UNIVERSIDAD DEL SINÚ
Elías Bechara Zainúm
Seccional Cartagena

OPTIMIZACIÓN DE UN MODELO DE CLASIFICACIÓN DE ENFERMEDADES CARDIOVASCULARES UTILIZANDO TÉCNICAS DE APRENDIZAJE PROFUNDO SUPERVISADO Y DESPLIEGUE DE DASHBOARD WEB

JESUS DAVID PEREZ TATIS

UNIVERSIDAD DEL SINÚ ELÍAS BECHARA ZAINÚM

FACULTAD DE CIENCIAS EXACTAS E INGENIERÍAS

ESCUELA DE INGENIERÍA DE SISTEMAS

CARTAGENA-COLOMBIA

Diciembre 2021

UNIVERSIDAD DEL SINÚ
Elías Bechara Zainúm
Seccional Cartagena

OPTIMIZACIÓN DE UN MODELO DE CLASIFICACIÓN DE ENFERMEDADES CARDIOVASCULARES UTILIZANDO TÉCNICAS DE APRENDIZAJE PROFUNDO SUPERVISADO Y DESPLIEGUE DE DASHBOARD WEB

Estudiante:
Jesús David Pérez Tatis

Director de trabajo de grado:
Eugenia Arrieta Rodriguez

UNIVERSIDAD DEL SINÚ ELÍAS BECHARA ZAINÚM
FACULTAD DE CIENCIAS EXACTAS E INGENIERÍAS
ESCUELA DE INGENIERÍA DE SISTEMAS
CARTAGENA-COLOMBIA
Diciembre 2021

AGRADECIMIENTOS

Debo agradecer primeramente a Dios, a mi familia, a la profesora Eugenia Arrieta por aceptarme para hacer este proyecto quien fue la asesora y tutora principal en el desarrollo de este proyecto, su apoyo y confianza. Como también la profesora María Claudia Las ideas propias, continuamente enmarcadas en su orientación, así también agradecer al grupo interdisciplinar de docentes de la Universidad del Sinú, que me impulsaron a forjar un carácter sólido como estudiante, como profesional. Al grupo de compañeros aprendimos que trabajar en equipo no es fácil, pero dando un poco más de cada uno de nosotros lo pudimos lograr y cumplir nuestras metas y propósitos.

CONTENIDO

TABLA DE ILUSTRACIONES

Resumen

El presente trabajo de grado presenta la optimización y despliegue del modelo del pregrado de título “Modelo predictivo de hipertensión para el diagnóstico de pacientes con factores de riesgo cardiovascular en el departamento de bolívar, mediante técnicas de Deep Learning” presentado por Gisella Rojas y Cristian Ruiz en el año 2020, el trabajo se centró en la elaboración de un modelo basado en Deep Learning que fuese capaz de apoyar el diagnóstico de los pacientes que en sus condiciones de salud y otros aspectos puedan elevar la vulnerabilidad de la hipertensión.

El desarrollo de este proyecto se basa en la optimización de un modelo previo de predicción de hipertensión del cual se mejora el proceso de análisis de los datos ya que el modelo inicial no generaba la predicción exacta de clasificación de enfermedades cardiovasculares debido a la mala interpretación y la falta de limpieza de los datos. Asimismo la optimización consistió en hacer ajustes en el modelo realizando mejoras en la construcción de la red neuronal y en el proceso de entrenamiento identificando técnicas de activación y épocas adecuadas para lograr obtener resultados óptimos.

Finalmente la preparación de los datos permitió formular, entrenar y probar un modelo de clasificación de enfermedades cardiovasculares construido en el entorno de TensorFlow desarrollado por Google el cual se obtuvo una red neuronal recurrente compuesta con una capa de entrada con 18 nodos, dos capas ocultas con 128 nodos cada capa y una capa de salida de 3 nodos en cual fueron los más óptimos para la predicción la cual obtuvo una precisión de 97% en la validación del modelo, y posteriormente se desplegó haciendo uso de una aplicación web para la consulta del personal médico el modelo elaborado por los mencionado anteriormente arrojó como resultados, una precisión superior al 86%, viendo los porcentajes en las métricas de evaluación que se utilizaron como: Precisión, Recall, F1 Score y accuracy, las cuales arrojaron resultados que superan el 80% en la clasificación de los riesgos.

Cabe destacar que el modelo desarrollado no disponía de una interfaz gráfica que permitiera la toma de decisiones por parte del personal médico usuario de la aplicación.

tiene como enfoque realizar la clasificación del código del diagnóstico de las personas que posiblemente sufran patologías como hipertensión esencial, diabetes mellitus no insulinodependiente sin mencion de complicacion, diabetes mellitus insulinodependiente sin mencion de complicacion, así también integrar una interfaz gráfica mediante un aplicativo web la cual será desarrollado en el framework DJANGO (Python) que se integre el modelo construido para realizar clasificación diferencial del diagnóstico de hipertensión.

Introducción

La hipertensión arterial (HTA) es una de las enfermedades más frecuentes del mundo. [1]. Su frecuencia aumenta de manera exponencial con la edad, a partir de los cincuenta años, lo cual, unido al aumento de la esperanza de vida en los países desarrollados, hace que se convierta en un problema sanitario de primera magnitud. Además presenta estrecha relación con las enfermedades cardiovasculares, primera causa de mortalidad en dichos países ha sido demostrada en múltiples estudios de observación, así como la disminución de la morbimortalidad con un adecuado control, en estudios de intervención[2].

Por otra parte, la hipertensión es la 1ª causa de patología en las naciones desarrollados; la 2ª causa de patología, luego del tabaquismo, en las naciones en desarrollo; la 1ª causa de ataque cerebrovascular e insuficiencia cardiaca; y la 2ª causa de síndrome coronario agudo[3].

Las predicciones de los códigos de diagnóstico y lo que representa cada uno de ellos como por ejemplo:

- I10X : Hipertension esencial.
- E119: Diabetes mellitus no insulinodependiente sin mencion de complicacion.
- E109: Diabetes mellitus insulinodependiente sin mencion de complicacion.

Se basarán en la toma de algunas variables que podrían influir. Se implementaron técnicas de análisis de variables para determinar cuáles son las que más relevancia e incidencia tiene respecto a la variable de respuesta.

Basándose en las variables de respuesta que genera el modelo con las posibles predicciones del diagnóstico se implementó la arquitectura principal para el desarrollo de un aplicativo web utilizando tecnologías de desarrollo más usadas en el mercado y mejor valoradas para poder acelerar el flujo de trabajo.

La información de los pacientes será alojada en una base de datos donde principalmente tendrá los datos médicos necesarios para realizar la clasificación como también la información personal de cada uno y llevar el proceso de cada uno de ellos.

Por otra parte, el sector salud está poniendo interés especial en los proyectos de Machine Learning que permiten obtener conocimiento, reglas o hallazgos no triviales a partir de datos clínicos[4], no solo por la información objetiva que se deriva sino también por la posibilidad de predecir eventos futuros, proporcionado

seguridad para realizar el diagnóstico y tratamiento de enfermedades cardiovasculares en pacientes[5].

En virtud de lo anterior, se propone un trabajo de culminación de pregrado que optimice un modelo previo de hipertensión para facilitar el trabajo al área de la salud, este trabajo se estructura de las siguiente forma análisis del modelo, optimización del modelo de hipertensión, análisis, diseño y desarrollo de aplicación web, inicialmente se hace la primera fase que fue el análisis del modelo donde se hizo un proceso que se describen, se exploran y se verifica la calidad de los mismos.

De seguido se realizó la optimización del modelado la cual se aplicaron técnicas de filtrado de variables como la técnica RFE y *StandardScaler*, balanceo de SMOTE estas técnicas fueron aplicadas para conocer cuáles eran las variables más significativas del conjunto de datos, lo cual arrojó 19 columnas más significativas y se utilizaron para el proceso del entrenamiento de la red neuronal.

Por último se realiza el guardado del modelo interconectando con una aplicación web, dicha aplicación también se despliega en un contenedor docker con una infraestructura óptima y se permite seguir escalando.

1. DISEÑO METODOLÓGICO DEL PROYECTO

1.1 Planteamiento del problema

La Hipertensión Arterial perjudica alrededor del 20% de la población adulta, siendo esta la primera causa de morbilidad y es el motivo del mayor número de consultas en las afecciones del artefacto circulatorio[16]. Asimismo es el componente de peligro de mayor relevancia para la patología cardio cerebrovascular, y constantemente se asocia con otros componentes de peligro bien conocidos como por ejemplo diabetes mellitus no insulinodependiente sin mención de complicación o diabetes mellitus insulinodependiente sin mención de complicación. Por otra parte, varias personas que poseen presión arterial alta causada por una patología no diagnosticada.

Sin embargo, el personal del área de la salud presenta dificultad a la hora de realizar diagnósticos, tomar decisiones correctas en el menor tiempo posible para garantizar el mejoramiento de la calidad de vida del paciente. Atendiendo a esta necesidad se sugiere utilizar plataformas, arquitecturas o algoritmos que permitan desarrollar un modelo efectivo para optimizar el trabajo del médico durante el proceso de atención de pacientes.

Esta investigación plantea el desarrollo de un modelo de predicción hipertensión desplegado en un entorno web apoyado por tecnologías de desarrollo libre y de bajo. Para realizar la clasificación del código de diagnóstico que puedan presentar los pacientes al sufrir estas patologías como: hipertensión esencial, diabetes mellitus no insulinodependiente sin mencion de complicacion, diabetes mellitus insulinodependiente sin mencion de complicacion. Se basa en una predicción mediante el uso de variables clínicas como: la presión de la sangre, los niveles de azúcar en la sangre y niveles de oxígeno, edad, peso, estilos de vida.

Este proyecto retoma las recomendaciones del trabajo de culminación de pregrado de la escuela Ingeniería de Sistemas de la Universidad del Sinú Seccional Cartagena presentado por Gisella Rojas y Cristian Ruiz en el 2020, el cual tenían un modelo predictivo realizado para el aporte de la investigación, por tal motivo no generó los resultados esperados ya que la información suministrada en el modelo obtenían datos erróneos y no era la adecuada para obtener los resultados esperados por el modelo

1.2 Justificación

Actualmente la Inteligencia Artificial está siendo aplicada en el campo de la salud abarcando trabajos relacionados con Sistemas Expertos, Redes Bayesianas, Minería de Datos que es útil para extraer conocimientos para la toma de decisiones y para generar hipótesis a partir de grandes datos médicos.

De igual forma Machine Learning que es una rama de la inteligencia artificial ha aportado modelos de recomendación, diagnóstico precoz, y predicción de enfermedades cómo: Diabetes, cerebrovascular, cardiacas, infracción cerebral, cáncer de mama, cáncer de próstata, Parkinson, infartos, artritis reumatoide y la hipertensión que es el interés especial de este trabajo de grado.

Debido a que estudios sobre la prevalencia de Hipertensión en Colombia[1], que permiten visualizar el panorama de esta patología crónica en el país, observando su tendencia al alza a lo largo del tiempo y su variabilidad según sexo, edad y área geográfica. La estimación de la prevalencia de Hipertensión en Colombia es un facilitador, tanto de la vigilancia en salud pública en el país, como del desarrollo de estrategias de intervención y evaluación.

Entonces, se hace necesario establecer un método estándar para llevar a cabo más estudios sobre hipertensión en la población, con el fin de caracterizarlos y diseñar intervenciones. Una de las explicaciones posibles de este aumento ha sido la migración constante de grupos poblacionales del área rural al área urbana, lo cual conduce a cambios importantes en los comportamientos de salud y al aumento de los factores de riesgo cardiovascular [2]. A pesar de la eficacia de los medicamentos disponibles para su tratamiento y del bajo costo para controlarla, existen bajas tasas de control de la enfermedad en pacientes diagnosticados.

Por otra parte, los profesionales de la salud están poniendo mayor interés en los proyectos de automatización de tareas repetitivas, generación de conocimiento a través de datos clínicos, y están poniendo su confianza en los algoritmos y técnicas de Machine Learning para el análisis y toma de decisiones de enfermedades cardiovasculares en los pacientes, entre ellas la hipertensión que posibiliten la predicción, recomendación o diagnóstico. El Machine Learning (ML) ofrece la oportunidad de mejorar la precisión mediante la explotación de interacciones complejas entre factores de riesgo cardiovascular, con el objeto de mejorar la predicción de enfermedades a través de datos clínicos o dataset

Ante esta situación se propone un trabajo de grado, del programa de Ingeniería de Sistemas de la Universidad del Sinú, que le aparece a la línea de investigación en Inteligencia Artificial y que sirva de base para futura investigaciones en el área, y que le aporte a su vez a la prevención y detección temprana de los factores de riesgos que aumentan la probabilidad de que una persona padezca de hipertensión.

Este proyecto se centra en diseñar y desarrollar un modelo de Deep learning capaz de ayudar a clasificar pacientes cuyas condiciones de salud, hábitos, entre otros aspectos aumente su vulnerabilidad ante la Hipertensión. Como también su despliegue en una arquitectura de software con tecnologías muy bien valoradas en el mercado, se implementó un aplicativo web interactivo que contiene las utilidades necesarias para poder introducir los datos requeridos en un formulario y realizar dicha predicción del diagnóstico.

1.3 Alcance

Optimizar el modelo predictivo del proyecto “Modelo predictivo de hipertensión para el diagnóstico de pacientes con factores de riesgo cardiovascular en el departamento de bolívar, mediante técnicas de Deep Learning” el cual tomaba como referencias variables para poder hacer el análisis y predicciones en los pacientes. Del total de pacientes encontrados en los registros clínicos proporcionados por el hospital local de Arjona Bolívar, el cual determinaba que el 70% de estos se utilizarían para el entrenamiento del modelo en fase de construcción, el 30% restante se tomó para realizar las pruebas del modelo.

Además, al realizar un despliegue en una arquitectura de software con tecnologías muy bien valoradas en el mercado se desarrolla la implementación de un aplicativo web teniendo la información necesaria para poder introducir los datos requeridos en un formulario y realizar dicha predicción del diagnóstico.

1.4 Pregunta problema

¿Cómo optimizar un modelo previo de clasificación hipertensión y su despliegue en la web?

1.5 Objetivos

1.5.1 Objetivo general

Optimizar un modelo de clasificación de hipertensión utilizando técnicas de aprendizaje profundo supervisado, desplegado en un tablero de mando para la toma de decisiones.

1.5.2 Objetivo específico

- Analizar el modelo previamente construido para determinar su nivel de precisión e identificar opciones de mejora.

- Optimizar el proceso de construcción del conjunto de datos y del modelo de red neuronal con la finalidad de minimizar la cantidad de variables de entrada y la precisión del modelo.

- Construir una interfaz gráfica mediante un aplicativo web en el framework DJANGO (Python) que se integre el modelo construido para realizar clasificación de enfermedades cardiovasculares.

- Desplegar el dashboard en un servidor en la nube para la consulta de los profesionales de la salud.

1.6 Estado del arte

Proyectos relacionados

A continuación, se explica de manera más detallada las investigaciones que tienen mayor relación con el proyecto, detalla los métodos en que se realizaron dichas investigaciones.

En [5] tiene como objetivo la predicción para el desarrollo de la cardiopatía hipertensiva, a partir de factores hemodinámicos y no hemodinámicos.

Se realizó el diseño y validación de un árbol de predicción del desarrollo de la cardiopatía hipertensiva mediante el procedimiento de descubrimiento de conocimientos en bases de datos, conocido internacionalmente como proceso KDD (del inglé Knowledge Discovery in Database) y minería de datos (Data Mining - DM), en pacientes hipertensos atendidos en la consulta especializada de HTA de

la Policlínica de Especialidades del Hospital General Universitario " Carlos Manuel de Céspedes " del municipio Bayamo, provincia Granma, Cuba, desde el 1ro de enero de 2004 hasta el 31 de diciembre de 2009. Como resultado de este método el árbol predijo el riesgo de desarrollar la cardiopatía hipertensiva a 82,598 % de los pacientes; con un área bajo la curva ROC de 0,861 y una tasa de verdaderos positivos de 0,733 y de 0,921 para las clases 1 y 2, respectivamente. El factor más importante lo constituyó la proteína C reactiva, seguida en orden de importancia por la glucemia, el ácido úrico, el colesterol y la microalbuminuria.

En [6] la autora explica la importancia de la prevención de la hipertensión arterial (HTA), exponiendo los diagnósticos y las causas que llegan a provocar esta enfermedad, además del tratamiento farmacológico para la HTA:

Los medicamentos más usados para el tratamiento de la HTA los podemos agrupar en:

Diuréticos: se denominan a veces «píldoras de agua». Se utilizan para tratar la insuficiencia cardíaca congestiva (ICC), la presión arterial alta (hipertensión) o el edema (retención de líquidos). Los diuréticos también se recetan para ciertos tipos de enfermedades del riñón o hígado. Disminuyen la cantidad de Na y por tanto el volumen sanguíneo, disminuyendo la carga cardiaca por vasodilatación.

Alfabloqueantes: Los alfa bloqueadores, alfas bloqueantes, antagonistas alfa adrenérgicos o bloqueantes α son agentes farmacológicos que actúan como antagonistas de los receptores alfa adrenérgicos. bloquean de manera selectiva y competitiva los receptores alfa1 adrenérgicos postsinápticos vasoconstrictores, produciendo vasodilatación arteriovenosa, reducción de las resistencias vasculares periféricas y de la PA.

Betabloqueantes: bloquean competitiva y reversiblemente los receptores betaadrenérgicos, disminuyendo la frecuencia y el gasto cardiaco además de bloquear la liberación de renina.

Antagonistas del calcio: se fijan a los canales de calcio tipo L voltajes dependientes eliminando la corriente de calcio que provoca la contracción muscular, produciendo la relajación del músculo liso vascular

Agentes que bloquean la producción o acción de la angiotensina:

- Inhibidores de la enzima convertidora de la angiotensina (IECAs): bloquean la síntesis de angiotensina II por inhibición competitiva de la enzima convertidora de la angiotensina (ECA) produciendo vasodilatación arteriovenosa además de nutrieres.

Antagonistas de los receptores de la angiotensina (ARA II): Bloquean de forma competitiva y selectiva los receptores AT1 inhibiendo la acción de la angiotensina II.

El objetivo es especificar el tratamiento farmacológico para el tratamiento de la HTA, esto con el fin de controlar la presión arterial del paciente y más a largo plazo reducir la morbimortalidad, fundamentalmente de las enfermedades cardiovasculares, cerebrovasculares y renales asociadas a la HTA.

Otro trabajo es [7], donde el enfoque está basado en la prevención del riesgo cardiovascular haciendo uso de la escala de Framingham . El método fue un estudio de campo observacional, descriptivo y correlacional, en el que se incluyeron 400 adultos de ambos sexos (M: 310 y H: 90), con una edad La estratificación del riesgo permitió determinar el porcentaje de riesgo para enfermedad cardiovascular e hipertensión Se tomaron medidas antropométricas según la OMS para el IMC y perímetro abdominal. Y así mismo, se identificaron los factores de riesgo cardiovascular en la población de estudio.

Los resultados luego de la investigación, el 10% de los estudiados presentó algo de padecer la enfermedad los sus próximos 10 años, 14% y 75% presentó moderado y bajo. La clasificación para el riesgo de hipertensión bajo es de 1,2 y 4 años. Llevar una vida sedentaria aparece como un factor frecuente, El 21% presentó HTA; 7% Diabetes; 6,7% Tabaquismo; y el 89% bebedores.

con lo anterior, teniendo en cuenta de que las investigaciones encontradas, entre ellas la técnica que es común es usar árboles de decisión para el análisis de los datos. Teniendo en cuenta esto, se opta por usar redes neuronales para hacer el algoritmo que permita hacer la clasificación con los datos de los pacientes obtenidos del hospital local de Arjona Bolívar.

1.7 Marco referencial

1.7.1 Marco Teórico

Machine learning

El término machine learning engloba al conjunto de algoritmos que permiten identificar patrones presentes en los datos y crear con ellos estructuras (modelos) que los representan. Una vez que los modelos han sido generados, se pueden emplear para predecir información sobre hechos o eventos que todavía no se han observado. Es importante recordar que, los sistemas de machine learning, "aprender" patrones que estén presentes en los datos con los que se entrenan, por lo tanto, solo pueden reconocer escenarios similares a lo que han visto antes. Al emplear sistemas entrenados con datos pasados para predecir futuros se está asumiendo que, en el futuro, el comportamiento será el similar, cosa que no siempre ocurre[40].

Deep learning

Podemos definir como DEEP LEARNING la rama de Aprendizaje Máquina basada en un conjunto de algoritmos que intentan modelar abstracciones de alto nivel en los datos usando múltiples capas de procesamiento con estructuras complejas, o bien compuestas de múltiples transformaciones no lineales.

Entonces se considera parte de una familia más amplia de métodos de Aprendizaje Máquina basados en aprendizaje de representaciones de los datos.[7]

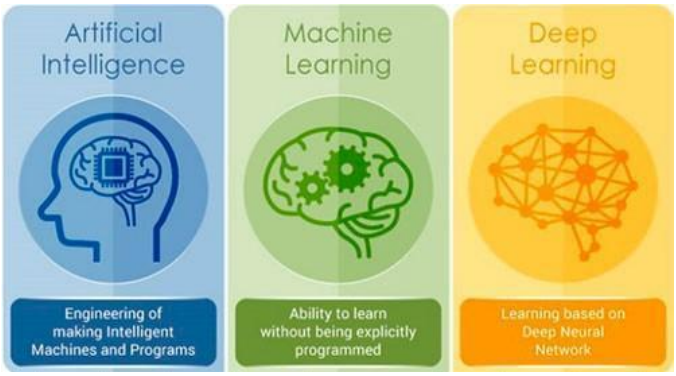


Ilustración 1 Inteligencia Artificial vs Aprendizaje Automático vs Deep Learning vs Ciencia de Datos. Tomado de [8]

La forma como funciona en el cerebro humano esta arquitectura de redes neuronales es que la información recibida del exterior pasa a través de un número grande de capas antes de convertirse en una respuesta

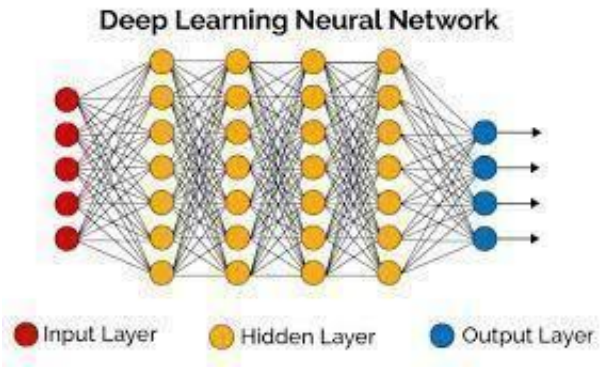


Ilustración 2 (Neural networks and deep learning). Tomado de [9]

La forma en la que trabaja el Deep Learning es usando estas cascadas de capas con unidades de procesamiento que permiten la extracción y transformación de variables. Cada red, dentro de su jerarquía, aplica una transformación en su capa de entrada y utiliza esa información de aprendizaje para crear un modelo estadístico de salida que itera las veces necesarias hasta lograr un nivel de precisión en el aprendizaje y respuesta aceptable. [10]

El campo de la inteligencia artificial es, esencialmente, cuando las máquinas pueden realizar tareas que generalmente requieren de inteligencia humana. Abarca el aprendizaje automático, donde las máquinas pueden aprender por experiencia y adquirir habilidades sin la participación de los humanos.

Herramientas del despliegue para aplicativo

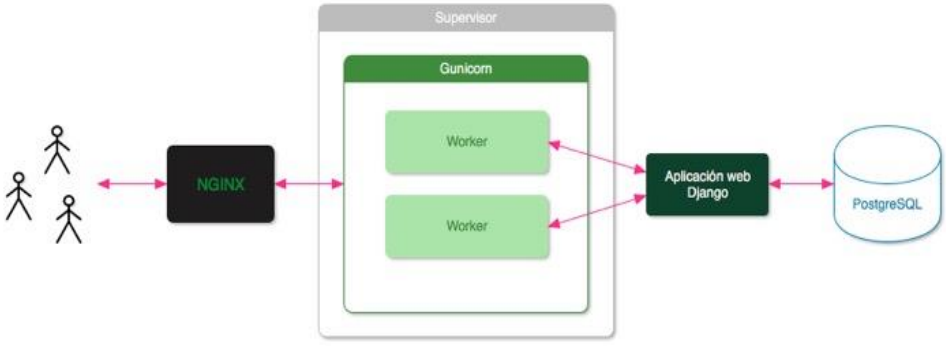


Ilustración 3 Arquitectura de despliegue. Tomado de [11]

Podemos observar que Nginx es inicialmente el encargado de resolver la petición web realizada por el cliente, re-enviándola para que Gunicorn pueda procesarla utilizando varios workers. Cada worker es capaz de comunicarse con la aplicación Django de manera que pueden atenderse múltiples peticiones al mismo tiempo, manteniendo un buen tiempo de respuesta.

Por otro lado, Supervisor se encargará de controlar la ejecución del proceso Gunicorn de tal forma que podamos iniciar, detener o reiniciar la aplicación de Django en el momento que sea necesario. [11]

Contenedores en la nube

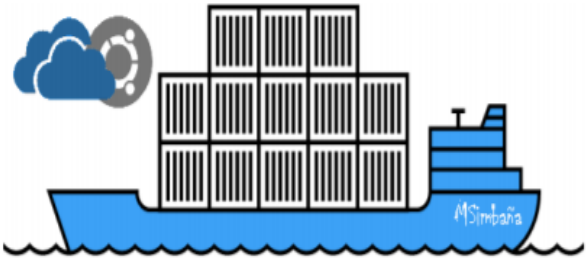


Ilustración 4 Contenedores Docker.

Esta nueva tendencia tecnológica de virtualizar contenedores brinda soluciones óptimas al administrador de sistemas y adicionalmente al desarrollador de aplicaciones al momento de crear infraestructuras virtualizadas.

Un contenedor simplemente es un proceso para el sistema operativo que, internamente, contiene la aplicación que queremos ejecutar y todas sus dependencias usando indirectamente el kernel del sistema operativo que se esté usando.

Al realizar un paralelismo entre el contenedor y una máquina virtual se define que ambos son sistemas autocontenidos que tienen una gran diferencia en que una máquina virtual necesita contener todo el sistema operativo mientras un contenedor aprovecha el sistema operativo en el que se ejecute.

Ejemplo:

Una máquina virtual necesita contener todo el sistema operativo mientras que un contenedor Docker aprovecha el sistema operativo sobre el cual se ejecuta, comparte el kernel del sistema operativo anfitrión e incluso parte de sus bibliotecas.

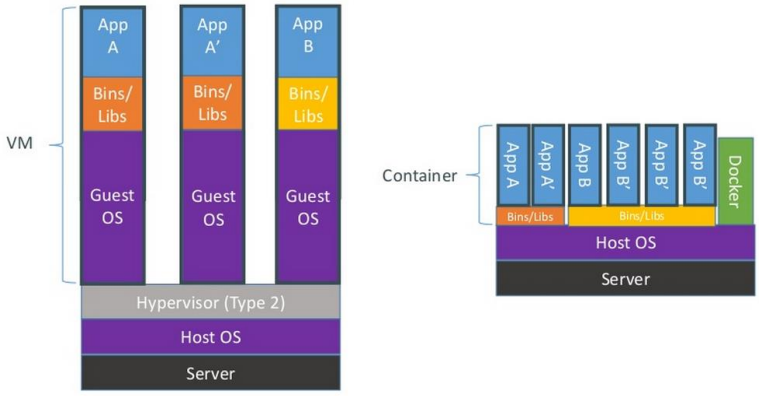


Ilustración 5 Diferencia Máquina Virtual y un Contenedor Docker

1.8 Marco conceptual

Framework: Es un conjunto de herramientas generalmente utilizado por programadores para realizar el desarrollo de software.

Python: Es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

R: Es un entorno y lenguaje de programación con un enfoque al análisis estadístico. R nació como una implementación de software libre del lenguaje S, adicionado con soporte para alcance estático.

Entorno Anaconda: es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático. Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos.

Jupyter: es un software de código abierto, estándares abiertos y servicios para computación interactiva en docenas de lenguajes de programación, entre ellos Julia, Python y R.

Spyder: Es un entorno de desarrollo integrado multiplataforma de código abierto para programación científica en lenguaje Python.

Google Colab: es un servicio cloud, basado en los Notebooks de Jupyter, que permite el uso gratuito de las GPUs y TPUs de Google, con librerías como: Scikit-learn, PyTorch, TensorFlow, Keras y OpenCV. Todo ello con bajo Python 2.7 y 3.6, que aún no está disponible para R y Scala.

TensorFlow: es una biblioteca de código abierto que se basa en un sistema de redes neuronales. Esto significa que puede relacionar varios datos en red simultáneamente, de la misma forma que lo hace el cerebro humano. Por ejemplo, puede reconocer varias palabras del alfabeto porque relaciona las letras y fonemas.

Keras: es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano. Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje Profundo.

Scikit Learn: Scikit-learn es una biblioteca de aprendizaje automático gratuita para Python. Cuenta con varios algoritmos como máquina de vectores de soporte, bosques aleatorios y vecinos k, y también admite bibliotecas numéricas y científicas de Python como NumPy y SciPy.

Nginx : Es un servidor web open source de alta performance que ofrece el contenido estático de un sitio web de forma rápida y fácil de configurar. Ofrece recursos de equilibrio de carga, proxy inverso y streaming, además de gestionar miles de conexiones simultáneas.

Gunicorn: es un servidor HTTP para sistemas Unix que cumple la especificación WSGI. Nos permite servir nuestra aplicación Flask con múltiples workers para incrementar el rendimiento de nuestra aplicación.

Django: es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en Python. Un framework web es un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente.

PostgreSQL: Es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. Las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON.

1.9 Marco Legal

El presente proyecto tiene sus bases legales sobre los siguientes pilares de normas, decretos y leyes del estado colombiano:

I. Decreto 846 de 2016; Por el cual se modifica la estructura del Departamento Administrativo de Ciencia, Tecnología e Innovación - COLCIENCIAS.

II. Decreto 591 del 26 de febrero de 1991 por el cual se regulan las modalidades específicas de contratos de fomento de actividades científicas y tecnológicas.

III. Decreto 585 del 26 de febrero de 1991 por el cual se crea el Consejo Nacional de Ciencia y Tecnología, se reorganiza el Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología-Colciencias- y se dictan otras disposiciones.

IV. Decreto 584 del 26 de febrero de 1.991, por el cual se reglamentan los viajes de estudio al exterior de los investigadores nacionales.

V. Decreto 393 del 26 de febrero de 1991 por el cual se dictan normas sobre asociación para actividades científicas y tecnológicas, proyectos de investigación y creación de tecnologías.

VI. Decreto 1467 del 2018 por el cual adiciona y modifica el Decreto 1082 de 2015 con el objeto de reglamentar la Ley 1923 de 2018 y se dictan otras.

VII. Decreto 293 del 2017; Por el cual se reglamenta el artículo 7 de la Ley 1753 de 2015 en lo relacionado con los Planes y Acuerdos Estratégicos Departamentales en Ciencia, Tecnología e Innovación y se dictan otras.

1.10 Metodología

Línea de Investigación

La Universidad del Sinú Seccional Cartagena cuenta con varios Grupos de Investigación que trabajan con el fin de mostrar avances tecnológicos a la optimización de procesos y generación de nuevos conocimientos. Este proyecto aportará a la línea de investigación de inteligencia artificial del grupo de investigación de artica. Debido a que este proyecto involucra técnicas de Machine learning y Deep Learning que son unas de las temáticas de gran impacto a nivel de investigaciones en el campo de la ingeniería.

Tipo de investigación

Este trabajo corresponde a una investigación aplicada, de cohorte retrospectiva en el cual los sujetos se estudian posteriormente de haberse producido la enfermedad. Los datos se obtendrán de los pacientes atendidos en el hospital Hospital Local de Arjona y el diagnóstico de estas realizadas por un profesional de la salud; en el que se aplican los conocimientos y las técnicas de inteligencia artificial para contribuir en la solución de un problema de la vida real, como es el apoyo diagnóstico en el departamento de cardiología, específicamente aplicado a la hipertensión. En este tipo de investigación el énfasis del análisis está en la aplicación efectiva de las técnicas de inteligencia artificial para obtener resultados positivos en términos de sensibilidad y especificidad.

Población y Muestra

Como población se tomaron los registros clínicos de los pacientes proporcionados del el Hospital Local de Arjona Bolívar, y como muestra se selecciona de la atención de pacientes entre enero de 2018 hasta diciembre de 2019, estos permitieran encontrar las variables que mayor determinan riesgos cardiovasculares.

Estos datos fueron llevados a un término de pacientes anónimos para salvaguardar la identidad real de estos, respetando los principios de ética profesional. Del total de pacientes encontrados en los registros clínicos se determinó que el 80% de estos se utilizarían para el entrenamiento del modelo en fase de construcción, el 20% restante se tomó para realizar las pruebas del modelo.

Variables

En la tabla 5 se muestran las descritas algunas de las variables presentes en el estudio de la hipertensión.

Tabla 5: variables de estudio

Nombre	Descripción	Unidad de medida	Tipo de variable	Valores normales
Edad	Edad de la persona	Años	Numérica	0-120 años

Talla	Medida de la persona en estatura	Centímetros	Numérica	1-200 centímetros
Peso	El peso es una medida de la fuerza gravitatoria que actúa sobre un objeto.	Kg	Numérica	IMC es entre 18.5 y 24.9, está dentro de los valores "normales" o de peso saludable. Si su IMC es entre 25.0 y 29.9, está dentro de los valores correspondientes a "sobrepeso". Si su IMC es 30.0 o superior, está dentro de los valores de "obesidad".

Selección de la metodología

En la tabla 4 se muestran las fases con las que se llevó a cabo el proyecto de investigación, haciendo una descripción de las tareas en cada una de las fases.

Tabla 4: Fases del proyecto

Fase	Objetivo	Actividad
Estudio de proyecto anterior	Adquirir conocimiento del tema en cuestión	Revisión de la base de datos que manejaba el anterior proyecto.

		Estudio del modelo y corrección de errores encontrados.
		Filtrado de variables para el nuevo modelo.
Optimización del modelo.	Construcción de nuevo modelo optimizado.	Escogencia de las herramientas en la nube para contener los datos.
		Optimización del anterior modelo.
		Escogencia de las columnas más óptimas para el estudio del nuevo modelo.
		usar técnicas para la normalización de los datos.
		Comprobar el modelo construido usando el 30% de los datos de pruebas, hacer la comparación con el error obtenido en entrenamiento vs el error de pruebas.
		Redacción de los capítulos que corresponden a la construcción del modelo con relación, además de la documentación de los escenarios de experimentación y pruebas efectuadas en la identificación de la técnica de Deep Learning.

Evaluación	Evaluar el funcionamiento del modelo	Verificar la funcionalidad y la exactitud del modelo utilizando el conjunto de datos destinados para las pruebas.
		Haciendo uso de las métricas, evaluar que tan preciso es a la hora de arrojar los resultados de las predicciones.
		Redacción de los capítulos finales de la monografía en la relación a los resultados obtenidos productos de la investigación
Diseño, Desarrollo y Despliegue	Diseñar arquitectura de la aplicación y desplegarla	Con base al proyecto presentado se realiza una aplicación web partiendo de unos requerimiento en base a la problemática como también el despliegue de la misma utilizando contenedor docker

2. ANÁLISIS DEL MODELO

En este capítulo se describe la construcción del conjunto de datos, en el cual se detalla paso a paso cómo está construido el conjunto de datos. Se identificaron una serie de etapas que determinan la ejecución de una tarea tanto para procesamiento de los datos como los resultados.

En esta fase se recopilan los datos iniciales, se describen, se exploran y se verifica la calidad de los mismos. Para la obtención se realizó extracción de los datos de la historia clínica cardiovascular con todos los campos o variables recolectados durante la atención entre el mes de enero de 2018 hasta Diciembre de 2019

El diagrama de proceso para la construcción del modelo de análisis de los datos está conformado por las tres fases que se muestran en la ilustración 6 y se detallan de la siguiente manera:

2.1 Comprensión del Negocio

El principal cliente de esta herramienta serán las clínicas y hospitales, con la cual los profesionales de la salud permitirán identificar pacientes con riesgos los cuales representan una población valiosa.

Los datos requeridos para construir el modelo fueron obtenidos del historial clínico de los pacientes atendidos en el Hospital Local de Arjona Bolívar, correspondientes al año 2017-2019, los registros hacen parte del programa prevención de riesgo cardiovascular que maneja la clínica.

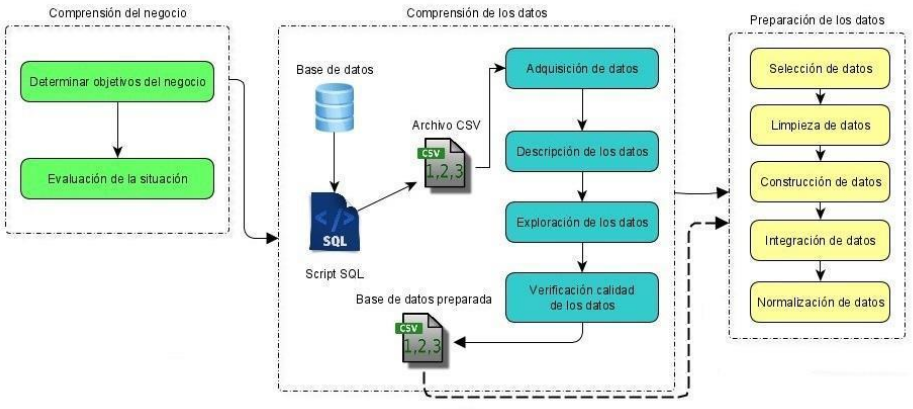


Ilustración 6 Diagrama proceso análisis de datos

2.2 Comprensión de los datos

En esta fase se recopilan los datos iniciales, se describen, se exploran y se verifica la calidad de los mismos. Para la obtención se construyó un script en SQL que permite extraer los datos de la historia clínica cardiovascular con todos los campos o variables recolectados durante la atención entre el mes de enero de 2018 hasta diciembre de 2019.

2.3 Preparación de Datos

En esta fase los datos para este proyecto fueron proporcionados por un hospital del departamento de Bolívar, el cual cuenta con base de datos de tipo SQL Server. Para el proceso de obtención de datos el primer paso fue el entendimiento de las estructuras de la base de datos. Para esto proporcionaron una base de datos de prueba, luego de conocerlas un poco pudimos realizar una consulta en lenguaje SQL, está consulta fue enviada al área de sistema del hospital en donde se validó y luego se realizó dicha consulta y los resultados obtenidos se enviaron en archivos con extensión CSV.

Los datos brutos se convierten en información con datos estructurados y a partir de esta base de datos se realiza un análisis descriptivo y exploratorio para identificar y seleccionar las variables más significativas inicialmente se realizó la importación de las librerías reflejada en la ilustración (7)

```
#Importando librerías
import pandas as pd
import numpy as np
```

Ilustración 7 Importación de librerías.

Se ejecutó la lectura de los datos donde inicialmente se tenían 187 columnas y 7454 registros de las atenciones cardiovasculares de la clínica este proceso se representa en la ilustración (8).

```
xls=pd.ExcelFile("base_de_datos_unificada.xlsx")
sheet_to_df_map={}

for sheet_name in xls.sheet_names:
    sheet_to_df_map[sheet_name] = xls.parse(sheet_name)

copy=sheet_to_df_map
all_sheets=[]

for sheet_name in copy:
    copy[sheet_name]=copy[sheet_name].assign(date=sheet_name)
    copy[sheet_name]=copy[sheet_name].reset_index().set_index(['date','index'])
    all_sheets.append(copy[sheet_name])

db=pd.concat(all_sheets)
db.head(5)
```

Ilustración 8 Lecturas de datos con excel.

2.3.1 Identificando datos faltantes

Se realizó la identificación de la columnas duplicada y se segmentan las variables categóricas y variables continuas de cada columna, obteniendo las siguientes cantidades

- El número de columnas categóricas es: 129
- El número de columnas continuas flotantes es: 51
- El número de columnas continuas enteras es: 7

En la siguiente ilustración (9) se puede evidenciar como se realizó la identificación de las columnas duplicadas

```
listaColumnasDuplicadas = [c for c in db.columns if c.find('.') != -1]
listaColumnasOrig = [];

for dup in listaColumnasDuplicadas :
  tempDup = dup[0:dup.index('.')]
  listaColumnasOrig.append(tempDup)

listaColumnasDuplicadas = listaColumnasDuplicadas+ listaColumnasOrig

listaColumnasDuplicadas = np.unique(np.array(listaColumnasDuplicadas))
print(listaColumnasDuplicadas)
```

Ilustración 9 Identificación de las columnas duplicadas.

El cual se identificaron 21 columnas duplicadas como: (Creatinina, Creatinina.1 ,DM1DM2, DM1DM2.1, Dislipidemia, Dislipidemia.1, Edad, Edad.1, IMC, IMC.1, Papiledema, Papiledema.1, Pulso, Pulso.1, ResultHDL, ResultHDL.1, Resultado, Resultado.1, Resultado.2, Resultado.3, Resultado.4), Un vez identificada se realizó el hallazgo de datos faltantes y se identificó el porcentaje de cada una, como se muestra en la siguiente ilustración (10)

```
nombrescol = []

porcentajePorColumna = {}

for columna in df.columns:
    faltantes = np.mean(df[columna].isnull())
    print('{} - {}%'.format(columna, round(faltantes*100)))
    if round(faltantes*100) > 0:
      nombrescol.append(columna)
    if columna in listaColumnasDuplicadas:
      porcentajePorColumna[columna] =faltantes
```

```
Peso - 7%
Talla - 7%
IMC - 7%
```

Ilustración 10 Porcentaje de datos faltantes con columnas correspondientes.

De acuerdo con análisis realizado, se puede destacar la siguiente información, la cual relaciona el porcentaje de datos faltantes y el nombre de las columnas correspondientes

Tabla 1. Porcentaje de datos faltantes

Porcentaje	Nombre de la columna
7%	Peso, Talla, IMC, FC, FR, Temperatura, IMC.1
8%	SMC, SignosVitales, Pulso, Pulso.1, ClasificacionFinalRiesgoCardiovascular
26%	LOB
28%	Condiciones_clinicas_asociadas
31%	Perimetro_abdominal
32%	Diabetes_mellitus_(DM2)
38%	Reflejos
50%	ResultCreatinina y ResultColestTotal
52%	TrigliceridosResultado
57%	ResultLDL
60%	ResultHDL - ResultHDL.1

Eliminación de columnas duplicadas en base a porcentaje de datos faltantes se presenta en la siguiente ilustración (11)

```
#print(porcentajePorColumna)
tempAsig = {}
for col in  [c for c in porcentajePorColumna if c in listaColumnasDuplicadas]:
  colName = col
  if  col.find('.') != -1:
    colName = col[0:col.index('.')]

  if colName in tempAsig:
    valorAsg = porcentajePorColumna[col]
  else:
    valorAsg = 1

  if col in df.columns:
    if porcentajePorColumna[col] >= valorAsg:
      del df[col]
      print('Se eliminó la columna ', col)
    else  :
      print('Se asigna la columna ', col)
      df[colName] = df[col]
      tempAsig[colName] = porcentajePorColumna[col]

      if  col.find('.') != -1:
        del df[col]
        print('Se eliminó la columna: ', col)
```

Ilustración 11 Eliminación de columnas duplicadas.

Correspondiente al seguimiento de las consultas de los pacientes, podemos observar que luego de eliminar las columnas duplicadas, los datos quedan de la siguiente manera:

No. Columnas = 175
No. Filas = 7454

La distribución de acuerdo con el tipo de datos es:

- 124 columnas son de tipo categóricas
- 44 columnas son de tipo continuas - flotante
- 7 columnas son de tipo continuas – enteras

A partir de los resultados obtenidos anteriormente se hace la eliminación de las columnas como se presenta en la ilustración (12) y podemos observar que luego de eliminar las columnas con porcentajes de datos faltantes mayor a 45% los datos quedan de la siguiente manera:

```
porcentajePorColumna = {}

for columna in db.columns:
    faltantes = np.mean(db[columna].isnull())
    if round(faltantes*100) > 0:
      #print('{} - {}%'.format(columna, round(faltantes*100)))
      if round(faltantes*100) > 45:
        del db[columna]
        print('Se eliminó la columna ', columna, 'porque su porcentaje de datos faltantes supera el 45%')
```

Ilustración 12 Eliminación de datos faltantes > 45%.

No. Columnas = 170
No. Filas = 7454

La distribución de acuerdo con el tipo de datos es:

- 121 columnas son de tipo categóricas
- 42 columnas son de tipo continuas - flotante
- 7 columnas son de tipo continuas - enteras

Después de realizar la eliminación se realizó la una gráfica de calor para identificar el proceso y observar su comportamiento y validar qué variables presentan pérdida de datos en la siguiente ilustración (13) se representará.

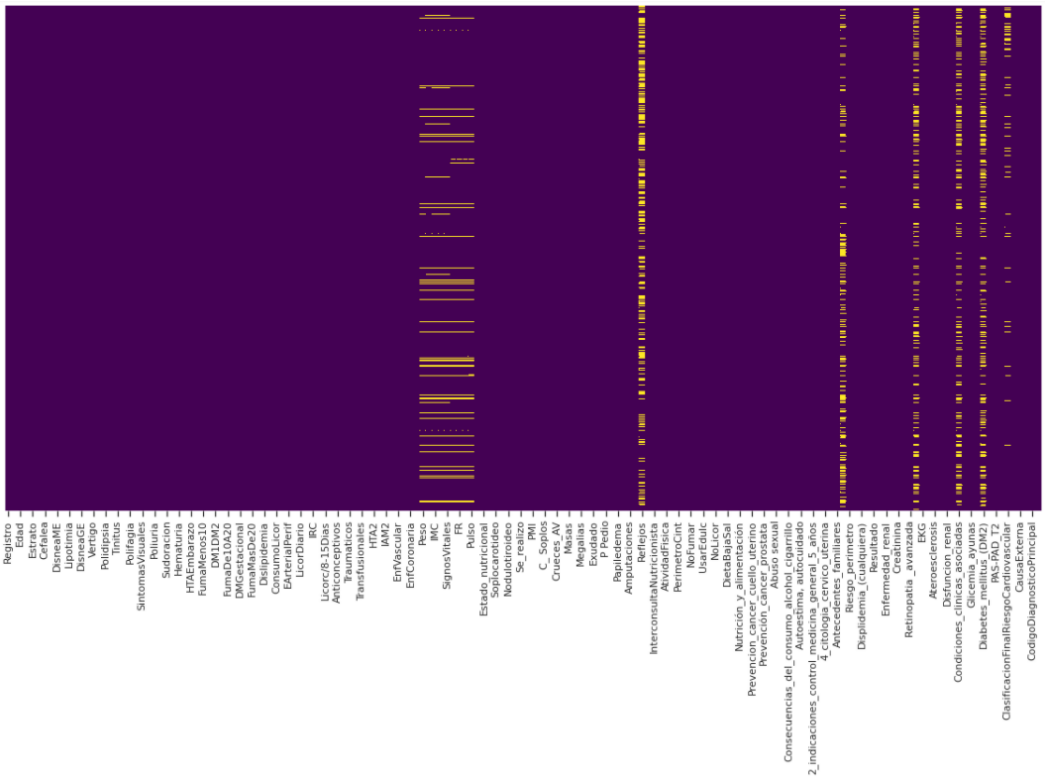


Ilustración 13 Columnas que presentan pérdida de datos.

2.3.2 Visualización de los datos

Para este proceso se grafican algunas variables para observar su comportamiento y cómo están agrupadas.

En la columna edad se puede ver que la mayoría de los pacientes se encuentra entre las edades de 65 a 75 años con más de 2000 registros, seguido a esto están los paciente entre los rangos de 55 a 64 años y 76 a 85 años con más de 1500 registros, los registros que se encuentran poco más de 1000 pertenecen a los rangos entre 45 a 54 años. En última medida los registros que son inferiores a 500 en los cuales se encuentran los rangos de 86 a 95 años, 35 a 44 años, 96 a 105 años y 25 a 34 años. Los resultados se pueden ver en la ilustración (14).

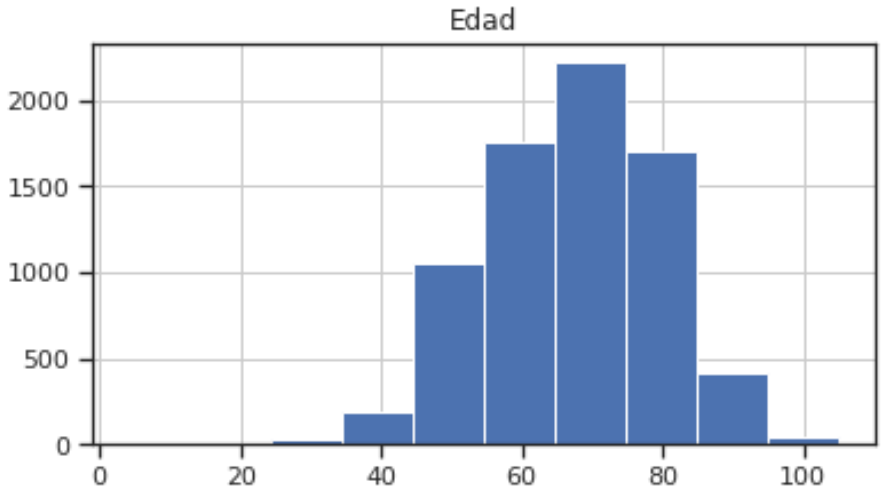


Ilustración 14 Gráfica edades

Así mismo, en la columna estrato se puede ver que todos pertenecen al estrato 1, lo cual muestra la Ilustración (15).

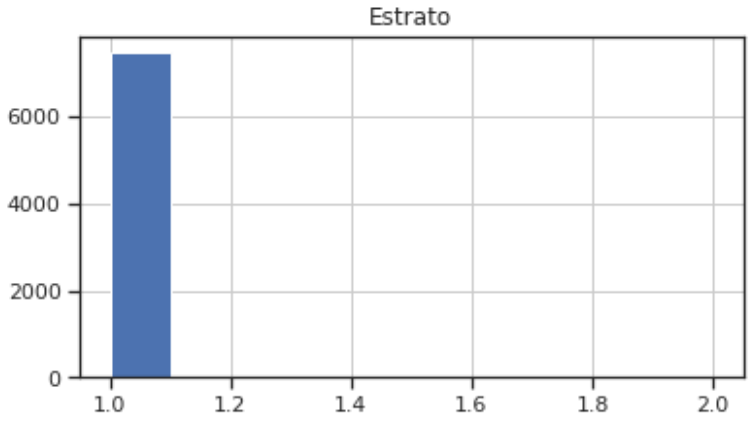


Ilustración 15 Gráfica estrato

También se graficó la variable peso, los cuales presentaron que la mayoría de los pacientes se encuentra entre el rango de 55 y 65 Kg con más de 2000 registros, entre el rango de 66 a 79 kg se encuentran los registros que superan los 1500 datos, además están los rangos de 45 a 55 kg y 80 a 90 kg que están por encima de 1000 registros, por último se encuentran 90 a 103, 35 a 44 kg y 104 a 115 kg por debajo de los 500 datos. Ilustración (16).

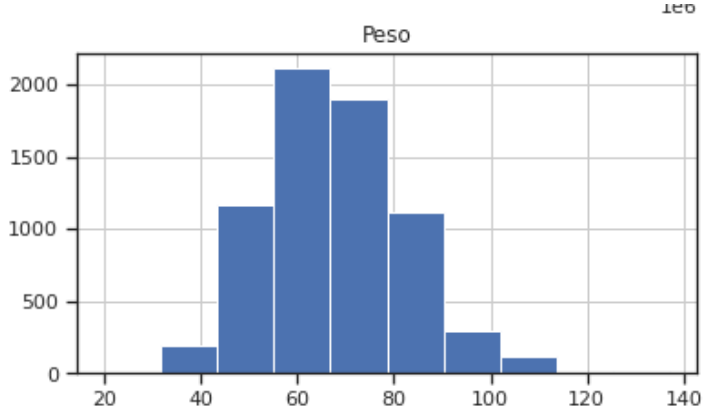


Ilustración 16 Gráfica peso

Del mismo modo se hizo con la columna de temperatura, aquí se muestra que los pacientes se ubican en un rango que va de 37 a 45 grados con más de 6000 registros. Se muestra en la ilustración (17).

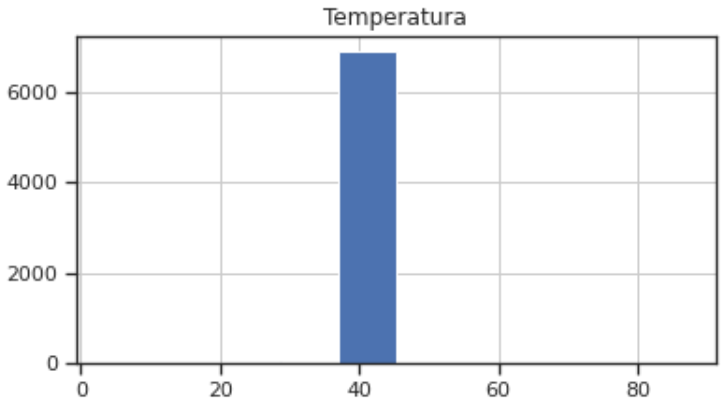


Ilustración 17 Gráfica temperatura

Por último se graficó la columna del diagnóstico los cuales arrojaron los siguientes resultados, para el diagnóstico 3 hay más de 4000 datos, seguido a este se encuentra el diagnóstico 1 con más de 1000 datos y por último está el diagnóstico 2 con un número de registros inferior a 1000. Se puede ver en la ilustración (18).

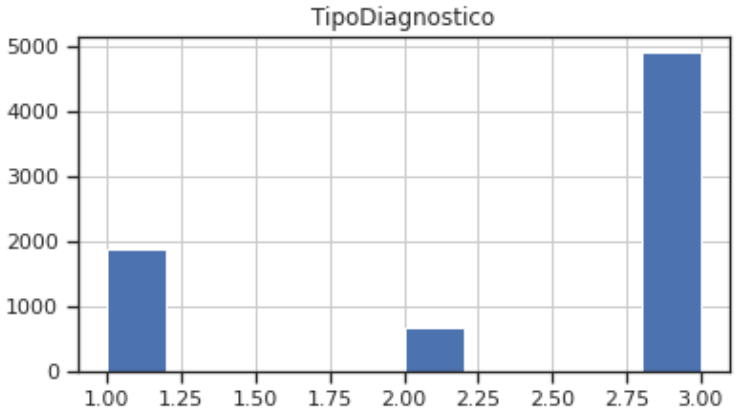


Ilustración 18 Gráfica DiagnosticoPrincipal

2.3.3 Gráficas de variables categóricas

La primera variable RSSS, la mayoría de los pacientes pertenecen a la categoría de subsidiado, esto se puede observar en la ilustración (19).

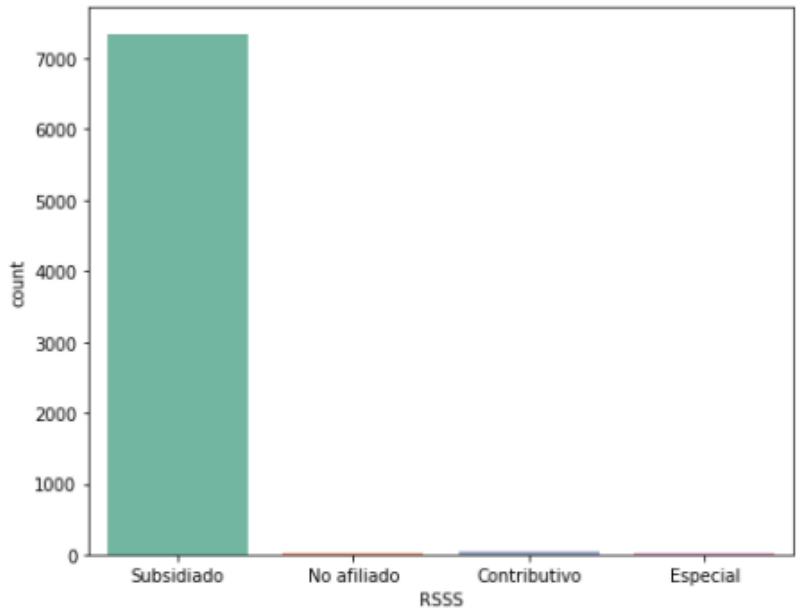


Ilustración 19 Categoría Rsss

Para la variable estrato, gran parte de los pacientes pertenecen al estrato 1. Ilustración (20).

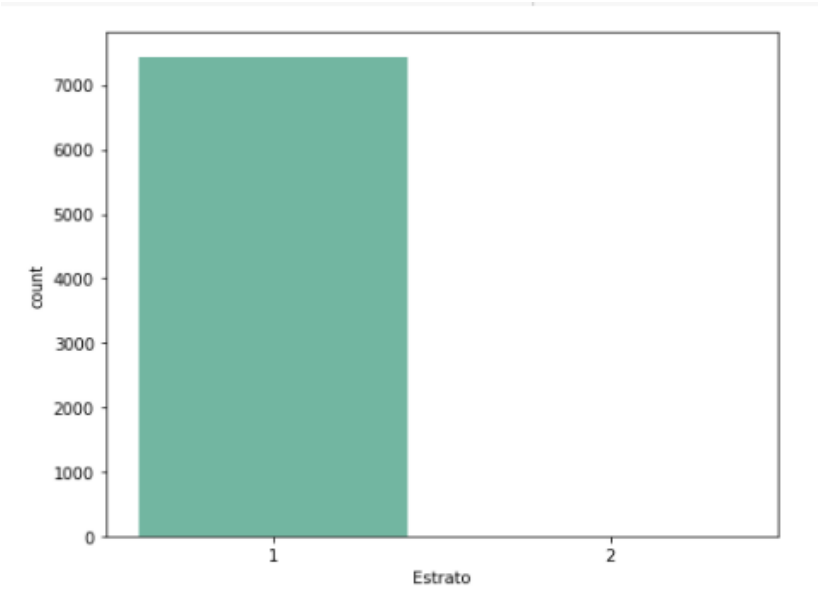


Ilustración 20 Categoría estrato

Así mismo, se graficó el estado civil, ilustración (21), en donde se muestra que la mayor parte de los datos se agrupan en la categoría soltero (color naranja) el cual

cuenta con más de 5000 registros de pacientes, seguido se encuentran union libre y casado (se muestran con los colores verde y púrpura respectivamente), por último viuda (rosado) y separado (verde limón).

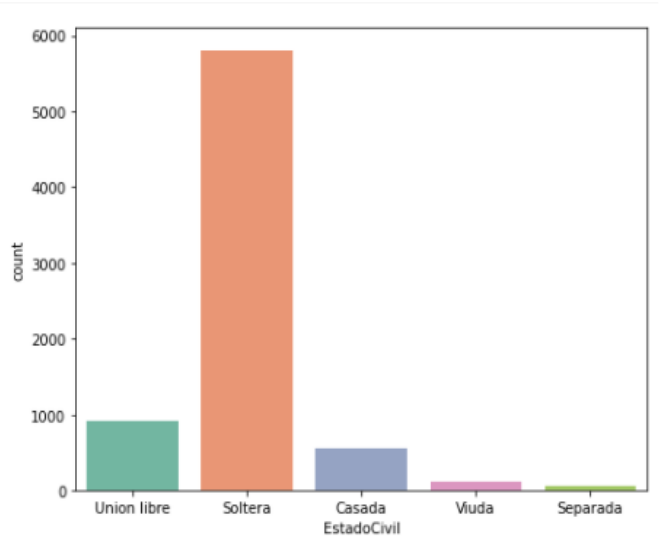


Ilustración 21 Categoría estado civil

Ademas, podemos ver la distribución de los datos en la ilustración (22), aquí se muestra el estado nutricional donde la mayoría de los pacientes tienen el peso adecuado (barra de color verde), en segundo lugar están los que tienen sobrepeso (barra de color purpura);

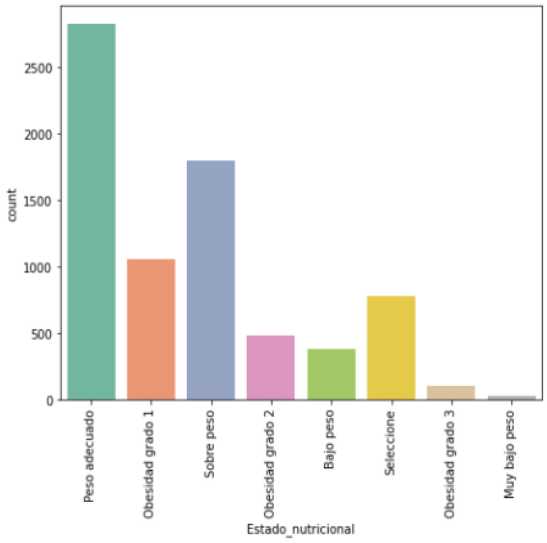


Ilustración 22 Categoría estado nutricional

Los siguientes son obesidad en primer grado (naranja), variable no identificada y nombrada así como “selecciones” que representa una cantidad por debajo de los 1000 datos hallados pertenecientes a esa categoría.

también se puede relacionar a datos vacíos o faltantes (color amarillo), obesidad en segundo grado se encuentra por encima de los 500 datos (color rosa), pacientes que presentan un peso bajo que está por debajo de los 500 datos (barra de color verde lima), y los dos últimos que son obesidad en tercer grado (color cafe) que está por encima de muy bajo peso (color gris).

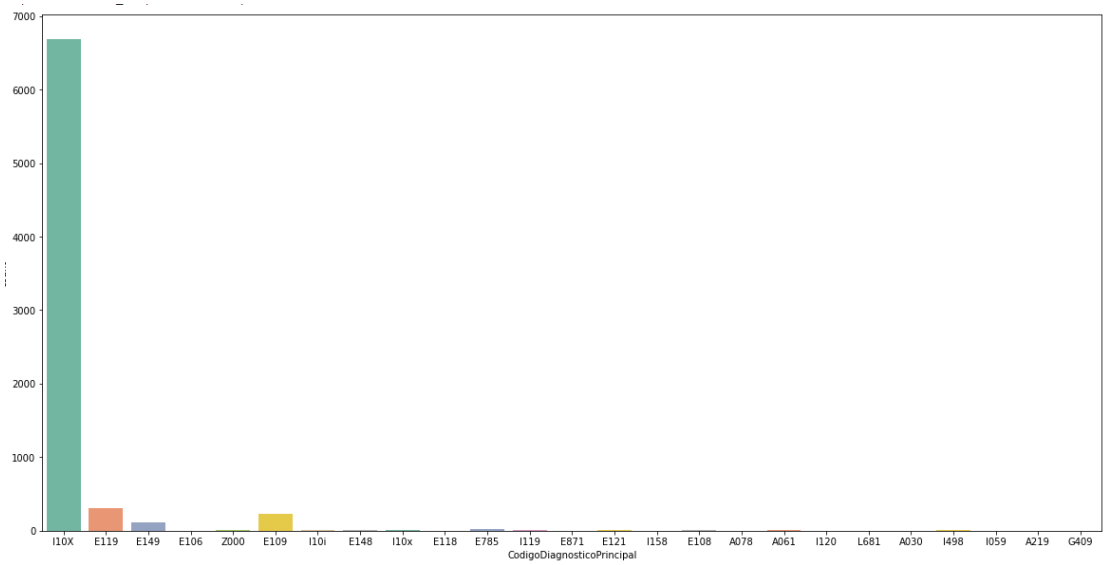


Ilustración 23 Diagnóstico principal

En la ilustración(23) se puede observar que la mayoría de los datos del diagnóstico principal pertenecen a I10X (representado por la barra de color verde), seguido a este están los datos de E119 (representado por el color zapote), los dos restantes que son E109 (color amarillo) y E149 (por el color morado).

2.3.4 Verificado y rellenado de datos faltantes

Antes de hacer el rellenado en las columnas que presentaban datos faltantes, se realizó una representación gráfica para ver de forma visual que columnas presentan la mayor parte de datos faltantes. En la ilustración (24) se puede ver las columnas que presentan datos faltantes, estas se pueden diferenciar porque presentan mayor número de franjas de color amarillo.

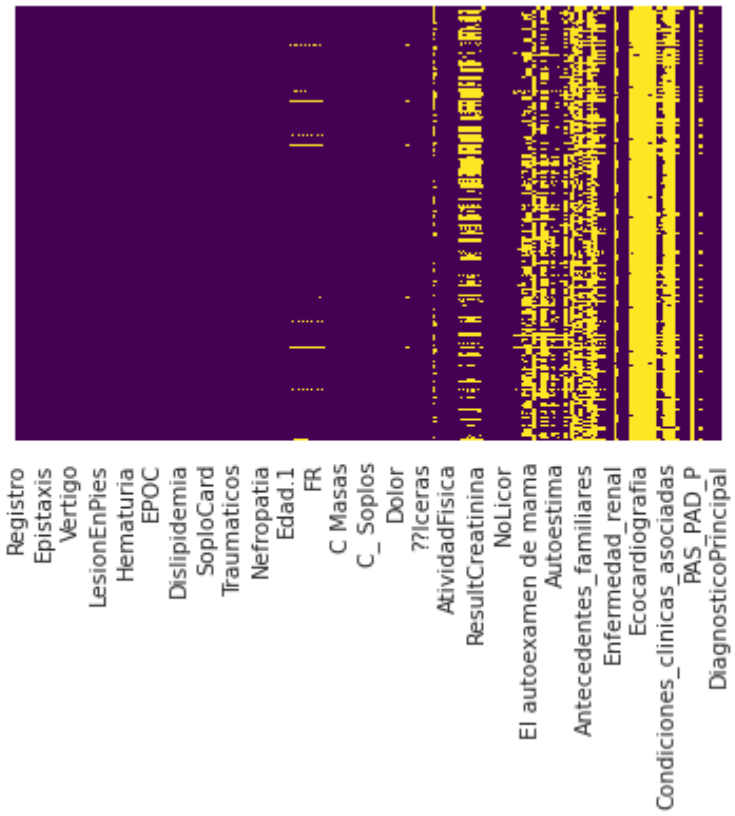


Ilustración 24 Gráfica de datos faltantes mapa de calor

A continuación, se realizó el llenado de datos faltantes en las columnas del DATASET, basados en los métodos interpolación lineal para aquellas variables de tipo numéricas y para las categorías el método PAD, se hizo las respectivas correcciones de igual manera las que presentaban errores en los datos, por ejemplo, en las columnas:

- **DiagnosticoPrincipal**: se encontraron espacios vacíos en algunos de los registros (NaN) y se rellenaron con: “No especificada sin mención de complicación”.

- **PAS_PAD_T1_1:** error en algunos registros que mostraban “oct-80”, los cuales se reemplazaron por el número “80”.

- **ResultCreatinina:** presentó errores en algunos registros, estos llevaban caracteres que no pertenecían al conjunto de datos y los cuales presentaban dificultades a la hora de procesarlos: “1-0, 0-7, 1-0|, 1-T, 0’.7, .0.8”, estos datos se reemplazaron por: “1.0, 0.7, 1.0, 1.0, 0.7 y 0.8” respectivamente.

Al finalizar, el llenado de datos nuevamente se realizó una representación gráfica para ver si se efectuaron los cambios en las columnas que contenían datos faltantes. En la ilustración (25) se muestra que las columnas que antes aparecían con datos faltantes se han corregido.

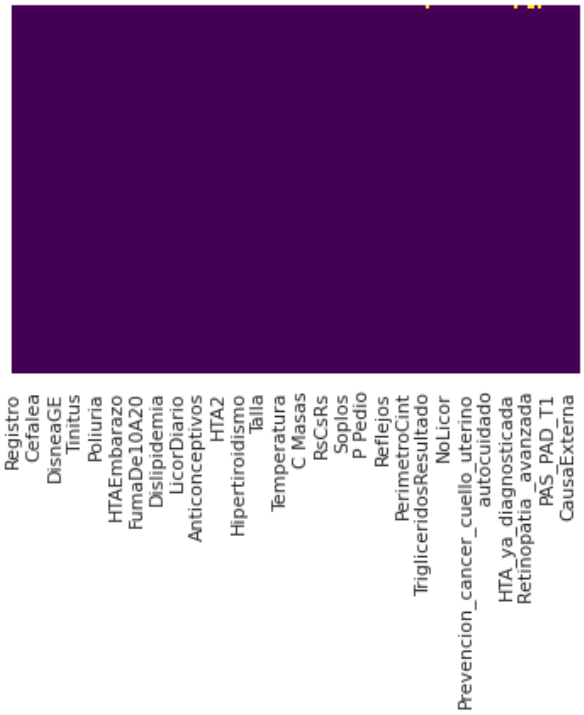


Ilustración 25 Gráfica de datos faltantes

2.3.5 Transformando columnas

Se realizó el reemplazamiento de todas aquellas variables que eran tipo categóricas o que tenían información representadas en letras como se representa en la siguiente ilustración (26).

```
for categorica in varcategoricas:
  if 0 not in df[categorica].astype('category').cat.categories:
    df[categorica] = df[categorica].cat.add_categories(0)
  df[categorica] = np.where(df[categorica]=='Si', 1, df[categorica])
  df[categorica] = np.where(df[categorica]=='No', 0, df[categorica])
  df[categorica] = np.where(df[categorica]==True, 1, df[categorica])
  df[categorica] = np.where(df[categorica]==False, 0, df[categorica])
  df[categorica] = np.where(df[categorica]=='True', 1, df[categorica])
  df[categorica] = np.where(df[categorica]=='False', 0, df[categorica])
  df[categorica] = np.where(df[categorica]=='0', 0, df[categorica])
  df[categorica] = np.where(df[categorica]=='1', 1, df[categorica])
  df[categorica].fillna(0, inplace =True)
```

Ilustración 26 Transformando columna

Todas aquellas que contenían como valor “Si” o “No” se le se le realizó la conversión a binario si=1 y no=0 y así sucesivamente con los demás valores a las cuales se les efectuó dicha transformación se muestra en la ilustración (27).

Cefalea	Epistaxis	DisneaME	Disuria	Lipotimia	Palpitaciones	DisneaGE	Edemas	Vertigo	Precordialgia
No	No	No	No	No	Si	Si	Si	No	No
Si	No	No	No	No	Si	No	No	Si	No
Si	No	No	No	No	Si	Si	No	Si	Si
Si	Si	No	No	No	Si	Si	No	No	No
Si	No	Si	No	No	Si	Si	Si	Si	Si

Ilustración 27 Muestra de algunas columnas categóricas

Como se puede evidenciar en la tabla se realizó la conversión de las columnas representadas en la siguiente ilustración (28) donde se le asignó el valor binario.

EstadoCivil_Casada	EstadoCivil_Separada	EstadoCivil_Soltera	EstadoCivil_Union libre	EstadoCivil_Viuda	CodigoDiagnosticoPrincipal_A030
0	0	1	0	0	0
0	0	1	0	0	0
1	0	0	0	0	0
0	0	1	0	0	0
0	0	1	0	0	0

Ilustración 28 Dummies a variables categóricas

Posterior a este proceso se procede a eliminar las variables que poseen menor importancia, al tener las variables óptimas se procede a eliminar las demás, las cuales no presentaban mucha relevancia. A continuación, la Ilustración (29) muestra dichas variables.

```
#Eliminando las variables que no son optimas para el modelo
pacientes_df_dummies=pacientes_df_dummies.drop(['Paciente','P Pedio','Alergicos','Dislipidemia','ConsumoLicor','EstadoCivil_Separada',
'ReducPeso','Palpitaciones','DisneaME','EstadoCivil_Viuda','CodigoDiagnosticoPrincipal_E149','Disuria','Deficitpulso','Epistaxis','Claudicacion',
'Tinitus','ConsumeVerdFrutas','Estrato','CodigoDiagnosticoPrincipal_G459','Polifagia','DMGestacional','CodigoDiagnosticoPrincipal_E871','Polidipsia',
'Cancer','EstadoCivil_Soltera','DisneaPxNoc','Licorc/8-15Dias','Soplos','1_indicaciones_sobre_sus_factores_riesgo_predisponente','NoAzucares',
'CodigoDiagnosticoPrincipal_G409','Lipotimia','IRC','LesionEnPies','Obesidad','CodigoDiagnosticoPrincipal_I059','Anticonceptivos','Hematuria','LicorDiario',
'Poliuria','Papiledema','TB','Amputaciones','FumaMenos10','Hipertiroidismo','??lceras','Ortopnea','EPOC','DisneaPE','Hemorragias',
'CodigoDiagnosticoPrincipal_E785','PAS_PAD_P_2','Prevencion_enfermedades_transmisi??n_sexual','Traumaticos','2_indicaciones_control_medicina_general_5_a?±os',
'HTAEmbarazo','Anticoagulacion','PerimetroCint','HTA_ya_diagnosticada','Estilosde_vida_saludable','Transfusionales','Autoestima',
'Prevencion_cancer_cuello_uterino','Resultado.2','Temperatura','Nutrici??n_y_alimentaci??n','Micro_albuminuria','Complicaciones _drogas_psicoactivas',
'El autoexamen de mama','Como_prevenir_diabetes_hipertensi??n_osteoporosis','I_Yugular','Consecuencias_del_consumo_alcohol_cigarrillo',' autocuidado',
'Glicemia_postprandial','IAM2','EnfVascular','4_citologia_cervico_uterina','Nefropatia','Soplocarotideo','CodigoDiagnosticoPrincipal_A078','C Masas',
'Displidemia_(cualquiera)','Retinopatia _avanzada','DM1DM2.1','Nodulotiroideo','Enfermedad_arterial_periferica','Fondo_ojo','Enfermedad_renal',
'CodigoDiagnosticoPrincipal_E121','EArterialPerif','Se_realizo','CodigoDiagnosticoPrincipal_E148','Cardiopulmonar','CodigoDiagnosticoPrincipal_A030',
'CodigoDiagnosticoPrincipal_E119','Creatinina','CodigoDiagnosticoPrincipal_A061','CodigoDiagnosticoPrincipal_E106','CodigoDiagnosticoPrincipal_A219',
'CodigoDiagnosticoPrincipal_E118','CodigoDiagnosticoPrincipal_E109','CodigoDiagnosticoPrincipal_E108','Crueces_AV','CodigoDiagnosticoPrincipal_I10i',
'Abdomen','CodigoDiagnosticoPrincipal_I10x','Masas','CodigoDiagnosticoPrincipal_I119','CodigoDiagnosticoPrincipal_I158','CodigoDiagnosticoPrincipal_I498',
'Megalias','CodigoDiagnosticoPrincipal_L681','CodigoDiagnosticoPrincipal_Z000','Extremidades','Exudado','EstadoCivil_Union libre','Estado_nutricional',
'HTA2','FumaDe10A20','Vertigo','AtividadFisica','NoLicor','IAM1','Angina','Sudoracion','Hospitalarios','FinalidadConsulta','Sensibilidad',
'PredominioIngestaDeGrasa','Retinopatia','HTA1','UsarEdulc','Dolor','C_ Soplos','Diabetes_mellitus_(DM2)_1','Hipotiroidismo','UtilizaSalero','Menopausia',
'DisneaGE','PAS_PAD_T1_1','Diabetes_mellitus_(DM2)_2','DolorNeuritico','Asma','Reflejos','EstadoCivil_Casada','InterconsultaNutricionista','NoFumar',
'DisminuGrasa','HacerEjer','PAS_PAD_T2_1','EnfCoronaria','SoploCard','Precordialgia','FumaMasDe20','SintomasVisuales','Registro'
],axis=1)
```

Ilustración 29 Variables con poca relevancia

2.3.6 Normalización de datos

Teniendo los resultados una vez realizado el proceso de transformación de los datos se procede a hacer la normalización del Dataframe, para esto se utiliza el método *StandardScaler*, este método permite estandarizar los datos, lo cual es útil para lo que son negativos .

```
from sklearn import preprocessing
dftp = df[[v for v in df.columns if v not in ['CodigoDiagnosticoPrincipal']]]

X = preprocessing.StandardScaler().fit(dftp).transform(dftp)
X[0:5]
```

```
X.shape
```

Ilustración 30 método StandardScale

Se organizan los datos en una distribución estándar. Para hacer la normalización se dividió el Dataframe en dos partes dos variables diferentes). La variable x como la variable independiente contiene una sección de los datos a normalizar. Mientras que la variable Y como la variable dependiente contienen la columna de respuesta (CodigoDiagnosticoPrincipal).

Luego de aplicar el método de estandarización en la variable independiente (x) se puede observar en la ilustración (31) algunas de las columnas cuyos datos se le aplicó la normalización.

	Edad	SignosVitales	Antecedentes_familiares	IMC	TrigliceridosResultado	ResultColestTotal
0	0.216020	0.128713	0.440305	-0.031218	-0.069531	-0.021812
1	0.190701	0.129540	0.443134	-0.031418	-0.069978	-0.021952
2	-0.009944	0.134162	-0.067714	-0.032539	-0.072475	-0.022735
3	0.170895	-0.260553	-0.074172	-0.035643	-0.079387	-0.024903
4	0.111372	0.130467	-0.065850	-0.031643	-0.070479	-0.022109

Ilustración 31 Estandarización de datos

De las características en base al coeficiente de correlación de Pearson (>0.5) para indicar si dos variables están relacionadas entre sí o no y para determinar si la correlación entre las variables es significativa, se compara el valor p con su nivel de

significancia al cual se le asigna un valor de (>0.14), de este proceso quedan entonces 56 Columnas.

Se aplica el algoritmo de regresión logística que es un algoritmo de clasificación de aprendizaje automático que se utiliza para predecir la probabilidad de una variable dependiente categórica. En la regresión logística, la variable dependiente es una variable binaria que contiene datos codificados como 1 (sí, éxito, etc.) o 0 (no, fracaso, etc.)[14]

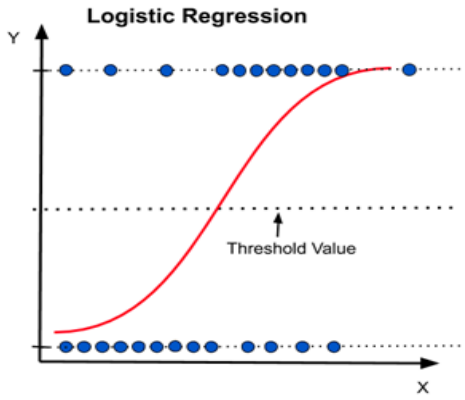


Ilustración 32 Variables con poca relevancia(Harshita Yadav).[15]

Se utilizó el método RFE (Recursive Feature Elimination)[12],[13], el cual se implementó con la biblioteca scikit-learn para la selección de variables relevantes, el cual resultó fiable y eficaz para la eliminación de variables irrelevantes y de esta forma reducir los datos que no son muy significativos para el entrenamiento.

Finalmente, quedó un total de 19 columnas y 6.499 filas MC, Edad, Peso, Talla,HTA1,DM1DM2,Menopausia,Quirúrgicos,Estado_nutricional,PerimetroCint,Creatinina,clasificacionFinalRiesgoCardiovascular,EstadoCivil_Soltera,Riesgo_edad,prevencion_enfermedades_transmisión_sexual,Antecedentes_familiares,riesgo_perimetro,HTA_ya_diagnosticada y CodigoDiagnosticoPrincipal.

3. OPTIMIZACIÓN DEL MODELO DE HIPERTENSIÓN

En en este capítulo se explica cómo se realizó la optimización del modelo de hipertensión, el partió del análisis de datos del modelo anterior, donde se encontró que en algunas variables se encontraron valores que no corresponden lo que estaba generando una mala predicción, porque existen valores erróneos que no eran significativo para poder realizar la clasificación, consecuentemente se logró conseguir otro dataset (Hospital Local de Arjona Bolívar), al cual se le tuvo que aplicar nuevamente el proceso de limpieza de los datos, e identificar cuáles son las variables de mayor correlación y de mayor diagnóstico.

Se aplica la fase de modelado ya que se hizo el entrenamiento del algoritmo para lo cual fue necesario aplicar técnicas de filtrado de variables como la técnica RFE y *StandardScaler*, estas técnicas fueron aplicadas para conocer cuáles eran las variables más significativas del conjunto de datos, lo cual arrojó 19 columnas más significativas y así mismo fue necesario normalizar los datos para transformar los valores de los datos a un rango determinado de entre 0 a 1 de forma que el escalador organice por medio de la función *fit.transform()* transforma las columnas arrojadas por el algoritmo de filtrado *RFE* en un array bidimensional.

Con los datos filtrados y normalizados se realiza la división de los datos o *Split* el conjunto de datos se dividió en dos partes una para entrenamiento con el 80% y una parte para prueba del 20% con un total de 6.499 datos y 19 columnas de entrada y 3 de salida para el entrenamiento del modelo. A continuación, se representará el paso paso del proceso de la lectura de datos ya optimizados y exportados de lo elaborado en el (capítulo 2)

Inicialmente se subió el archivo xlsx a una carpeta en google Drive para luego ser llamada desde Google Colab donde nos permite ejecutar y programar en Python en tu navegador

A continuación en la siguiente ilustración(33) se mostrará los procesos realizados desde la limpieza de los datos hasta en el entrenamiento y creación de la red neurona

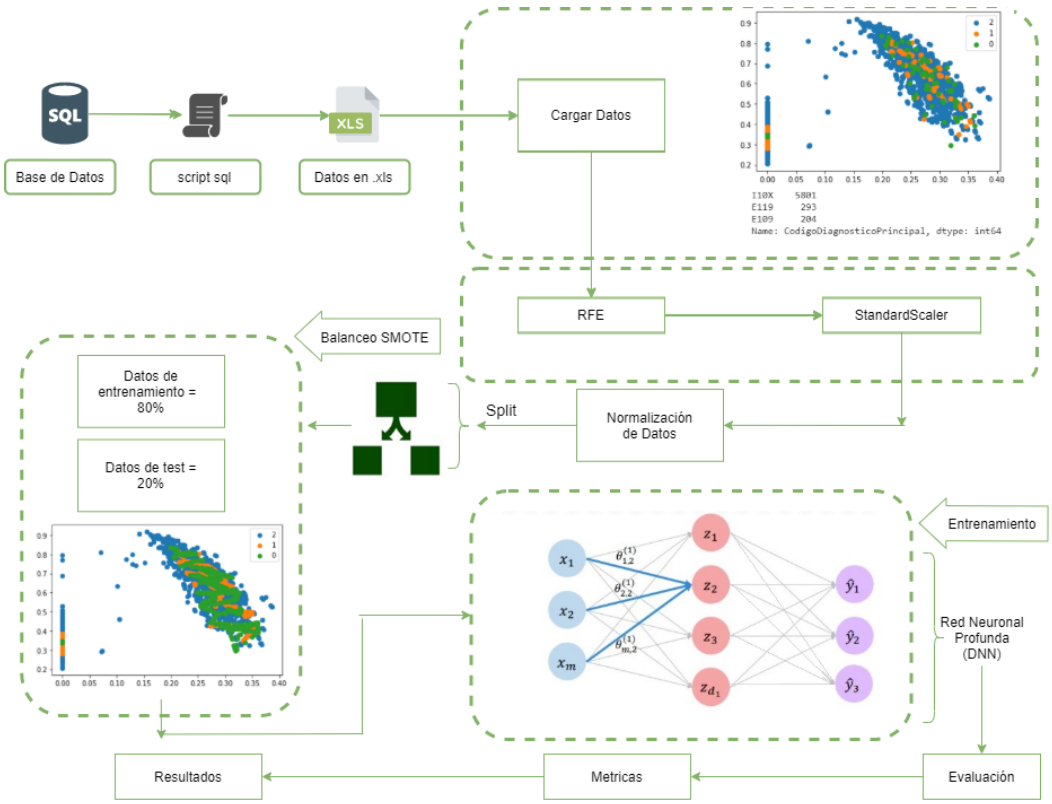


Ilustración 33 Clasificación de enfermedades cardiovasculares

Se realizó el análisis del mapa de calor para validar que efectivamente no exista ningún dato faltante ya una vez realizado la limpieza de los datos, esto se hace solamente para corroborar que efectivamente se así como se puede evidenciar en siguiente ilustración (34).

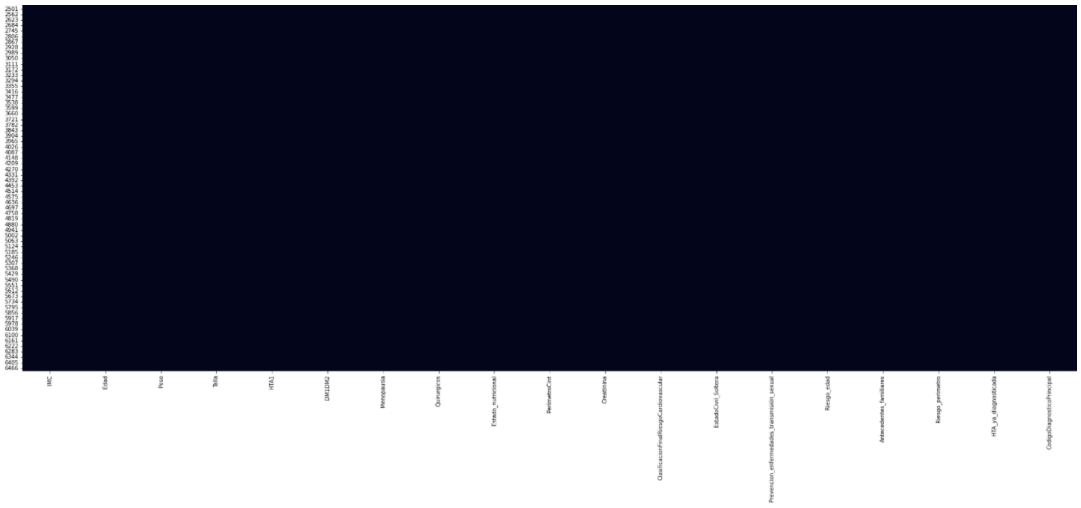


Ilustración 34 Mapa de calor Datos Faltantes

Se realiza una distribución de los códigos de diagnósticos que en este caso es nuestra variable de respuesta para hallar los porcentajes que más relevancia tienen para así identificar cuáles clases son más efectivas para la predicción como se muestra en la siguiente ilustración(35)

```
(dataset['CodigoDiagnosticoPrincipal'].value_counts()/dataset.shape[0])*100
```

```
I10X    89.259886
E119     4.508386
E109     3.138944
E149     1.523311
E785     0.292353
Z000     0.230805
I10i     0.184644
I119     0.169257
E148     0.107709
E108     0.092322
I10x     0.076935
I498     0.061548
E121     0.061548
A061     0.061548
A219     0.046161
I059     0.046161
E106     0.046161
I158     0.030774
E118     0.015387
A078     0.015387
L681     0.015387
A030     0.015387
Name: CodigoDiagnosticoPrincipal, dtype: float64
```

Ilustración 35 Porcentajes de Codigo de CodigoDiagnosticoPrincipal

Como se puede evidenciar en la ilustración(35) podemos notar que las clases con mayor porcentajes son las siguientes: I10X - 89.25%, E119 - 4.50%, E109 - 3.13% Por los que después se procede a seleccionar esas 3 que fueron las 3 clases con más datos.

```
new_df=dataset.loc[(dataset['CodigoDiagnosticoPrincipal']=='I10X') | (dataset['CodigoDiagnosticoPrincipal']=='E119') | (dataset['CodigoDiagnosticoPrincipal']=='E109')
```

Ilustración 36 selección por clase

Se hizo el conteo de muestra por clase que en este caso fueron las 3 mencionadas anteriormente para identificar qué cantidad existe en cada una de las clases, en al siguiente ilustración(37) podemos identificar las 3 clase con su respectiva cantidad

```
new_df['CodigoDiagnosticoPrincipal'].value_counts()

I10X    5801
E119     293
E109     204
Name: CodigoDiagnosticoPrincipal, dtype: int64
```

Ilustración 37 Conteo de muestra por clase

Un vez realizada las muestras por clases se produce una codificación usando LabelEncoder, referenciando las clases con un valor numérico como se puede evidenciar en la siguiente ilustración (38)

```
labelencoder = LabelEncoder()
new_df['CodigoDiagnosticoPrincipal']=labelencoder.fit_transform(new_df['CodigoDiagnosticoPrincipal'])
new_df['CodigoDiagnosticoPrincipal'].value_counts()

2    5801
1     293
0     204
Name: CodigoDiagnosticoPrincipal, dtype: int64
```

Ilustración 38 Codificación LabelEncode

Definición de variables independientes y dependientes

La variable *X* representa el conjunto de atributos sin tener en cuenta la variable de respuesta. Mientras la variable *y* contiene sólo los datos de la variable de respuesta.

```
X=new_df.drop(['CodigoDiagnosticoPrincipal'],axis=1)
Y=new_df['CodigoDiagnosticoPrincipal']
```

Ilustración 39 Definición de variables

Se divide el dataset en conjuntos típicamente en los siguientes porcentajes: 80% de los datos para entrenar la red neuronal, el 20% para probar cómo se comporta el algoritmo contrastando. Aunque en este caso solo se dividió en dos conjuntos y para dividir el dataset se hace uso de la función “train_test_split” que simplifica la tarea: Así se obtiene (x_train, y_train) con los datos de entrenamiento y por otro lado (x_test, y_test) con los datos que se reservan para validar posteriormente el modelo.

En tercer lugar, se construye una red neuronal de arquitectura secuencial con dos capas ocultas densamente conectadas con todas las salidas de cada neurona de la capa anterior, la estructura final quedó de 128 neuronas en cada capa oculta, 18 neuronas de entrada y una capa de salida de 3 neuronas. Esta configuración de capas y neuronas se llegó empíricamente debido a que encaja bien para obtener una buena precisión en este problema.

Asimismo, se agrega una capa de exclusión entre la entrada (o capa visible) y la primera capa oculta. La tasa de abandono o la omisión de neuronas se establece del 34%, lo que significa que es una técnica de regularización para reducir el sobreajuste o sobre entrenamiento de la red neuronal.

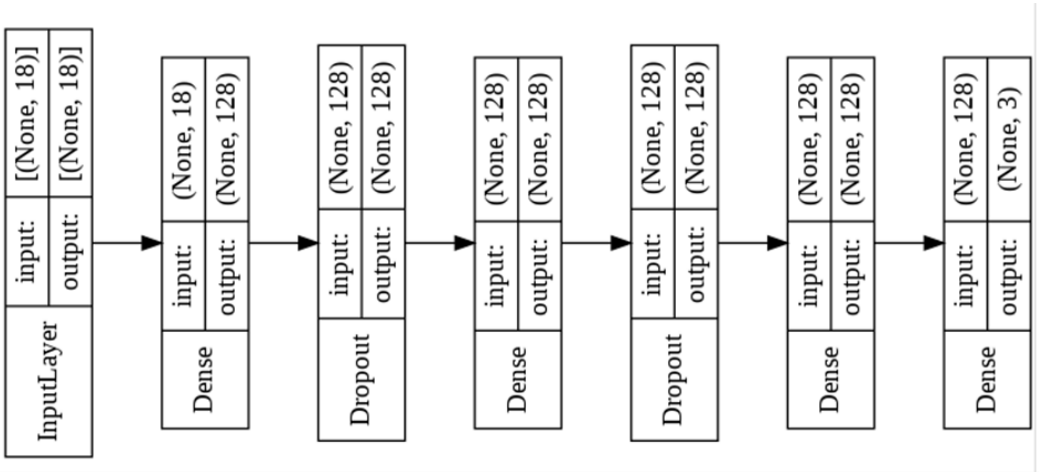


Ilustración 40 Red Neuronal

Cómo se muestra en el código anterior, todos los nodos de diferentes capas ocultas utilizaron funciones de activación Rectified Linear Unit (ReLU)[26] debido a su efectividad ya que requiere cálculos simples, la velocidad de entrenamiento es más rápida, menos problemas de saturación, menor número de épocas y muestras.

Simplemente reemplaza los valores negativos con cero y los valores positivos permanecen sin cambios.

El siguiente código permite compilar el modelo con la función model.compile, especificando la función de optimización, la función de coste o pérdida y las métricas que se utilizará. En este caso, se utilizó el optimizador Adam, que es un optimizador estocástico basado en gradientes, para el ajuste de pesos entre las conexiones, la función de pérdida Categorical_crossentropy para cuando hay dos o más clases de etiquetas y para la métrica se utiliza accuracy (o tasa de acierto).

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Ilustración 41 Compilación de modelo

Se entrena el modelo con la función model.fit que recibe dos parámetros: variable independiente y la variable dependiente destinadas al proceso de entrenamiento inicial, 50 interacciones o épocas, el resultado del entrenamiento se guarda en la variable history, de la cual, se extrae el histórico de los datos del entrenamiento.

```
history=model.fit(x_train, y_train ,epochs=50, validation_split = 0.1)
```

Ilustración 42 Entrenamiento de modelo

Se guarda el proceso de entrenamiento para mostrar y validar algunas métricas cómo la precisión, la pérdida en entrenamiento, la validación y las épocas. y se representa gráficamente en la siguiente figura donde se muestran las épocas y la pérdida que hubo.

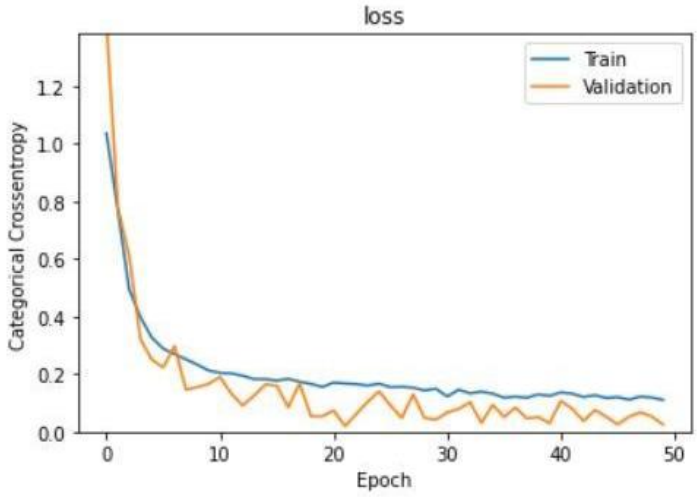


Ilustración 43 Gráfica de perdida y precisión

Una vez realizada y evaluada la red neuronal procedemos a realizar el guardado del modelo de dicha red construida, para poder ser usada y pasarle las variables independientes para que realice su proceso de predicción, en la siguiente ilustración(44) se refleja la función que fue utilizada para realizar el guardado.

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")
```

Ilustración 44 Guardado de modelo

El cual genera dos tipos de archivos llamados model.json y la serialización del modelo que en este caso es el que se llama model.h5

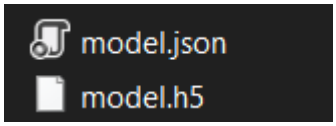


Ilustración 45 Archivos de modelo

Una vez realizado el proceso anterior cargamos los dos archivos generados en la ilustración (46)

```
# load json and create model
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
```

Ilustración 46 Cargado del modelo

El siguiente paso es la predicción, utilizando la función predict la cual recibe el conjunto de variables independientes definidas anteriormente para pruebas.

```
loaded_model.load_weights("model.h5")
Xnew = array([[28.44,69,71.0,1.58,0,0,0,0,0,0,0,1,0,1,1,0,0,0]])
ynew = loaded_model.predict_classes(Xnew)
print("Predicted= %s" % (ynew[0]))
```

Ilustración 47 Lectura del modelo

Evaluación: Antes de proceder con el despliegue de la solución, se debe evaluar objetivamente el rendimiento del modelo.

Para comprobar la precisión del modelo, se usa el conjunto de datos de test que se había reservado. Se le pedirá al modelo que haga una clasificación para cada registro que tenemos en ese 20% de datos que se guardaron aparte y se compara con la predicción del algoritmo con el etiquetado inicial.

```
results = model.evaluate(x_test, y_test)
print('prueba')
print('Final test set loss: {:4f}'.format(results[0]))
print('Final test set accuracy: {:4f}'.format(results[1]))
```

Ilustración 48 Evaluación del modelo

El proceso realizado anteriormente se obtuvo una precisión de este modelo del 97%, y la pérdida es 9%, para poder realizar el diagnóstico de las tres (3) categorías de enfermedades

Código de Enfermedad	Código Clasificación	Diagnóstico Principal
I10X	2	HIPERTENSIÓN ESENCIAL
E119	1	DIABETES MELLITUS NO INSULINODEPENDIENTE SIN MENCION DE COMPLICACION
E109	0	DIABETES MELLITUS INSULINODEPENDIENTE SIN MENCION DE COMPLICACION

4. ANÁLISIS, DISEÑO Y DESARROLLO DE APLICACIÓN WEB

En este capítulo se explica cómo se desarrolló la aplicación web para el despliegue del modelo de predicción de hipertensión, para lo cual se seleccionó el framework DJANGO debido a su desarrollo en Python y por ser compatible con el lenguaje con el cual está desarrollado el modelo y además se seleccionó la herramienta Docker, necesaria para realizar la virtualización, el desarrollo, pruebas y análisis de rendimiento.

Django usa un componente basado en la arquitectura “shared-nothing” (cada parte de la arquitectura es independiente de las otras, y por lo tanto puede ser reemplazado o cambiado si es necesario). En la siguiente imagen se muestra la arquitectura anteriormente mencionada.

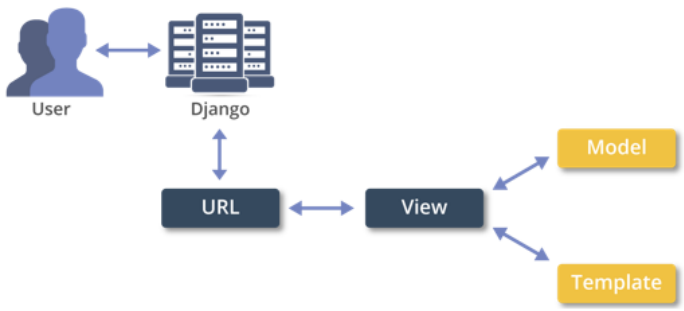


Ilustración 49 Arquitectura de framework Django.[11]

4.1 Identificación de requerimientos de la Aplicación

Para identificar los requerimientos se utilizaron la siguiente plantilla

R1: Loguearse al Sistema

Plantilla de Requerimientos	
Función	Iniciar sesión con las credenciales para hacer uso del sistema
Descripción	Todos los usuarios que tengan acceso a una credencial(Usuario y contraseña) podrán acceder al sistema.
Entradas	Usuario y contraseña
Fuentes	Teclado
Salida	Confirmación de ingreso al sistema

Proceso	El sistema deberá validar si el usuario que desea ingresar al sistema está registrado en la base de datos.
Restricciones	Si el usuario está registrado, el sistema automáticamente da el acceso a la aplicación web. Si el usuario no está registrado, el sistema indicará que los datos son erróneos y no podrá ingresar.
Fecha [Tipo]	10-06-2021 Esencial y Primario

Fuente: Elaboración propia

R2: Registrar los pacientes

Plantilla de Requerimientos	
Función	Registrar en el sistema al paciente
Descripción	El usuario administrador es el encargado de registrar cada paciente al sistema
Entradas	Datos personales obligatorios: Nombres, Apellidos, Teléfono, Dirección, Fecha de nacimiento, Sexo y Estado civil.
Fuentes	Paciente
Salida	Confirmación de paciente registrado
Proceso	El sistema despliega el formulario para el registro del paciente, en el cual se deben ingresar los datos personales descritos en la entrada.
Restricciones	El paciente solo debe estar registrado una sola vez.
Contactos	Por determinar
Fecha [Tipo]	10-06-2021 Esencial y Primario

Plantilla de Requerimientos	
Función	Realizar una consulta de predicción de un paciente
Descripción	Se mostrará una ventana en donde se elige al paciente que se le va a realizar la predicción y se deben responder unas preguntas de si o no.
Entradas	El usuario administrador debe elegir al paciente y responder unas preguntas relacionadas con la salud del paciente.
Fuentes	Base de datos del sistema y el paciente.
Salida	Mostrará el tipo de predicción de acuerdo a los datos suministrados por el paciente
Proceso	El usuario administrador debe elegir al paciente que desea realizar la predicción y este deberá responder unas preguntas relacionadas con su salud.
Restricciones	
Fecha [Tipo]	10-06-2021 Esencial y Primario

R3: Visualización del Diagnóstico de Predicción de Hipertensión

Plantilla de Requerimientos	
Función	Visualizar la información que se ha manejado en la aplicación.
Descripción	Se visualizará una estadística de todo lo que se ha realizado en el sistema en cuanto a pacientes registrados, usuarios administradores, Cantidad de predicciones hechas.
Entradas	
Fuentes	Base datos
Salida	Visualización de las estadísticas.

Proceso	El usuario administrador deberá dirigirse dentro del sistema en la opción Dashboard para obtener estos datos de visualización.
Restricciones	Solo el usuario administrador tendrá acceso a estos datos
Fecha [Tipo]	10-06-2021 Esencial y Primario

4.2 Diseño del Sistema

Con base a los requerimientos que se realizaron para la elaboración de la aplicación web se diseñaron los siguientes mockup para luego proceder el desarrollo de diseños y funcionalidad de acuerdo a los requerimientos, La siguiente ilustración(50) hace referencia al requerimiento R1 en donde la priemera fila llamada funcion de la plantilla de requerimiento explica brevemente que funcionalidad tiene.

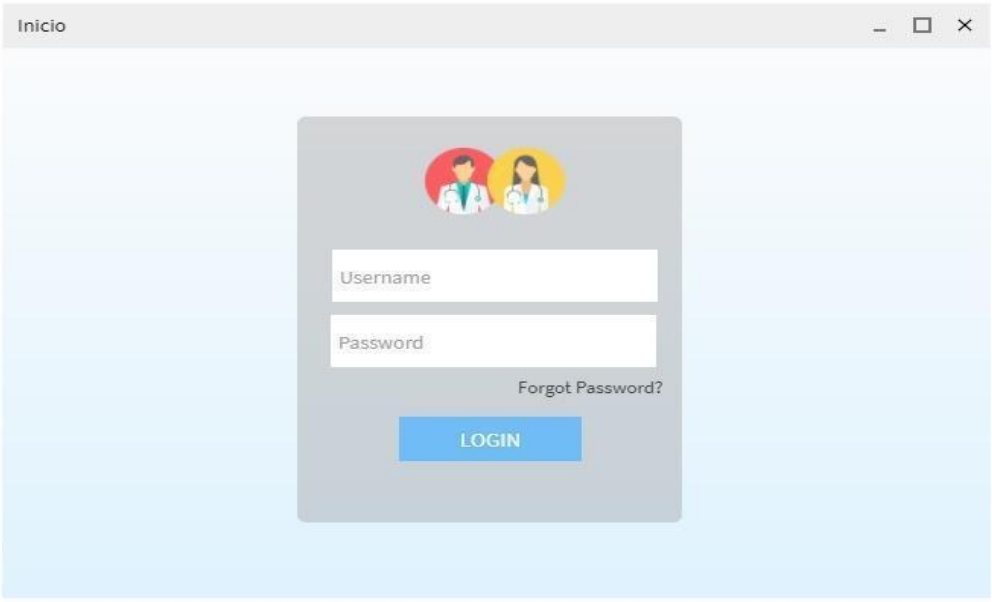


Ilustración 50 Loguearse al Sistema

La siguiente ilustración(51) hace referencia al requerimiento R2 en donde la primera fila llamada función de la plantilla de requerimiento explica brevemente qué funcionalidad tiene.

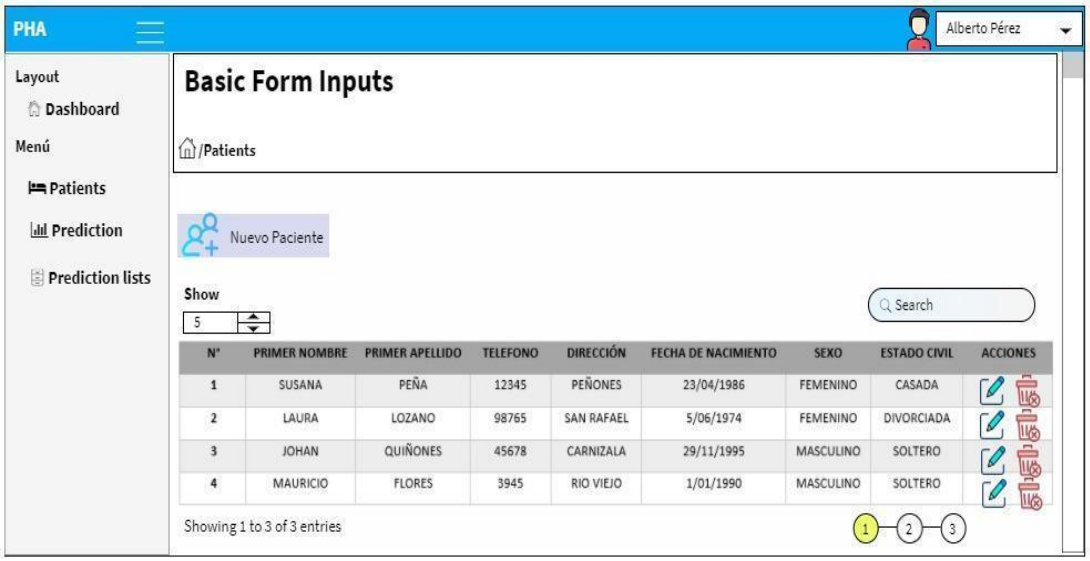


Ilustración 51 Registrar los pacientes

La siguiente ilustración(52) hace referencia al requerimiento R3 en donde la primera fila llamada función de la plantilla de requerimiento explica brevemente qué funcionalidad tiene.

Ilustración 52 Dashboard del sistema

La siguiente ilustración(53) hace referencia al requerimiento R2 en donde la primera fila llamada función de la plantilla de requerimiento explica brevemente qué funcionalidad tiene

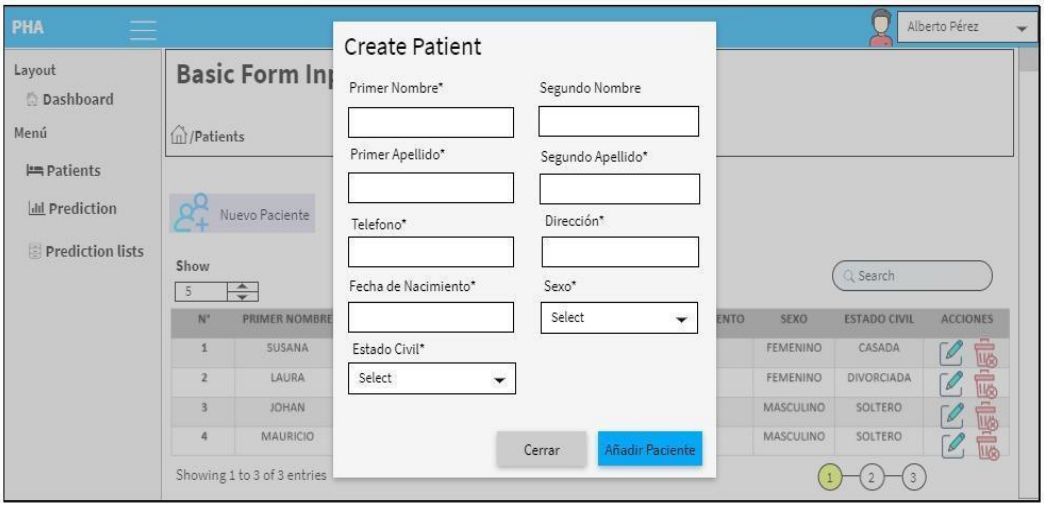


Ilustración 53 Registrar los pacientes

La siguiente ilustración(54) hace referencia al requerimiento R3 en donde la primera fila llamada función de la plantilla de requerimiento explica brevemente qué funcionalidad tiene

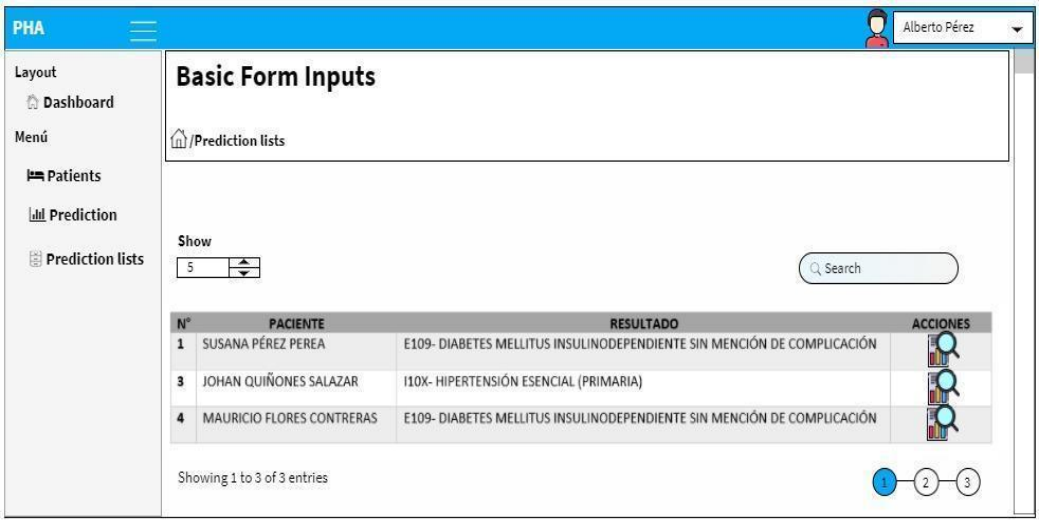


Ilustración 54 Resultado predicciones

4.3 Arquitectura de Base Datos

La siguiente ilustración(55) hace referencia al diagrama UML del sistema que se basa en un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, como se puede visualizar tenemos se crearon 3 tablas para manejar la información del sistema.

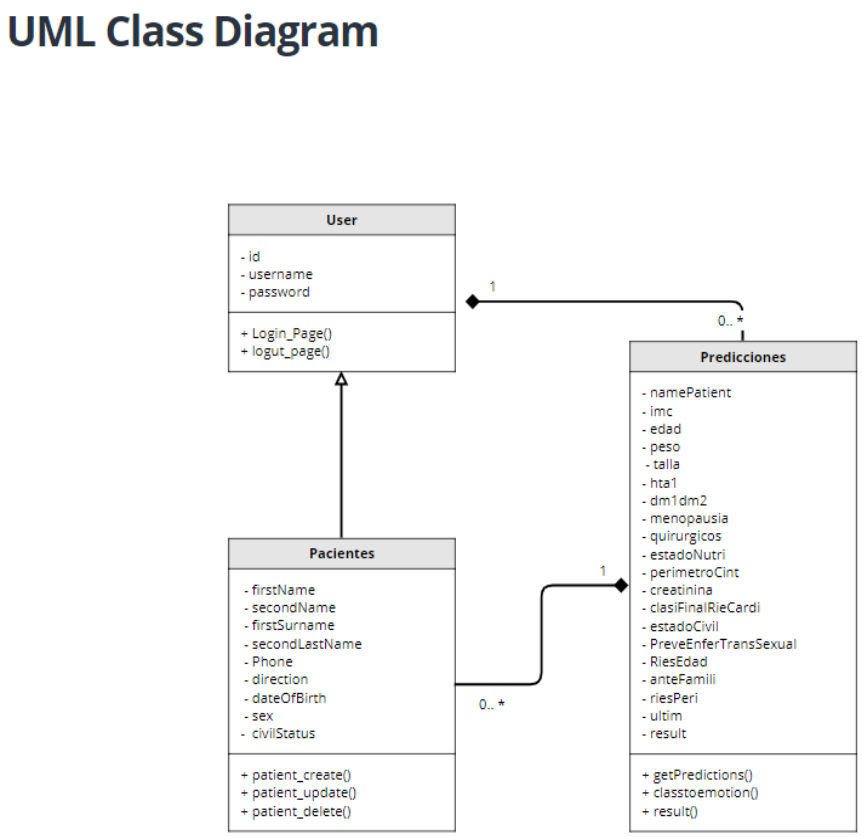


Ilustración 55 UML Diagrama de Clases de base de datos

User: En la siguiente tabla se manejan los usuarios del sistema que este caso sería el doctor que está realizando la diagnóstico de sus pacientes, las funciones que realiza esta clase obtiene los siguientes métodos:

- login_page();
  Este método es el encargado de realizar la autenticación del usuario

- logut_page();
  Este método es es el encargado de cerrar la autenticación del usuario

Pacientes: En la siguiente tabla se manejan la información básica de los pacientes, la cual se encuentra relacionada con la de User ya que el usuario que este caso va a ser el doctor tiene permitido crear pacientes, por eso se centra una relación entre ellas, las funciones que realiza esta clase obtiene los siguientes métodos:

- patient_create();
  Este método es el encargado de crear los pacientes y guardarlos en la base de datos.

- patient_update();
  Este método es el encargado de actualizar los datos del paciente y guardarlos en la base de datos.

- patient_delete
  Este método es el encargado de eliminar los pacientes y quitarlos de la base de datos.

Predicciones: En la siguiente tabla se manejan la información de las variables independientes que se usaron para entrenar evaluar la red neuronal que en este caso son las variables que quedaron después de haber realizado en proceso de limpieza y entrenamiento de la red neuronal esta se encuentra relacionada con la de User y Pacientes ya que el usuario que este caso va a ser el doctor tiene permitido realizar las predicciones de dichos pacientes ya haya creado anteriormente, las funciones que realiza esta clase obtiene los siguientes métodos:

- getPredictions();
  Este método recibe los parámetros de las variables independientes y carga el modelo que se generó después de haber hecho la red neuronal.

- classtoemotion();
  Este método es el encargado de procesar las 3 clases que quedaron definidas para arrojar el resultado de la predicción, que sería el código del diagnóstico del paciente con la definición de dicho código.

- result();
  Este método es el encargado de procesar los datos que se digitan en el formulario de predicciones y da el resultado si fue exitosa la predicción o ocurre algún inconveniente.

4.4 Casos de uso

En la siguiente ilustración(56) se presenta un diagrama de caso de uso que será la descripción de las actividades que deberá realizar alguien o algo para llevar a cabo los proceso utilizado del aplicativo web.

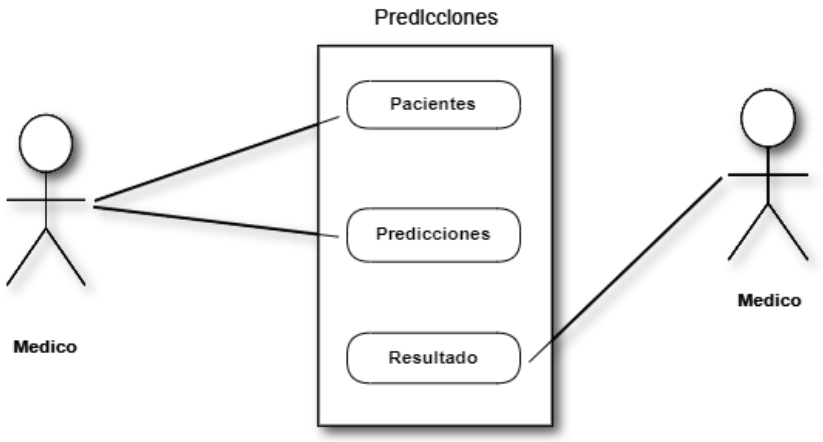


Ilustración 56 Casos de uso

En la definición del caso de uso podemos observar que un médico realiza dos procesos el cual tiene un resultado final en este caso el médico puede ser capaz de crear pacientes y a la vez realizar las predicciones y se obtiene un resultado que sería una de las 3 clases de los código de diagnóstico de la hipertensión el resultado se basará de la información suministrada en la predicción que le hacen al paciente y saber que clasficacion posee.

4.5 Despliegue con Docker

Para configurar Django y que se ejecute en Docker con Postgres. Para entornos de producción, agregaremos Nginx y Gunicorn. Donde Nginx ofrece el contenido estático de un sitio web de forma rápida y fácil y el Gunicorn nos permite servir nuestra aplicación Django con múltiples workers (trabajadoras) como podemos evidenciar en la ilustración (57) se identifican estas tecnologías mencionadas.

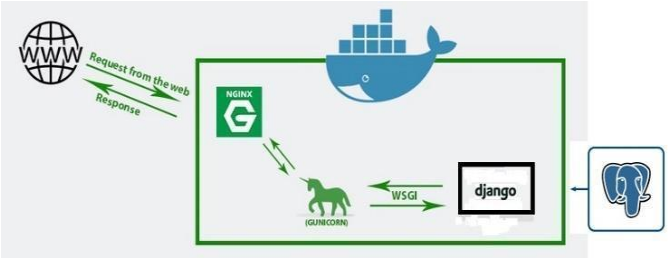


Ilustración 57 Arquitectura de despliegue.[11]

El despliegue se realizó en el sistema operativo window 10 con una tecnología de window que es subsystemforlinux que se basa en crear un subsistema dentro del sistema operativo original que en este caso es window 10 y el subsistema seria Linux el cual es utilizado para poder instalar docker en el proceso que se llevo a cabo para realizar esta configuración fue la siguiente.

Primeramente para realizar la instalación de docker en este sistema operativo primero se debe validar que tengamos activa la característica de virtualización de nuestro computador el cual se puede verificar en el administrador de tareas en la siguiente ilustración(58) se evidencia esta configuración.

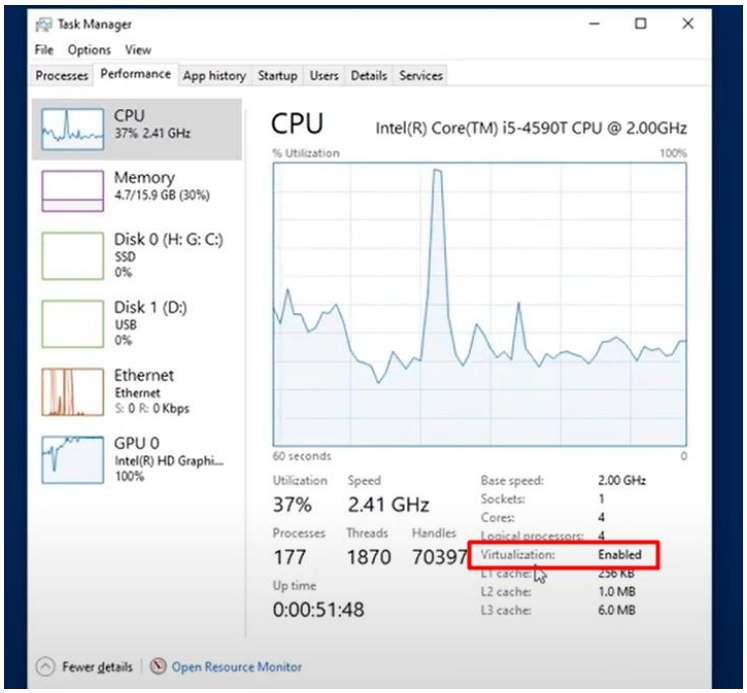


Ilustración 58 Verificacion de virtualización Window 10.

Una vez verificado que efectivamente tengamos activada las característica de virtualización procedemos a ejecutar las instrucciones que se presentan en la ilustración (59) en el PoweShell, este debe ser ejecutado como administrador.

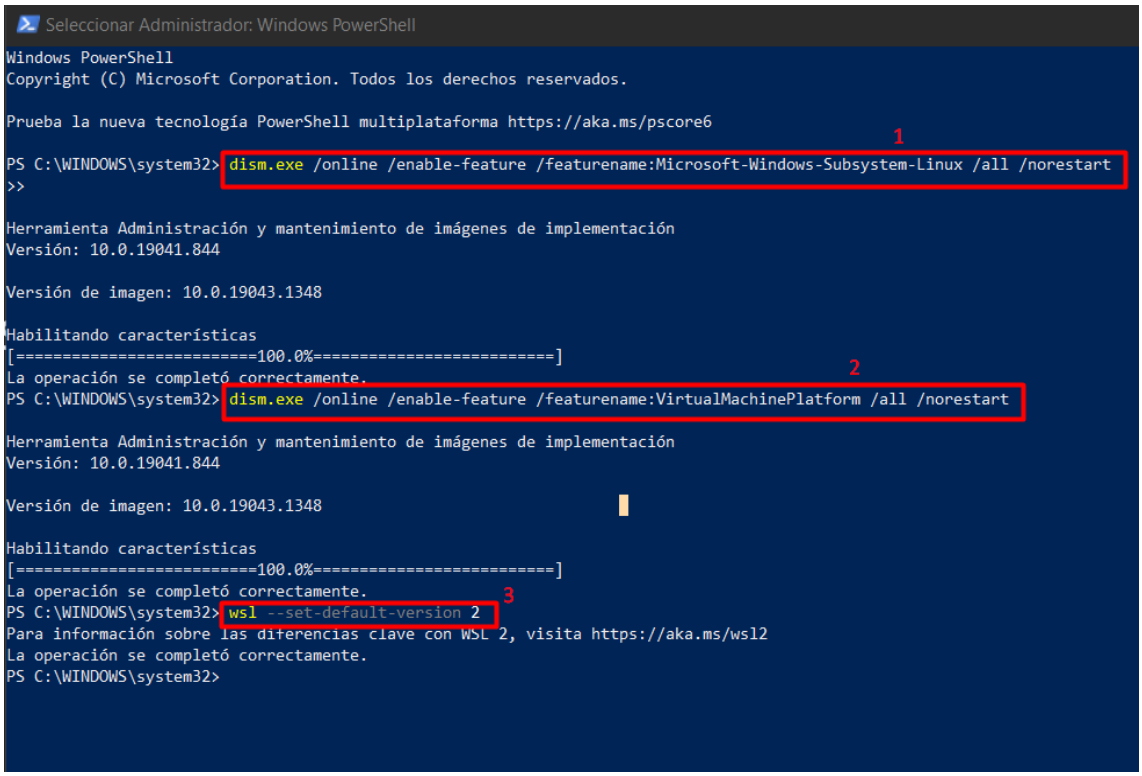
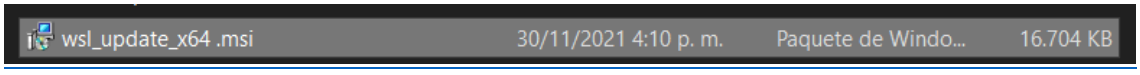


Ilustración 59 Instrucciones de instalación PowerShell.

- Habilitar subsystem for linux WSL2
- Activa las características de virtualización.

Una vez ejecutado el comando 1 y 2 enumerados en la ilustración (59) procedemos a realizar la actualización del kernel de linux dentro de nuestro sistema window que este caso es window 10 con el siguiente programa el cual se podrá descargar en el siguiente link

https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi



- Colocar por defecto subsystem for linux WSL 2

Una vez realizado los 3 primeros pasos procedemos a instalar una versión de linux que la podemos obtener de la tienda de aplicación de window en este caso se instalo Ubuntu 20.04 LTS comos se puede evidenciar en la ilustración (60)

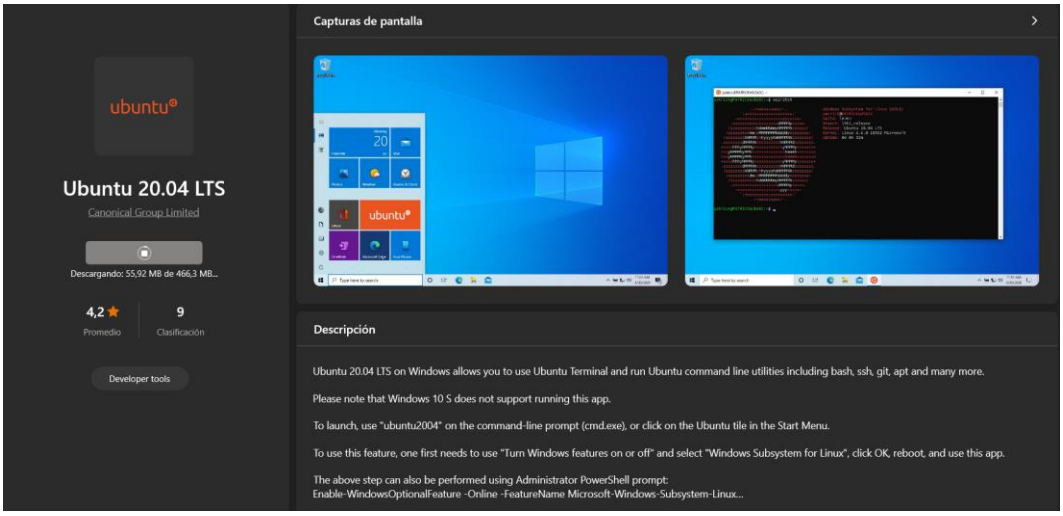


Ilustración 60 Instalación Ubuntu 20.04 LTS.

Ya cuando se tenga instalado el ubuntu ingresamos y se le coloca un usuario y contraseña para el acceso, este por defecto cuando lo inicias te pide colocar el usuario y contraseña después, ejecutamos el comando **Sudo apt update && sudo apt upgrade** para realizar las actualizaciones de los paquetes esto se hace básicamente para tener todo al día y nuestro sistema este seguro.

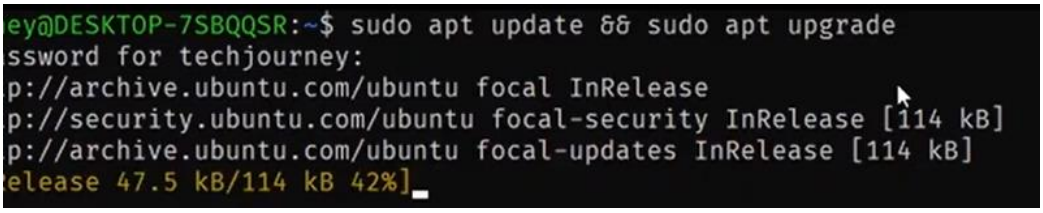


Ilustración 61 Actualización de paquetes Ubuntu 20.04 LTS.

Una vez se realiza la actualización de los paquetes de ubuntu se procede a la instalación del docker desktop ya que es una forma más sencilla de ejecutar, compilar, depurar y probar las aplicaciones Dockerized.

Cuando se hizo la instalación del docker desktop ejecutamos la siguiente línea de comando **wsl --set-default ubuntu-20.04** para así colocar por defecto Ubuntu como la distribución por defecto en docker desktop y correr los contenedores en ese sistema.

Ya cuando iniciamos el programa docker desktop podemos apreciar en la ilustración (62) que está listo para poder empezar a construir nuestros contenedores

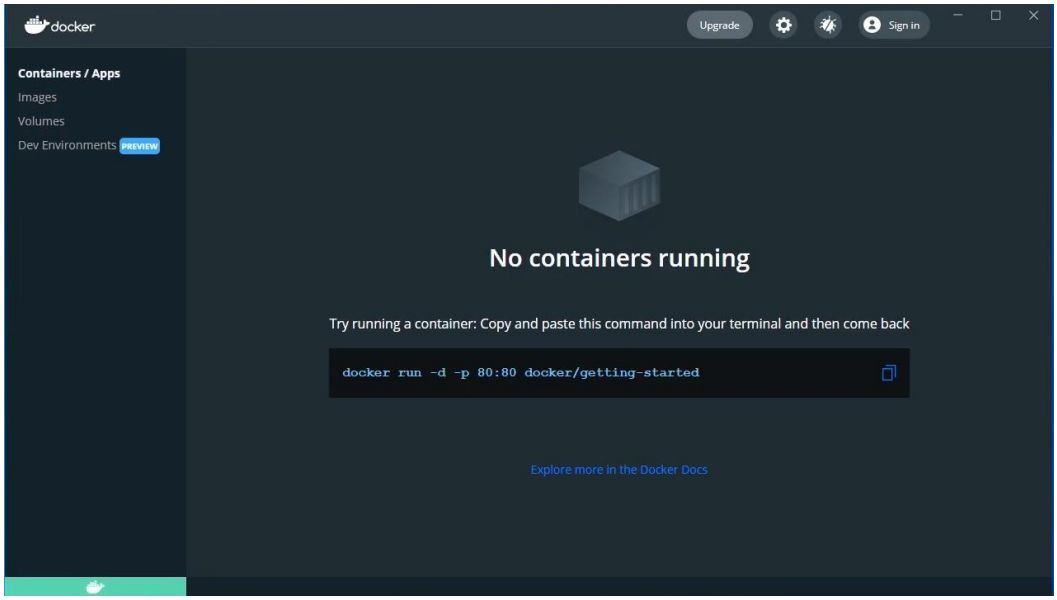


Ilustración 62 Docker Desktop.

El siguiente paso será crear un Dockerfile dentro de una carpeta que contenga todo el proyecto en el cual se construirá imágenes Docker. Un Dockerfile describe los elementos de software que componen una imagen.

Sin embargo, un Dockerfile puede describir qué entorno usar o qué comandos puede ejecutar en el contenedor. De la misma forma desplegamos imágenes de Dockerfile las cuales podrán contener bases de datos, un servidor web etc.

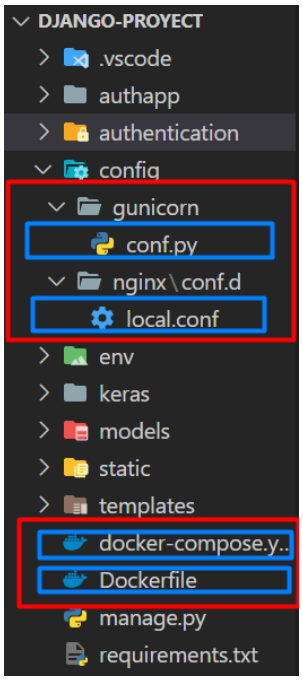


Ilustración 63 Archivos para poder desplegar con Docker.

En la ilustración (63) se puede identificar que los archivos creados son aquellos que se encuentran trazados con la línea azul a continuación se explicará brevemente lo que significa cada uno de ellos.

Se crearon las carpetas llamadas Nginx y Gunicorn donde Nginx contiene un archivo llamado **local.conf** el cual este ofrece el contenido estático del sitio web de forma rápida y fácil y la carpeta llamada Gunicorn contiene un archivo el cual es **conf.py** este nos permite servir nuestra aplicación Django con múltiples trabajadoras.

También se crearon los archivos llamados **docker-compose.yaml** y **Dockerfile** el archivo docker-compose.yaml es donde definimos los servicios que serán utilizados validando el volumen de imagen y configuraciones de cada servicio.

El archivo Dockerfilees es donde definimos un conjunto de comandos o instrucciones. Estos comandos / instrucciones se ejecutan sucesivamente para realizar acciones en la imagen base para crear una nueva imagen de la ventana acoplable.

Una vez configurado y creado todos los archivos anteriormente mencionados se procede a la creación del contenedor con todos los parámetros y configuraciones realizadas que en este caso se realiza la creación de las imágenes de cada servicio como podemos identificar en la ilustración (64) que se creó la imagen del proyecto que sería la aplicación, el gestor de base de datos postgres y el Nginx que es el que se encarga del contenido estático del sitio web.

Para poder llegar al resultado de la creación de las imágenes que se muestran en la ilustración (64) primero debemos ejecutar los siguientes comandos en el CMD o powerShell que este ubicado en la raiz del proyecto **docker-compose up –build** una vez se hallan Dockerizado los servicios ejecutamos el siguiente comando **docker-compose run django_app python django-proyect/manage.py migrate** El cual se encarga de realzar las migraciones de las tablas para que se puedan instanciar y reflejar en la base de datos.

Por último ejecutamos el siguiente comando **docker-compose run django_app python django-proyect/manage.py collectstatic** este se encargará de realizar la configuración donde mostrarnos los archivos estáticos

Al ejecutar los comandos se iniciará la instalación de todas las configuraciones realizadas anteriormente y se obtendrá el siguiente resultado que es la aplicación Dockerized.

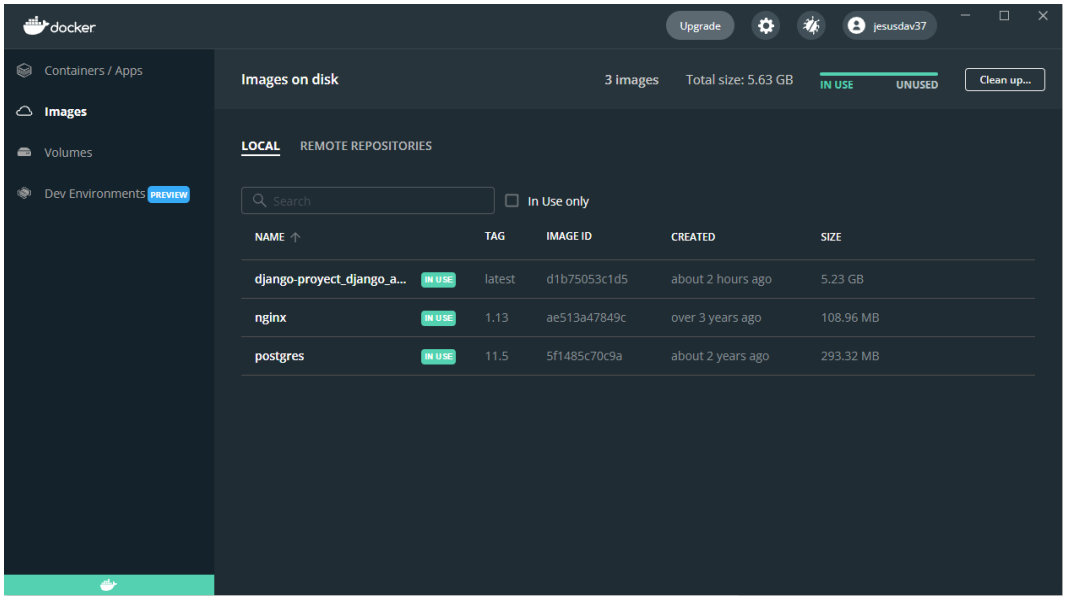


Ilustración 64 Contenedor Docker.

5. RESULTADOS Y DISCUSIONES

El ciclo de vida del modelo predicción para clasificar el riesgo cardiovascular en los pacientes del hospital municipal de Arjona, pasó por todas las etapas planteadas inicialmente. Con los datos proporcionados se realizó una limpieza rigurosa que incluyó; conocimiento de los datos, rellenados de datos faltantes, balanceo de clases, análisis estadístico y normalización general de estos.

Posteriormente se aplicó un modelo de clasificación con el algoritmo de árboles de decisión, librería S. Learn, el cual dio un valor de precisión superior al 95%. Para el desarrollo del modelo de predicción final se utilizó la implementación de redes neuronales con la ayuda de Keras – Tensor Flow.

Comparando el resultado obtenido luego de la construcción del modelo de predicción para clasificación de riesgo cardiovascular, tenemos que finalmente proporcionó un grado de satisfacción correcto frente a los resultados que obtuvieron los investigadores e instituciones reseñadas en el estado del arte del presente proyecto.

En comparación con el proyecto anterior elaborado por Gisella Rojas y Cristian Ruiz en el año 2020, el cual tenían un modelo predictivo realizado al cual se le hizo el seguimiento para evaluar que efectivamente el modelo estaba funcionando, por tal motivo no generó los resultados esperados ya que la información suministrada en el modelo obtenían datos erróneos e inventados y no era la adecuada para obtener los resultados esperados por el modelo.

Teniendo en cuenta lo anterior se volvió a realizar nuevamente el modelo con datos reales suministrados por el hospital local de arjona y se le dio un enfoque diferente al momento de realizar la predicción, inicialmente el modelo elaborado por lo mencionados anteriormente realizaba la predicción para clasificación de riesgo cardiovascular que básicamente mostraba el riesgo que tenía ya sea Alto, moderado y bajo. Las cuales mencionan en el proyecto anterior que arrojaron resultados que superan el 80% en la clasificación de los riesgos. Este al momento de realizar las pruebas no se logra poner en funcionamiento el modelo para así validar que efectivamente los resultados de este superan el porcentaje mencionado.

En el proceso de la validación del modelo se utilizó la función *predict* la cual recibe un conjunto de variables independientes y así poder identificar si efectivamente

realizaba o no la predicción, por este motivo se recurrió a volver a implementar la limpieza y construcción del modelo ya al momento realizar la predicción no estaba en funcionamiento.

En el proceso de la optimización y elaboración del nuevo modelo se utilizó muchas de las técnicas y funciones en el proceso de limpieza de los datos del modelo elaborado por Gisella Rojas y Cristian Ruiz en el año 2020 el proceso de la nueva versión del modelo fue construida de la siguiente manera, en la parte de la limpieza de los datos de explica en el (capítulo 2) y el desarrollo y optimización de la red neuronal se explica en el (capítulo 3)

El proyecto además de ser optimizado y elaborado nuevamente se le dio un enfoque de predictor al clasificador de enfermedades diferente ya que ahora no realiza la predicción para clasificación de riesgo cardiovascular si no que se realiza la clasificación de enfermedades cardiovasculares que básicamente se representa por unos códigos relacionados a estas dichas enfermedades uno de los resultados es como los que se muestran en la siguiente tabla donde representa el código de enfermedades y el diagnóstico principal que esta significa.

Código de Enfermedad	Diagnóstico Principal
I10X	HIPERTENSIÓN ESENCIAL
E119	DIABETES MELLITUS NO INSULINODEPENDIENTE SIN MENCION DE COMPLICACION
E109	DIABETES MELLITUS INSULINODEPENDIENTE SIN MENCION DE COMPLICACION

También se realizó el proceso de poder guardar el modelo para ser utilizado en una aplicación web que además también fue construida con varias tecnologías muy utilizadas en desarrollo de software y agregando toda esta información en una base de datos, como también el despliegue de esta aplicación en un contenedor Docker el cual proceso de análisis y elaboración de esta se explica en el (capítulo 4). Por último se hace la comparación de resultados del proyecto elaborado por Gisella Rojas y Cristian Ruiz en el año 2020 realizado por la escuela de ingeniería de la universidad del sinú, donde el modelo se basa en la clasificación de riesgo cardiovascular a comparación con este proyecto que lleva como título modelo de clasificación de enfermedades cardiovasculares.

Como se puede identificar la siguiente ilustración(65) vemos 5 columnas llamadas Loss, Accuracy, val_loss, val_accuracy, epoch de las cuales vamos a tomar como las más importante las siguientes: Loss (Pérdida), Accuracy (Precision), Epoch (Épocas) donde nos especifica la Perdida y Precision que esta recurriendo durante el entrenamiento en las épocas que se estipulados para el entrenamiento de la red neuronal la cual arroja muy buenos resultados en la precisión del modelo con un porcentaje del 96%.

	loss	accuracy	val_loss	val_accuracy	epoch
45	0.105387	0.963193	0.045178	0.987814	45
46	0.106756	0.961918	0.019510	0.992832	46
47	0.110067	0.963113	0.036742	0.994982	47
48	0.113713	0.960644	0.020688	1.000000	48
49	0.115378	0.960245	0.088093	0.966308	49

Ilustración 65 Precisión Red Neuronal.

Finalmente se realiza una evaluación final con la función llamada *evaluate* Para identificar la precisión final pero de las variables de prueba que en este caso son x_test y y_test el cual arroja muy buenos resultados.

```
[35]    loaded_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
        results = loaded_model.evaluate(x_test, y_test)
        print('prueba')
        print('Final test set loss: {:4f}'.format(results[0]))
        print('Final test set accuracy: {:4f}'.format(results[1]))

    110/110 [==============================] - 1s 3ms/step - loss: 0.2064 - accuracy: 0.9488
    prueba
    Final test set loss: 0.206383
    Final test set accuracy: 0.948784
```

Ilustración 66 Evaluación del modelo.

Conclusión

En la última década, las técnicas de aprendizaje profundo (Deep Learning) han desempeñado un papel importante en el dominio de la atención médica para el descubrimiento de conocimientos y la clasificación de enfermedades cardíacas. Asimismo, las técnicas de aprendizaje profundo se pueden utilizar para predecir óptimamente la hipertensión en bases de datos clínicas.

Se utilizó un proceso estructurado, el cual permitió antes de aplicar la técnica o modelado preparar los datos, lo cual consistió en la adquisición de datos, limpieza de datos a través de la identificación y corrección de erros en los datos, y selección de características, es decir, la identificación de variables de entrada que son más relevantes para el problema. Una vez seleccionada la técnica de modelado o algoritmo, su capacidad para realizar un buen ajuste dependerá de las características de los datos (número de variables, linealidad, normalidad, valores perdidos y variables continuas o categóricas).

Posteriormente, para el modelo se utilizó la técnica de clasificación supervisada de arquitectura Secuencial utilizando Tersor Flow de Google, la cual permitió crear las capas del modelo, entrenarlo y probarlo dándole los datos de un paciente, y el modelo responderá indicando en qué categoría lo clasifica y finalmente determinar si la clase o categoría de hipertensión pertenece una nueva muestra, teniendo en cuenta que la información se puede extraer del conjunto de datos de entrenamiento.

También se pudo aplicar el diseño de lo que fue un aplicativo web para la consulta de los profesionales de la salud el cual se encargó de poder integrarse al modelo construido para realizar clasificación de enfermedades cardiovasculares y poder ser desplegada en un servidor en la nube con contenedores docker con esta arquitectura de despliegue nueva y tendencia tecnológica de virtualizar contenedores brinda soluciones óptimas y crea infraestructuras virtualizadas para optimización.

Asimismo, se proponen las siguientes recomendaciones para mejorar la calidad de este trabajo de investigación:

- Que se ponga en producción la aplicación en la plataforma seleccionada y se realice capacitaciones al personal interesado en utilizarla, así como la digitación de información real que permita tomar las decisiones con respecto al personal médico encargado.

- Implementar la optimización de la aplicación interpretando los datos con gráficos estadísticos para la toma de mejores decisiones.

- Escalamiento de la predicción de clasificaciones llevarla a un nivel de tomar decisiones con respecto a poblaciones más grandes y no específicamente una persona.

Bibliografía

[1] [medicinatv. (s.f.). Hipertensión arterial. *medicinatv*, https://www.medicinatv.com/enfermedades/hipertension-arterial/.

[2] O. Suárez Landazábal, C. Villarreal Sotomayor, A. Parody Muñoz, A. Rodríguez Delgado, and R. Rebolledo Cobos, “Prevalencia de hipertensión arterial y de sus factores de riesgo en estudiantes universitarios de Barranquilla, Colombia,” *Rev. la Fac. Ciencias la Salud Univ. del Cauca*, vol. 21, no. 2, pp. 16–23, 2019, doi: 10.47373/rfcs.2019.v21.1372.

[3] «¿Qué es la hipertensión arterial? | CuídatePlus», *CuidatePlus*, mar. 26, 2009. https://cuidateplus.marca.com/enfermedades/enfermedades-vasculares-y-del-corazon/hipertension-arterial.html (accedido may 30, 2021).

[4] «Alarcón - SISTEMA DE JUEGOS DEPORTIVOS.pdf». Accedido: may 30, 2021. [En línea]. Disponible en:http://repositorio.utn.edu.ec/bitstream/123456789/5775/1/04%20ISC%20429%20TRABAJO%20DE%20GRADO.pdf

[5] «MARIA DEL CARMEN AVILA LILLO.pdf». Accedido: may 30, 2021. [En línea]. Disponible en: http://147.96.70.122/Web/TFG/TFG/Memoria/MARIA%20DEL%20CARMEN%20AVILA%20LILLO.pdf

[6] «Predicción del riesgo cardiovascular e hipertensión arterial según Framingham en pacientes de atención primaria en salud. Estudio FRICC | Revista Colombiana de Medicina Física y Rehabilitación». https://revistacmfr.org/index.php/rcmfr/article/view/159 (accedido may 30, 2021).

[7] Y. Morillo, «Deep learning o aprendizaje profundo | Qué es y cómo funciona», *Futuro Electrico*, ago. 27, 2020. https://futuroelectrico.com/deep-learning-aprendizaje-profundo/ (accedido may 30, 2021).

[8] «Inteligencia Artificial vs Aprendizaje Automático vs Deep Learning vs Ciencia de Datos», *EADIC - Cursos y Master para Ingenieros y Arquitectos*, nov. 24, 2019. https://www.eadic.com/inteligencia-artificial-vs-aprendizaje-automatico-vs-deep-learning-vs-ciencia-de-datos/ (accedido may 30, 2021).

[9] E. de I. en C. y S. ECYS, *Deep Learning y su increíble impacto en la realidad conocida | Decimotercera Edición - ECYS*. Accedido: may 30, 2021. [En línea]. Disponible en: https://revistaecys.github.io/13Edicion/03_ggiron.html

[10] «Aprendizaje Supervisado: Introducción a la Clasificación y Principales Algoritmos | by Victor Roman | Ciencia y Datos | Medium». https://medium.com/datos-y-ciencia/aprendizaje-supervisado-introducci%C3%B3n-a-la-clasificaci%C3%B3n-y-principales-algoritmos-dadee99c9407 (accedido may 31, 2021).

[11] P. J. M. V. Mendoza, «Django, Gunicorn, Nginx y Supervisor», *Rukbottoland*. https://rukbottoland.com/blog/django-gunicorn-nginx-supervisor/ (accedido may 31, 2021).

[12] G. Ruiz, Cristian; Rojas, “Modelo Predictivo de Hipertensión Para el Diagnostico de Pacientes con Factores de Riesgo Cardiovascular en el Departamento de Bolívar, Mediante Técnicas de Deep Learning,” p. 6, 2021, [Online]. Available: http://repositorio.unisinucartagena.edu.co:8080/xmlui/handle/123456789/26.

[13] A. R. Nair and B. R. Manju, “A RFE- Optimised Method for Predicting the Malignancy of the Breast Cancer,” *2021 Int. Conf. Nascent Technol. Eng. ICNET 2021 - Proc.*, no. Icnte, 2021, doi: 10.1109/ICNTE51185.2021.9487761.

[14] sitiobigdata. (24 de diciembre de 2019). Clasificación de Aprendizaje automático supervisado. págs. https://sitiobigdata.com/2019/12/24/clasificacion-de-aprendizaje-automatico-supervisado/.

[15] Yadav, H. (14 de Mayo de 2020). Logistic Regression Implementation in Python. págs. https://medium.com/machine-learning-with-python/logistic-regression-implementation-in-python-74321fafa95c.

[16] Salud, O. P. (s.f.). Hipertensión. pág. https://www.paho.org/es/temas/hipertension. [40] https://www.cienciadedatos.net/documentos/41_machine_learning_con_r_y_caret