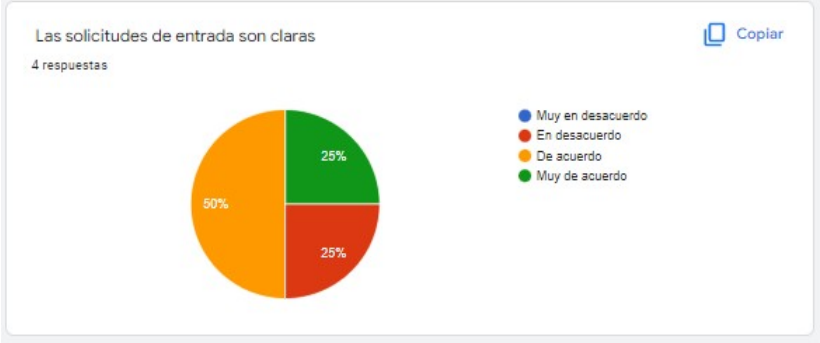


Figura 3-23: Nivel de acuerdo con la premisa: Las solicitudes de entrada son claras. Elaboración propia.

Figura 3-24: Nivel de acuerdo con la premisa: La interfaz software cumple las expectativas y es una buena herramienta a implementar en el campo de la salud. Elaboración propia.

Figura 3-25: Nivel de acuerdo con la premisa: El usuario administrador ha sido un valor agregado significativo para la comunidad educativa. Elaboración propia.

Ahora teniendo en cuenta la votación respecto a la dificultad de uso de la interfaz, dos evaluadores indicaron que es muy fácil valorándola en 5 y dos evaluadores indicaron que es fácil valorándola en 4 como se puede ver en la Figura **3-26**. Para la velocidad del sistema el resultado de la evaluación fue bueno como se puede ver en la Figura **3-27**, tres evaluadores la valoraron en 5 como rápida y solo un evaluador en 3 con una velocidad media. Finalmente las dos preguntas abiertas se respondieron de la siguiente manera: ¿La ubicación de los logos es adecuada? para lo cual, tres evaluadores respondieron que si y un evaluador no respondió. Los comentarios de los evaluadores o sugerencias que pudieran ayudarnos a mejorar la interfaz de usuario fueron dados de manera oral y en la siguiente sección son destacados.

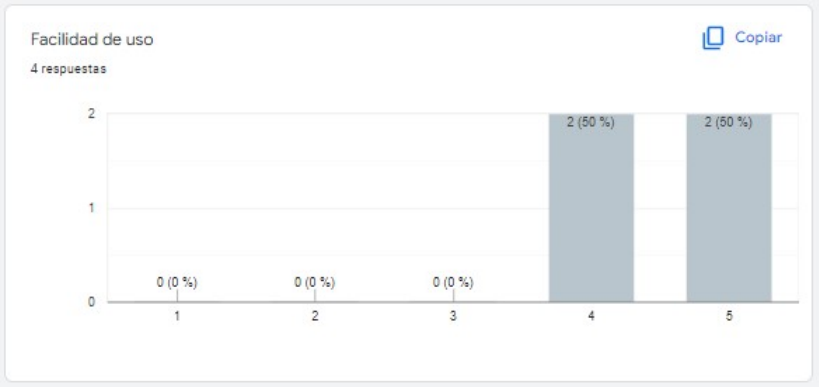


Figura 3-26: Valoración de la dificultad de uso. Elaboración propia.

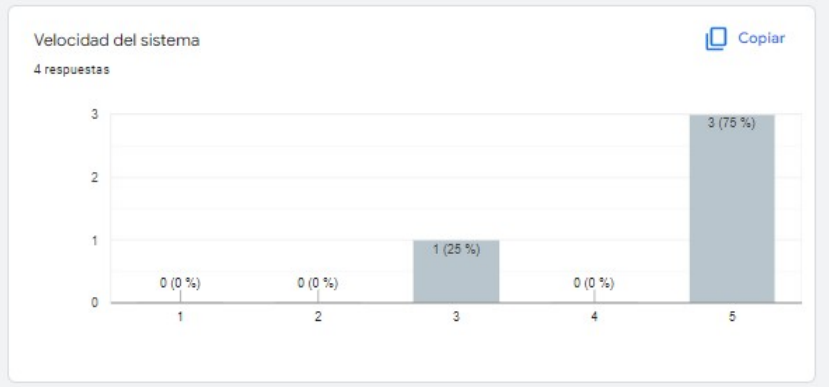
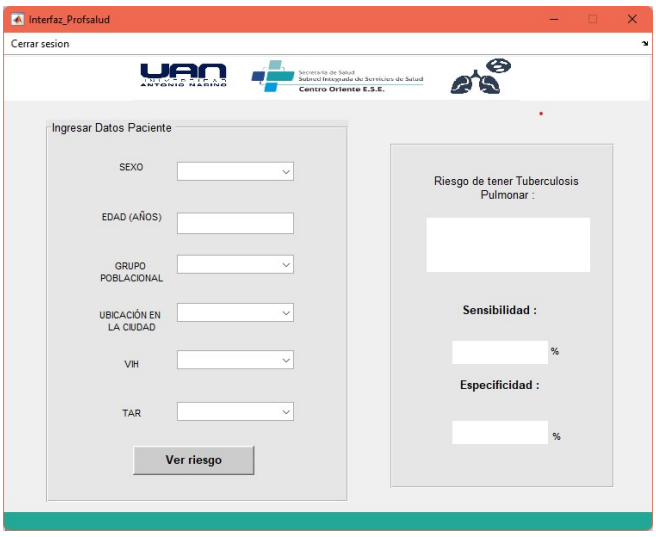


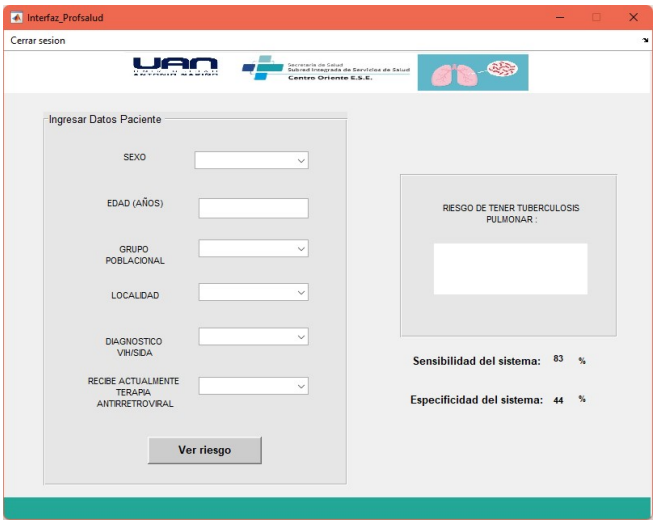
Figura 3-27: Valoración de la velocidad del sistema. Elaboración propia.

3.5.3. Sugerencias realizadas por el personal científico del proyecto.

Sugerencia número 1: Los médicos expresaron que al solicitar el ingreso de las seis variables de entrada: edad, sexo, grupo poblacional, localidad, diagnostico VIH y si recibe actualmente tratamiento antirretroviral, no se estaba siendo demasiado explicito con el mensaje puesto en pantalla para las ultimas tres variables. Por consiguiente y como se puede ver en la Figura **3-28** se cambió ubicación en la ciudad por localidad, VIH por diagnostico VIH/sida, y TAR por recibe actualmente tratamiento antirretroviral.



(a) Interfaz diseñada antes de la evaluación realizada por los médicos.



(b) Interfaz atendiendo las recomendaciones realizadas por los médicos.

Figura 3-28: Correspondientes modificaciones en la interfaz software ante la evaluación de los médicos. Elaboración propia.

Sugerencia número 2: La recomendación se hizo respecto al logo de TB que se muestra en la portada de la interfaz, el cual es un pulmón acompañado de la Mycobacterium Tuberculosis a blanco y negro, ellos comentaron que en la pruebas de laboratorio la bacteria se reconoce con un color fucsina carbólica y que seria interesante que se colocara en el logo ese color característico o uno parecido, por esta razón se modificó como se puede ver en la Figura **3-28** (b).

Sugerencia número 3: Esta sugerencia esta relacionada con la forma de mostrar la sen-

sibilidad y especificidad del sistema, debido a que durante la reunión con los médicos se detecto una confusión, ya que ellos preguntaron si la sensibilidad y la especificidad era por cada paciente ingresado o era de todo el sistema. Debido a lo anterior, se decidió modificar la forma en que se debían mostrar estos dos índices de desempeño, al eliminar el color blanco de fondo para los números y al mostrar los valores cuando se abre la ventana como se puede ver en la Figura **3-28** (b).

4 Resultados del proyecto y Discusión.

4.1. Validación de la Interfaz.

Durante las reuniones con el personal científico del proyecto, los médicos solicitaron validar algunos de los casos de la base de datos de pacientes sospechosos de TB. Cabe aclarar, adicionalmente de la validación que se realiza en las fases de entrenamiento y de test, usadas para la selección de la red neuronal. Destacamos que para el entrenamiento de la red se usaron únicamente los años 2017 y 2018, que corresponden al 70 % de los datos. En la Tabla **4-1** se muestran las variables ingresadas en la interfaz software y el riesgo de TB indicado.

CASOS	RIESGO
Sexo: Masculino, Edad: 52, Grupo poblacional: Habitante de calle, Localidad: Sin dato, Diagnostico VIH/SIDA: Positivo, Recibe actualmente terapia antirretroviral: Negativo.	Alto
Sexo: Femenino, Edad: 30, Grupo poblacional: Otros, Localidad: Sin dato, Diagnostico VIH/SIDA: Positivo, Recibe actualmente terapia antirretroviral: Positivo.	Alto
Sexo: Masculino, Edad: 23, Grupo poblacional: Otros, Localidad: Los martires, Diagnostico VIH/SIDA: Positivo, Recibe actualmente terapia antirretroviral: Positivo.	Alto
Sexo: Femenino, Edad: 91, Grupo poblacional: Otros, Localidad: Usme, Diagnostico VIH/SIDA: Negativo, Recibe actualmente terapia antirretroviral: Sin dato.	Bajo
Sexo: Femenino, Edad: 55, Grupo poblacional: Otros, Localidad: Tunjuelito, Diagnostico VIH/SIDA: Negativo, Recibe actualmente terapia antirretroviral: Negativo.	Bajo
Sexo: Femenino, Edad: 66, Grupo poblacional: Otros, Localidad: Antonio Nariño, Diagnostico VIH/SIDA: Sin dato, Recibe actualmente terapia antirretroviral: Sin dato.	Bajo

Tabla 4-1: Casos seleccionados para la validación de la red con 154 neuronas. Elaboración propia.

En la primera reunión se presentó a los médicos la ventana del profesional de la salud de

la interfaz software, con una sensibilidad del 95 %, una especificidad del 10 % y un error de predicción del 17 %. Sin embargo y debido a la baja especificidad obtenida, la interfaz indicó que el riesgo era alto para casi todos los casos. De acuerdo a lo anterior se decidió volver a entrenar la red, para lo cual se amplio el numero de neuronas de 2 a 200 durante el entrenamiento. Finalmente se escogió una red neuronal con 154 neuronas en la capa oculta, obteniendo una sensibilidad de 83 %, una especificidad de 44 % y un error de predicción del 22 %. Este resultado aun sigue sin ser del todo satisfactorio, ya que no se logro obtener una mayor especificidad, sin afectar la sensibilidad de la red. Dado que la selección de un mejor algoritmo no hace parte de este proyecto, se decidió actualizar la red para mostrar las mejoras realizadas en la interfaz en una segunda reunión.
En la segunda reunión para la validación, se ingresaron algunos pacientes de la base de datos de sospechosos de TB del año 2019 y el sistema respondió de manera adecuada. En la Tabla **4-1** se pude ver que de seis casos que fueron ingresados, tres arrojaron riesgo alto y tres riesgo bajo correctamente.

4.2. Resultados para la interfaz software.

En este proyecto se realizo una interfaz capaz de predecir el riesgo de un paciente con sospecha de tener TB, la cual clasifica al paciente en riesgo alto o riesgo bajo. La sensibilidad del sistema automático es de 83 %, con especificidad de 44 % y un error de predicción del 22 %. La interfaz software se compone por 8 ventanas principales: una ventana de inicio, una ventana para el profesional de la salud, una ventana para el administrador, una ventana para el algoritmo Fuzzy c-means con 2 cluster, una ventana para el algoritmo Fuzzy c-means con 3 cluster, una ventana para el algoritmo k-means con 2 cluster, una ventana para el algoritmo k-means con 3 cluster y una ventana para la red neuronal. Las ventanas tienen color blanco de fondo y en la parte inferior una barra de color verde, además cada una tiene una portada en la cual se presentan tres logos representativos: el logo de la Universidad Antonio Nariño, el logo de la subred CO y un logo representativo de la TB. La interfaz inicia con una ventana de bienvenida la cual presenta las opciones de ingresar y salir. Además en la parte superior izquierda, se despliega una opción en caso de querer ingresar a el entorno educativo del proyecto como administrador (ver Figura **4-1**).

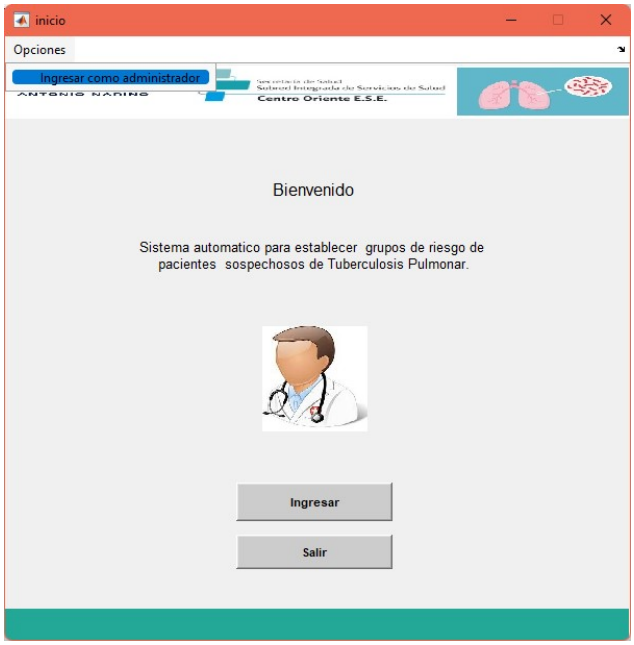


Figura 4-1: Ventana de bienvenida e ingreso de usuario. Elaboración propia.

Cuando el usuario decide entrar ya sea como profesional de la salud o como administrador, la interfaz como sistema de seguridad despliega una ventana para el ingreso de un usuario y contraseña, siendo para cada caso valores de entrada diferentes, (ver Figura **4-2**).

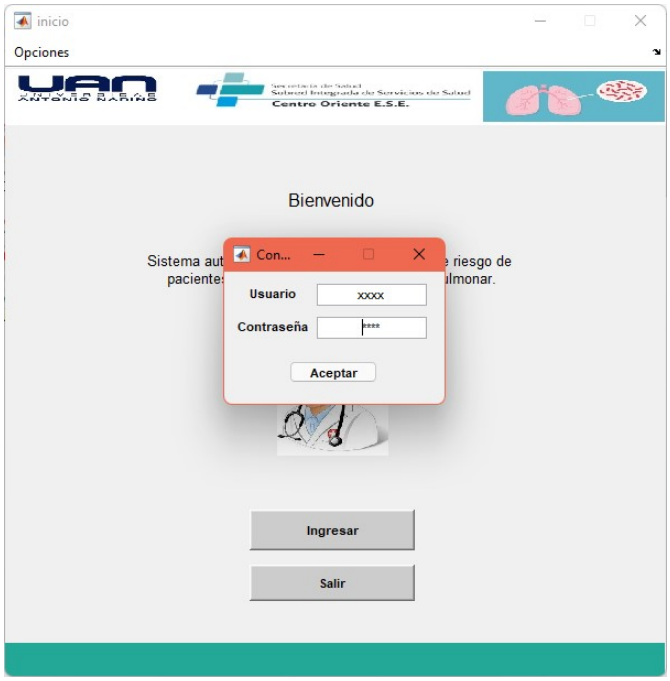


Figura 4-2: Ingreso de usuario y contraseña, como sistema de seguridad. Elaboración propia.

Si el usuario es un profesional de la salud, el entorno de trabajo que encuentra se compone de una sección para ingresar los datos del paciente y una sección para observar el riesgo arrojado. Además, el usuario encontrará en pantalla los índices de desempeño del sistema automático como se puede ver en la Figura **4-3**. Cuando el profesional de la salud desee ingresar los datos del paciente deberá desplegar una lista de opciones para cada variable de entrada y dar click sobre el valor de su interés, excepto para la edad en la que se deberá escribir un numero dentro del rango 1 a 100. De acuerdo a lo anterior la lista de opciones desplegable para cada variable es, para Sexo : Masculino, Femenino (ver Figura **4-4**). Para el Grupo poblacional: Habitante de calle, Desplazado, Migrante, Población carcelaria, Víctima de violencia armada, Indigena, y otros, como se aprecia en la Figura **4-5**. Para la Localidad: Antonio Nariño, Barrios Unidos, Bosa, Chapinero, Ciudad Bolívar, Engativá, Fontibón, Kennedy, La Candelaria, Los Mártires, Puente Aranda, Rafael Uribe Uribe, San Cristóbal, Santa Fe, Suba, Teusaquillo, Tunjuelito, Usaquén, Usme, No aplica, y Sin dato, como se muestra en la Figura **4-6**. Para Diagnostico VIH/sida: Positivo, Negativo y Sin dato, como se indica en la Figura **4-7** y para Recibe actualmente terapia antirretroviral: Positivo, Negativo, y Sin dato, como se indica en la Figura **4-8**.

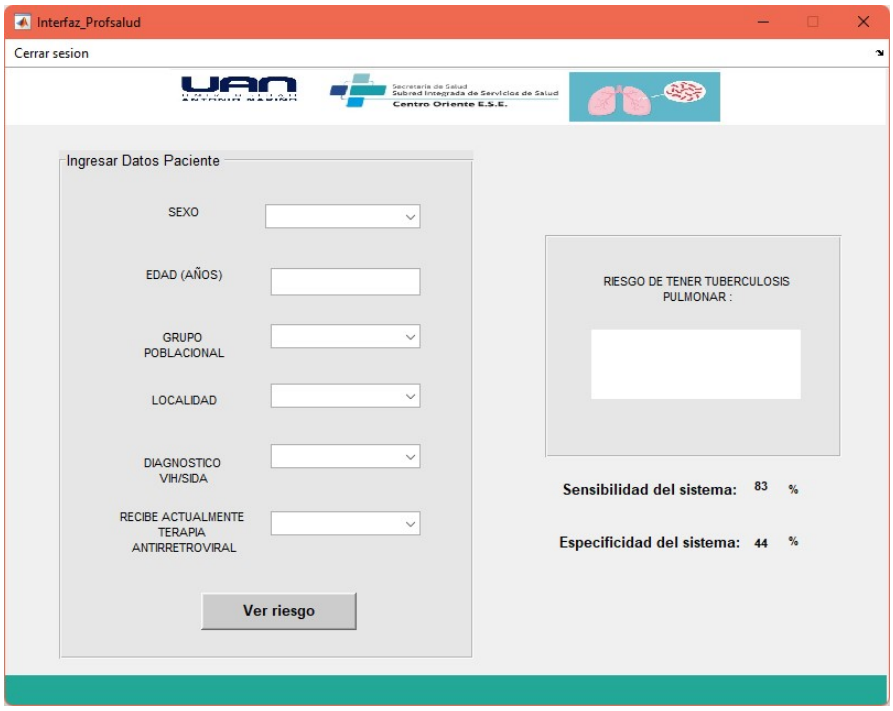


Figura 4-3: Ventana principal para obtener el riesgo de Tuberculosis. Elaboración propia.

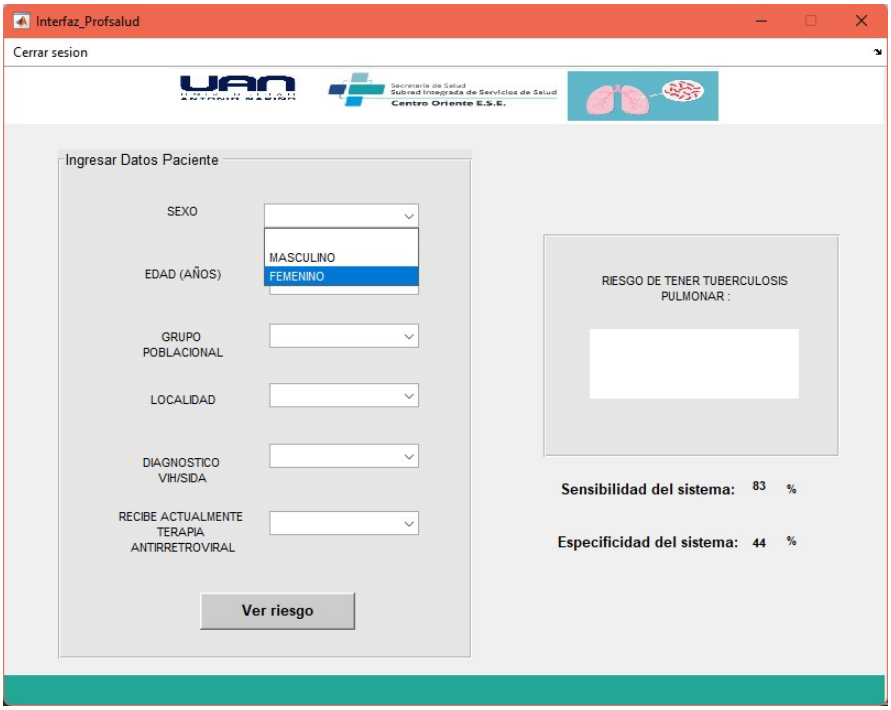


Figura 4-4: Lista de opciones desplegable para sexo. Elaboración propia.

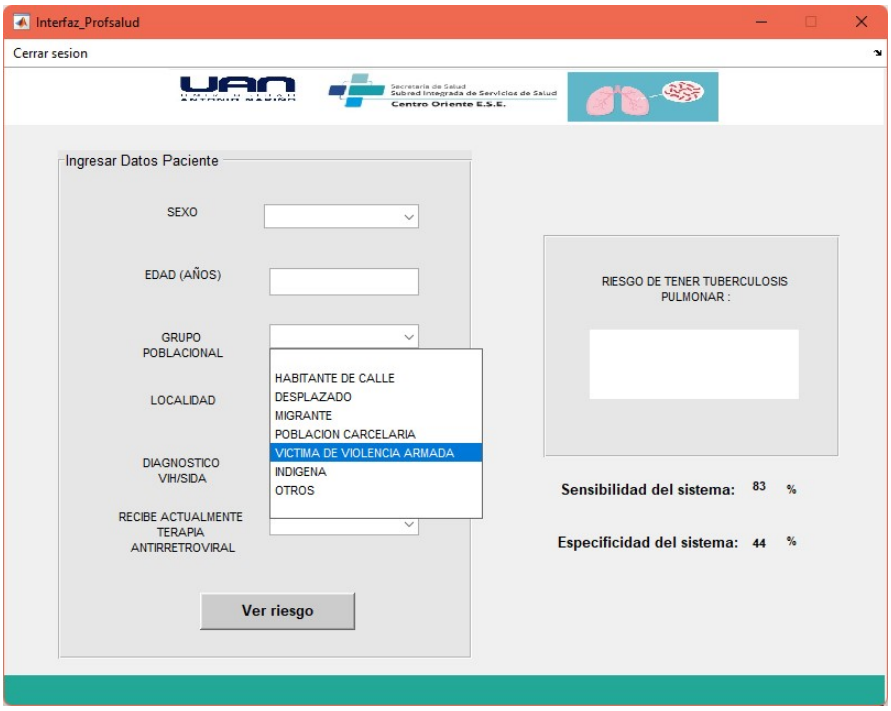


Figura 4-5: Lista de opciones desplegable para grupo poblacional. Elaboración propia.

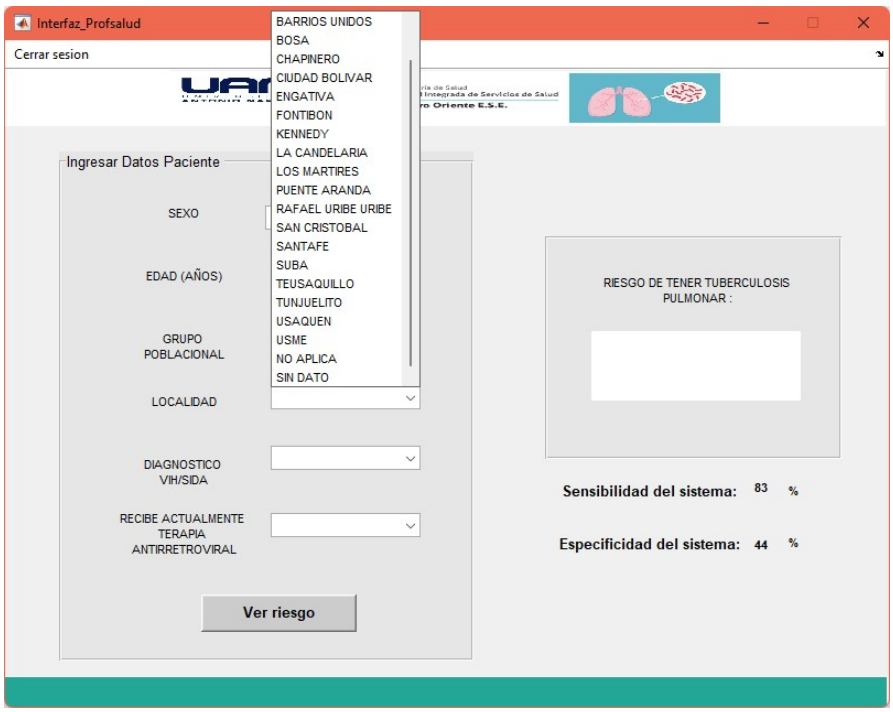


Figura 4-6: Lista de opciones desplegable para localidad. Elaboración propia.

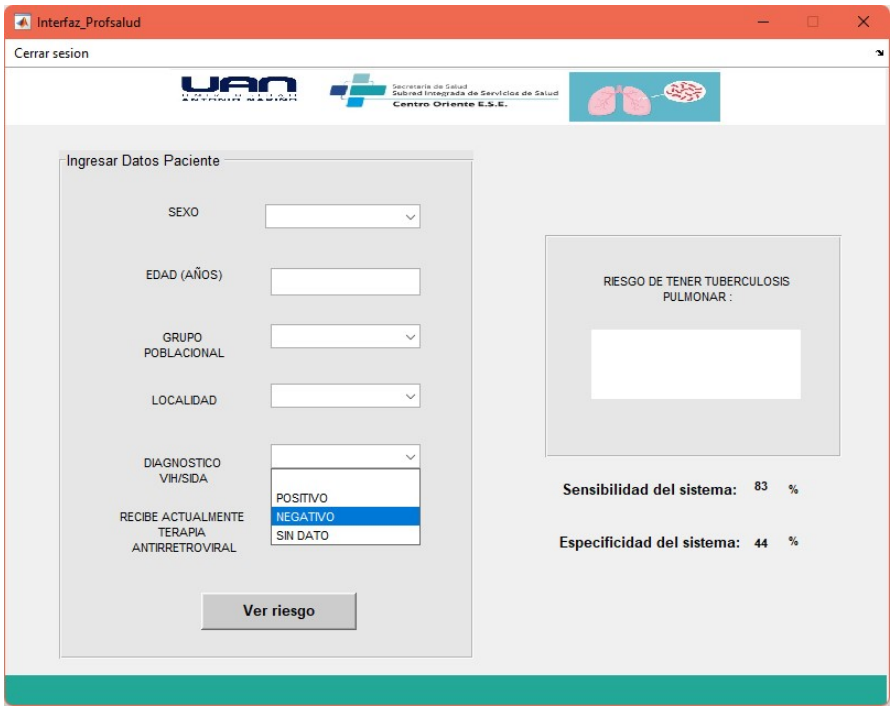


Figura 4-7: Lista de opciones desplegable para diagnostico VIH/sida. Elaboración propia.

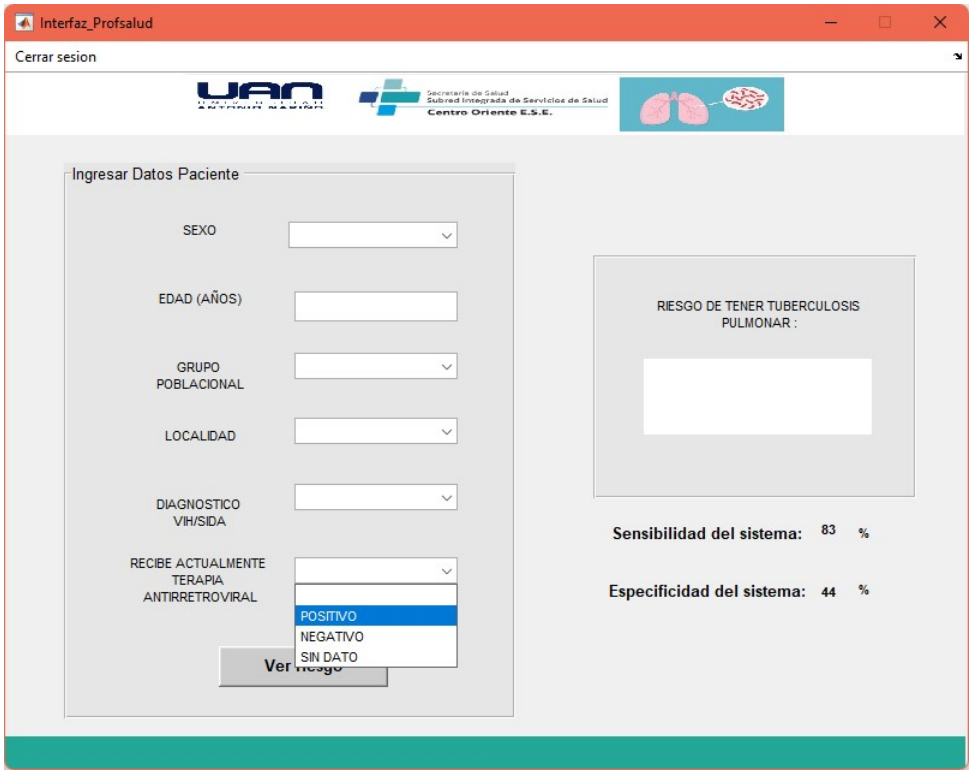
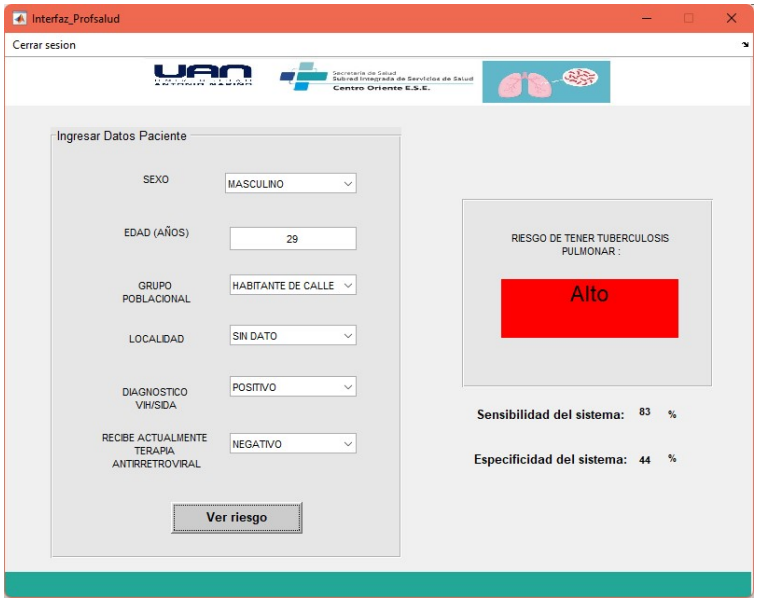
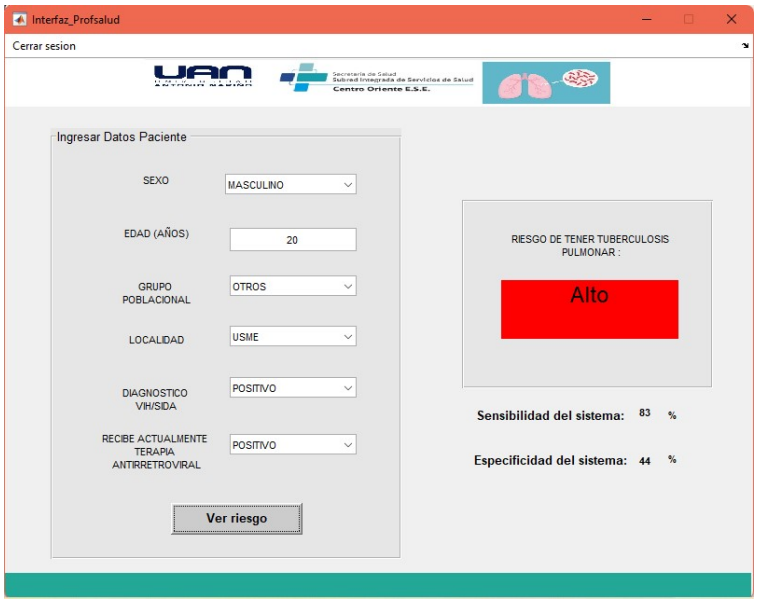


Figura 4-8: Lista de opciones desplegable para terapia antirretroviral. Elaboración propia.

Teniendo en cuenta cada una de las opciones para cada variable, se procede a mostrar el funcionamiento de la interfaz, el cual consiste en ingresar las 6 variables, dar click en el botón "Ver riesgo" e inmediatamente el sistema arrojará en tiempo real el riesgo en el que se encuentra el paciente por medio de un mensaje y un color indicativo. Para riesgo alto el mensaje enviado es ALTO y el color es rojo, para riesgo bajo el mensaje enviado es BAJO y el color es verde. Como se puede ver en las Figuras **4-9** y **4-10** por medio dos ejemplos para riesgo alto y dos ejemplos para riesgo bajo. Los 4 pacientes pertenecen al año 2019 de la base de datos.

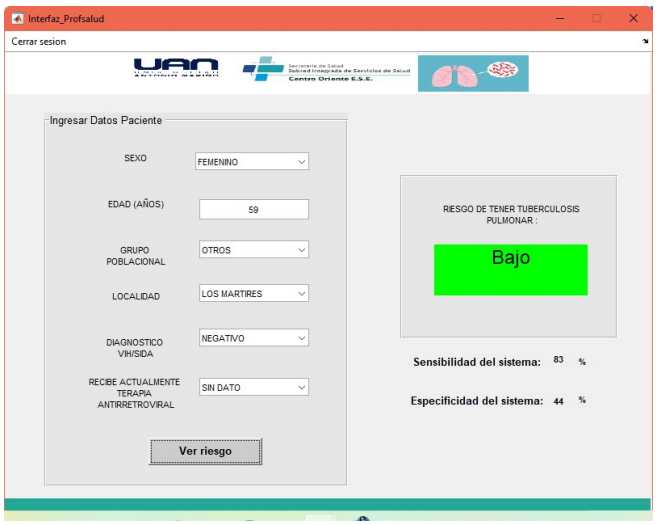


(a) Ejemplo número 1 de riesgo alto.

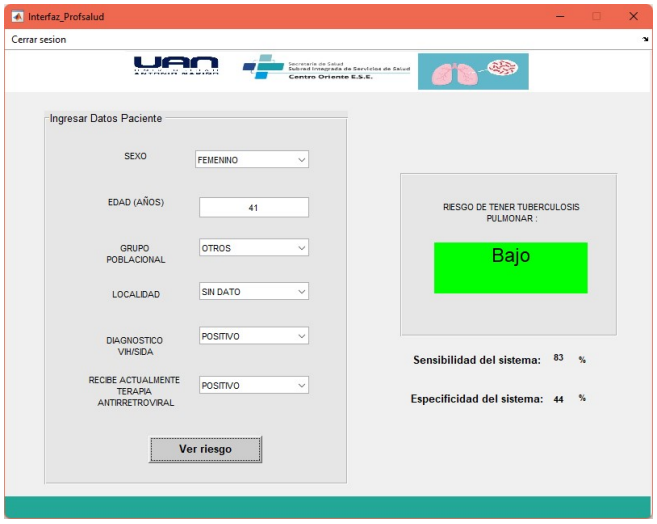


(b) Ejemplo número 2 de riesgo alto.

Figura 4-9: Interfaz software arrojando riesgo alto en dos casos diferentes. Elaboración propia.



(a) Ejemplo número 1 de riesgo bajo.



(b) Ejemplo número 2 de riesgo bajo.

Figura 4-10: Interfaz software arrojando riesgo bajo en dos casos diferentes. Elaboración propia.

Ahora, si el usuario ingresa como administrador, encontrará la ventana de trabajo con una tabla limpia y las opciones para correr cada uno de los algoritmos Fuzzy C-means con 2 clústers, Fuzzy C-means con 3 clústers, K-means con 2 clústers, K-means con 3 clústers y la red neuronal como se puede ver en la Figura **4-11**. Para poder ver la predicción de cada uno de los algoritmos se debe importar la base de datos, en la opción que se encuentra en parte superior izquierda IMPORTAR e inmediatamente se desplegará en la tabla la base de datos como se puede ver en la Figura **4-12**.

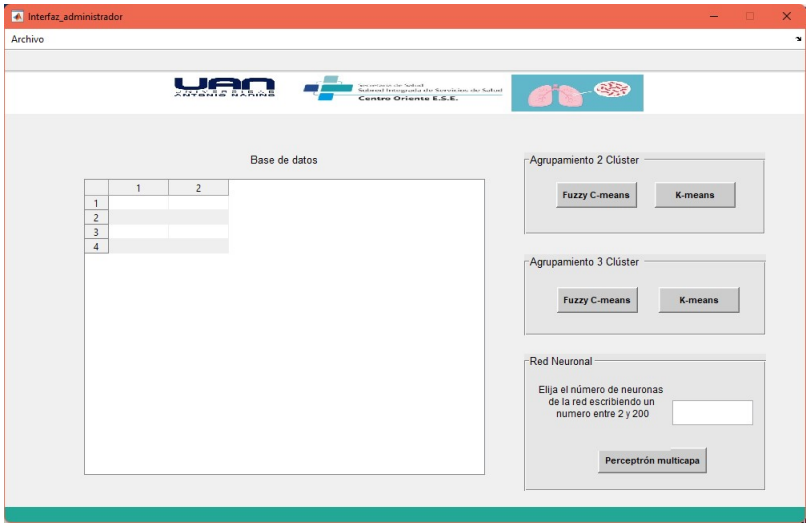


Figura 4-11: Ventana administrador antes de ingresar los datos. Elaboración propia.

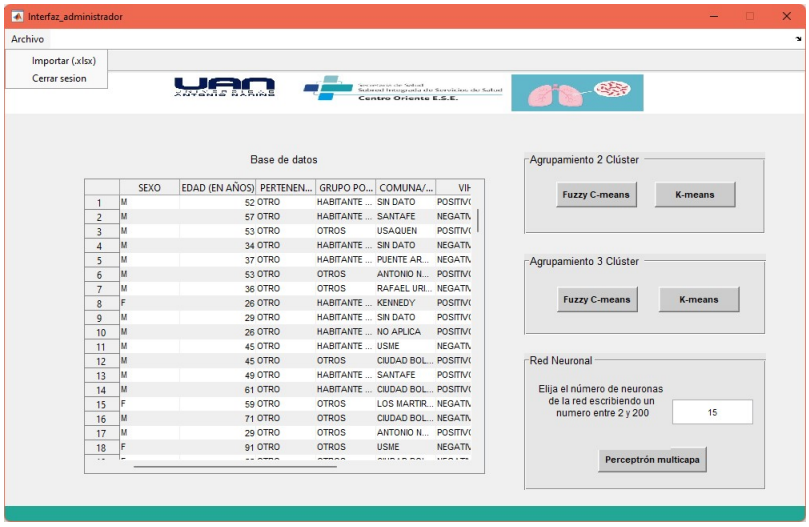


Figura 4-12: Importar base de datos a la interfaz. Elaboración propia.

En las figuras **4-13**, **4-14**, **4-15** y **4-16** se muestra la ventana desplegada para cada algoritmo, en las cuales se puede ver: los grupos de riesgo conformados a través de un plano en el espacio, en los cuales cada grupo corresponde a un dominio, además se cuenta con las opciones de hacer zoom, alejarse y de ver el valor las coordenadas que toma cada punto. La correspondiente matriz de confusión, en la cual se puede observar una diagonal de color azul como los pacientes predichos correctamente y alrededor de la diagonal en color rojo los pacientes predichos incorrectamente, donde los valores del eje Y corresponden a los grupos de riesgo reales y los valores del eje X a los grupos de riesgo predichos y por ultimo. La sensibilidad, la especificidad y el error de predicción en porcentaje.

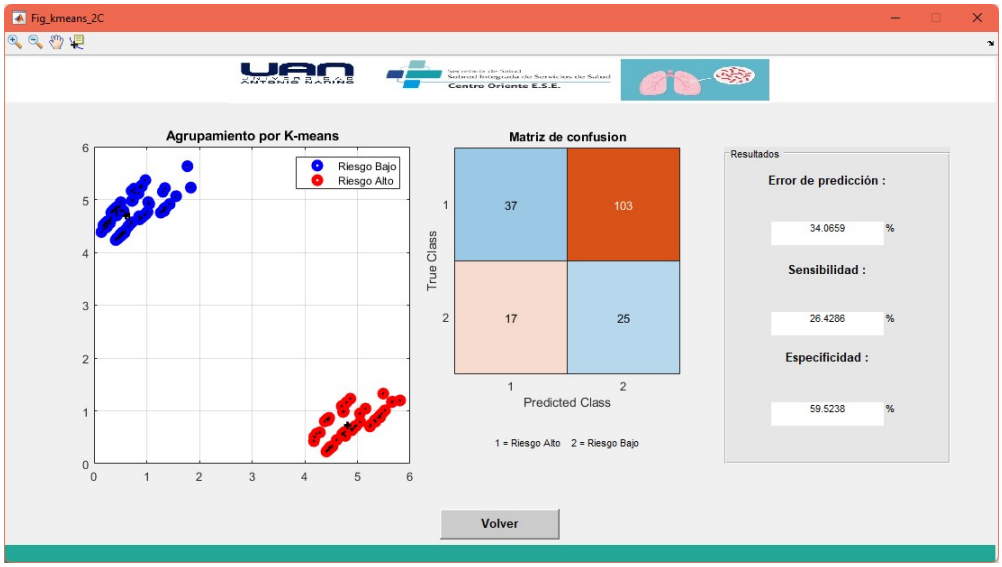


Figura 4-13: Evaluación del algoritmo K-means con 2 Clúster Elaboración propia.

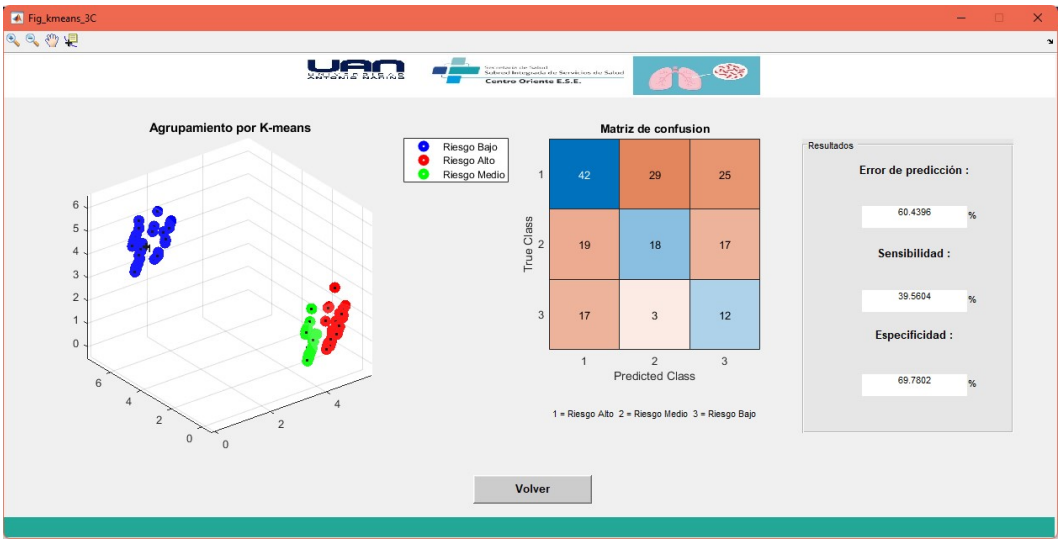


Figura 4-14: Evaluación del algoritmo K-means con 3 Clúster. Elaboración propia.

Figura 4-15: Evaluación del algoritmo Fuzzy C-means con 2 Clúster. Elaboración propia.

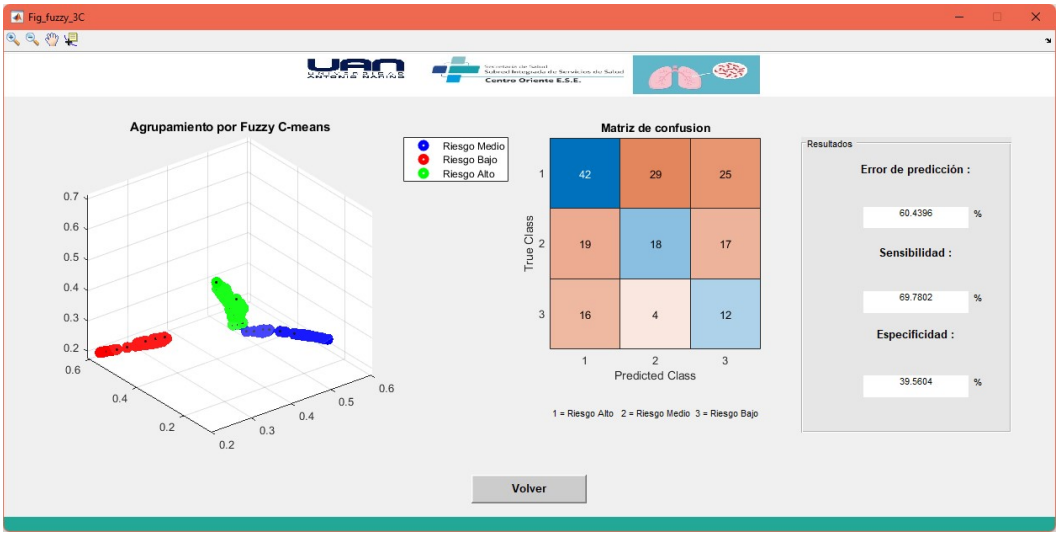


Figura 4-16: Evaluación del algoritmo Fuzzy C-means con 3 Clúster. Elaboración propia.

Para ver la predicción con la red neuronal el usuario debe escribir un numero entre 2 y 200 para indicar el número de neuronas que tendrá el algoritmo en su capa oculta, de esta forma, para cada número que escriba el usuario se entrenará una red. Inmediatamente se desplegará la ventana con sus respectivas variables de evaluación del algoritmo como se puede ver en la Figura **4-17**.

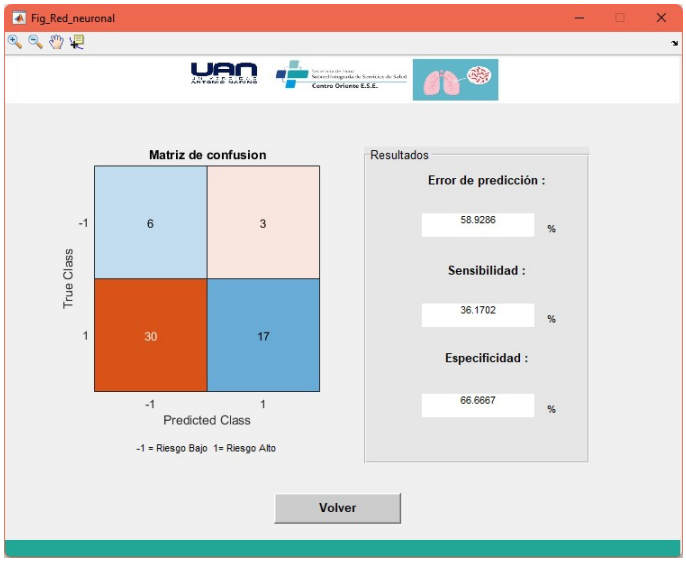


Figura 4-17: Evaluación de la red neuronal Perceptrón multicapa con 15 neuronas. Elaboración propia.

4.3. Discusión

A partir de la segunda reunión realizada con el personal científico del proyecto, se generaron algunas sugerencias para aplicar en trabajos futuros relacionados con la interfaz software para establecer grupos de riesgo de pacientes sospechosos de TB. El médico Carlos Awad afirmo que la interfaz software que se esta desarrollando el grupo de investigación es pertinente y va por buen camino para llegar a ser implementada como una herramienta de apoyo diagnostico a los médicos, en regiones de tipo rural y con poca infraestructura. Sin embargo, aun no es una herramienta que remplace a las pruebas de laboratorio en las grandes ciudades. El doctor resaltó que la TB no es una enfermedad de fácil diagnostico, dado que la incidencia es muy baja, estamos hablando de 127 casos por cada 100 000 habitantes, razón por la cual la toma de decisiones respecto al tratamiento inicial de esta enfermedad es compleja, sobre todo en lugares con baja densidad poblacional y recursos limitados [45]. Seguidamente, el médico resalta que existe una gran diferencia en la incidencia de esta enfermedad respecto de las enfermedades cardiacas o de la diabetes. Dada esta situación el médico expuso una serie de propuestas futuras para el proyecto, las cuales ayudaran a ir perfeccionando la interfaz software. En primer lugar se propone ampliar lista de opciones de entrada a la interfaz para sexo y grupo poblacional. De acuerdo a lo anterior, para la variable sexo, el médico indica que es importante incluir a la personas transexuales, dado que estas se consideran como un grupo con riesgo alto para contraer VIH. Para el caso de los grupos poblaciones el médico se baso en un informe de Tuberculosis presentado por Minsalud en el año 2021, para exponer los grupos que no se tuvieron en cuenta en este proyecto, pero que ya se trabajan el marco de la salud. Dentro de los cuales se encuentran: los afrodescendientes, los trabajadores de la salud, las

personas con discapacidad, los gitanos, las mujeres gestantes y las personas pertenecientes a centros psiquiátricos [45]. En segundo lugar para que los índices de desempeño de la interfaz (sensibilidad y especificidad) mejoren y aumenten la confianza de los médicos respecto de la interfaz para apoyo a diagnostico de TB, el doctor Carlos propone volver a entrenar los algoritmos con una base de datos más amplia, donde se tenga en cuenta un mayor numero de variables soscieconomicas y un mayor numero de pacientes control, es decir, que se añadan pacientes que no tengan TB confirmada ya que muchos de los pacientes que conforman la actual base de datos eran diagnosticados con TB, lo cual puede significar un sesgo en el entrenamiento de los algoritmos.

Para finalizar, es importante aclarar que las recomendaciones presentadas anteriormente, se están desarrollando en el marco de un TIP de la Maestría en Instrumentación y Automatización, denominado "Automatización de toma de datos para procesamiento del lenguaje natural aplicado al diagnóstico de tuberculosis pulmonar", en el cual se tiene acceso anonimizado a las historias clínicas del hospital, con el cual se espera completar la base de datos.

5 Conclusiones.

- Se desarrolló una interfaz software en Matlab, que permite establecer el grupo de riesgo alto o bajo al que pertenecen los pacientes sospechosos de Tuberculosis pulmonar, usando información de la base de datos de pacientes sospechosos de TB de los años 2017, 2018 y 2019, proporcionada por la Subred CO.

- Las variables seleccionadas para realizar la predicción del riesgo de tuberculosis son información de tipo limitada, sin embargo aportan información suficiente para los algoritmos, esto es debido a que la tuberculosis es una enfermedad que se relaciona directamente con el sistema inmunitario y las condiciones socio-económicas.

- La interfaz cumple las expectativas de los médicos como herramienta de apoyo diagnostico para instituciones prestadoras de servicios de salud, en particular esta pensada para ser usada en regiones distantes a las grandes ciudades y con recursos limitados, ya que solo se requiere del uso de un computador para predecir el riesgo de tuberculosis de un paciente.

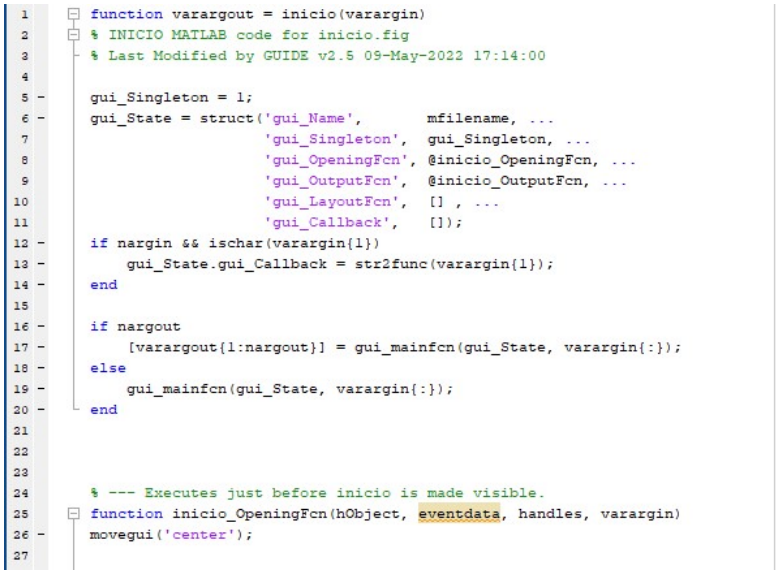
- La red neuronal perceptrón multícapa fue el mejor algoritmo de aprendizaje automático usado en este proyecto. Sin embargo, con una mayor cantidad de datos de pacientes sospechosos de TB se podrían usar algoritmos de aprendizaje profundo que permitan una mayor generalización en los resultados, esto puede ser posible si mas instituciones de salud apoyan este marco investigativo.

- La interfaz software logra ser amigable con el usuario ya que su manejo es muy fácil e indicativo y se elaboró con el fin de que no se presentara ningún error durante su uso que pudiera intervenir en el alcance de su objetivo.

- Se realizaron dos reuniones con el personal científico (personal de la salud) del proyecto de investigación “Generación de modelos alternativos basados en inteligencia computacional para tamización y diagnóstico de Tuberculosis pulmonar”, en las cuales se evaluó y valido el funcionamiento de la interfaz.

6 Anexos.

6.1. Código en Matlab de la ventana Inicio.



```
function varargout = inicio(varargin)
% INICIO MATLAB code for inicio.fig
% Last Modified by GUIDE v2.5 09-May-2022 17:14:00

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end



% --- Executes just before inicio is made visible.
function inicio_OpeningFcn(hObject, eventdata, handles, varargin)
movegui('center');

```

Figura 6-1: Inicio, parte 1 del código. Elaboración propia.

```
27
28 -    axes(handles.axes1);
29 -    [x,map]=imread('Portada.jpg');
30 -    image(x);
31 -    colormap(map);
32 -    axis off
33 -    hold on
34
35 -    axes(handles.axes6);
36 -    [x,map]=imread('Icono_usuario.jpg');
37 -    image(x);
38 -    colormap(map);
39 -    axis off
40 -    hold on
41
42      % Choose default command line output for inicio
43 -    handles.output = hObject;
44      % Update handles structure
45 -    guidata(hObject, handles);
46
47
48      % --- Outputs from this function are returned to the command line.
49      function varargout = inicio_OutputFcn(hObject, eventdata, handles)
50
51      % Get default command line output from handles structure
52 -    varargout{1} = handles.output;
```

Figura 6-2: Inicio, parte 2 del código. Elaboración propia.

```
52 -    varargout{1} = handles.output;
53
54
55      % --- Executes on button press in boton_usuario_salud.
56      function boton_usuario_salud_Callback(hObject, eventdata, handles)
57 -    passwordfig('salud','1234');
58 -    uiwait();
59 -    close(handles.output);
60 -    Interfaz_Profsalud;
61
62      % --- Executes on button press in boton_salir.
63      function boton_salir_Callback(hObject, eventdata, handles)
64 -    close(handles.output);
65
66
67      % --------------------------------------------------------------------
68      function Opciones_Callback(hObject, eventdata, handles)
69      % hObject    handle to Opciones (see GCBO)
70      % eventdata  reserved - to be defined in a future version of MATLAB
71      % handles    structure with handles and user data (see GUIDATA)
72
73
74      % --------------------------------------------------------------------
75      function Ingresar_administrador_Callback(hObject, eventdata, handles)
76 -    passwordfig('administrador','2022');
77 -    uiwait();
78 -    close(handles.output);
79 -    Interfaz_administrador;
```

Figura 6-3: Inicio, parte 3 del código. Elaboración propia.

6.2. Código en Matlab de la ventana Profsalud.

```
function varargout = Interfaz_Profsalud(varargin)
% INTERFAZ_PROFSALUD MATLAB code for Interfaz_Profsalud.fig

% Last Modified by GUIDE v2.5 07-May-2022 18:43:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Interfaz_Profsalud_OpeningFcn, ...
                   'gui_OutputFcn',  @Interfaz_Profsalud_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Interfaz_Profsalud is made visible.
function Interfaz_Profsalud_OpeningFcn(hObject, eventdata, handles, varargin)
movegui('center');
```

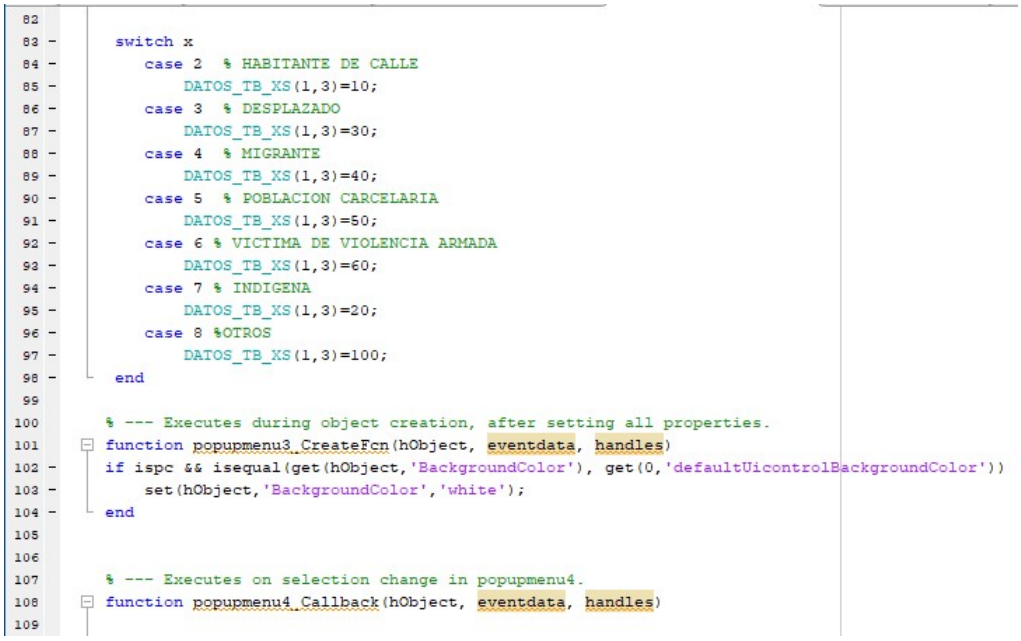
Figura 6-4: Profsalud, parte 1 del código. Elaboración propia.

```
movegui('center');

axes(handles.axes2)
[x,map]=imread('Portada.jpg');%leer la imagen
image(x);
colormap(map);
axis off
hold on

global DATOS_TB_XS

DATOS_TB_XS =zeros(1,6);

handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Interfaz_Profsalud_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

global DATOS_TB_XS
```

Figura 6-5: Profsalud, parte 2 del código. Elaboración propia.

```
global DATOS_TB_XS

 x = get(hObject,'value');

    switch x
    case 2

        DATOS_TB_XS(1,1)= 1;
    case 3
        DATOS_TB_XS(1,1)= -1;
    end

% assignin('base', 'DATOS_TB_X',DATOS_TB_XS)

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)

global DATOS_TB_XS

 x = get(hObject,'value');

```

Figura 6-6: Profsalud, parte 3 del código. Elaboración propia.



```
  switch x
     case 2  % HABITANTE DE CALLE
         DATOS_TB_XS(1,3)=10;
     case 3  % DESPLAZADO
         DATOS_TB_XS(1,3)=30;
     case 4  % MIGRANTE
         DATOS_TB_XS(1,3)=40;
     case 5  % POBLACION CARCELARIA
         DATOS_TB_XS(1,3)=50;
     case 6 % VICTIMA DE VIOLENCIA ARMADA
         DATOS_TB_XS(1,3)=60;
     case 7 % INDIGENA
         DATOS_TB_XS(1,3)=20;
     case 8 %OTROS
         DATOS_TB_XS(1,3)=100;
  end

  % --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


  % --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)

```

Figura 6-7: Profsalud, parte 4 del código. Elaboración propia.

```
109
110 -    global DATOS_TB_XS
111
112 -     x = get(hObject,'value');
113
114 -     switch x
115 -        case 2  % ANTONIO NARIÑO
116 -            DATOS_TB_XS(1,4)=1;
117 -        case 3  % BARRIOS UNIDOS
118 -            DATOS_TB_XS(1,4)=2;
119 -        case 4  % BOSA
120 -            DATOS_TB_XS(1,4)=3;
121 -        case 5  % CHAPINERO
122 -            DATOS_TB_XS(1,4)=4;
123 -        case 6 % CIUDAD BOLIVAR
124 -            DATOS_TB_XS(1,4)=5;
125 -        case 7 % ENGATIVA
126 -            DATOS_TB_XS(1,4)=6;
127 -        case 8 %FONTIBON
128 -            DATOS_TB_XS(1,4)=7;
129 -        case 9 %KENNEDY
130 -            DATOS_TB_XS(1,4)=8;
131 -        case 10 %LA CANDELARIA
132 -            DATOS_TB_XS(1,4)=9;
133 -        case 11 %LOS MARTIRES
134 -            DATOS_TB_XS(1,4)=10;
135 -        case 12 %PUENTE ARANDA
136 -            DATOS_TB_XS(1,4)=11;
```

Figura 6-8: Profsalud, parte 5 del código. Elaboración propia.

```
136 -            DATOS_TB_XS(1,4)=11;
137 -        case 13 %RAFAEL URIBE URIBE
138 -            DATOS_TB_XS(1,4)=12;
139 -        case 14 %SAN CRISTOBAL
140 -            DATOS_TB_XS(1,4)=13;
141 -        case 15 %SANTAFE
142 -            DATOS_TB_XS(1,4)=14;
143 -        case 16 %SUBA
144 -            DATOS_TB_XS(1,4)=15;
145 -        case 17 %TEUSAQUILLO
146 -            DATOS_TB_XS(1,4)=16;
147 -        case 18 %TUNJUELITO
148 -            DATOS_TB_XS(1,4)=17;
149 -        case 19 %USAQUEN
150 -            DATOS_TB_XS(1,4)=18;
151 -        case 20 %USME
152 -            DATOS_TB_XS(1,4)=19;
153 -        case 21 %NO APLICA
154 -            DATOS_TB_XS(1,4)=20;
155 -        case 22 %SIN DATO
156 -            DATOS_TB_XS(1,4)=21;
157 -     end
158
159     % --- Executes during object creation, after setting all properties.
160     function popupmenu4_CreateFcn(hObject, eventdata, handles)
161 -   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
162 -       set(hObject,'BackgroundColor','white');
163 -   end
```

Figura 6-9: Profsalud, parte 6 del código. Elaboración propia.

```
163 -   end
164
165     % --- Executes on selection change in popupmenu5.
166     function popupmenu5_Callback(hObject, eventdata, handles)
167
168 -   global DATOS_TB_XS
169
170 -    x = get(hObject,'value');
171
172 -       switch x
173 -       case 2 %POSITIVO
174 -           DATOS_TB_XS(1,5)= 1;
175 -       case 3 %NEGATIVO
176 -           DATOS_TB_XS(1,5)= -1;
177 -       case 4 %SIN DATO
178 -           DATOS_TB_XS(1,5)= -10;
179 -       end
180
181     % --- Executes during object creation, after setting all properties.
182     function popupmenu5_CreateFcn(hObject, eventdata, handles)
183 -   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
184 -       set(hObject,'BackgroundColor','white');
185 -   end
186
187
188     % --- Executes on selection change in popupmenu6.
189     function popupmenu6_Callback(hObject, eventdata, handles)
190 -   global DATOS_TB_XS
```

Figura 6-10: Profsalud, parte 7 del código. Elaboración propia.

```
190 -   global DATOS_TB_XS
191
192 -    x = get(hObject,'value');
193
194 -       switch x
195 -       case 2 %POSITIVO
196 -           DATOS_TB_XS(1,6)= 1;
197 -       case 3 %NEGATIVO
198 -           DATOS_TB_XS(1,6)= -1;
199 -       case 4 %SIN DATO
200 -           DATOS_TB_XS(1,6)= -10;
201 -       end
202
203     % --- Executes during object creation, after setting all properties.
204     function popupmenu6_CreateFcn(hObject, eventdata, handles)
205 -   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
206 -       set(hObject,'BackgroundColor','white');
207 -   end
208
209     % --------------------------------------------------------------------
210     function Cerrar_sesion_Callback(hObject, eventdata, handles)
211 -   seleccion = questdlg('¿Desea cerrar la interfaz?','Pregunta','Si','No','No');
212 -   if strcmp(seleccion,'No')
213 -       return
214 -   else
215 -      close(handles.output);
216 -   end
217
```

Figura 6-11: Profsalud, parte 8 del código. Elaboración propia.

```
217
218
219       % --- Executes on button press in pushbutton1.
220     function pushbutton1_Callback(hObject, eventdata, handles)
221
222 -     global DATOS_TB_XS
223
224 -     x = num2str(get(handles.edit4, 'string'));
225 -     m =100;
226
227 -     for in=1:1:m
228 -         l = num2str(in);
229 -         y = strcmp(x,l);%recorre la columna comparando con NumID
230
231 -         if y==1
232 -             edad = in;
233 -         end
234 -     end
235
236 -     DATOS_TB_XS(1,2)= edad;
237       % assignin('base', 'DATOS_TB_XS',DATOS_TB_XS)
238
239 -     xl=DATOS_TB_XS./max(abs(DATOS_TB_XS));
240 -     load('red_final.mat');
241
242 -      y2_test = red_final{1}(xl')';
243 -      y2s_test=(y2_test >= 0.5)*2 -1;
244
```

Figura 6-12: Profsalud, parte 9 del código. Elaboración propia.

```
244
245 -     txt1 = cellstr('Alto');
246 -     txt2 = cellstr('Bajo');
247 -     if y2s_test==1
248 -        set(handles.text14,'string',txt1,'Fontsize',18,'BackgroundColor','r');
249 -     else
250 -        set(handles.text14,'string',txt2,'Fontsize',18,'BackgroundColor','g');
251 -     end
252
253 -   Sensibilidad=(0.95)*100;
254 -   Especificidad=(0.1)*100;
255
256 -   set (handles.text20, 'String',Sensibilidad);
257 -   set (handles.text21, 'String',Especificidad);
258
259       % error=(0.17)*100D
260
261       function edit4_Callback(hObject, eventdata, handles)
262       % hObject    handle to edit4 (see GCBO)
263       % eventdata  reserved - to be defined in a future version of MATLAB
264       % handles    structure with handles and user data (see GUIDATA)
265
266       % --- Executes during object creation, after setting all properties.
267       function edit4_CreateFcn(hObject, eventdata, handles)
268       % Hint: edit controls usually have a white background on Windows.
269       %       See ISPC and COMPUTER.
270 -     if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
271 -         set(hObject,'BackgroundColor','white');
272 -     end
```

Figura 6-13: Profsalud, parte 10 del código. Elaboración propia.

6.3. Código en Matlab de la ventana administrador.

```
function varargout = Interfaz_administrador(varargin)
% INTERFAZ_ADMINISTRADOR MATLAB code for Interfaz_administrador.fig
% Last Modified by GUIDE v2.5 07-May-2022 21:16:25

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Interfaz_administrador_OpeningFcn, ...
                   'gui_OutputFcn',  @Interfaz_administrador_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Interfaz_administrador is made visible.
function Interfaz_administrador_OpeningFcn(hObject, eventdata, handles, varargin)

movegui('center');
```

Figura 6-14: Administrador, parte 1 del código. Elaboración propia.

```
movegui('center');

set(handles.uitable1);%limpia la tabla

%Mostrar la imagen de portada
axes(handles.axes1)
[x,map]=imread('Portada.jpg');%leer la imagen
image(x);
colormap(map);
axis off
hold on


% Choose default command line output for Interfaz_administrador
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = Interfaz_administrador_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


%Menu
% --------------------------------------------------------------------
function Archivo_Callback(hObject, eventdata, handles)
```

Figura 6-15: Administrador, parte 2 del código. Elaboración propia.

```
function Archivo_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Importar_Callback(hObject, eventdata, handles)
%Importar una base de datos en formato Excel.
[nombre, ruta] = uigetfile({'*.xlsx', 'Archivo de Excel'}, ...
                           'Escoge un archivo','C:\Users\Tatiana\Documents\Pregrado Ing Biomédica\T

[M, string, Datos] = xlsread([ruta, nombre]);
NomColumnas = Datos(1,:); %Separar el encabezado del arreglo y guardarlo como un arreglo nuevo.
Datos(1,:) = []; %Borrar el encabezado de los datos del arreglo.
NomFilas = 1:size(Datos, 1);

set(handles.uitable1, 'Data', Datos, 'ColumnName', NomColumnas, ...
    'ColumnEditable', logical(1:size(Datos,2)), 'RowName', NomFilas);

%Extraer datos de la tabla
Datos = get(handles.uitable1, 'Data');
% assignin('base', 'Datos',Datos)

Sexo = Datos(:,1);
Edad = Datos(:,2);
Pertenencia_etnica= Datos(:,3);
Grupo_poblacional = Datos(:,4);
```

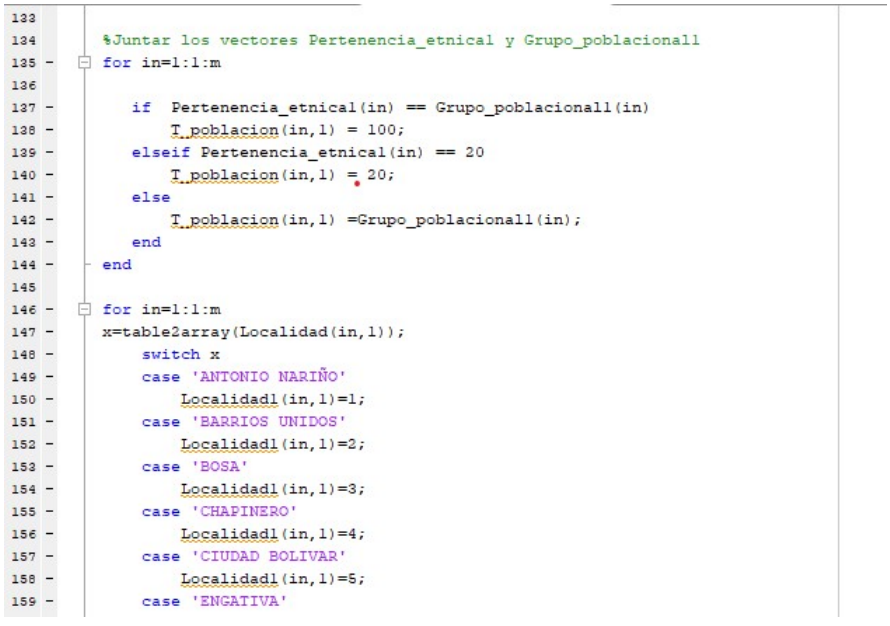
Figura 6-16: Administrador, parte 3 del código. Elaboración propia.

```
Grupo_poblacional = Datos(:,4);
Localidad = Datos(:,5);
VIH = Datos(:,6);
TAR = Datos(:,7);
Nivel = Datos(:,8);
Nivel2 = Datos(:,9);


[m,~]=size(Datos);


for in=1:1:m
x=table2array(Sexo(in,1));
    switch x
    case 'M'
        Sexo1(in,1)=1;
    case 'F'
        Sexo1(in,1)= -1;
    end
end


for in=1:1:m
Edad1(in,1)=table2array(Edad(in,1));
end
```

Figura 6-17: Administrador, parte 4 del código. Elaboración propia.

```
106
107 -   for in=1:1:m
108 -   x=table2array(Pertenencia_etnica(in,1));
109 -       switch x
110 -       case 'INDIGENA'
111 -           Pertenencia_etnical(in,1)=20;
112 -       case 'OTRO'
113 -           Pertenencia_etnical(in,1)=100;
114 -       end
115 -   end
116 -   for in=1:1:m
117 -   x=table2array(Grupo_poblacional(in,1));
118 -       switch x
119 -       case 'HABITANTE DE CALLE'
120 -           Grupo_poblacionall(in,1)=10;
121 -       case 'OTROS'
122 -           Grupo_poblacionall(in,1)=100;
123 -       case 'DESPLAZADO'
124 -           Grupo_poblacionall(in,1)=30;
125 -       case 'MIGRANTE'
126 -           Grupo_poblacionall(in,1)=40;
127 -       case 'POBLACION CARCELARIA'
128 -           Grupo_poblacionall(in,1)=50;
129 -        case 'VICTIMA DE VIOLENCIA ARMADA'
130 -           Grupo_poblacionall(in,1)=60;
131 -       end
132 -   end
```

Figura 6-18: Administrador, parte 5 del código. Elaboración propia.



```
133
134     %Juntar los vectores Pertenencia_etnical y Grupo_poblacionall
135 -   for in=1:1:m
136
137 -      if  Pertenencia_etnical(in) == Grupo_poblacionall(in)
138 -          T_poblacion(in,1) = 100;
139 -      elseif Pertenencia_etnical(in) == 20
140 -          T_poblacion(in,1) = 20;
141 -      else
142 -          T_poblacion(in,1) =Grupo_poblacionall(in);
143 -      end
144 -   end
145
146 -   for in=1:1:m
147 -   x=table2array(Localidad(in,1));
148 -       switch x
149 -       case 'ANTONIO NARIÑO'
150 -           Localidadl(in,1)=1;
151 -       case 'BARRIOS UNIDOS'
152 -           Localidadl(in,1)=2;
153 -       case 'BOSA'
154 -           Localidadl(in,1)=3;
155 -       case 'CHAPINERO'
156 -           Localidadl(in,1)=4;
157 -       case 'CIUDAD BOLIVAR'
158 -           Localidadl(in,1)=5;
159 -       case 'ENGATIVA'
```

Figura 6-19: Administrador, parte 6 del código.. Elaboración propia.

```
159 -        case 'ENGATIVA'
160 -            Localidadl(in,1)=6;
161 -        case 'FONTIBON'
162 -            Localidadl(in,1)=7;
163 -        case 'KENNEDY'
164 -            Localidadl(in,1)=8;
165 -        case 'LA CANDELARIA'
166 -            Localidadl(in,1)=9;
167 -        case 'LOS MARTIRES'
168 -            Localidadl(in,1)=10;
169 -        case 'PUENTE ARANDA'
170 -            Localidadl(in,1)=11;
171 -        case 'RAFAEL URIBE URIBE'
172 -            Localidadl(in,1)=12;
173 -        case 'SAN CRISTOBAL'
174 -            Localidadl(in,1)=13;
175 -        case 'SANTAFE'
176 -            Localidadl(in,1)=14;
177 -        case 'SUBA'
178 -            Localidadl(in,1)=15;
179 -        case 'TEUSAQUILLO'
180 -            Localidadl(in,1)=16;
181 -        case 'TUNJUELITO'
182 -            Localidadl(in,1)=17;
183 -        case 'USAQUEN'
```

Figura 6-20: Administrador, parte 7 del código. Elaboración propia.

```
184 -            Localidadl(in,1)=18;
185 -        case 'USME'
186 -            Localidadl(in,1)=19;
187 -        case 'NO APLICA'
188 -            Localidadl(in,1)=20;
189 -        case 'SIN DATO'
190 -            Localidadl(in,1)=21;
191 -        end
192 -    end
193
194 -    for in=1:1:m
195 -    x=table2array(VIH(in,1));
196 -        switch x
197 -        case 'POSITIVO'
198 -            VIH1(in,1)=1;
199 -        case 'NEGATIVO'
200 -            VIH1(in,1)= -1;
201 -        case 'SIN DATO'
202 -            VIH1(in,1)= -10;
203 -        end
204 -    end
205
206 -    for in=1:1:m
207 -    x=table2array(TAR(in,1));
208 -        switch x
209 -        case 'POSITIVO'
210 -            TAR1(in,1)=1;
```

Figura 6-21: Administrador, parte 8 del código. Elaboración propia.

```
        case 'NEGATIVO'
            TAR1(in,1)= -1;
        case 'SIN DATO'
            TAR1(in,1)= -10;
        end
    end

    for in=1:1:m
    x=table2array(Nivel(in,1));
        switch x
        case 'A'
            Nivel1(in,1)=1;
        case 'M'
            Nivel1(in,1)=2;
        case 'B'
            Nivel1(in,1)=3;
        end
    end

    for in=1:1:m
    x=table2array(Nivel2(in,1));
        switch x
        case 'A'
            Nivel2_1(in,1)=1;
        case 'B'
            Nivel2_1(in,1)=2;

```

Figura 6-22: Administrador, parte 9 del código. Elaboración propia.

```
        end
    end

    DATOS_TB_X =[Sexo1 Edad1 T_poblacion Localidad1 VIH1 TAR1];


    save('DATOS_TB_X','DATOS_TB_X');
    save('Nivel1','Nivel1');
    save('Nivel2_1','Nivel2_1');
    %
    % assignin('base', 'DATOS_TB_X',DATOS_TB_X)
    guidata(hObject, handles);


    % --------------------------------------------------------------------
    function Cerrar_sesion_Callback(hObject, eventdata, handles)
    % Cerrar_sesion  la GUI

    seleccion = questdlg('¿Desea cerrar la interfaz?','Pregunta','Si','No','No');
    if strcmp(seleccion,'No')
        return
    else
       close(handles.output);
    end

    % --- Fuzzy C-means
    function pushbutton7_Callback(hObject, eventdata, handles)
```

Figura 6-23: Administrador, parte 10 del código. Elaboración propia.

```
265
266 -   load('DATOS_TB_X.mat');
267
268 -   x=DATOS_TB_X./max(abs(DATOS_TB_X)); %normalizar los valores
269
270 -   Me=4.1; %exponente para la matriz de particion fuzzy
271 -   it =100; %Maximum number of iterations
272 -   Mof=1e-19;%Minimum improvement in objective function between two consecutive iterations, specified as a positive scalar.
273 -   options = [Me it Mof true];
274 -   Nc=2;  %numero de clusters
275 -   [C,Particion_difusa,objFunc] = fcm(x,Nc,options); % fuzzy C-Means
276
277 -   save('Particion_difusa','Particion_difusa');
278 -   Fig_fuzzy_2C;
279
280     % --- Fuzzy C-means
281     function pushbutton5_Callback(hObject, eventdata, handles)
282
283 -   load('DATOS_TB_X.mat');
284 -   x=DATOS_TB_X./max(abs(DATOS_TB_X)); %normalizar los valores
285
286 -   Me=4.1; %exponente para la matriz de particion fuzzy
287 -   it =100; %Maximum number of iterations
288 -   Mof=1e-17;%Minimum improvement in objective function between two consecutive iterations, specified as a positive scalar.
289 -   options = [Me it Mof true];
290 -   Nc=3;  %numero de clusters
291 -   [C,Particion_difusa,objFunc] = fcm(x,Nc,options); % fuzzy C-Means
```

Figura 6-24: Administrador, parte 11 del código. Elaboración propia.

```
292
293 -   save('Particion_difusa','Particion_difusa');%U
294
295 -   Fig_fuzzy_3C;
296
297     % --- K-means
298     function pushbutton8_Callback(hObject, eventdata, handles)
299
300 -   load('DATOS_TB_X.mat');
301
302 -   x=DATOS_TB_X./max(abs(DATOS_TB_X)); %normalizar los valores
303 -   Nc=2;
304 -   [idx,C,sumd,Distancias_Pcentral] = kmeans(x,Nc);% K-Means
305
306 -   save('Distancias_Pcentral','Distancias_Pcentral');%D
307 -   save('idx','idx');
308
309 -   Fig_kmeans_2C;
310
311     % --- K-means
312     function pushbutton6_Callback(hObject, eventdata, handles)
313
314 -   load('DATOS_TB_X.mat');
315 -   x=DATOS_TB_X./max(abs(DATOS_TB_X)); %normalizar los valores
316 -   Nc=3;
317 -   [idx,C,sumd,Distancias_Pcentral] = kmeans(x,Nc);% K-Means
318
```

Figura 6-25: Administrador, parte 12 del código. Elaboración propia.

```
291 -     [C,Particion_difusa,objFunc] = fcm(x,Nc,options); % fuzzy C-Means
292
293 -     save('Particion_difusa','Particion_difusa');%U
294
295 -     Fig_fuzzy_3C;
296
297       % --- K-means
298       function pushbutton8_Callback(hObject, eventdata, handles)
299
300 -     load('DATOS_TB_X.mat');
301
302 -     x=DATOS_TB_X./max(abs(DATOS_TB_X)); %normalizar los valores
303 -     Nc=2;
304 -     [idx,C,sumd,Distancias_Pcentral] = kmeans(x,Nc);% K-Means
305
306 -     save('Distancias_Pcentral','Distancias_Pcentral');%D
307 -     save('idx','idx');
308
309 -     Fig_kmeans_2C;
310
311       % --- K-means
312       function pushbutton6_Callback(hObject, eventdata, handles)
313
314 -     load('DATOS_TB_X.mat');
315 -     x=DATOS_TB_X./max(abs(DATOS_TB_X)); %normalizar los valores
316 -     Nc=3;
317 -     [idx,C,sumd,Distancias_Pcentral] = kmeans(x,Nc);% K-Means
```

Figura 6-26: Administrador, parte 13 del código. Elaboración propia.

```
317 -     [idx,C,sumd,Distancias_Pcentral] = kmeans(x,Nc);% K-Means
318
319 -     save('Distancias_Pcentral','Distancias_Pcentral');%D
320 -     save('idx','idx');
321
322 -     Fig_kmeans_3C;
323
324       % --- Red neuronal.
325       function pushbutton11_Callback(hObject, eventdata, handles)
326
327 -     x = num2str(get(handles.edit4, 'string'));
328 -     m =200;
329
330 -     for in=1:1:m
331 -         l = num2str(in);
332 -         y = strcmp(x,l);%recorre la columna comparando con NumID
333
334 -         if y==1
335 -             Num_neuronas= in;
336 -         end
337 -     end
338
339 -     save('Num_neuronas','Num_neuronas');
340 -     Fig_Red_neuronal;
341
342       function edit4_Callback(hObject, eventdata, handles)
343       % hObject    handle to edit4 (see GCBO)
344       % eventdata  reserved - to be defined in a future version of MATLAB
```

Figura 6-27: Administrador, parte 14 del código. Elaboración propia.

```
329
330 -   for in=1:1:m
331 -       l = num2str(in);
332 -       y = strcmp(x,l);%recorre la columna comparando con NumID
333
334 -       if y==1
335 -           Num_neuronas= in;
336 -       end
337 -   end
338
339 -   save('Num_neuronas','Num_neuronas');
340 -   Fig_Red_neuronal;
341
342     function edit4_Callback(hObject, eventdata, handles)
343     % hObject    handle to edit4 (see GCBO)
344     % eventdata  reserved - to be defined in a future version of MATLAB
345     % handles    structure with handles and user data (see GUIDATA)
346
347
348     % --- Executes during object creation, after setting all properties.
349     function edit4_CreateFcn(hObject, eventdata, handles)
350     % Hint: edit controls usually have a white background on Windows.
351     %       See ISPC and COMPUTER.
352 -   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
353 -       set(hObject,'BackgroundColor','white');
354 -   end
355
```

Figura 6-28: Administrador, parte 15 del código. Elaboración propia.

6.3.1. Código en Matlab de la ventana Fuzzy C-means con 2 clúster

```
1     function varargout = Fig_fuzzy_2C(varargin)
2     % FIG_FUZZY_2C MATLAB code for Fig_fuzzy_2C.fig
3     % Last Modified by GUIDE v2.5 07-May-2022 23:14:42
4
5     % Begin initialization code - DO NOT EDIT
6 -   gui_Singleton = 1;
7 -   gui_State = struct('gui_Name',       mfilename, ...
8                        'gui_Singleton',  gui_Singleton, ...
9                        'gui_OpeningFcn', @Fig_fuzzy_2C_OpeningFcn, ...
10                       'gui_OutputFcn',  @Fig_fuzzy_2C_OutputFcn, ...
11                       'gui_LayoutFcn',  [] , ...
12                       'gui_Callback',   []);
13 -   if nargin && ischar(varargin{1})
14 -       gui_State.gui_Callback = str2func(varargin{1});
15 -   end
16
17 -   if nargout
18 -       [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
19 -   else
20 -       gui_mainfcn(gui_State, varargin{:});
21 -   end
22
23    % --- Executes just before Fig_fuzzy_2C is made visible.
24    function Fig_fuzzy_2C_OpeningFcn(hObject, eventdata, handles, varargin)
25 -   movegui('center');
26
27 -   axes(handles.axes4);
28 -   [x,map]=imread('Portada.jpg');%leer la imagen
```

Figura 6-29: Fuzzy 2C, parte 1 del código. Elaboración propia.

```
28 -    image(x);
29 -    colormap(map);
30 -    axis off
31 -    hold on
32
33 -    load('Particion_difusa.mat');
34 -    load('Nivel2_1.mat');
35 -    Nc=2;
36
37 -    maxU = max(Particion_difusa);
38 -    index1 = find(Particion_difusa(1,:) == maxU);
39 -    index2 = find(Particion_difusa(2,:) == maxU);
40
41 -    ohm1=[Particion_difusa(1,index1);Particion_difusa(2,index1)];
42 -    ohm2=[Particion_difusa(1,index2);Particion_difusa(2,index2)];
43
44      % calculo de la desviacion standar
45 -    std1=std(ohm1');
46 -    std2=std(ohm2');
47
48      %calculo de los centros
49 -    cen1=mean(ohm1');
50 -    cen2=mean(ohm2');
51
52 -    tam=size(Particion_difusa,2);
53
54 -    nivel_hat= zeros(tam,1);
55 -    for in=1:1:Nc
```

Figura 6-30: Fuzzy 2C, parte 2 del código. Elaboración propia.

```
55 -    for in=1:1:Nc
56 -    dec=0;
57 -    nivel_hat(index2)=in;
58
59 -    if in == 2
60 -        dec=2;
61 -    end
62
63 -    nivel_hat(index1)=in+1-dec;
64 -    Error_c(in)=sum(Nivel2_1~= nivel_hat);
65
66 -    end
67
68 -    Error_prediccion=(min(Error_c)/tam)*100;
69
70 -    axes(handles.axes3)
71 -    plot(ohm1(1,:),ohm1(2,:),'ob','LineWidth',4)
72 -    hold on
73 -    plot(ohm2(1,:),ohm2(2,:),'or','LineWidth',4)
74 -    plot(Particion_difusa(1,:),Particion_difusa(2,:),'.k')
75
76 -    plot(cen1(:,1),cen1(:,2),'k+','LineWidth',2)
77 -    plot(cen2(:,1),cen2(:,2),'+k','LineWidth',2)
78 -    text(cen1(:,1),cen1(:,2),int2str(1),'FontWeight','bold');
79 -    grid on
80
81 -    Niv_predichos = zeros(tam,1);
```

Figura 6-31: Fuzzy 2C, parte 3 del código. Elaboración propia.

```
80
81 -    Niv_predichos = zeros(tam,1);
82
83 -    if  Error_c(1) < Error_c(2)
84 -    legend('Riesgo Alto','Riesgo Bajo');
85 -    Niv_predichos(index1)=1;
86 -    Niv_predichos(index2)=2;
87 -    else
88 -    legend('Riesgo Bajo', 'Riesgo Alto');
89 -    Niv_predichos(index1)=2;
90 -    Niv_predichos(index2)=1;
91 -    end
92 -    title('Agrupamiento por Fuzzy C-means')
93
94 -    axes(handles.axes5)
95 -    confusionchart(Nivel2_1,Niv_predichos,'Title','Matriz de confusion');
96 -    matriz_confu=confusionmat(Nivel2_1,Niv_predichos);%Matriz de confusion
97
98 -    TP = matriz_confu(1,1);% Verdaderos positivos
99 -    TN = matriz_confu(2,2);%Verdaderos negativos
100 -   FP = matriz_confu(2,1);%Falsos positivos
101 -   FN= matriz_confu(1,2);%Falsos negativos
102
103 -   Sensibilidad = (TP/(FN+TP))*100;
104 -   Especificidad = (TN/(TN+FP))*100;
105
106
```

Figura 6-32: Fuzzy 2C, parte 4 del código. Elaboración propia.

```
97
98 -    TP = matriz_confu(1,1);% Verdaderos positivos
99 -    TN = matriz_confu(2,2);%Verdaderos negativos
100 -   FP = matriz_confu(2,1);%Falsos positivos
101 -   FN= matriz_confu(1,2);%Falsos negativos
102
103 -   Sensibilidad = (TP/(FN+TP))*100;
104 -   Especificidad = (TN/(TN+FP))*100;
105
106
107 -   set (handles.text20, 'String',Error_prediccion);
108 -   set (handles.text21, 'String',Sensibilidad);
109 -   set (handles.text22, 'String',Especificidad);
110
111 -   handles.output = hObject;
112 -   guidata(hObject, handles);
113
114     % --- Outputs from this function are returned to the command line.
115     function varargout = Fig_fuzzy_2C_OutputFcn(hObject, eventdata, handles)
116
117     % Get default command line output from handles structure
118 -   varargout{1} = handles.output;
119
120
121     % --- Executes on button press in pushbutton3.
122     function pushbutton3_Callback(hObject, eventdata, handles)
123 -   close(handles.output);
124
```

Figura 6-33: Fuzzy 2C, parte 5 del código. Elaboración propia.

6.3.2. Código en Matlab de la ventana Fuzzy C-means con 3 clúster

```
function varargout = Fig_fuzzy_3C(varargin)
% FIG_FUZZY_3C MATLAB code for Fig_fuzzy_3C.fig
% Last Modified by GUIDE v2.5 07-May-2022 23:44:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fig_fuzzy_3C_OpeningFcn, ...
                   'gui_OutputFcn',  @Fig_fuzzy_3C_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fig_fuzzy_3C is made visible.
function Fig_fuzzy_3C_OpeningFcn(hObject, eventdata, handles, varargin)
movegui('center');

```

Figura 6-34: Fuzzy 3C, parte 1 del código. Elaboración propia.

```

axes(handles.axes2)
[x,map]=imread('Portada.jpg');%leer imagen de portada
image(x);
colormap(map);
axis off
hold on

load('Particion_difusa.mat');
load('Nivell.mat');
Nc=3;

maxU = max(Particion_difusa);
index1 = find(Particion_difusa(1,:) == maxU);
index2 = find(Particion_difusa(2,:) == maxU);
index3 = find(Particion_difusa(3,:) == maxU);

ohm1=[Particion_difusa(1,index1);Particion_difusa(2,index1);Particion_difusa(3,index1)];
ohm2=[Particion_difusa(1,index2);Particion_difusa(2,index2);Particion_difusa(3,index2)];
ohm3=[Particion_difusa(1,index3);Particion_difusa(2,index3);Particion_difusa(3,index3)];

% calculo de la desviacion standar
std1=std(ohm1');
std2=std(ohm2');
std3=std(ohm3');

%calculo de los centros
cen1=mean(ohm1');
```

Figura 6-35: Fuzzy 3C, parte 2 del código. Elaboración propia.

```
55 -    cen1=mean(ohm1');
56 -    cen2=mean(ohm2');
57 -    cen3=mean(ohm3');
58
59 -    tam=size(Particion_difusa,2);
60
61 -    nivel_hat= zeros(tam,1);
62 -    for in=1:1:Nc
63 -    dec=0;
64 -    nivel_hat(index1)=in;
65
66 -    if in == 3
67 -        dec=3;
68 -    end
69
70 -    nivel_hat(index2)=in+1-dec;
71
72 -    if in >=2
73 -        dec=3;
74 -    end
75
76 -    nivel_hat(index3)=in+2-dec;
77 -    Error_c(in)=sum(Nivel1~=nivel_hat);
78 -    end
79
80 -    Error_prediccion=(min(Error_c)/tam)*100;
```

Figura 6-36: Fuzzy 3C, parte 3 del código. Elaboración propia.

```
82      %
83 -    axes(handles.axes1)
84
85 -    scatter3(ohm1(1,:),ohm1(2,:),ohm1(3,:),'ob','LineWidth',4)
86 -    hold on
87 -    scatter3(ohm2(1,:),ohm2(2,:),ohm2(3,:),'or','LineWidth',4)
88 -    scatter3(ohm3(1,:),ohm3(2,:),ohm3(3,:),'og','LineWidth',4)
89 -    scatter3(Particion_difusa(1,:),Particion_difusa(2,:),Particion_difusa(3,:),'.k')
90
91 -    scatter3(cen1(:,1),cen1(:,2),cen1(:,3),'k+','LineWidth',2)
92 -    scatter3(cen2(:,1),cen2(:,2),cen2(:,3),'+k','LineWidth',2)
93 -    scatter3(cen3(:,1),cen3(:,2),cen3(:,3),'+k','LineWidth',2)
94 -    text(cen1(:,1),cen1(:,2),cen1(:,3),int2str(1),'FontWeight','bold');
95 -    grid on
96
97 -    n=30;
98 -    [x, y, z] = sphere(n);
99 -    hold on
100 -   mesh(3*std1(:,1)*x+cen1(:,1), 3*std1(:,2)*y+cen1(:,2), 3*std1(:,3)*z+cen1(:,3),'LineWidth',0.01, 'FaceAlpha', 0.1)
101 -   mesh(3*std2(:,1)*x+cen2(:,1), 3*std2(:,2)*y+cen2(:,2), 3*std2(:,3)*z+cen2(:,3),'LineWidth',0.01, 'FaceAlpha', 0.1)
102 -   mesh(3*std3(:,1)*x+cen3(:,1), 3*std3(:,2)*y+cen3(:,2), 3*std3(:,3)*z+cen3(:,3),'LineWidth',0.01, 'FaceAlpha', 0.1)
103
104 -   Niv_predichos = zeros(tam,1);
105
106 -   if  Error_c(1) < Error_c(2) &&  Error_c(1) < Error_c(3)
107 -   legend('Riesgo Alto','Riesgo Medio','Riesgo Bajo')
108 -   Niv_predichos(index1)=1;
109 -   Niv_predichos(index2)=2;
```

Figura 6-37: Fuzzy 3C, parte 4 del código. Elaboración propia.

```
109 -    Niv_predichos(index2)=2;
110 -    Niv_predichos(index3)=3;
111 -    end
112
113 -    if Error_c(2) < Error_c(1) && Error_c(2) < Error_c(3)
114 -    legend('Riesgo Medio','Riesgo Bajo','Riesgo Alto')
115 -    Niv_predichos(index1)=2;
116 -    Niv_predichos(index2)=3;
117 -    Niv_predichos(index3)=1;
118 -    end
119
120 -    if Error_c(3) < Error_c(1)&& Error_c(3) < Error_c(2)
121 -    legend('Riesgo Bajo', 'Riesgo Alto','Riesgo Medio')
122 -    Niv_predichos(index1)=3;
123 -    Niv_predichos(index2)=1;
124 -    Niv_predichos(index3)=2;
125 -    end
126
127 -    hold off
128 -    title('Agrupamiento por Fuzzy C-means')
129 -    axis tight
130
131 -    axes(handles.axes3)
132 -    confusionchart(Nivel1,Niv_predichos,'Title','Matriz de confusion');
133 -    matriz_confu=confusionmat(Nivel1,Niv_predichos);%Matriz de confusion
134      % Verdaderos positivos
135 -    TP=[matriz_confu(1,1)+matriz_confu(2,2)+matriz_confu(3,3)];
136
```

Figura 6-38: Fuzzy 3C, parte 5 del código. Elaboración propia.

```
136
137      %Verdaderos negativos
138 -    TN_1=[matriz_confu(2,2)+matriz_confu(2,3)+matriz_confu(3,2)+matriz_confu(3,3)];
139 -    TN_2=[matriz_confu(1,1)+matriz_confu(1,3)+matriz_confu(3,1)+matriz_confu(3,3)];
140 -    TN_3=[matriz_confu(1,1)+matriz_confu(1,2)+matriz_confu(2,1)+matriz_confu(2,2)];
141 -    TN=[TN_1+TN_2+TN_3];
142
143      %Falsos positivos
144 -    FP_1=[matriz_confu(2,1)+matriz_confu(3,1)];
145 -    FP_2=[matriz_confu(1,2)+matriz_confu(3,2)];
146 -    FP_3=[matriz_confu(1,3)+matriz_confu(2,3)];
147 -    FP=[FP_1+FP_2+FP_3];
148
149      %Falsos negativos
150 -    FN_1=[matriz_confu(1,2)+matriz_confu(1,3)];
151 -    FN_2=[matriz_confu(2,1)+matriz_confu(2,3)];
152 -    FN_3=[matriz_confu(3,1)+matriz_confu(3,2)];
153 -    FN=[FN_1+FN_2+FN_3];
154
155
156 -     Sensibilidad = (TP/(FN+TP))*100;
157 -     Especificidad = (TN/(TN+FP))*100;
158
159 -    set (handles.text13, 'String',Error_prediccion);
160 -    set (handles.text14, 'String',Sensibilidad);
161 -    set (handles.text15, 'String',Especificidad);
162
163 -    handles.output = hObject;
```

Figura 6-39: Fuzzy 3C, parte 6 del código. Elaboración propia.

```
153 -   FN=[FN_1+FN_2+FN_3];
154
155
156 -    Sensibilidad = (TP/(FN+TP))*100;
157 -    Especificidad = (TN/(TN+FP))*100;
158
159 -   set (handles.text13, 'String',Error_prediccion);
160 -   set (handles.text14, 'String',Sensibilidad);
161 -   set (handles.text15, 'String',Especificidad);
162
163 -   handles.output = hObject;
164     % Update handles structure
165 -   guidata(hObject, handles);
166
167
168     % --- Outputs from this function are returned to the command line.
169     function varargout = Fig_fuzzy_3C_OutputFcn(hObject, eventdata, handles)
170
171     % Get default command line output from handles structure
172 -   varargout{1} = handles.output;
173
174
175     % --- Executes on button press in pushbutton1.
176     function pushbutton1_Callback(hObject, eventdata, handles)
177
178 -   close(handles.output);
179
```

Figura 6-40: Fuzzy 3C, parte 7 del código. Elaboración propia.

6.3.3. Código en Matlab de la ventana K-means con 2 clúster

```
1     function varargout = Fig_kmeans_2C(varargin)
2     % FIG_KMEANS_2C MATLAB code for Fig_kmeans_2C.fig
3     % Last Modified by GUIDE v2.5 07-May-2022 23:43:42
4
5     % Begin initialization code - DO NOT EDIT
6 -   gui_Singleton = 1;
7 -   gui_State = struct('gui_Name',       mfilename, ...
8                        'gui_Singleton',  gui_Singleton, ...
9                        'gui_OpeningFcn', @Fig_kmeans_2C_OpeningFcn, ...
10                       'gui_OutputFcn',  @Fig_kmeans_2C_OutputFcn, ...
11                       'gui_LayoutFcn',  [] , ...
12                       'gui_Callback',   []);
13 -   if nargin && ischar(varargin{1})
14 -       gui_State.gui_Callback = str2func(varargin{1});
15 -   end
16
17 -   if nargout
18 -       [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
19 -   else
20 -       gui_mainfcn(gui_State, varargin{:});
21 -   end
22    % End initialization code - DO NOT EDIT
23
24
25    % --- Executes just before Fig_kmeans_2C is made visible.
26    function Fig_kmeans_2C_OpeningFcn(hObject, eventdata, handles, varargin)
27 -  movegui('center');
28
```

Figura 6-41: kmeans 2C, parte 1 del código. Elaboración propia.

```
28
29 -    axes(handles.axes3)
30 -    [x,map]=imread('Portada.jpg');%leer la imagen
31 -    image(x);
32 -    colormap(map);
33 -    axis off
34 -    hold on
35
36 -    load('Distancias_Pcentral.mat');
37 -    load('idx.mat');
38 -    load('Nivel2_1.mat');
39
40 -    Nc=2;
41 -    Distancias_Pcentral = Distancias_Pcentral';
42 -    index1 = find(idx == 1);
43 -    index2 = find(idx == 2);
44
45 -    ohm1=[Distancias_Pcentral(1,index1);Distancias_Pcentral(2,index1)];
46 -    ohm2=[Distancias_Pcentral(1,index2);Distancias_Pcentral(2,index2)];
47
48      % calculo de la desviacion estandar
49 -    std1=std(ohm1');
50 -    std2=std(ohm2');
51
52      %calculo de los centros
53 -    cen1=mean(ohm1');
54 -    cen2=mean(ohm2');
55
```

Figura 6-42: kmeans 2C, parte 2 del código. Elaboración propia.

```
55
56 -    tam=size(Distancias_Pcentral,2);
57
58 -    nivel_hat= zeros(tam,1);
59 -    for in=1:1:Nc
60 -    dec=0;
61 -    nivel_hat(index2)=in;
62
63 -    if in == 2
64 -        dec=2;
65 -    end
66
67 -    nivel_hat(index1)=in+1-dec;
68 -    Error_c(in)=sum(Nivel2_1~=nivel_hat);
69
70 -    end
71 -    Error_prediccion=(min(Error_c)/tam)*100;
72
73 -    axes(handles.axes2)
74 -    plot(ohm1(1,:),ohm1(2,:),'ob','LineWidth',4)
75 -    hold on
76 -    plot(ohm2(1,:),ohm2(2,:),'or','LineWidth',4)
77 -    plot(Distancias_Pcentral(1,:),Distancias_Pcentral(2,:),'.k')
78
79 -    plot(cen1(:,1),cen1(:,2),'k+','LineWidth',2)
80 -    plot(cen2(:,1),cen2(:,2),'+k','LineWidth',2)
81 -    text(cen1(:,1),cen1(:,2),int2str(1),'FontWeight','bold');
82 -    grid on
```

Figura 6-43: kmeans 2C, parte 3 del código. Elaboración propia.

```
grid on

Niv_predichos = zeros(tam,1);

if  Error_c(1) < Error_c(2)
legend('Riesgo Alto','Riesgo Bajo');
Niv_predichos(index1)=1;
Niv_predichos(index2)=2;
else
legend('Riesgo Bajo', 'Riesgo Alto');
Niv_predichos(index1)=2;
Niv_predichos(index2)=1;
end
title('Agrupamiento por K-means')

axes(handles.axes4)
confusionchart(Nivel2_1,Niv_predichos,'Title','Matriz de confusion');
matriz_confu=confusionmat(Nivel2_1,Niv_predichos);%Matriz de confusion

TP = matriz_confu(1,1);% Verdaderos positivos
TN = matriz_confu(2,2);%Verdaderos negativos
FP = matriz_confu(2,1);%Falsos positivos
FN= matriz_confu(1,2);%Falsos negativos

Sensibilidad = (TP/(FN+TP))*100;
Especificidad = (TN/(TN+FP))*100;

set (handles.text17, 'String',Error_prediccion);
```

Figura 6-44: kmeans 2C, parte 4 del código. Elaboración propia.

```
confusionchart(Nivel2_1,Niv_predichos,'Title','Matriz de confusion');
matriz_confu=confusionmat(Nivel2_1,Niv_predichos);%Matriz de confusion

TP = matriz_confu(1,1);% Verdaderos positivos
TN = matriz_confu(2,2);%Verdaderos negativos
FP = matriz_confu(2,1);%Falsos positivos
FN= matriz_confu(1,2);%Falsos negativos

Sensibilidad = (TP/(FN+TP))*100;
Especificidad = (TN/(TN+FP))*100;

set (handles.text17, 'String',Error_prediccion);
set (handles.text19, 'String',Sensibilidad);
set (handles.text18, 'String',Especificidad);

handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Fig_kmeans_2C_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
close(handles.output);
```

Figura 6-45: kmeans 2C, parte 5 del código. Elaboración propia.

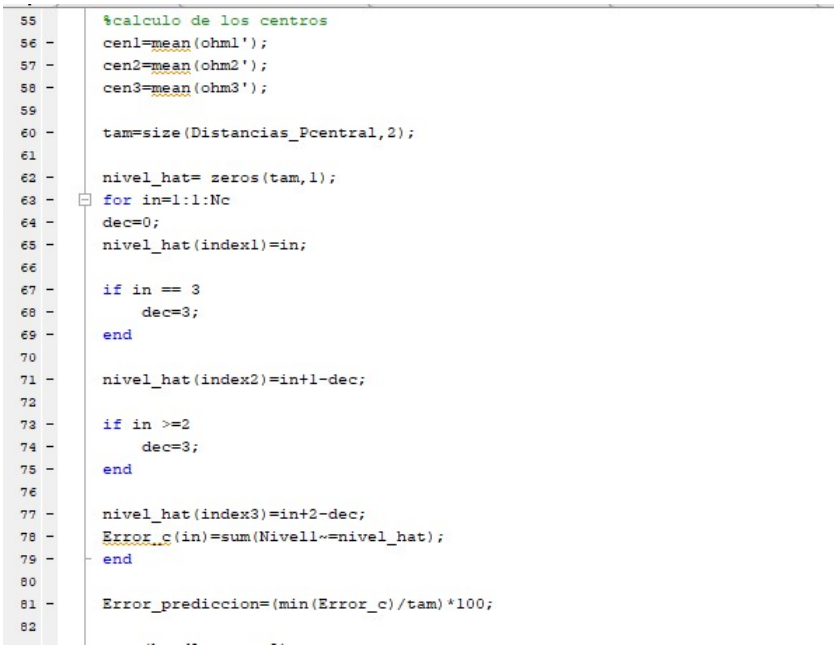
6.3.4. Código en Matlab de la ventana K-means con 3 clúster

```
function varargout = Fig_kmeans_3C(varargin)
% FIG_KMEANS_3C MATLAB code for Fig_kmeans_3C.fig
% Last Modified by GUIDE v2.5 07-May-2022 23:43:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fig_kmeans_3C_OpeningFcn, ...
                   'gui_OutputFcn',  @Fig_kmeans_3C_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Fig_kmeans_3C is made visible.
function Fig_kmeans_3C_OpeningFcn(hObject, eventdata, handles, varargin)
movegui('center');

```

Figura 6-46: kmeans 3C, parte 1 del código. Elaboración propia.

```

axes(handles.axes3)
[x,map]=imread('Portada.jpg');%leer la imagen
image(x);
colormap(map);
axis off
hold on

load('Distancias_Pcentral.mat');
load('idx.mat');
load('Nivell.mat');

Nc=3;
Distancias_Pcentral = Distancias_Pcentral';
index1 = find(idx == 1);
index2 = find(idx == 2);
index3 = find(idx == 3);

ohm1=[Distancias_Pcentral(1,index1);Distancias_Pcentral(2,index1);Distancias_Pcentral(3,index1)];
ohm2=[Distancias_Pcentral(1,index2);Distancias_Pcentral(2,index2);Distancias_Pcentral(3,index2)];
ohm3=[Distancias_Pcentral(1,index3);Distancias_Pcentral(2,index3);Distancias_Pcentral(3,index3)];

% calculo de la desviacion standar
std1=std(ohm1');
std2=std(ohm2');
std3=std(ohm3');

%calculo de los centros
```

Figura 6-47: kmeans 3C, parte 2 del código. Elaboración propia.



```
%calculo de los centros
cen1=mean(ohm1');
cen2=mean(ohm2');
cen3=mean(ohm3');

tam=size(Distancias_Pcentral,2);

nivel_hat= zeros(tam,1);
for in=1:1:Nc
dec=0;
nivel_hat(index1)=in;

if in == 3
    dec=3;
end

nivel_hat(index2)=in+1-dec;

if in >=2
    dec=3;
end

nivel_hat(index3)=in+2-dec;
Error_c(in)=sum(Nivel1~=nivel_hat);
end

Error_prediccion=(min(Error_c)/tam)*100;
```

Figura 6-48: kmeans 3C, parte 3 del código. Elaboración propia.

```
axes(handles.axes2)

scatter3(ohm1(1,:),ohm1(2,:),ohm1(3,:),'ob','LineWidth',4)
hold on
scatter3(ohm2(1,:),ohm2(2,:),ohm2(3,:),'or','LineWidth',4)
scatter3(ohm3(1,:),ohm3(2,:),ohm3(3,:),'og','LineWidth',4)
scatter3(Distancias_Pcentral(1,:),Distancias_Pcentral(2,:),Distancias_Pcentral(3,:),'.k')

scatter3(cen1(:,1),cen1(:,2),cen1(:,3),'k+','LineWidth',2)
scatter3(cen2(:,1),cen2(:,2),cen2(:,3),'+k','LineWidth',2)
scatter3(cen3(:,1),cen3(:,2),cen3(:,3),'+k','LineWidth',2)
text(cen1(:,1),cen1(:,2),cen1(:,3),int2str(1),'FontWeight','bold');
grid on

n=30;
[x, y, z] = sphere(n);
hold on
mesh(3*std1(:,1)*x+cen1(:,1), 3*std1(:,2)*y+cen1(:,2), 3*std1(:,3)*z+cen1(:,3),'LineWidth',0.01, 'FaceAlpha', 0.1)
mesh(3*std2(:,1)*x+cen2(:,1), 3*std2(:,2)*y+cen2(:,2), 3*std2(:,3)*z+cen2(:,3),'LineWidth',0.01, 'FaceAlpha', 0.1)
mesh(3*std3(:,1)*x+cen3(:,1), 3*std3(:,2)*y+cen3(:,2), 3*std3(:,3)*z+cen3(:,3),'LineWidth',0.01, 'FaceAlpha', 0.1)

Niv_predichos = zeros(tam,1);

if  Error_c(1) < Error_c(2) &&  Error_c(1) < Error_c(3)
legend('Riesgo Alto','Riesgo Medio','Riesgo Bajo')
Niv_predichos(index1)=1;
Niv_predichos(index2)=2;
```

Figura 6-49: kmeans 3C, parte 4 del código. Elaboración propia.

```
109 -     Niv_predichos(index2)=2;
110 -     Niv_predichos(index3)=3;
111 -     end
112
113 -     if Error_c(2) < Error_c(1) && Error_c(2) < Error_c(3)
114 -     legend('Riesgo Medio','Riesgo Bajo','Riesgo Alto')
115 -     Niv_predichos(index1)=2;
116 -     Niv_predichos(index2)=3;
117 -     Niv_predichos(index3)=1;
118 -     end
119
120 -     if Error_c(3) < Error_c(1)&& Error_c(3) < Error_c(2)
121 -     legend('Riesgo Bajo', 'Riesgo Alto','Riesgo Medio')
122 -     Niv_predichos(index1)=3;
123 -     Niv_predichos(index2)=1;
124 -     Niv_predichos(index3)=2;
125 -     end
126
127 -     hold off
128 -     title('Agrupamiento por K-means')
129 -     axis tight
130
131
132 -     axes(handles.axes4)
133 -     confusionchart(Nivel1,Niv_predichos,'Title','Matriz de confusion');
134
135 -     matriz_confu=confusionmat(Nivel1,Niv_predichos);%Matriz de confusion
136
```

Figura 6-50: kmeans 3C, parte 5 del código. Elaboración propia.

```
136
137       % Verdaderos positivos
138 -     TP=[matriz_confu(1,1)+matriz_confu(2,2)+matriz_confu(3,3)];
139
140       %Verdaderos negativos
141 -     TN_1=[matriz_confu(2,2)+matriz_confu(2,3)+matriz_confu(3,2)+matriz_confu(3,3)];
142 -     TN_2=[matriz_confu(1,1)+matriz_confu(1,3)+matriz_confu(3,1)+matriz_confu(3,3)];
143 -     TN_3=[matriz_confu(1,1)+matriz_confu(1,2)+matriz_confu(2,1)+matriz_confu(2,2)];
144 -     TN=[TN_1+TN_2+TN_3];
145
146       %Falsos positivos
147 -     FP_1=[matriz_confu(2,1)+matriz_confu(3,1)];
148 -     FP_2=[matriz_confu(1,2)+matriz_confu(3,2)];
149 -     FP_3=[matriz_confu(1,3)+matriz_confu(2,3)];
150 -     FP=[FP_1+FP_2+FP_3];
151
152       %Falsos negativos
153 -     FN_1=[matriz_confu(1,2)+matriz_confu(1,3)];
154 -     FN_2=[matriz_confu(2,1)+matriz_confu(2,3)];
155 -     FN_3=[matriz_confu(3,1)+matriz_confu(3,2)];
156 -     FN=[FN_1+FN_2+FN_3];
157
158 -     Sensibilidad =(TP/(FN+TP))*100;
159 -     Especificidad =(TN/(TN+FP))*100;
160
161 -     set (handles.text15, 'String',Error_prediccion);
162 -     set (handles.text17, 'String',Sensibilidad);
163 -     set (handles.text16, 'String',Especificidad);
```

Figura 6-51: kmeans 3C, parte 6 del código. Elaboración propia.

```
152     %Falsos negativos
153 -   FN_1=[matriz_confu(1,2)+matriz_confu(1,3)];
154 -   FN_2=[matriz_confu(2,1)+matriz_confu(2,3)];
155 -   FN_3=[matriz_confu(3,1)+matriz_confu(3,2)];
156 -   FN=[FN_1+FN_2+FN_3];
157
158 -   Sensibilidad =(TP/(FN+TP))*100;
159 -   Especificidad =(TN/(TN+FP))*100;
160
161 -   set (handles.text15, 'String',Error_prediccion);
162 -   set (handles.text17, 'String',Sensibilidad);
163 -   set (handles.text16, 'String',Especificidad);
164
165 -   handles.output = hObject;
166     % Update handles structure
167 -   guidata(hObject, handles);
168
169     % --- Outputs from this function are returned to the command line.
170     function varargout = Fig_kmeans_3C_OutputFcn(hObject, eventdata, handles)
171
172     % Get default command line output from handles structure
173 -   varargout{1} = handles.output;
174
175     % --- Executes on button press in pushbutton2.
176     function pushbutton2_Callback(hObject, eventdata, handles)
177 -   close(handles.output);
178
```

Figura 6-52: kmeans 3C, parte 7 del código. Elaboración propia.

6.3.5. Código en Matlab de la ventana Red neuronal

```
1     function varargout = Fig_Red_neuronal(varargin)
2     % FIG_RED_NEURONAL MATLAB code for Fig_Red_neuronal.fig
3     % Last Modified by GUIDE v2.5 07-May-2022 22:02:25
4
5     % Begin initialization code - DO NOT EDIT
6 -   gui_Singleton = 1;
7 -   gui_State = struct('gui_Name',       mfilename, ...
8                        'gui_Singleton',  gui_Singleton, ...
9                        'gui_OpeningFcn', @Fig_Red_neuronal_OpeningFcn, ...
10                       'gui_OutputFcn',  @Fig_Red_neuronal_OutputFcn, ...
11                       'gui_LayoutFcn',  [] , ...
12                       'gui_Callback',   []);
13 -   if nargin && ischar(varargin{1})
14 -       gui_State.gui_Callback = str2func(varargin{1});
15 -   end
16
17 -   if nargout
18 -       [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
19 -   else
20 -       gui_mainfcn(gui_State, varargin{:});
21 -   end
22    % End initialization code - DO NOT EDIT
23
24
25    % --- Executes just before Fig_Red_neuronal is made visible.
26    function Fig_Red_neuronal_OpeningFcn(hObject, eventdata, handles, varargin)
27 -  movegui('center');
28
```

Figura 6-53: Red neuronal, parte 1 del código. Elaboración propia.

```
28
29 -    axes(handles.axes5)
30 -    [x,map]=imread('Portada.jpg');%leer la imagen
31 -    image(x);
32 -    colormap(map);
33 -    axis off
34 -    hold on
35
36 -    load('Nivel2_1.mat');
37 -    load('DATOS_TB_X.mat');
38 -    load('Num_neuronas.mat');
39
40 -    d_t=-(Nivel2_1==2)+(Nivel2_1==1);%salidas codificadas
41
42      %Definicion de la red
43 -    tam_train=round(0.7*size(DATOS_TB_X,1)); % entrenar con 70% de datos
44 -    DATOS_TB_X_t=DATOS_TB_X(1:tam_train,:);
45 -    dtrain=d_t(1:tam_train,:);
46 -    xa=DATOS_TB_X_t;
47 -    x=xa./max(abs(xa));%normalizar los datos
48 -    D=[dtrain]; % 70% datos salida
49 -    net = feedforwardnet(Num_neuronas);%numero de neuronas
50 -    net = configure(net,x',D');
51 -    y1 = net(x');
52
53      %Entrenamiento
54 -    net = train(net,x',D');
55 -    y2 = net(x')';          % resultado red entrenada
```

Figura 6-54: Red neuronal, parte 2 del código. Elaboración propia.

```
56 -    y2s=(y2 >= 0.5)*2 -1;
57
58      %Prediccion
59      %probar la red con el 30% de los datos restantes
60 -    DATOS_TB_X_test=DATOS_TB_X(tam_train:end,:);
61 -    dtrain_test=d_t(tam_train:end,:);
62 -    xa_test=DATOS_TB_X_test;
63 -    x_test=xa_test./max(abs(xa_test));   %normalizo los datos
64 -    y2_test = net(x_test')';          % resultado red en el test
65 -    y2s_test=(y2_test >= 0.5)*2 -1;
66
67 -    axes(handles.axes8)
68 -    confusionchart(dtrain_test,y2s_test,'Title','Matriz de confusion')
69
70      %Matriz de confusion
71 -    matriz_confu = confusionmat(dtrain_test,y2s_test);
72 -    TP_test=matriz_confu(2,2);
73 -    TN_test=matriz_confu(1,1);
74 -    FP_test=matriz_confu(1,2);
75 -    FN_test=matriz_confu(2,1);
76
77 -    Sensibilidad= (TP_test/(TP_test+FN_test))*100;
78 -    Especificidad=(TN_test/(TN_test+FP_test))*100;
79 -    Error_prediccion=((FN_test+FP_test)/(TP_test+TN_test+FN_test+FP_test))*100;
80
81 -    set (handles.text16, 'String',Error_prediccion);
82 -    set (handles.text17, 'String',Sensibilidad);
```

Figura 6-55: Red neuronal, parte 3 del código. Elaboración propia.

```
71 -    matriz_confu = confusionmat(dtrain_test,y2s_test);
72 -    TP_test=matriz_confu(2,2);
73 -    TN_test=matriz_confu(1,1);
74 -    FP_test=matriz_confu(1,2);
75 -    FN_test=matriz_confu(2,1);
76
77 -    Sensibilidad= (TP_test/(TP_test+FN_test))*100;
78 -    Especificidad=(TN_test/(TN_test+FP_test))*100;
79 -    Error_prediccion=((FN_test+FP_test)/(TP_test+TN_test+FN_test+FP_test))*100;
80
81 -    set (handles.text16, 'String',Error_prediccion);
82 -    set (handles.text17, 'String',Sensibilidad);
83 -    set (handles.text18, 'String',Especificidad);
84
85 -    handles.output = hObject;
86      % Update handles structure
87 -    guidata(hObject, handles);
88
89      % --- Outputs from this function are returned to the command line.
90      function varargout = Fig_Red_neuronal_OutputFcn(hObject, eventdata, handles)
91      % Get default command line output from handles structure
92 -    varargout{1} = handles.output;
93
94      % --- Executes on button press in pushbutton1.
95      function pushbutton1_Callback(hObject, eventdata, handles)
96 -    close(handles.output);
97
```

Figura 6-56: Red neuronal, parte 4 del código. Elaboración propia.

Bibliografía

[1] Organización Mundial de la Salud. Tuberculosis. Salud, Instituto Nacional de Salud, 10 2020.

[2] Sistema Nacional de Vigilancia en Salud Pública. boletín epidemiológico semanal. Salud, Instituto Nacional de Salud, 07 2020.

[3] B. Tortora, G.and Derrickson. *Anatomía y fisiología*.

[4] World Health Organization's 2021 Global TB report. Las muertes por tuberculosis aumentan por primera vez en más de una década por la covid. *Agencia de noticias científicas de la Fundación Española para la Ciencia y la Tecnología SINC*, 14(30), 10 2021.

[5] R. Natalia Programa Nacional de Control y Eliminación de la Tuberculosis M. Tania, M. Fabiola. Manual operativo implementación del genexpert mtb/rif en el programa de tuberculosis. 2017.

[6] Vu D. Phan and Janet M. Poponick. *Tuberculosis*. McGraw-Hill Education, New York, NY, 2018.

[7] Linux. Gnu/linux. la terminal o línea de comandos. *Blog para los amantes de Linux y el software libre*, 07 2019.

[8] Ian. Sommerville. *Ingenieria del software*. PEARSON EDUCACIÓN S.A., 2005.

[9] S. Kumar. Agrupamiento difuso de c-means: ¿es mejor que el agrupamiento de k-means? *Towards Data Science*, 04 2021.

[10] S. Kumar. Comprensión de los algoritmos de agrupación en clústeres k-means, k-means++ y k-medoids. *Towards Data Science*, 06 2020.

[11] Z.Carlos. Evaluación de modelos de clasificación. *RPubs*, 05 2017.

[12] J.D. Munera, L A. Montoya, J.A. Mosquera, et al. Casos de tuberculosis pulmonar y extrapulmonar notificados al programa de tuberculosis en el departamento del Chocó, Colombia, periodo 2012-2015. *Enf Infec Microbiol*, 39(3):93–102, 09 2019.

[13] M.J. Gonzalez, B. Gonzales, J.A. Sotolongo, et al. Programa de intervencion comunitaria dirigido a pacientes con riesgo de tuberculosis pulmonar. *Revista Cubana de Salud Publica*, 45(3), 09 2019.

[14] J.E. Polanco-Pasaje, L. Rodriguez Marquez, K.Y. Tello-Hoyos, et al. Cascada de atención de la tuberculosis para la población indígena en colombia: una investigación operativa. *Pan American Journal of Public Health*, 44, 12 2020.

[15] M.A. Sanchez, J. Pino, R. Pacheco, et al. Análisis de letalidad en pacientes con diagnóstico de tuberculosis en un centro de alta complejidad en cali, colombia. Ingenieria clinica, Universidad Icesi, Cali-Colombia, 02 2018.

[16] Ke Yuan, Yabing Huang, and Qian Tang. The impact of social and economic development on the spread of infectious respiratory diseases, push or constrain? empirical research from china based on machine learning methods. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1364–1369, 2020.

[17] Sophie Khaddaj and Hussain Chrief. Prevention and control of emerging infectious diseases in human populations. In *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 336–339, 2020.

[18] PMA. Alvarez, PXA. Morales, A. Rodriguez-Ramirez, et al. Costo-efectividad de tres pruebas diagnósticas de tuberculosis pulmonar en dos ciudades de colombia, 2015. *Enf Infec Microbiol*, 39(4):129–133, 2015.

[19] Anuradha D. Gunasinghe, Achala C. Aponso, and Harsha Thirimanna. Early prediction of lung diseases. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–4, 2019.

[20] Richard Joseph, Yohan Mahajan, Sanjib Naha Biswas, Karan Patowary, and Dhanashri Asai. Contagious disease propagation study using machine learning. In *2018 3rd International Conference on Inventive Computation Technologies (ICICT)*, pages 724–728, 2018.

[21] Shivam Karn, Shubham Sangole, Abhishek Gawde, and Jyoti Joshi. Prediction and classification of vector-borne and communicable diseases through artificial neural networks. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1011–1015, 2019.

[22] Adnan Fojnica, Ahmed Osmanović, and Almir Badnjević. Dynamical model of tuberculosis-multiple strain prediction based on artificial neural network. In *2016 5th Mediterranean Conference on Embedded Computing (MECO)*, pages 290–293, 2016.

[23] P. SMARTSON and NYONI. Prediction of tb notifications at gweru provincial hospital using artificial neural networks. *INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY*, 7(6), 06 2020.

[24] Mehera Binte Mizan, Md. Al Mehedi Hasan, and Syed Rahat Hassan. A comparative study of tuberculosis detection using deep convolutional neural network. In *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, pages 157–161, 2020.

[25] H.M. Gaviria, M.B.and Henao, Martínez T., and Bernal E. Papel del personal de salud en el diagnóstico tardío de la tuberculosis pulmonar en adultos de medellín. *Revista Panamericana de Salud Publica/Pan American Journal of Public Health*, 27(2):83–92, 2015.

[26] M. Jaramillo Grajales, R. Torres Villa, E. Pabon Gelves, et al. *La Clínica y el laboratorio Diagnóstico de tuberculosis: desde lo tradicional hasta el desarrollo actual*, page 311–331. Medicina y laboratorio, 2015.

[27] M. Ibrahim, I.and Iqbal and Abdulazeez2 A. The role of machine learning algorithms for diagnosing diseases. *Journal of Applied Science and Technology Trends*, 2(01):10–19, 01 2021.

[28] Sacyl. Portal de salud. 2018.

[29] Union Europea.and GOIB. Misistemainmune.

[30] Dirección de Regulación de Patógenos. and Agencia de Salud Pública de Canadá. Fichas de datos de seguridad de patógenos: Sustancias infecciosas: Mycobacterium tuberculosis y mycobacterium tuberculosis complex. *Government of Canada*, 09 2012.

[31] Mario C. Raviglione. *Tuberculosis*. McGraw-Hill Education, New York, NY, 2018.

[32] MD David A. Smith. *Urgencias pulmonares*. McGraw-Hill Education, New York, NY, 2015.

[33] Asha N. Chesnutt, Mark S. Chesnutt, Niall T. Prendergast, and Thomas J. Prendergast. *Tuberculosis pulmonar*. McGraw-Hill Education, New York, NY, 2021.

[34] Lawrence M. Tierney Jr., Sanjay Saint, and Mary A. Whooley. *Neumopatías*. McGraw-Hill Education, New York, NY, 2011.

[35] redactores y equipo de editores médicos de la Sociedad Americana Contra El Cáncer. Las muertes por tuberculosis aumentan por primera vez en más de una década por la covid. *American cancer society*, 9 2021.

[36] Texas Tech University Health Sciences Center. L. Aimee Hechanova, MD. Dialisis. *Manual MSD*, 12 2020.

[37] Biblioteca nacional de medicina. Cortiocides. *Medline Plus*, 07 2021.

[38] J. Corrales. Interfaz de usuario o ui: ¿qué es y cuáles son sus características? *Blog rockcontent*, 08 2019.

[39] J. Manjarres. 8 algoritmos de agrupación en clústeres en el aprendizaje automático que todos los científicos de datos deben conocer. *freeCodeCamp*, 04 2021.

[40] S.Virtual. Etapas en un proyecto de machine learning. 06 2020.

[41] R.España. Etapas del proceso de machine learning. 11 2020.

[42] J. Sanchez. ¿cómo aprenden las máquinas? machine learning y sus diferentes tipos. *datos.job.es*, 08 2020.

[43] V. Raschka, S.and Mirjalili. *Python Machine Learning*. Packt Publishing Ltd., 2017.

[44] I. Silva, D.H Spatti, R.A Flauzino, L.H Bartocci, and S.F Reis. *Artificial Neural Networks*.

[45] Msc O.A Martínez. Informe de evento de tuberculosis aÑo 2021. Salud, Ministerio de Salud y Protección Social, 11 2021.