UNIVERSIDAD DEL BIO BIO

FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN Y TECNOLOGÍAS DE LA INFORMACIÓN

"ANÁLISIS DE DATOS EN PACIENTES POST ACV-ISQUÉMICO, USANDO TÉCNICAS CLÁSICAS DE MACHINE LEARNING"

AUTOR

ABRAHAM MARIANJEL SEPÚLVEDA

AÑO ACADÉMICO 2022

9 de marzo de 2023

Profesora Guía	**Dr. Carola Andrea Figueroa Flores** Doctor en Informática, mención Cum Loude Dpto. Ciencias de la Computación y Tecnologías de la Información Universidad del Bio Bio
Profesores Co-Guía	**Dr. Carlos Alonso Escudero Orozco** Doctor en Ciencias Médicas Dpto. Ciencias Básicas Universidad del Bio Bio **Dr. Andrés Ignacio Rodríguez Morales** Doctor en Ciencias Biológicas Dpto. Ciencias Básicas Universidad del Bio Bio
Profesora Informante	**Mg. María Antonieta Soto Chico** Magíster en Ciencias de la Ingeniería, mención Industrial Dpto. Ciencias de la Computación y Tecnologías de la Información Universidad del Bio Bio
Comité Proyecto de Título	**Mg. Marlene Elena Muñoz Sepúlveda** Magíster en Informática Educativa Dpto. Ciencias de la Computación y Tecnologías de la Información Universidad del Bio Bio

UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

Este documento fue elaborado por el autor usando LaTeX.
El trabajo en este documento se llevó a cabo para obtener el título de Ingeniero Civil en Informática de la Universidad del Bío Bío, sede Chillán.

"Nunca olvides que basta una persona o una idea para cambiar tu vida para siempre, ya sea para bien o para mal"

J Brown

Agradecimientos

A Dios en primera instancia, que ha sido bueno conmigo durante toda mi vida, agradezco lo que hizo, lo que está haciendo y lo que hará en mí. Gracias por permitirme sonreír frente a todas las personas y adquirir virtudes de todas ellas, haciendo que crezca como ser humano y me expanda de distintas formas.

A mi profesora guía Carola Figueroa Flores, Doctora en Informática con Mención Cum Loude. Sus concejos, ideas fueron siempre útiles cuando mis soluciones a mis problemas se volvían confusas. Gracias por sus palabras de aliento, historias de experiencias vividas y sus orientaciones.

A mis profesores, quienes con sus conocimientos formaron al profesional que soy actualmente. Gracias por su paciencia, dedicación y tolerancia, haciendo una mención especial a mi Jefa de Carrera Marlene Muñoz Sepúlveda que estuvo siempre brindándome apoyo y consejo.

A mi madre, quien siempre me ha dado un amor inconmensurable, todo lo que podía dar y más y esperanza en todas las cosas que me esperan a futuro. Gozoso de tenerte como madre y que me acompañe en todos mis momentos. Gracias por ser quien eres y por creer en mí.

A mis amigos y compañeros, que dentro de esta maravillosa aventura dimos nuestro mejor esfuerzo para cumplir con todas las exigencias de la carrera y cooperando mutuamente para lograr el mejor resultado para todos. Especialmente, Diego Garrido y Daniel Gonzáles por su apoyo, constancia y amistad, que con ustedes la carrera fue menos pesada, más divertida y con un largo historial de aventuras que estuvieron en ella. Gracias por estar allí siempre.

Resumen

Los Accidentes Cerebro Vascular, también llamados ACV o ictus, son de las principales causas de muerte en hombres y mujeres en Chile y el mundo. Los ACV podemos encontrarlos de dos tipos: ACV Isquémico por obstrucción de un vaso sanguíneo y ACV Hemorrágico por rotura de un vaso sanguíneo.

El proyecto utiliza una de las técnicas más conocidas de la Inteligencia Artificial (IA), como lo es el aprendizaje automático (Machine Learning) para realizar clasificaciones sobre pacientes que han sufrido un ACV y poder tomar decisiones prematuramente gracias a los modelos predictivos. La recopilación de datos de los pacientes fue una colaboración entre la investigadora e Informática Dra. Carola Figueroa con el médico e investigador Dr. Carlos Escudero que obtuvieron los datos del Hospital Herminda Martin de Chillán.

Se plantea la posibilidad de clasificar al paciente para predecir si tendrá un buen pronóstico cuando este sea dado de alta, por medio de la escala internacional de NIHSS (Escala de Accidentes Cerebrovasculares de los Institutos Nacionales de Salud), escala que mide el daño neurológico en los pacientes.

Como veremos a continuación, existen muchos tipos de algoritmos de Machine Learning, pero hay algunos que se repiten en el área de salud. Escogeremos 4 algoritmos y los desarrollaremos lo más simple posible, para que se pueden comparar con las mismas métricas y ningún algoritmo sufra una ventaja significativa sobre otro.

Palabras Claves: Accidente Cerebro Vascular, Machine Learning, Predicción

Abstract

Cerebro Vascular Accident, also called ACV or stroke, are one of the main causes of death in men and women in Chile and the world. CVA can be found in two types: Ischemic CVA due to obstruction of a blood vessel and Hemorrhagic CVA due to rupture of a blood vessel.

The project uses one of the best-known Artificial Intelligence (AI) techniques, such as Machine Learning, to classify patients who have suffered a stroke and to make premature decisions thanks to predictive models. The collection of patient data was a collaboration between the researcher and IT Dr. Carola Figueroa with the physician and researcher Dr. Carlos Escudero who obtained the data from the Herminda Martin de Chillán Hospital.

The possibility of classifying the patient to predict that he will have a good prognosis when he is discharged is raised, using the international scale of NIHSS (Cerebrovascular Accident Scale of the National Institutes of Health), a scale that measures neurological damage in patients. patients.

As we will see below, there are many types of Machine Learning algorithms, but there are some that are repeated in the health area. We will choose 4 algorithms and we will develop them as simple as possible, so that they can be compared with the same metrics and no algorithm suffers a significant advantage over another.

Keywords: Stroke, Machine Learning, Prediction

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCIÓN

En nuestros tiempos los computadores o similares son indispensables en las tareas del día a día, ayudando a mejorar la calidad de los trabajos, provocando un gran impacto en la sociedad hoy más que nunca. Las múltiples ramas de las ciencias de la computación han ayudado al desarrollo progresista de diversas áreas de trabajo e investigación, como lo es medicina, economía, estadística, entre otras. Al tener un gran número de datos disponibles, esto permite combinar y utilizar esta información de diferentes maneras, aumentando así las posibilidades de combinación entre las áreas. Esto a su vez puede mejorar la precisión y eficacia de los modelos en IA en tareas específicas.

Durante las últimas décadas se ha incrementado enormemente nuestra capacidad para recoger información en cualquier actividad, por ejemplo, en los negocios, recopilando información sobre procesos de producción, ventas, servicios, campañas de marketing, entre varios. La gran cantidad de información ha crecido el interés de las personas por utilizarlos, para extraer la información que represente una ventaja competitiva, lo que es particularmente relevante para las organizaciones.

El campo interdisciplinario de Ciencias de la Computación involucra métodos científicos, procesos y sistemas para extraer conocimiento o mejor entendimiento de datos en sus diferentes formas, ya sean información útil o no útil, idealmente estructurado. Dentro de Ciencias de la Computación, se encuentra la Inteligencia Artificial, que posee una variedad de metodologías para resolver problemas de

forma eficaz y eficiente.

La IA ha experimentado un rápido crecimiento en los últimos años debido a su capacidad para resolver problemas con una alta precisión y su aplicabilidad en una amplia gama de disciplinas y áreas de trabajo. Muchos sectores, incluyendo el comercio, la salud y la tecnología social, han adoptado herramientas basadas en IA para mejorar su eficiencia y productividad. Esto ha llevado a un aumento en el mercado de productos y servicios relacionados con la IA, y se espera que esta tendencia continúe en el futuro.

Una de las principales ramas de la Inteligencia Artificial es el Machine Learning (ML), destacándose en áreas como análisis de datos, minería de datos, reconocimiento de patrones, entre varias. ML, constituye una de las ramas más atractivas para los sectores que trabajan con grandes cantidades de datos.

Este trabajo abordara una problemática referente al ACV Isquémico y la IA, siendo el ACV una de las principales causas de muerte a nivel mundial y la segunda causa de muerte en nuestra región.

1.1. Descripción del Problema

El ACV, hoy en día, es una de las principales causas de muerte, asimismo las personas que logran sobrevivir quedan con secuelas o discapacidades en la mayoría de los casos, presentándose con más frecuencia, sorpresivamente, en adultos jóvenes, pero también está aumentando en los adultos mayores [2].

En Chile, solo el año 2021 hubo 29.542 egresos hospitalarios por ACV, siendo la segunda causa de mortalidad a nivel país, después de las enfermedades isquémicas del corazón y no considerando la pandemia por el SARS-CoV-2. El registro de defunciones por ACV llegó a los 7.501 casos ese mismo año, lo que equivale a una muerte cada 1 hora y 12 minutos [3].

La salud es vital disminuir las posibles enfermedades en el menor plazo posible. El ACV, por su parte posee una alta tasa de morbimortalidad, es decir, el ACV en sí o por las secuelas que genera, provoca muertes muy rápidas en poco tiempo [4]. El impacto negativo de las secuelas en los pacientes tiene un alto costo sanitario, tanto en lo intrahospitalario como extra hospitalario. En lo físico, la pérdida de la movilidad de una parte del cuerpo (discapacidad) y, en lo social, quizás la pérdida de un sentido (hablar o escuchar) o disminución de la calidad de vida de la persona [5].

En Ñuble, se cuenta con una red hospitalaria que lleva un registro de las enfermedades y pacientes que ingresan a los hospitales, de este modo se determinó que una de las principales causas de muerte en la región es consecuencia de los ACV. En términos de probabilidades, el ACV representa una persona muerta al día, esto la hace la primera causa de muerte de la zona y una de las más importantes del país, ya que el escenario se repite para las demás zonas. Esta enfermedad se trata de una urgencia, donde acceder a un tratamiento oportunamente, puede establecer la diferencia en el pronóstico de salud. Las acciones deben estar destinadas a preservar la integridad del tejido cerebral [6].

Para obtener resultados óptimos, el médico debe realizar una serie de exámenes, los cuales ayudarían a obtener un pronóstico más certero y lograr manejar adecuadamente el estado de salud del paciente. La IA y el ML aporta a la predicción del diagnóstico a través de la obtención de resultados en un tiempo acotado, además de lograr una mayor acertividad de este. Es por esta razón que el rol de la IA jugará un papel importante en el futuro de la salud, esto se puede observar en los países más desarrollados, los cuales presentan estudios exitosos de exploración y ejecución en esta área. Al mismo tiempo, los errores de diagnósticos en los países en vías de desarrollo se ven incrementados, lo que nos alerta de emplear lo antes posible herramientas ligadas a la IA, esperando lograr con estas una mayor acertibidad en el diagnóstico de enfermedades y perfeccionar la medicina preventiva [7].

1.2. Descripción del proyecto

La presente sección tiene como propósito dar a conocer la hipótesis y los objetivos del proyecto.

1.2.1. Hipótesis

Es posible determinar la técnica de ML más precisa, con el fin de clasificar según el tipo de secuela a los pacientes que han sufrido ACV Isquémico del Hospital Herminda Martín

1.2.2. Objetivo General

Detectar la mejor técnica de ML para identificar los factores de mal pronóstico en adultos con diagnóstico de ACV Isquémico a través del análisis comparativo de su precisión para pacientes del Hospital Herminda Martín.

1.2.3. Objetivos Específicos

1. Estudiar las distintas técnicas de ML que existen en la literatura.
2. Determinar qué técnicas de ML se van a comparar.
3. Elegir un mecanismo adecuado, para aplicar un modelo de ML al problema.
4. Implementar las técnicas seleccionadas de ML, para identificar la similitud entre las variables clínicas en los diferentes grupos y las variables que tienen mayor importancia en el pronóstico de los usuarios con diagnóstico ACV isquémico.
5. Realizar una comparación de las distintas técnicas de ML, en base a su precisión.

Capítulo 2

MARCO TEÓRICO

El presente capitulo tiene como propósito presentar al lector algunas investigaciones y conceptos relacionados con IA, específicamente ML, y en salud particularmente de ACV Isquémico, conceptos claves para esta investigación.

La organización del capítulo se encuentra de la siguiente manera. Inicialmente, veremos ML y sus técnicas clásicas, pues consideramos estas podrían ser la respuesta a nuestra problemática junto a Deep Learning (DL), posteriormente, nos enfocaremos en algunos términos de salud que nos ayudarán a entender de mejor forma el problema.

2.1. Inteligencia Artificial

Las matemáticas nos han ayudado a interpretar y entender nuestro medio ambiente y ser capaz de predecir algunos sucesos en áreas como la cosmología o naturaleza o problemas incomprendidos por los humanos [8]. Hoy en día contamos con el área de las ciencias de la computación que fue creada hace solo un par de décadas atrás, siendo una de las ciencias más nuevas y que tiene gran valoración en la actualidad. En las ciencias de la computación podemos encontrar varias áreas aplicadas, de las cuales la IA destaca por su relación con la matemática, biología, lingüística, entre otras. [8].

La IA llegó para resolver tareas que el ser humano realiza como tareas cotidianas básicas y complejas. Los algoritmos de IA se basan en el aprendizaje automático, siendo que cada vez las máquinas pueden aprender por ellas mismas algunas cosas, es así que los humanos dejaremos de perder nuestro tiempo programando reglas para lidiar con muchas combinaciones de datos y situaciones que se presentan a diario. Pues es así, que el aprendizaje de los propios algoritmos mejora el rendimiento de estos sin la necesidad de programación adicional por parte humana. Esto es logrado a través del uso de técnicas de aprendizaje automático, en las que el algoritmo es entrenado con un conjunto de datos, luego, se ajusta a medida que recibe nuevos datos y realiza más tareas. [1].

Dentro de la IA, convive unsubconjunto llamada ML y su vez, dentro de ML vive un subconjunto llamado DL. Estos subconjunto interactúan con lo que es el Data Science y el Big Data, haciendo que se relacionan entre sí para procesar y analizar grandes cantidades de datos con el objetivo de extraer información valiosa y conocimientos útiles [9].

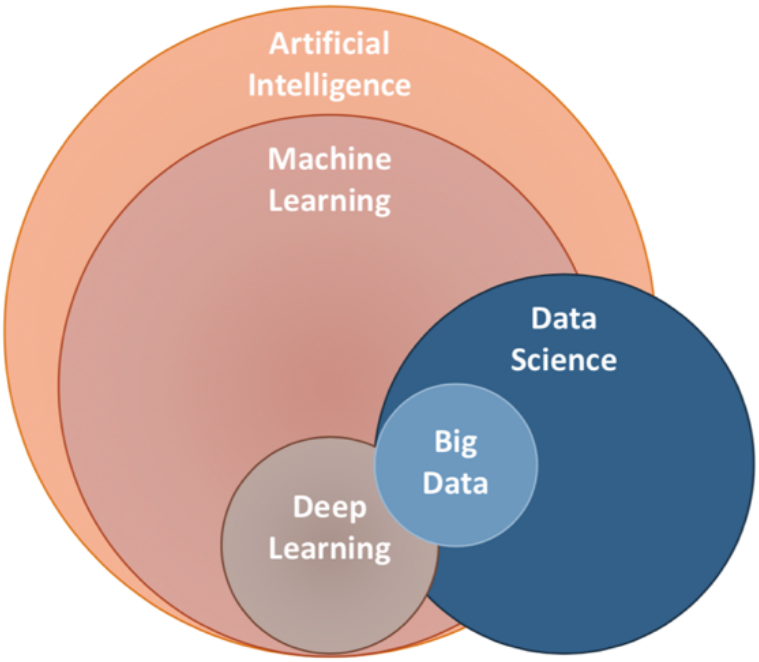

Figura 2.1: Diagrama de Venn con las subáreas de la IA

2.2. Machine Learning

Los problemas por computadora se solucionan con un algoritmo que le indica a la máquina los pasos a seguir. En un ejemplo clásico podemos usar un algoritmo de búsqueda, donde el dato a buscar sería la entrada y su salida sería si se encontró o no y en cuánto tiempo. Es bien conocido que para resolver el problema de búsqueda existen variados algoritmos, algunos más eficientes que otros, aplicables a un contexto u otro de mejor forma, solucionando el mismo problema.

Asimismo, la publicidad en el celular es un ejemplo de cómo los algoritmos de ML están aplicándose en la vida cotidiana. Estos algoritmos utilizan técnicas de análisis de datos y aprendizaje automático para comprender los patrones de comportamiento y preferencias de los usuarios a partir de la información de navegación y otros datos recopilados en sus dispositivos móviles. Además de la publicidad en el celular, los algoritmos de ML también están siendo utilizados en una variedad de aplicaciones, como la recomendación de productos y servicios, la detección de fraudes, la mejora de la eficiencia en la toma de decisiones y la optimización de la experiencia del usuario.

En el año 1959, Arthur Samuel [10], definió el concepto de ML como:

> "*Un campo de estudio que le entrega a los computadores la habilidad de aprender sin haber sido explícitamente programado para eso*"

Tom M. Mitchell en 1977 [11], define el ML, en uno de sus libros, como:

> *"El estudio de algoritmos de computación que mejoran automáticamente su rendimiento gracias a la experiencia. Se dice que un programa informático aprende sobre un conjunto de tareas, gracias a la experiencia y usando una medida de rendimiento, si su desempeño en estas tareas mejora con la experiencia"*

Es decir, estos algoritmos aprenden y mejoran solos gracias a las experiencias pasadas, a diferencia de modelos en los que un experto puede asignar reglas y modela gracias a sus conocimientos.

ML utiliza algoritmos para analizar grandes cantidades de datos y descubrir patrones y relaciones en ellos. Una vez que el algoritmo ha aprendido de estos datos, puede aplicar lo que ha aprendido a un nuevo conjunto de datos y utilizarlo para tomar decisiones o hacer predicciones. [12].

En este sentido, el ML permite a las máquinas aprender de los datos y utilizar ese conocimiento para mejorar sus decisiones y predicciones. Esto es una gran ventaja en comparación con los enfoques tradicionales, en los que un programador humano debe escribir una serie de reglas y lógica para tomar decisiones. Con el aprendizaje automático, las máquinas pueden aprender por sí mismas y mejorar con el tiempo sin la necesidad de programación adicional.

2.2.1. Machine Learning: Tipos de Aprendizajes

ML contempla dos enfoques o tipos de aprendizaje bastante usados, el aprendizaje supervisado y el aprendizaje no supervisado.

Aprendizaje Supervisado

Este enfoque es un método de análisis de datos que necesita de algoritmos que aprendan a través de entrenamiento, en el cual el algoritmo es alimentado con datos etiquetados, atributos y la variable objetivo en cada iteración.

El aprendizaje supervizado se divide en dos tareas comunes: clasificación y regresión.

La tarea de clasificación se utiliza para predecir una categoría o clase para un nuevo conjunto de datos. Por ejemplo, un algoritmo de clasificación puede ser entrenado con datos sobre diferentes tipos de frutas y sus características para predecir si una nueva fruta es una manzana o una pera.

La tarea de regresión se utiliza para predecir un número o valor continuo. Por ejemplo, un algoritmo de regresión puede ser entrenado con datos sobre la relación entre la edad de una persona y su salario para predecir el salario de una persona en función de su edad.

En resumen, el aprendizaje supervisado es una técnica muy útil y ampliamente utilizada en el ML para hacer predicciones precisas sobre nuevos datos. A través de la tarea de clasificación y regresión, se pueden solucionar una amplia variedad de problemas y aplicaciones en una amplia gama de industrias [12].

Aprendizaje No Supervisado

El aprendizaje no supervisado tiene datos sin etiquetar que el algoritmo tiene que entender por si mismo, el propósito es descubrir patrones ocultos en ellos. Si nosotros le pedimos a nuestro programa *"predecir $\boldsymbol{Y}$ para nuestros datos $\boldsymbol{A}$"*, nosotros deberíamos *"pedirle que nos provea de la información de nuestros datos $\boldsymbol{A}$"* [12].

Las tareas más comunes dentro del aprendizaje no supervisado son el *clustering* y *dimension reduction*, los cuales no son objeto de nuestro estudio por lo que solo son mencionados en esta parte del documento.

2.2.2. Técnicas de Clasificación

Para que nuestro modelo de ML funcione, debemos categorizar los datos de entrada y salida, reconociendo atributos del elemento a clasificar y utilizando el conocimiento adquirido durante el entrenamiento del algoritmo para asignar un valor a la variable objetivo de dicho elemento.

La clasificación de datos es un proceso que consta de dos etapas, la etapa de aprendizaje donde es construido el modelo, y la etapa de clasificación donde el modelo es usado para predecir las etiquetas de clases de los datos dados [13].

En la *primera* etapa de aprendizaje o entrenamiento del algoritmo, se construye el modelo utilizando una serie de datos que sirven de base para el conocimiento del algoritmo, para lograr esto, es que se le entrena utilizando un set de entrenamiento, que no son más que tuplas de datos etiquetados tanto en sus atributos como en su variable objetivo. Una tupla X, generalmente es representada por un vector n-dimensional llamado vector de atributos $X = (x_1, x_2, ..., x_n)$.

En la *segunda* etapa, el algoritmo ya ha sido entrenado con los datos de entrenamiento y está listo para la clasificación de una variable x. El algoritmo implementado será capaz de tomar los atributos de dicha variable, analizarlos y tomar una decisión en los resultados, basándose en los datos que fueron entregados anteriormente.

En la clasificación veremos algunas de las técnicas clásicas de ML, pertenecientes al enfoque de los algoritmos de aprendizaje supervisado, definiendo brevemente cada una de ellas.

2.2.3. Decision Tree (Árbol de Decisión)

Este algoritmo que se utiliza como herramienta de apoyo gráfico o modelo de decisiones con sus posibles consecuencias, también a veces son representados los costos y su posible utilidad (CART, Classification and Regression Trees). Este método se crea particionando la entrada recursivamente en distintas ramas, siendo que la idea es crear un camino desde la raíz hasta las hojas, donde cada nodo podría ser una condición así, si se cumple la condición se sigue por el camino de decisión, o por la otra rama si no se llegara a cumplir la condición [14].

La creación del modelo se puede representar en la ecuación 2.1:

$$f(x) = E[y|x] = \sum_{m=1}^{M} w_m I(x \in R_m) = \sum_{m=1}^{M} w_m \phi(x; v_m) \quad (2.1)$$

Donde:

- R_m representa la región de m.
- w_m es respuesta media a esa región.
- v_m codifica la elección de la variable por la que dividir y el valor límite de la división.

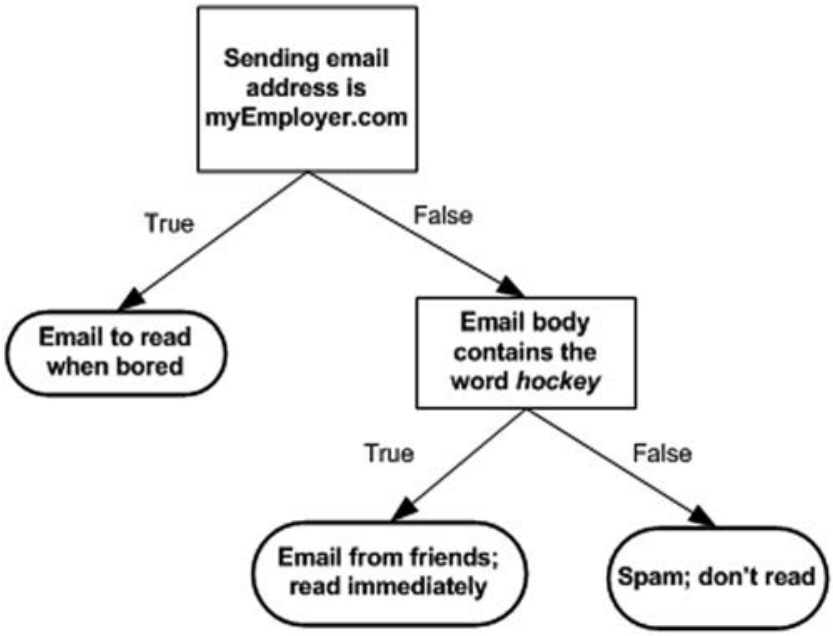

Figura 2.2: Ejemplo de un problema aplicando Árbol de Decisiones

En la Figura 2.2 se presenta un ejemplo del modelo que se puede generar. Las condiciones se dan como alternativas de caminos a seguir.

Usos para los Decision Tree

Al ser de fácil implementación, los Decision Tree son usados en diversas áreas, siendo las instituciones financieras las más comunes, ayudando a clasificar clientes,

estableciendo sus riesgos o posibilidades financieras [14].

En el área de la salud son empleados para diagnósticos de infecciones a la sangre o predicción de ataque al corazón en pacientes de alto riesgo [14].

En el seguimiento de movimiento, los árboles de decisión se utilizan para analizar los movimientos de un objeto en un video y predecir su ubicación en el siguiente cuadro. Esto se logra mediante la creación de un modelo de decisión que tome en cuenta factores como la velocidad, la dirección y el tamaño del objeto. [14].

En el reconocimiento facial, los árboles de decisión se utilizan para identificar a una persona en un video o imagen. Esto se logra mediante la creación de un modelo de decisión que tome en cuenta características como la forma de la cara, el tamaño de la nariz y la distancia entre los ojos [14].

Ventajas y Desventajas

Las ventajas de esta técnica de ML es su bajo costo computacional y simple interpretación de resultados.

La mayor desventaja es que la técnica es propensa a caer en el sobreajuste, siendo a veces manipulada por quien lo implemente [14].

2.2.4. Random Forest (Bosque Aleatorio)

El algoritmo de Random Forest es una técnica de aprendizaje supervisado que genera múltiples árboles de decisión sobre un conjunto de datos de entrenamiento: los resultados obtenidos se combinan para obtener un modelo único más robusto en comparación con los resultados de cada árbol por separado [15].

Cada árbol se obtiene mediante un proceso de dos etapas:

1-. Se genera un número considerable de árboles de decisión con el conjunto de datos. Cada árbol contiene un subconjunto aleatorio de variables m (predictores) de forma que $m < M$ (donde M = total de predictores).

2-. Cada árbol crece hasta su máxima extensión.

En la Figura 2.3 se muestran las etapas mencionadas anteriormente:

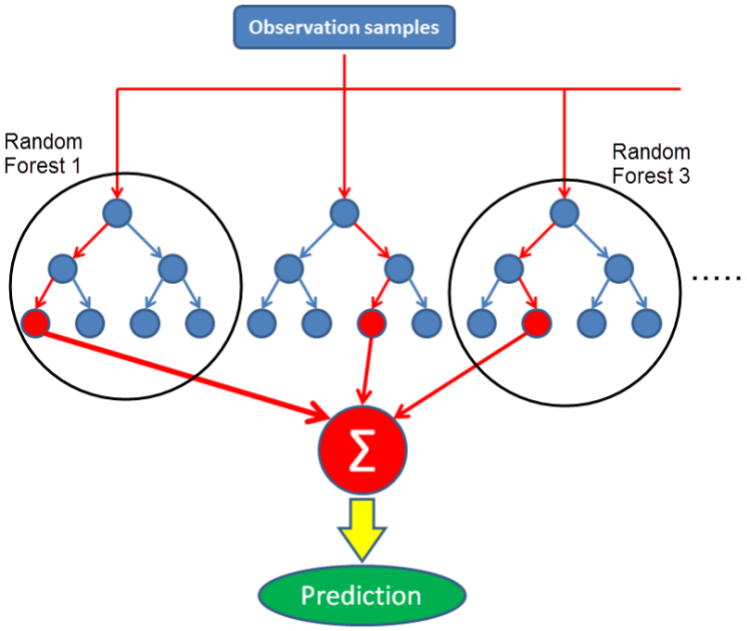

Figura 2.3: Ejemplo de la secuencia en un Random Forest

Ventajas y Desventajas

Las principales ventajas de Random Forest son su simpleza al entrenar el modelo, desempeño muy eficiente, certera en base de datos grandes, mantiene su precisión con proporciones grandes de datos perdidos [16].

Algunas desventajas son la visualización gráfica donde los resultados pueden ser difíciles de interpretar, tiene poco control sobre lo que hace el modelo (en cierto sentido es como una caja negra) [16].

2.2.5. Naïve Bayes (Redes de Bayes)

Naïve Bayes es un algoritmo de clasificación probabilístico que utiliza la teoría de probabilidad y estadística para realizar clasificaciones. Es llamado *Naive* porque supone que todas las características son independientes entre sí, lo que en la mayoría de los casos no es verdad. Este teorema calcula la probabilidad de una clase dado un conjunto de características. La probabilidad se calcula multiplicando la probabilidad a priori de cada clase con la probabilidad condicional de cada característica dada esa clase[17].

Este algoritmo es muy eficiente y rápido que se utiliza en una amplia gama de aplicaciones, incluyendo la detección de spam, la categorización de documentos y la clasificación de texto. Además, es uno de los algoritmos más simples y fáciles de implementar en Machine Learning [17].

La fórmula condicional se expresa en la ecuación 2.2:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{(\frac{\#casosfavorablesA \cap B}{\#casosposibles})}{(\frac{\#casosfavorablesB}{\#casosposibles})} \tag{2.2}$$

Desarrollando la ecuación nos queda en la ecuación 2.3

$$P(A_i|B) = \frac{P(B|A_i) * P(A_i)}{P(B)} \tag{2.3}$$

Donde cada evento es tomado como se demuestra en la ecuación 2.4

$$P(A|B) = P(B_1|A) \times P(B_2|A) \times \ldots \times P(B_n|A) \times P(A) \tag{2.4}$$

Donde:

- $P(A_i)$ son las probabilidades a priori.
- $P(B|A_i)$ es la probabilidad de B dado A_i.
- $P(A_i|B)$ son las probabilidades a posteriori.
- $P(B)$ es la probabilidad total de B.

Las Redes de Bayes se generan de reglas de decisión donde participan activamente las probabilidades que ocurren referente a eventos, siendo que en base a esas probabilidades y resultados obtenidos, se toman decisiones sobre cuál arco de red moverse. Al final del proceso, el resultado será dado por el valor del nodo final de la red [18].

Ventajas y Desventajas

Las ventajas de esta técnica de ML son su fácilidad para implementarla, eficiencia y rápidez de tiempo de entrenamiento, buen rendimiento en textos y documentos, además no requiere mucha información para ser entrenado.

Las desventajas son la suposición de independencia de las variables (siendo que en muchos casos las características no son independientes), vulnerable al ruido, no es óptimo para características no correlacionadas [18].

2.2.6. Logistic Regression (Regresión Logística)

El algoritmo Logistic Regression modela la relación entre distintas variables utilizando una medida de error que se intentará minimizar en un proceso iterativo para poder realizar predicciones acertadas, llevando a cabo una clasificación binaria con una distribución Bernoulli en vez de Gaussian y después realiza una combinación lineal de las variables en un rango de 0 a 1 [19].

La ecuación lineal 2.5 se debe ajustar:

$$y(X) = W^T X + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon \tag{2.5}$$

$W^T X$ representa el producto escalar de entrada X.
W e Y son vectores de pesos $\in \{0, 1\}$.

Ahora mostrando la ecuación 2.6 la distribución de Bernoulli:

$$p(y|x, w) = Ber(y|u(x)) \tag{2.6}$$

El resultado del intervalo quedaría $0 \leq u(x) \leq 1$.

El resultado de la ecuación nos ayudará a predecir valores con la mejor respuesta a partir del menor error posible, teniendo un valor continuo entre 0 y 1. Si existe un valor mayor o igual a 0.5, la clase será 1, en cambio si es menor será 0. Todo esto ocurre porque el algoritmo de Logistic Regression predice un valor en vez de una clase en función de las variables utilizadas.

Ventajas y Desventajas

Logistic Regression al igual que las técnicas anteriores es fácil de implementar, interpretar y muy eficiente al momento de entrenar, incluyendo que no hace suposiciones sobre distribuciones de clases en el espacio de características y es muy rápido para clasificar registros desconocidos.

Las principales desventajas de esta técnica radican en si el número de observaciones es menor que el número de características, no se debe utilizar la regresión logística; de lo contrario, puede provocar un sobreajuste y la difícil obtención de

relaciones complejas. Para trabajos más potentes y compactos existen la Artificial Neural Networks, las cuales pueden superar fácilmente este algoritmo. [14].

2.2.7. Support Vector Machine (Máquina de Vectores de Soportes)

Esta técnica no será parte de los algoritmos que se analizarán en este trabajo, sin embargo, se explicará en qué consiste su proceso, debido a que es una de las técnicas clásicas de ML.

El algoritmo Support Vector Machine (SVM, por sus siglas en inglés) pertenece al ML supervisado que se utiliza para clasificación y regresión. Es un algoritmo de ML basado en el aprendizaje de modelos. El objetivo de SVM es encontrar un hiperplano que separe los datos en dos clases, de manera que los datos de una clase se encuentren en un lado del hiperplano y los datos de la otra clase se encuentren en el otro lado. El hiperplano es elegido de tal manera que maximice la margin, es decir, la distancia entre el hiperplano y los datos más cercanos. Estos datos más cercanos son conocidos como vectores de soporte. En este método, una función elige la predicción del valor esperado del caso mediante una entrada de datos [20].

2.2.8. Artificial Neural Networks (Redes Neuronales Artificiales)

Esta técnica no será parte de los algoritmos que se analizarán en este trabajo, sin embargo, se explicará en qué consiste su proceso, debido a que es una de las técnicas clásicas de ML.

La Artificial Neural Networks (ANN, por sus siglas en inglés) son un tipo de modelo de aprendizaje automático que simula la estructura y función de las redes neuronales en el cerebro humano. Una ANN está compuesta por nodos o "neuronas"que están conectados entre sí y transmiten información a través de las conexiones. Posee una arquitectura de procesadores múltiples interconectados para simular la estructura humana [21]. Las ANN se utilizan para realizar tareas como

la clasificación, la regresión, la traducción de idiomas, la generación de texto y la identificación de patrones en grandes conjuntos de datos.

Los métodos de aprendizaje que se emplean por lo regular son arquitecturas de redes neuronales tradicionales, donde solo se tienen dos o tres capas ocultas, imitando la operación que realiza el cerebro (Azath et al., 2020), en cambio, el DL aprenden sobre la marcha y su arquitectura puede llegar a tener 150 capas ocultas.

2.2.9. Deep Belief Network (Red de creencias profundas)

Esta técnica no será parte de los algoritmos que se analizarán en este trabajo, sin embargo, se explicará en qué consiste su proceso, debido a que es una de las técnicas clásicas de ML.

La Deep Belief Network (DBN, por sus siglas en inglés) son un tipo de ANN que se utiliza para modelar relaciones complejas entre variables. Una DBN es una combinación de varias redes neuronales artificiales y se entrena de forma no supervisada para aprender patrones y relaciones en los datos. La DBN tiene múltiples niveles de capas y variables ocultas, ellas están conectadas entre las capas visibles y ocultas, pero no en las capas visibles – visible u oculta – oculta [22].

Las DBN se utilizan en aplicaciones como la clasificación de imágenes, la detección de fraudes, la recomendación de productos y la identificación de tendencias en los datos.

2.2.10. Feedforward Artificial Neural Network (Red Neuronal de Retroalimentación)

Esta técnica no será parte de los algoritmos que se analizarán en este trabajo, sin embargo, se explicará en qué consiste su proceso, debido a que es una de las técnicas clásicas de ML.

Las Feedforward Artificial Neural Network (FANN) es la sucesora de ANN, trabajándose en diversos campos y aplicándose más en DL. Este algoritmo se caracteriza por tener una estructura en la que los datos fluyen en una dirección, desde las entradas hasta las salidas, sin retroalimentación o retroalimentación en ciclo. En la FANN, los datos se introducen en la red en la capa de entrada y se procesan a través de varias capas intermedias, cada una compuesta por una serie de nodos o neurones. Los nodos realizan cálculos simples en base a las entradas recibidas y generan una salida, que a su vez es procesada por la capa siguiente. La salida final de la red se produce en la capa de salida [21].

La FANN se utiliza en una amplia variedad de aplicaciones, incluyendo la clasificación de imágenes, la detección de fraudes, la recomendación de productos y la identificación de tendencias en los datos.

2.2.11. Recurrent Neural Networks (Redes Neuronales Recurrentes)

Esta técnica no será parte de los algoritmos que se analizarán en este trabajo, sin embargo, se explicará en qué consiste su proceso, debido a que es una de las técnicas clásicas de ML.

La Recurrent Neural Networks (RNN, por sus siglas en inglés) son un tipo de red neuronal artificial que se utiliza en el ML. A diferencia de la FANN, las RNN tienen una estructura que permite la retroalimentación, lo que les permite tomar en cuenta la secuencia temporal de los datos. Las RNN asigna parámetros únicos para representar a cada dato en una secuencia [23].

Para poder tener el control y no sufrir interrupciones sobre la secuencia. Su arquitectura es multicapa que comparte pasos entre los datos espaciados secuencialmente para poder unir la información. Su arquitectura se va incrementando

con la conexión de nodos adyacentes a través de la adición de ciclos dentro de la red.

Las RNN son reconocidas por obtener información de datos secuenciales como lo son el procesamiento del lenguaje natural, videos y subtitulación de imágenes.

2.3. Deep Learning (Aprendizaje Profundo)

Las técnicas de ML están limitadas en el procesamiento de los datos naturales en forma cruda y para dar solución a la problemática se creó el aprendizaje profundo. La comprensión de la IA y cómo puede llegar a igualar los comportamientos humanos, inclusive en el aprendizaje, a veces superándonos, fue un gran acontecimiento que se debe a la gran contribución de Alan Turing [1]. El DL como subárea del ML entra en acción cuando los datos tienen demasiadas características, son enormes, se requiere de un nivel de precisión altísima y el ML no puede ofrecer completamente los resultados deseados.

2.3.1. Convolutional Neural Networks (Redes Neuronales Convolucionales)

El DL ha demostrado muy buenos resultados para la resolución de problemas, en cambio, las limitaciones, sobre todo en el campo de la imagenología, ha hecho que se elabore un método diferente para que exista un análisis más preciso al momento de analizar una imagen. Este método se llama Convolutional Neural Networks (CNN), que tiene una arquitectura con mejor rendimiento para las tareas de relaciones complejas [24].

Desde el año 2012, las arquitecturas basadas en CNN para visión artificial han crecido muchísimo; sin embargo, no todas han sido eficientes para ocuparse en tareas de visión artificial (Figueroa Flores, 2021), siendo el Grupo de Geometría Visual (VGG) de la Universidad de Oxford [25] una de las arquitecturas más

utilizadas para las tareas de procesamiento de visión artificial.

Las CNN se forman usando tres tipos de capas, los cuales son capas convolucionales, capas de pooling y capas totalmente conectadas, como se muestra en la Figura 2.4.

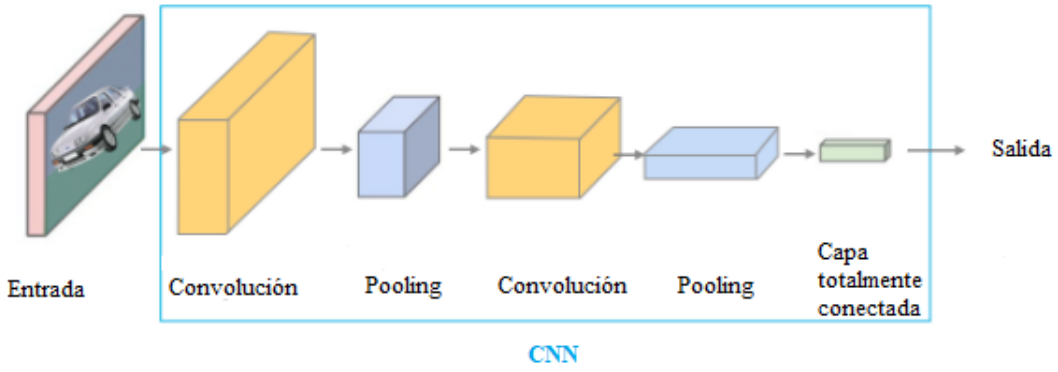

Figura 2.4: Descripción del funcionamiento de una CNN [1]

Para la CNN existen dos arquitecturas básicas, las cuales son CNN que entrega una salida para toda la imagen, como en la Figura 2.4 y la Fully Convolutional networks que posee un codificador y decodificador, entrega una compresión de la información y su salida es por pixel. En las arquitecturas CNN exiten varios ejemplos, como lo son AlexNet, entrenado con ImageNet, GoogLeNet que posee 22 capas y sus neuronas son más complejas, entre otras [26].

2.3.2. Neural Networks for Scarce Data Domains (Redes neuronales para dominios de datos escasos)

Las redes neuronales pueden ser muy efectivas para abordar problemas en los que los datos son escasos [1]. Sin embargo, cuando los datos son escasos, es esencial utilizar un enfoque diferente para entrenar las redes neuronales. Fei-Fei et al. [27] demostraron que es posible aprender nuevas categorías, una o pocas muestras por clase, aprovechando las categorías aprendidas anteriormente.

2.4. Otros conceptos importantes para la investigación relacionados con la IA

Existen muchos conceptos importantes que tienen relación con la IA, en esta sección definimos conceptos relevantes para lo que sigue de este trabajo.

2.4.1. Dataset

Un Dataset se refiere a una colección de datos que generalmente tiene la misma forma que una tabla de base de datos o una hoja de cálculo [28]. Un conjunto de datos consiste en un conjunto de ejemplos o casos. Una instancia también se denomina fila en una tabla de base de datos o, a veces, caso en estadística. Las funciones (columnas de la tabla) también tienen muchos nombres diferentes. Los estadísticos llaman atributos a las variables independientes o predictoras que se proporcionan como entradas. En investigación de operaciones, se habla de variables explicativas. La variable objetivo, cuyo valor se va a predecir, generalmente se denomina variable dependiente en estadística. La terminología puede ser un poco confusa; las variables independientes pueden no ser independientes entre sí (o de nada), y las variables dependientes no necesariamente dependen de todas las variables independientes. Es importante ser claro: la variable objetivo no se utiliza para predecirse a sí misma. Sin embargo, los valores anteriores de la variable objetivo pueden ser útiles para predecir valores futuros, por lo que estos valores pasados pueden incluirse como función [29].

En el Dataset la reducción de datos es importante y éste intenta tomar un gran conjunto de datos y reemplazarlo con un conjunto de datos más pequeño que contiene la mayor parte de la información importante en el conjunto más grande. Los conjuntos de datos más pequeños pueden ser más fáciles de manejar. Cuanto más pequeño sea el conjunto de datos, mejor información se podrá descubrir. Por ejemplo, grandes conjuntos de datos sobre preferencias de visualización de películas, se pueden reducir a un conjunto de datos más pequeño, que revela las preferencias de gustos del consumidor ocultas en los datos de visualización (como

las preferencias de género de la audiencia). La reducción de datos a menudo se asocia con la pérdida de información [29].

2.4.2. Matriz de Confusión

En el campo de la IA, en especial en el problema de la clasificación estadística, una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado .

La estructura de la matriz de confusión 2x2 o más (dependiendo del número de clases), donde cada fila representa una clase real y cada columna representa una clase predicha por el modelo [14]:

- **Positivo (P)**: La observación es positiva (por ejemplo, el paciente *tiene* COVID).
- **Negativo (N)**: La observación no es positiva (por ejemplo, el paciente *no tiene* Covid).
- **Verdadero Positivo (TP)**: Resultado en el que el modelo *predice correctamente la clase positiva*.
- **Verdadero Negativo (TN)**: Resultado donde el modelo *predice correctamente la clase negativa*.
- **Falso Positivo (FP)**: También llamado error de tipo 1, resultado donde el modelo *predice incorrectamente la clase positiva cuando en realidad es negativa*.
- **Falso Negativo (FN):** También llamado error de tipo 2, un resultado en el que el modelo *predice incorrectamente la clase negativa cuando en realidad es positiva*.

La matriz de confusión se utiliza para calcular métricas de rendimiento como la precisión, la exhustividad y la medida F1.

2.5. Aspectos de la salud y la enfermedad

La salud es uno de los aspectos más importantes en la vida, muchas veces nuestra salud se ve afectada por factores externos o internos a nuestra persona. “Este

concepto involucra un estado completo de bienestar físico, mental y social, y no solamente la ausencia de afecciones o enfermedades" (Wold Health Organization, 1946).

2.5.1. Accidente Cerebro Vascular

Un Accidente Cerebro Vascular es la detención del flujo de sangre a una parte del cerebro. En ocasiones se le llama "Ataque Cerebral". Si el flujo sanguíneo se detiene por más de pocos segundos, el cerebro no recibirá nutrientes ni oxígeno, causando una muerte y un daño permanente en la zona afectada [30].

Existen dos tipos de ACV, Isquémicos y Hemorrágicos.

2.5.2. Accidente Cerebro Vascular Isquémico

Los ACV Isquémicos son los más comunes, generalmente, son causados por un coágulo sanguíneo (masas que se presentan cuando la sangre se endurece, pasando de líquida a sólida) que bloquea el vaso sanguíneo del cerebro, lo que provoca la muerte de células cerebrales. Esto puede causar daño cerebral y resultar en una variedad de síntomas y discapacidades, incluyendo debilidad en un lado del cuerpo, problemas de habla, visión doble, pérdida de la capacidad de caminar y, en casos graves, la muerte [31].

Existen dos tipos de ACV Isquémicos: transitorios y permanentes [30]. Los transitorios se producen cuando la sangre no llega al cerebro por unos instantes, en cambio, el permanente es producido cuando la sangre no llega al cerebro por un tiempo prolongado, también es llamado Infarto Cerebral.

2.5.3. Síntomas ACV Isquémico

Los síntomas de un ACV Isquémico pueden ser:

- Entumecimiento o debilidad repentina de la cara, brazo o pierna (especialmente en un lado del cuerpo).
- Confusión repentina, dificultad para hablar o entender el lenguaje.
- Dificultad repentina para ver con uno o ambos ojos.
- Problemas para caminar repentino, mareos, pérdida de equilibrio o coordinación.

2.5.4. Categorías de los ACV

En los ACV para ayudar a optimizar el tratamiento específico, existen categorías que son identificadas por la escala de TOAST [31].

La primera categoría es la enfermedad Aterotrombótica aterosclerótica de gran vaso, se basa en la reducción de tejido sanguíneo medio o grande en el cerebro, con ubicación cortical o subcortical, con localización vertebrobasilar o carotídea, donde se encuentra presente una aterosclerosis u obstrucción con estenosis u oclusión de las arterias craneales. También la aterosclerosis sin estenosis con menos factores de riesgo se puede encontrar presente en esta categoría. La segunda categoría es el Cardioembolismo, es una reducción de tejido sanguíneo medio o grande, de localización cortical, en la que existe una cardiopatía embolígena [32]. La tercera categoría es la enfermedad oclusiva de pequeño vaso infarto lacunar, es una reducción de tejido sanguíneo de tamaño pequeño, en el sector de una arteria perforante cerebral que puede provocar una oclusión en el transporte de nutrientes. La cuarta categoría se debe a otras causas, de tamaño o localización variable que no están en las tres categorías anteriores, y que pueden producir enfermedades metabólicas, alteraciones de la coagulación, displacia fibromuscular, etc. La quinta categoría hace énfasis a los orígenes desconocidos con estudios incompletos o completos, por más de una etiología [33].

2.5.5. Tratamiento en los ACV

Las ayudas diagnósticas proveen información sobre el grado de lesión y la identificación de la lesión como las imágenes, escogiendo el tratamiento más adecuado para la lesión [34]. En el tratamiento como recomendación general, el soporte de la vía aérea y la asistencia ventilatoria es fundamental, ya que los pacientes pueden presentar alteración en el estado de conciencia o disfunción bulbar que afecte la vía aérea. Además, se recomienda lograr saturaciones de oxígeno mayores a 94 % aún si implica oxígeno suplementario. Agregando a lo anterior se debe monitorizar la hiperglicemia, porque si llega a perdurar por más de 24 horas, el pronóstico se asocia a un peor desenlace [30].

El tratamiento específico para un ACV Isquémico depende del tamaño y la ubicación del coágulo, así como de la gravedad de los síntomas. Algunos pacientes pueden requerir terapia médica, como anticoagulantes o medicamentos que disuelvan el coágulo, mientras que otros pueden necesitar cirugía o procedimientos endovasculares para extraer o disolver el coágulo. Además del tratamiento médico, los pacientes que han sufrido un ACV isquémico pueden necesitar rehabilitación para recuperar la función cerebral y física perdida. Esto puede incluir terapia física, terapia ocupacional, terapia de habla y otros tipos de rehabilitación [30].

2.5.6. Escala de ACV del National Institute of Health (NIHSS)

La escala de NIHSS mide el daño neurológico ocasionado en el paciente con ACV. Esta escala se ha convertido en una de las herramientas más útiles para monitorear neurológicamente en Unidades de ACV, tanto en la evaluación inicial del paciente, como para su seguimiento. El seguimiento puede arrojar mejoría o empeoramiento neurológico [35].

La escala contiene 11 items (desarrollada serían 15), que permiten valorar de forma rápida: funciones corticales, pares craneales superiores, función motora, sensibilidad, coordinación y lenguaje [35]. La escala tiene un minimo de 0 puntos y

un máximo de 42 puntos.

La clasificación de la puntación se presenta de la siguiente manera (Montaner et al, 2006) [36]:

- 0 punto: sin déficit.
- 1 punto: déficit mínimo.
- 2-5 puntos: leve.
- 6-15 puntos: moderado.
- 15-20 puntos: déficit importante.
- >20 puntos: grave.

La puntuación inicial tiene buen valor pronóstico [37], considerando que un NIHSS ≤ 6 corresponde con una excelente recuperación neurológica y cada incremento en un punto empeoraría la evolución [31]. Pacientes con fibrilación auricular, una NIHSS ≥ 16 ya se considera de muy mal pronóstico [38].

Capítulo 3

TRABAJOS RELACIONADOS

El análisis del estado del Arte contempla una revisión bibliográfica, contextualizando los avances en la investigación acerca de la IA y el área de salud. A continuación daremos a conocer algunos trabajos de los últimos años.

Los avances en hospitales de países desarrollados han permitido la implementación de la IA en sus sistemas, lo que implica nuevos desafíos, como lo son el procesamiento de grandes volúmenes de información, datos incompletos debido a la incompatibilidad de los sistemas en que se registran o incluso la presión de entregar una investigación o producto sin prolijidad y beneficio real para las personas [39]. Google es un ejemplo claro de esta situación donde sus algoritmos con IA presentan los problemas mencionados anteriormente.

Con el aumento de las tecnologías en el campo médico y avance de las técnicas de ML, se ha desarrollado un interés por el mundo científico para predecir algún proceso o secuela después de un ACV. Investigadores en el 2019 [40], realizaron estudio utilizando Random Forest, Redes Neuronales y Logistic Regression para predecir la prognosis de un paciente de ACV Isquémico tres meses después del evento inicial. Los autores deseaban predecir la mortalidad a los 3 años luego de salir de la rehabilitación con un algoritmo basado en Decision Tree. El mejor modelo fue Random Forest con la implementación del minority oversampling technique, el cual logró un nivel de predicción del modelo de 0.928 [41]. Yu et

al. [42] usaron técnicas de ML considerando el Decision Tree, siendo el objetivo clasificar la severidad del ACV Isquémico. El árbol se construyó originalmente con 13 variables, de 18 propuestas, los datos usados fueron de personas mayores de 65 años del National Institutes of Health Stroke Scale. Con esta técnica se logró tener un accuracy del 91.11 %, prediciendo el nivel de discapacidad, dentro de las 24 horas y posteriormente a los 90 días [43]. Los predictores incluían información de exámenes de escáner, demografía e información clínica de los pacientes.

Con la herramienta de las imágenes nace la posibilidad de generar algoritmos para un método más actualizado en la detección de esta enfermedad y su futura prevención, puesto que existen variadas técnicas [34] como la RM con DWI para la evaluación de presencia y extensión de isquemia posterior y la CTA y DSA para la trombosis de arteria. Continuando con las investigaciones en imágenes, un estudio en Estados Unidos [30] analizó 610.000 casos y 185.000 con recurrencia en el año 2019, los cuales mostraban lesiones visibles en sus exámenes de imagenología, con dicha información se pudo sustentar que el manejo médico y la prevención secundaria son vitales para mermar las secuelas de los pacientes. Además de la necesidad de educar a la comunidad para reconocer algunos síntomas del ACV y así acudir al centro médico más cercano. Con base en el estudio anteriormente señalado podemos predecir que esta enfermedad llegará a un 6,2 % de la población en países desarrollados, por esta razón es importante crear un modelo que nos permita predecir las secuelas o futuros problemas en el tratamiento, llevando a cabo una prevención exitosa, evitando la recurrencia. Por lo anteriormente expuesto es conveniente que el algoritmo para la atención deba estar basado en experiencias nacionles e internacionales [30]

La IA puede generar investigaciones DL relacionados a los estudios de Aprendizaje Profundo como estudio de revisión sistemática del diseño, estándares de informes y afirmaciones de los estudios de aprendizaje profundo [44]. El objetivo de un modelo con DL es que logre ser efectivo y eficaz y para ello es necesario trabajar con médicos expertos, a fin de que puedan evaluar los diagnósticos mediante imágenes y contrastar los resultados obtenidos con la IA. Esta investigación posee

una gran cantidad de datos como Ensayos controlados, datos Medline y ensayos que la Organización Mundial de la Salud posee desde 2010 hasta 2019, de los cuales se encontraron registros aleatorios de aprendizaje profundo con bajo nivel de sesgo. Por consiguiente la información que es emitida por los modelos de aprendizaje profundo para un diagnóstico más certero puede ser manipulada por los expertos, ya que los algoritmos [39] son experimentales y pioneros en la materia.

Las redes de datos convolucionales demostraron, en Corea, ser una herramienta que predice con precisión los cuidados intensivos en servicios médicos (Kang et al., 2020), asumiendo que el modelo predictivo, basado en DL, es superior a las otras herramientas de predicción y puntuaciones convencionales [45]. Cabe destacar, que el algoritmo de aprendizaje es muy eficiente por la cantidad de capas que puede poseer el modelo, puesto que entre más capas mayor puede ser el aprendizaje. Pese a las evidencias que demuestran la efectividad y el desempeño de las redes convolucionales, en términos de predicción, puede compararse o ser mejor al del humano, existe un miedo por la implementación en los sistemas de salud, lo que puede provocar que el crecimiento del DL, perezca de una base amplia para su desarrollo [39].

En el área de la implementación de un modelo con CNN, encontramos el trabajo de Chunjiao Dong , Chunfu Shao, Juan Li, and Zhihua Xiong del 2018, que desarrolla específicamente la predicción sobre los accidentes de tránsito [46]. Ellos demuestran una técnica novedosa con un modelo de regresión multivariable, que presenta la relación entre lo examinado y los accidentes de tránsito. Como resultados el modelo identifica las variables de entrada y representaciones de características de salida, aunque se haya reducido su magnitud, se conserva la información original. Además, el modelo propuesto explica mejor los problemas de heterogeneidad en predicción de accidentes de tráfico y puede ser aplicado a casos similares.

En este caso el modelo propuesto en contraste con el SVM en la categoría choque con daños menores es significativo (29.961 % versus 61.350 %), así es como

la predicción medida por el RMSD se puede mejorar un 84,58 % y un 158,27 % en comparación con el modelo de aprendizaje profundo sin la capa de regresión y el modelo SVM.

El trabajo "Intelligence versus clinicians: systematic review of design, reporting standards, and claims of deep learning studies" del año 2020 [39], que utilizó el Deep Learning con Redes Neuronales Convolucionales, tuvo como objetivo examinar sistemáticamente el diseño, los estándares de informes, el riesgo de sesgo y las afirmaciones de los estudios que comparan el rendimiento de los algoritmos de aprendizaje profundo de diagnóstico para imágenes médicas con el de médicos expertos. La investigación evaluó mediante estándares consolidados, informes de ensayo para estudios aleatorios de un modelo de predicción multivariable para pronóstico o diagnóstico individual, que comparan el rendimiento en imágenes médicas con un grupo contemporáneo de uno o más médicos expertos. Estos estudios seleccionados tenían como objetivo utilizar imágenes médicas para predecir el riesgo absoluto de enfermedad existente o la clasificación en grupos de diagnóstico (p. ej., enfermedad o no enfermedad). El estudio concluyó que hay una escasez de estudios prospectivos de DL y ensayos aleatorios en el campo de las imágenes médicas. La mayoría de los ensayos no aleatorios no fueron prospectivos, tuvieron un alto riesgo de sesgo y se desviaron de los estándares de información existentes. La mayoría de los estudios carecen de disponibilidad y código de datos, y los grupos de comparación humanos suelen ser pequeños. Otros estudios deberían reducir el riesgo de sesgo, mejorar la importancia clínica en el mundo real, mejorar los informes y la transparencia y corregir conclusiones moderadas.

El trabajo "Machine Learning–based model for prediction of outcomes in acute stroke", del año 2019 [40], utilizó Random Forest, Redes Neuronales y Logistic Regression para la predicción de prognosis de un paciente de ACV Isquémico tres meses después del evento inicial. El objetivo de este trabajo era buscar el mejor algoritmo para la problemática planteada. El estudio demostró que los algoritmos de ML, en particular la Red Neuronal Profunda, pueden mejorar la predicción de resultados a largo plazo para pacientes con ACV isquémico.

En “Machine learning to predict mortality after rehabilitation among patients with severe stroke” del año 2020 [41], se utilizó Logistic Regression y Random Forest con y sin implementación SMOTE (técnica estadística de sobremuestreo de minorías sintéticas para aumentar el número de casos de un conjunto de datos de forma equilibrada) para predecir la mortalidad después de la rehabilitación entre pacientes con ACV grave. El objetivo de este estudio era doble: evaluar el rendimiento relativo de los algoritmos basados en ML, con o sin la aplicación SMOTE, para predecir la mortalidad a largo plazo en pacientes con ACV con discapacidad grave y comparar el rendimiento de los algoritmos de ML con el de un modelo de Logistic Regression estándar. El estudio demostró que los algoritmos de ML superaron al modelo Logístico estándar para predecir la mortalidad a los 3 años, además, después de la implementación de SMOTE, los algoritmos de ML exhibieron un rendimiento general excelente, superando a los algoritmos sin la aplicación SMOTE, si bien las diferencias fueron pequeñas, el algoritmo RF exhibió el mejor rendimiento entre los algoritmos SMOTE.

La investigación “An elderly health monitoring system using machine learning and in-depth analysis techniques on the nihss stroke scale” del año 2020 [42], el cual utilizó Random Forest, Decision Tree, Logistic Regression y Artificial Neural Networks, propone un nuevo sistema de predicción y análisis en profundidad de la gravedad del ACV en personas mayores de 65 años basado en la escala de ACV de NIHSS y el mejor algoritmo de ML que es aplicable a la escala. Como conclusiones, el sistema clasifica y analiza de forma automática la gravedad de la apoplejía en cuatro clases que se utilizaron como clasificación, utilizando las funciones NIHSS de daño neurológico de los pacientes recopiladas en tiempo real. También el sistema proporciona a los pacientes y sus familias información de alarma sobre la gravedad del ACV en tiempo real, para que los pacientes puedan recibir visitas al centro médico y atención de emergencia. Con Decision Tree se realizó un análisis semántico con reglas adicionales destalladas.

En “Use of gradient boosting machine learning to predict patient outcome in

acute ischemic stroke on the basis of imaging, demographic, and clinical information" del año 2018 [43], se utilizó Decision Tree con aumento de gradiente (GBM) y refuerzo de gradiente extremo (XGB). El objetivo de este estudio fue integrar biomarcadores comunes de ACV utilizando métodos de ML y predecir el resultado de la recuperación del paciente a los 90 días. El estudio concluyó que los GBM basados en Decision Tree pueden predecir el resultado de la recuperación de los pacientes con ACV al ingreso con un AUC alto. Dividir los grupos de pacientes sobre la base de la recanalización y la no recanalización puede ayudar potencialmente con el proceso de decisión del tratamiento.

El trabajo "Imaging recommendations for acute stroke and transient ischemic attack patients: A joint statement by the american society of neuroradiology, the american college of radiology, and the society of neurointerventional surgery" del año 2013 [34], utilizó NCCT que es una técnica estándar de diagnóstico por imágenes aceptada para la exclusión de hemorragia intracraneal y se ha incorporado en los criterios de inclusión en ensayos clínicos aleatorizados, llevando a su uso generalizado continuado en imágenes de ACV agudos. Como resumen, en pacientes con ACV agudo que son candidatos para trombólisis IV, se recomiendan imágenes de NCCT para excluir hemorragia intracraneal y determinar la extensión de los cambios isquémicos, además, los resultados concordantes de, al menos, 2 técnicas de imagen no invasivas se pueden usar para determinar la elegibilidad del tratamiento para los procedimientos de revascularización.

El estudio "Actualización en diagnóstico y tratamiento del ataque cerebrovascular isquémico agudo" del año 2019 [30], tuvo como objetivo presentar una actualización sobre los métodos diagnósticos actuales y las distintas terapias disponibles según sea el caso de cada paciente, para el ACV isquémico agudo, con un enfoque clínico práctico, ordenado y aplicable al escenario actual de salud en Colombia. Como conclusión, los pacientes que son candidatos a un tipo de terapia especifica post ACV deben regirse con algunos criterios como escala de Ranking, NIHSS, entre otros, que señala el algoritmo de árbol de decisiones y es importante contar con políticas en salud pública enfocadas en educar a la comunidad en

reconocer de manera oportuna los síntomas de un ACV para acudir rápidamente a un centro médico.

El trabajo "A novel end-to-end classifier using domain transferred deep convolutional neural networks for biomedical images" del año 2017 [44] aplica el método de Redes Neuronales Convolucionales de DL. El objetivo es la clasificación de imágenes biomédicas y la identificación de enfermedades a partir de ellas. En el estudio se propuso un clasificador de extremo a extremo altamente confiable y preciso para todo tipo de imágenes biomédicas a través del DL y el aprendizaje por transferencia. Como conclusión, el clasificador de extremo a extremo automatizado basado en un modelo con Redes Neuronales Convolucionales es altamente confiable y preciso, que ha sido confirmado por varios conjuntos de datos de imágenes biomédicas públicas.

El trabajo "Desafios bioéticos do uso da inteligência artificial em hospitais" del año 2022 [45] plantea un análisis de los desafíos de la IA en los hospitales. El objetivo es la identificación de desafíos en el desarrollo de sistemas dotados de IA (fase prehospitalaria) y en la implementación y formación de equipos de salud (fase hospitalaria). Como conclusión presentó numerosas posibilidades para el uso de la IA en el área de la salud, destacando su uso en el soporte hospitalario y sopesando las ventajas y desafíos.

La investigación "An improved Deep learning model for traffic crash prediction" del año 2018 [46] utilizó DL con un modelo binomial negativo multivariable (MVNB). En este estudio se propone un modelo de DL mejorado para explorar las complejas interacciones entre las carreteras, el tráfico, los elementos ambientales y los accidentes de tráfico. Como conclusión, el modelo propuesto que incluye la capa de regresión MVNB en el módulo de ajuste fino supervisado puede explicar mejor los patrones de distribución diferencial en los accidentes de tráfico según la gravedad de las lesiones y proporciona mejores predicciones de accidentes de tráfico.

Los trabajos más citados y precisos en el área de la salud y otros campos dentro de la IA se atribuyen al DL, la utilidad y precisión de este modelo es en gran medida funcional y confirma hallazgos con características de vital importancia. Dentro de las técnicas clásicas la literatura nos hace referencia a las Artificial Neural Networks, Logistic Regression, Decision Tree, Random Forest y Naïve Bayes. En salud las escalas qué miden en que estado se encuentra el paciente juegan un rol importante para una atención primaria de rápida atención, es por eso que la literatura señala a la escala NIHSS como una de las más importante a tener en cuenta en un algoritmo de atención en caso de un ACV.

Capítulo 4

ESTUDIO EMPÍRICO

El presente capítulo tiene como finalidad la explicación del método realizado, donde se recopilaron datos de pacientes del Hospital Herminda Martín de Chillán y se realizó la implementación del método escogido.

4.1. Enfoque de la investigación

La investigación cuenta con un enfoque cuantitativo por los resultados que se quieren llegar a obtener, donde primará el análisis matemático de los datos presentes en la base de datos y entre los algoritmos a comparar. El tipo de investigación será de carácter experimental, ya que medirá tendencias en los resultados arrojados por los algoritmos.

La población estará conformada por todas las personas cuyos datos aparecen registrados en la base de datos a trabajar. En este caso específico, se trata de pacientes post ACV Isquémico del Hospital Herminda Martin de Chillán. La muestra serán los pacientes post ACV Isquémico registrados en la base de datos y que cumplan con algunos criterios médicos para su análisis.

4.2. Metodología

El diseño metodológico dará una guía con los pasos a seguir para la obtención de la finalidad del proyecto. Se tendrá en cuenta el tipo, el enfoque, la población y la muestra para iniciar el trabajo.

En este trabajo se tomará como referencia la propuesta del libro *Machine Leaning in Action* [14], en particular el procedimiento de la sección *"Steps in developing a machine learning application"* que establece 6 pasos para la implementación de una aplicación que utiliza técnicas de ML.

1. **Colección de los datos de entrada:** El primer paso para implementar una aplicación que trabaje utilizando técnicas de ML es coleccionar los datos que serán analizados.

2. **Preparación de datos de entrada:** Una vez que se obtienen los datos, es necesario asegurarse que estén en el formato correcto para ser procesados por el algoritmo de ML seleccionado. El formato que usaremos en este estudio es la lista de Python. El beneficio de tener este formato estándar es que puede mezclar, combinar algoritmos y fuentes de datos.

   Este paso involucra, si fuera necesario, formatear los datos para adaptarlos a la necesidad de cada algoritmo.

3. **Analizar los datos de entrada:** En este paso se debe observar los datos para reconocer algún patrón o si hay algo obvio, como algunos puntos de datos que son muy diferentes del resto del conjunto. Los pasos 1 y 2 deben realmente funcionar y no contener un montón de datos vacíos.

4. **Entrenamiento del algoritmo:** Aquí es donde tiene lugar el aprendizaje automático. Este paso y el siguiente paso es donde se encuentran los algoritmos "básicos". Se alimenta el algoritmo con los datos limpios de los primeros dos pasos y se extrae el conocimiento y la información dependiendo de la funcionalidad del algoritmo. El conocimiento se almacenará en un formato simple

de utilización para el algoritmo, en los siguientes pasos se utilizará este conocimiento.

5. **Testeo del algoritmo (Prueba del algoritmo):** La información aprendida por el algoritmo es testeada, es decir, se mide el nivel de acierto que tiene el algoritmo. Utilizando los datos de entrenamiento podremos establecer el grado de eficacia de la implementación de la técnica seleccionada. Si los resultados no son los esperados es probable que haya que volver a etapas previas para intentar identificar el error, que tal vez se encuentre en los datos de entrada o en el algoritmo en sí. Una vez realizados los cambios, hará falta volver a pasar por todos los pasos anteriores una vez más.

6. **Uso del algoritmo:** Una vez se han consumado todos los pasos previos, no queda más que usar el algoritmo, esta etapa implica tener que volver a ejecutar los pasos 1, 2, 3 y 5.

A continuación, en la Figura 4.1 se muestra un mapa de secuencia con los pasos a seguir para obtener los resultados esperados:

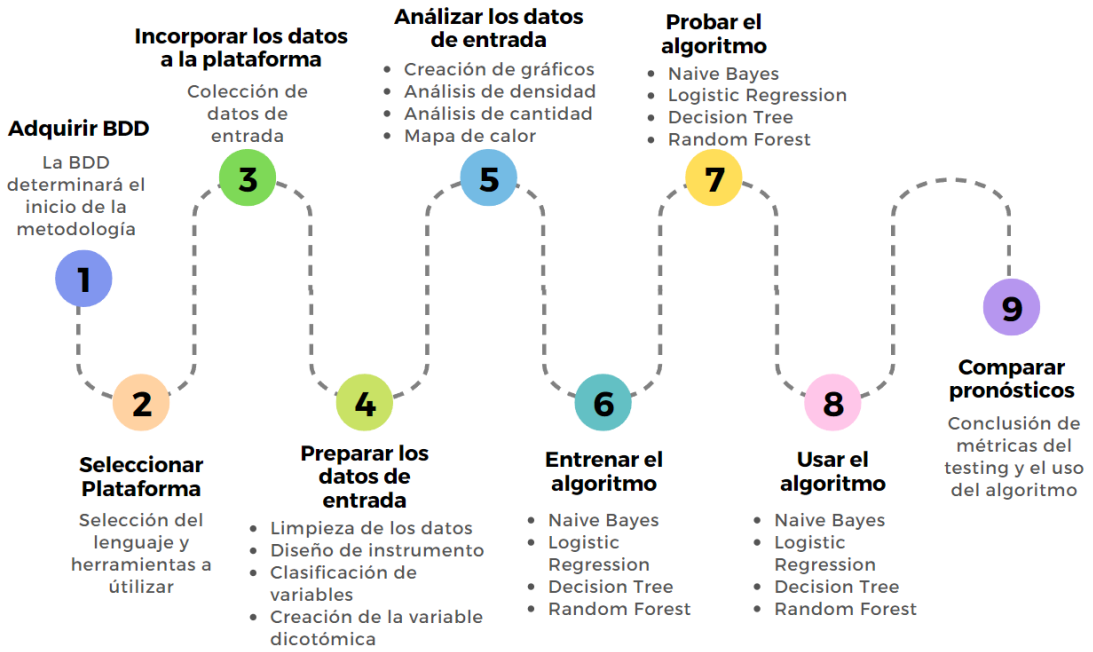

Figura 4.1: Mapa de secuencia metodológica

4.2.1. Lenguaje de programación y plataforma

Los lenguajes de programación más usados para el ML son Java, R o Python. Para este proyecto se escogió Python por cuatro motivos:

1. Librerías potentes: Python tiene una amplia gama de librerías y frameworks específicos para IA, como TensorFlow, PyTorch, scikit-learn, entre otros, que facilitan la creación y entrenamiento de modelos de IA.

2. Fácil de aprender: La sintaxis clara y legible de Python hace que sea fácil de aprender y usar para los desarrolladores, lo que significa que se pueden desarrollar modelos de IA de manera más eficiente.

3. Comunidad activa: La comunidad de Python es activa y colaborativa, lo que significa que hay una gran cantidad de recursos y soluciones disponibles en línea para ayudar a los desarrolladores a resolver cualquier problema que puedan tener.

4. Interoperabilidad: Python se integra fácilmente con otros lenguajes y tecnologías, lo que significa que se pueden utilizar conjuntamente con otros sistemas y herramientas en proyectos de IA.

Siguiendo con este razonamiento, para trabajar con el lenguaje Python se pudieron ocupar framework como Tensor Flow, Theano, Amazon Machine Learning, entre otros; al final se optó por la plataforma llamada Jupyter Notebook, popular en la Ciencia de Datos e IA debido a estas ventajas:

1. Interfaz interactiva: Jupyter permite a los usuarios ejecutar código en un entorno interactivo y ver los resultados en tiempo real, lo que facilita la exploración y experimentación con datos y modelos de IA.

2. Documentación integrada: Jupyter permite a los usuarios crear documentos que combinan código, texto, gráficos y otros tipos de contenido multimedia, lo que lo convierte en una excelente opción para la creación de informes y presentaciones de resultados.

3. Compartir y colaborar: Jupyter permite a los usuarios compartir y colaborar fácilmente en proyectos de IA mediante la creación de notebooks en línea y la posibilidad de trabajar en tiempo real con otros usuarios.

4. Amplia compatibilidad: Jupyter es compatible con una amplia gama de lenguajes de programación, incluyendo Python, R, Julia, entre otros, lo que significa que los usuarios pueden elegir el lenguaje que mejor se adapte a sus necesidades.

4.3. Aplicación

Para empezar, debemos implementar los pasos sugeridos en la sección 4.2 en el mismo orden en que se nos presenta. Los pasos 1, 2 y 3 son comunes para todos los algoritmos, por ende, solo se mostrará una vez en la presente investigación. Al finalizar esta sección se verán los pasos 4, 5 y 6 en cada Algoritmo presentado.

4.4. Colección de datos de entrada

El primer paso en el método es la Colección de datos de entrada, en este proceso capturaremos los datos provenientes de la BDD del Hospital Herminda Martín de Chillán y luego diseñaremos un nuevo instrumento.

[1]:
```
# Libreria para la manipulación de los datos
import pandas as pd
import numpy as np


# Leer el dataframe
dataframe = pd.read_excel('../bdd/bdd_final.xlsx')
print(dataframe)
```

CLAVE	COMUNA	TELEFONOS	EDAD	...	IL-6 corregida	log IL-6	IL-6/VEGF	IL-6/PlGF
1	san carlos		53	...	0,156952708	-0,804	-0,454839548	-0,825732298
2	coihueco	41723921-74822219	54	...	0,012817828	-1,892	-0,882899371	-1,081411325
3	chillan	71818219-50323843	78	...	0,436365262	-0,360	-0,230735564	-0,296817468
. . .	. . .	. . .	. . .	...	. . .	. . .	. . .	. . .
75	pinto	82525308	79	...	0,842328176	-0,075	-0,047317636	-0,08950326
76	chillan	50711724-87107760	54	...	1,273106319	0,105	0,054145766	0,137865611
TENS	chillan	85226191-96473857	69	...	0,795830574	-0,099	#¡NUM!	#¡NUM!

Tabla 4.1: Base de datos de pacientes post ACV Isquémico del Hospital Herminda Martin

[2]:
```
# Mostramos las variables que posee la base de datos
columns_names = dataframe.columns.values
print(columns_names)
```

```
['CLAVE' 'COMUNA' 'TELEFONOS ' 'FICHA CLINICA' 'CTA CTE'␣
↪'EDAD' 'PESO'
 'TALLA' 'HTA' 'DIABETES' 'OTRAS PATOLOGIAS' 'FUMA' 'FC'␣
↪'PAS' 'PAD'
 'GLUCOSA' 'Hb A/C  %' 'COL. TOTAL' 'TRIGLICERIDOS' 'LDL'␣
↪'HDL' 'HCTO'
 'HB' 'VCM' 'HCM' 'VHS' 'PLAQUETAS' 'INR' 'CONTEO G.B.' 'P.C.
↪R'
 'Nitrogeno Ureico' 'Uremia' 'Creatinina' 'TTPA' 'TP' 'NA'␣
↪'K' 'CL'
 'Fosfatasa Alcalina' 'Gamma glutamil' 'Transaminasa␣
↪piruvica'
 'Trans oxal' 'AREA DE LESION ' 'No  ESTENOSIS INTRACRANEAL'
 'No  ESTENOSIS EXTRACRANEAL' '%  ESTENOSIS INTRACRANEAL'
 '%  ESTENOSIS EXTRACRANEAL' 'GLASGOW AL INICO ACV' 'NIHSS␣
↪INICO ACV'
 'RANKIN INICIO ACV' 'NIHSS alta ACV' 'RANKIN alta ACV'␣
↪'NIHSS 6M'
 'RANKIN 6M' 'diag Elopez' 'DIAG. NEUROLOGICO' 'Diag2'␣
↪'Diag3'
 'FECHA TOMA MUESTRA' 'Estado paciente' 'Fecha defuncion'
```

```
 'CAUSA  DEFUNCION ' 'MOTIVO DE DESCARTE ' 'TROMBOLISIS'␣
↪'eduardo'
 'sirve ' 'escala' 'exosomes 1' 'exosomes 2' 'VEGF ab1'␣
↪'VEGF ab2'
 'VEGF prome' 'plgf mg prot' 'plgf mg prot.1' 'plgf␣
↪promedio' 'logVEGF'
 'logPlGF' 'logPCR' 'PCR/VEGF ratio' 'PCR/PLGF ratio' 'IL-6␣
↪(pg/ml)'
 'IL-6 corregida' 'log IL-6' 'IL-6/VEGF' 'IL-6/PlGF']
```

Mostramos la cantidad de pacientes y variables (columnas) que posee la BDD:

[3]:
```
print('Existen {} pacientes con {} variables.'.
 ↪format(*dataframe.shape))
print("Existen", dataframe.size, "elementos")
```

```
Existen 75 pacientes con 85 variables.
Existen 6375 elementos
```

Se observa que la BDD posee muchas variables y pocos pacientes registrados en las tuplas. Esto hará que sea más difícil la predicción para los algoritmos, así que necesitamos un nuevo instrumento.

4.4.1. Diseño del instrumento

Desde el punto de vista científico, para que un estudio resulte lo más certero posible necesitamos variables significativas para la investigación y con la menor pérdida de datos posible. En la BDD todas las variables presentan importancia, algunas son imprescindibles para la investigación, otras con pocos datos completados o simplemente las variables sujetas a interpretación médica (humana). Debido a lo anterior se seleccionaron las variables por dos motivos, el primero fue porque eran las que estaban más completas en la BDD y el segundo porque se determinó que eran más significativas para la investigación por estudios realizados al ACV y dataset presentes en internet. A continuación, se mostrarán las variables escogidas y una pequeña descripción de ellas.

HTA: "si" o "no", HIPERTENSIÓN

DIABETES: "si" o "no"

EDAD: Edad del paciente

GLUCOSA: Nivel de azúcar en la sangre

COL. TOTAL: Cantidad de Colesterol en la sangre

TRIGLICERIDOS: Cantidad de triglicéridos en la sangre

INR: Índice internacional normalizado (INR, por sus siglas en inglés) es un tipo de cálculo que se basa en los resultados de las pruebas de tiempo de protrombina

CONTEO G.B.: Conteo de glóbulos blancos en la sangre

GLASGOW AL INICO ACV: Escala de 15 puntos médica que es para medir el estado de conciencia. Esta pertence a la Inicial

NIHSS INICO ACV: Escala de 42 puntos más empleada para la valoración de funciones neurológicas básicas en la fase aguda del ictus isquémico, tanto al inicio como durante su evolución. Esta pertence a la Inicial

NIHSS alta ACV: Corresponde al NIHSS cuando es dado de alta el paciente

La Hipertensión Arterial y la Diabetes son factores de riesgo altos en cualquier enfermedad no trasmisible, por esto son de las primeras seleccionadas. Además, contaremos con las escalas de Glasgow y NIHSS que son escalas internacionales para la evaluación del ACV Isquémico.

4.5. Preparación de los datos de entrada

El segundo paso descrito en la metodología es un paso crítico y uno de los más extensos en el desarrollo de proyectos de ML. Aquí se llevará a cabo una limpieza de datos, como la eliminación de registros o rescatar datos nulos; selección de características que sean relevantes para la investigación; transformación de características, que serán para escalar datos para que se adecuen al modelo de ML.

4.5.1. Eliminación valores nulos

Como se indicó anteriormente, se dispone de 75 tuplas con 85 columnas, con un total de 6375 elementos, que se planean disminuir por indicación del médico que facilitó la base de datos. La indicación fue que había pacientes que fueron retirados del programa y estaban marcados con un "out" en la variable de "diag Elopez".

[4]:
```
dataframe.drop(dataframe[(dataframe['diag Elopez'] ==␣
↪'out')].index, inplace=True)


# mostramos 10 columnas
pd.options.display.max_columns = 10


# Mostramos las primeras 7 tuplas
dataframe.head(7)
```

CLAVE	COMUNA	TELEFONOS	EDAD	...	IL-6 corregida	log IL-6	IL-6/VEGF	IL-6/PlGF
1	san carlos		53	...	0,156952708	-0,804	-0,454839548	-0,825732298
2	coihueco	41723921-74822219	54	...	0,012817828	-1,892	-0,882899371	-1,081411325
3	chillan	71818219-50323843	78	...	0,436365262	-0,360	-0,230735564	-0,296817468
.	.	.	.		.	.	.	.
.	.	.	.	...	.	.	.	.
.	.	.	.		.	.	.	.
75	pinto	82525308	79	...	0,842328176	-0,075	-0,047317636	-0,08950326
76	chillan	50711724-87107760	54	...	1,273106319	0,105	0,054145766	0,137865611
TENS	chillan	85226191-96473857	69	...	0,795830574	-0,099	#¡NUM!	#¡NUM!

Tabla 4.2: Eliminación de pacientes que no aportan en la investigación

[5]:
```

```

```
print('Existen {} pacientes con {} variables.'.
 ↪format(*dataframe.shape))
print("Existen", dataframe.size, "elementos")
```

```
Existen 46 pacientes con 85 variables.
Existen 3910 elementos
```

4.5.2. Variables significativas para la investigación

Ahora asignamos las variables significativas, para esto se extrae la información del marco de trabajo, identificando las variables categóricas para un arreglo completamente nuevo y así empezar a trabajar sobre el nuevo archivo.

[6]:
```
# Tomaremos las variables más significativas para la␣
 ↪investigación
columnasMuestra = ['HTA', 'DIABETES', 'EDAD', 'GLUCOSA',␣
 ↪'COL. TOTAL', 'TRIGLICERIDOS', 'INR', 'CONTEO G.B.',␣
 ↪'GLASGOW AL INICO ACV', 'NIHSS INICO ACV', 'NIHSS alta␣
 ↪ACV']
dataset = dataframe[[*columnasMuestra]]

# Muestramos las columnas que se ajusten a la cantidad de␣
 ↪espacio
pd.options.display.max_columns = 0

dataset.head(5)
```

HTA	DIABETES	EDAD	GLUCOSA	COL. TOTAL	TRIGLICERIDOS	INR	CONTEO G.B.	GLASGOW AL INICO ACV	NIHSS INICO ACV	NIHSS alta ACV
		53	137,09	268	130	1,08	41,9	11	14	42
si	si	54		187	130		8,3	15	6	0
si	si	78	359,42	159	97	0,89	8,5	15	5	2
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
si	si	79	116,99	109	118		9,9		2	2
si	si	54	211,58			0,99	6,5	15	1	1
si	si	69	217,21	202	232	1,03	10,4	0		

Tabla 4.3: Dataset de variables para la investigación

Mostramos la cantidad de pacientes y variables/columnas que posee la BDD

después de la selección de variables significativas para la investigación:

[7]:
```
print('Existen {} pacientes con {} variables.'
 ↪format(*dataset.shape))
print("Existen", dataset.size, "elementos")
```

```
Existen 46 pacientes con 11 variables.
Existen 506 elementos
```

4.5.3. Descripción general de los datos

En la descripción de los datos, se muestran parámetros, pérdida de datos (Missing Data), forma y su descripción estadística.

[8]:
```
# Check Dataset:
def check_data(dataset,head=5):
    print(20*"-" + "Información".center(20) + 20*"-")
    print(dataset.info())
    print(20*"-" + "Forma de datos".center(20) + 20*"-")
    print(dataset.shape)
    print("\n" + 20*"-" + "Los primeros 5 datos".center(20)␣
 ↪+ 20*"-")
    print(dataset.head())
    print("\n" + 20 * "-" + "Los últimos 5 datos".
 ↪center(20) + 20 * "-")
    print(dataset.tail())
    print("\n" + 20 * "-" + "Missing Data".center(20) + 20␣
 ↪* "-")
    print(dataset.isnull().sum())
    print("\n" + 20 * "-" + "Describir los datos".
 ↪center(20) + 20 * "-")
    print(dataset.describe().T)
check_data(dataset)
```

```
--------------------    Información     --------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 46 entries, 0 to 74
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   HTA                   42 non-null     object
 1   DIABETES              34 non-null     object
 2   EDAD                  44 non-null     float64
 3   GLUCOSA               38 non-null     float64
 4   COL. TOTAL            40 non-null     float64
 5   TRIGLICERIDOS         40 non-null     float64
 6   INR                   37 non-null     float64
 7   CONTEO G.B.           46 non-null     float64
 8   GLASGOW AL INICO ACV  29 non-null     float64
 9   NIHSS INICO ACV       39 non-null     float64
 10  NIHSS alta ACV        32 non-null     float64
dtypes: float64(9), object(2)
memory usage: 4.3+ KB
None
--------------------   Forma de datos   --------------------
(46, 11)

--------------------Los primeros 5 datos--------------------
   HTA DIABETES  ...  NIHSS INICO ACV  NIHSS alta ACV
0  NaN      NaN  ...             14.0            42.0
1   si       si  ...              6.0             0.0
2   si       si  ...              5.0             2.0
3   si       si  ...              1.0             0.0
4   si       si  ...              3.0             2.0

[5 rows x 11 columns]
```

```
--------------------Los últimos 5 datos --------------------
   HTA DIABETES  ...  NIHSS INICO ACV  NIHSS alta ACV
68  si       no  ...              NaN             NaN
69  no       no  ...              2.0             0.0
72  si       si  ...              2.0             2.0
73  si       si  ...              1.0             1.0
74  si       si  ...              NaN             NaN

[5 rows x 11 columns]

--------------------    Missing Data    --------------------
HTA                      4
DIABETES                12
EDAD                     2
GLUCOSA                  8
COL. TOTAL               6
TRIGLICERIDOS            6
INR                      9
CONTEO G.B.              0
GLASGOW AL INICO ACV    17
NIHSS INICO ACV          7
NIHSS alta ACV          14
dtype: int64

--------------------Describir los datos --------------------
                      count        mean  ...       75%     ␣
 ↪max
EDAD                   44.0   71.522727  ...   80.5000   90.
 ↪00
GLUCOSA                38.0  135.529211  ...  162.5925  359.
 ↪42
```

```
COL. TOTAL              40.0  162.750000  ...  188.5000  342.
↪00
TRIGLICERIDOS           40.0  123.450000  ...  142.0000  232.
↪00
INR                     37.0    1.197568  ...    1.2000    3.
↪08
CONTEO G.B.             46.0   10.119783  ...   10.3750   41.
↪90
GLASGOW AL INICO ACV    29.0   13.482759  ...   15.0000   15.
↪00
NIHSS INICO ACV         39.0    5.564103  ...    6.5000   21.
↪00
NIHSS alta ACV          32.0    5.687500  ...    4.0000   42.
↪00

[9 rows x 8 columns]
```

La descripción de los datos nos ayuda a evaluar los casos particulares de las variables que serán tratadas en los siguientes títulos.

4.5.4. Missing data

Los datos que faltan ocurren cuando no se almacena ningún valor en la variable de observación. Aunque el Missing data es una ocurrencia muy común, estos pueden tener una presión significativa en los resultados de la aplicación del instrumento. Para corregir este problema, existen variadas técnicas estadísticas, siendo una de ellas la mediana y el redondeo como lo muestra [47] en algunas variables que utilizaremos para llenar las celdas faltantes. Tomaremos la información que mostramos anteriormente.

[9]:
```
dataset.isnull().sum()
```

	MISSING DATA
HTA	4
DIABETES	12
EDAD	2
GLUCOSA	8
COL. TOTAL	6
TRIGLICERIDOS	6
INR	9
CONTEO G.B.	0
GLASGOW AL INICO ACV	17
NIHSS INICO ACV	7
NIHSS alta ACV	14

Tabla 4.4: Missing data de variables

Como se demuestra en la tabla 4.4, casi todas las variables presentan missing data, así que el algoritmo para trabajar será el siguiente:

[10]:
```
valores_por_defecto = {
                        'HTA': "DESCONOCIDO",
                        'DIABETES' : "DESCONOCIDO",
                        'EDAD':dataset["EDAD"].median().
 ↪round(),
                        'GLUCOSA':dataset["GLUCOSA"].
 ↪median(),
                        'COL. TOTAL':dataset["COL. TOTAL"].
 ↪median(),
                        'TRIGLICERIDOS':
 ↪dataset["TRIGLICERIDOS"].median(),
                        'INR':dataset["INR"].median(),
                        'GLASGOW AL INICO ACV':
 ↪dataset["GLASGOW AL INICO ACV"].median().round(),
                        'NIHSS INICO ACV':dataset["NIHSS␣
 ↪INICO ACV"].median().round(),
                        'NIHSS alta ACV':dataset["NIHSS alta␣
 ↪ACV"].median().round(),
                       }
# Missing Data
dataset = dataset.fillna(value=valores_por_defecto)
```

Comprobamos si ahora existe missing data:

[11]:
```
dataset.isnull().sum()
```

	MISSING DATA
HTA	0
DIABETES	0
EDAD	0
GLUCOSA	0
COL. TOTAL	0
TRIGLICERIDOS	0
INR	0
CONTEO G.B.	0
GLASGOW AL INICO ACV	0
NIHSS INICO ACV	0
NIHSS alta ACV	0

Tabla 4.5: Missing data finalizado

Como se ve en la tabla 4.5, ya no existe missing data.

4.5.5. Procesamiento de los datos y clasificación

La preparación de los datos para el modelo de ML es lo que nos llevará a decidir qué modelo podremos utilizar, ya que trabajar con modelos de clasificación no es lo mismo que con los de regresión. Los pasos exactos para la preparación de los datos dependerán del modelo utilizado y de los datos recopilados.Cabe destacar que se requerirá cierta cantidad de manipulación de datos para cualquier aplicación de ML.

Análisis de variable Objetivo

En el apartado del Marco Teórico 2.5.6 se mencionó de la escala NIHSS y sus niveles de daño neurológico, considerándola como la variable más importante en la predicción. Dentro de nuestro dataset existe la escala NIHSS al iniciar el ACV, la escala NIHSS al alta del paciente y la escala NIHSS en el control de los 6 meses después del ACV.

La escala que se utilizará en la predicción es la escala NIHSS de alta.

[12]:
```
dataset["NIHSS alta ACV"].value_counts()
```

NIHSS alta ACV	Cantidad
1	19
0	12
2	6
42	2
4	2
8	1
10	1
18	1
32	1
5	1

Tabla 4.6: Cantidad de pacientes en la escala NIHSS en alta

En esta escala los valores menores suponen un mejor pronóstico para los pacientes, es por eso que se decide crear una variable con los resultados que da el test NIHSS.

Crear columna para NIHSS_alta_cat Esta columna nos servirá para la clasificación de los estados de la variable NIHSS.

[13]:
```
# Realizamos la clasificación de la escala
condicionesALTA = [
    (dataset['NIHSS alta ACV'] == 0),
    (dataset['NIHSS alta ACV'] == 1),
    (dataset['NIHSS alta ACV'] >= 2) & (dataset['NIHSS alta␣
↪ACV'] <= 5),
    (dataset['NIHSS alta ACV'] >= 6) & (dataset['NIHSS alta␣
↪ACV'] <= 15),
    (dataset['NIHSS alta ACV'] >= 16) & (dataset['NIHSS␣
↪alta ACV'] <= 20),
    (dataset['NIHSS alta ACV'] > 20),
]
valoresALTA = ["Sin Déficit", "Déficit Mínimo", "Leve␣
↪(Trombolisando)", "Moderado (Buen Pronóstico)", "Déficit␣
↪Importante", "Grave"]
```

```
dataset['NIHSS_alta_cat'] = np.select(condicionesALTA,␣
↪valoresALTA)
dataset['NIHSS_alta_cat'].value_counts()
```

NIHSS_alta_cat	Cantidad
Déficit Mínimo	19
Sin Déficit	12
Leve (Trombolisando)	9
Grave	3
Moderado (Buen Pronóstico)	2
Déficit Importante	1

Tabla 4.7: Cantidad de pacientes en la clasificación NIHSS de alta

[14]:
```
# Como son muchas columnas, muestro menos columnas
pd.options.display.max_columns = 6


dataset.head(5)
```

	HTA	DIABETES	EDAD	GLUCOSA	...	NIHSS INICO ACV	NIHSS alta ACV	NIHSS_alta_cat
0	DESCONOCIDO	DESCONOCIDO	53	137,09	...	14	42	Grave
1	si	si	54	119,995	...	6	0	Sin Déficit
2	si	si	78	359,42	...	5	2	Leve (Trombolisando)
.	.	.	.	.	...	.	.	.
.	.	.	.	.		.	.	.
.	.	.	.	.		.	.	.
72	si	si	79	116,99	...	2	2	Leve (Trombolisando)
73	si	si	54	211,58	...	1	1	Déficit Mínimo
74	si	si	69	217,21	...	4	1	Déficit Mínimo

Tabla 4.8: Actualización del dataset incorporando variable NIHSS_alta_cat

Crear columna para NIHSS_alta_ESTABLE_O_GRAVE Esta columna nos ayudará a observar si el paciente está estable o crítico. Nos interesa saber el pronóstico del paciente, por eso es necesaria una variable binaria sobre el alta del paciente. Luego de obtener la nueva columna, se comparará con las otras variables.

[15]:
```
dataset['NIHSS_alta_ESTABLE_O_GRAVE'] = np.
↪where(dataframe['NIHSS alta ACV'] <=6, 0, 1)


dataset['NIHSS_alta_ESTABLE_O_GRAVE'].value_counts()
```

NIHSS_alta_ESTABLE_O_GRAVE	Cantidad
0	26
1	20

Tabla 4.9: Clasificación binaria para la variable NIHSS_alta_cat

Existen 26 paciente estables y 20 pacientes no en buen estado medido según la tabla 4.9.

[16]:
```
dataset.head(5)
```

	HTA	DIABETES	EDAD	...	NIHSS alta ACV	NIHSS_alta_cat	NIHSS_alta_ESTABLE_O_GRAVE
0	DESCONOCIDO	DESCONOCIDO	53	...	42	Grave	1
1	si	si	54	...	0	Sin Déficit	0
2	si	si	78	...	2	Leve (Trombolisando)	0
. . .	. . .	. . .	. . .	...	. . .	. . .	. . .
72	si	si	79	...	2	Leve (Trombolisando)	0
73	si	si	54	...	1	Déficit Mínimo	0
74	si	si	69	...	1	Déficit Mínimo	0

Tabla 4.10: Actualización dataset con clasificación binaria

Se demuestra en la tabla 4.10 que si en la escala de NIHSS el paciente se encuentra con una valor leve o sin déficit obtendrá el valor 0 binario, al contrario todos los demás obtendrán el valor 1.

Análisis de las escalas de INICIO del ACV

NIHSS INICIO ACV Esta variable la transformaremos a la categoría que nos ofrece la literatura con sus 6 estados.

[17]:
```
# Análisis NIHSS INICO ACV
print(f'NIHSS INICO ACV Variable min: {dataset["NIHSS INICO␣
 ↪ACV"].min()}')
print(f'NIHSS INICO ACV Variable max: {dataset["NIHSS INICO␣
 ↪ACV"].max()}')
print(f'NIHSS INICO ACV Variable: {dataset["NIHSS INICO␣
 ↪ACV"].nunique()}')
```

```
NIHSS INICO ACV Variable min: 0.0
NIHSS INICO ACV Variable max: 21.0
```

```
NIHSS INICO ACV Variable: 14
```

[18]:
```
# Realizamos la clasificación de la escala
condicionesINICIO = [
    (dataset['NIHSS INICO ACV'] == 0),
    (dataset['NIHSS INICO ACV'] == 1),
    (dataset['NIHSS INICO ACV'] >= 2) & (dataset['NIHSS␣
 ↪INICO ACV'] <= 5),
    (dataset['NIHSS INICO ACV'] >= 6) & (dataset['NIHSS␣
 ↪INICO ACV'] <= 15),
    (dataset['NIHSS INICO ACV'] >= 16) & (dataset['NIHSS␣
 ↪INICO ACV'] <= 20),
    (dataset['NIHSS INICO ACV'] > 20),
]
valoresINICIO = ["Sin Déficit", "Déficit Mínimo", "Leve␣
 ↪(Trombolisando)", "Moderado (Buen Pronóstico)", "Déficit␣
 ↪Importante", "Grave"]
dataset['NIHSS_INICIO_cat'] = np.select(condicionesINICIO,␣
 ↪valoresINICIO)
dataset['NIHSS_INICIO_cat'].value_counts()
```

[18]:
```
Leve (Trombolisando)          28
Moderado (Buen Pronóstico)    12
Déficit Mínimo                 3
Grave                          1
Déficit Importante             1
Sin Déficit                    1
Name: NIHSS_INICIO_cat, dtype: int64
```

[19]:
```
# Para gráficos matpltlib
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
from matplotlib import style


sns.catplot(y="NIHSS_INICIO_cat",␣
 ↪hue="NIHSS_alta_ESTABLE_O_GRAVE", kind="count",
            palette="Set1", edgecolor=".9",
            data=dataset);
```

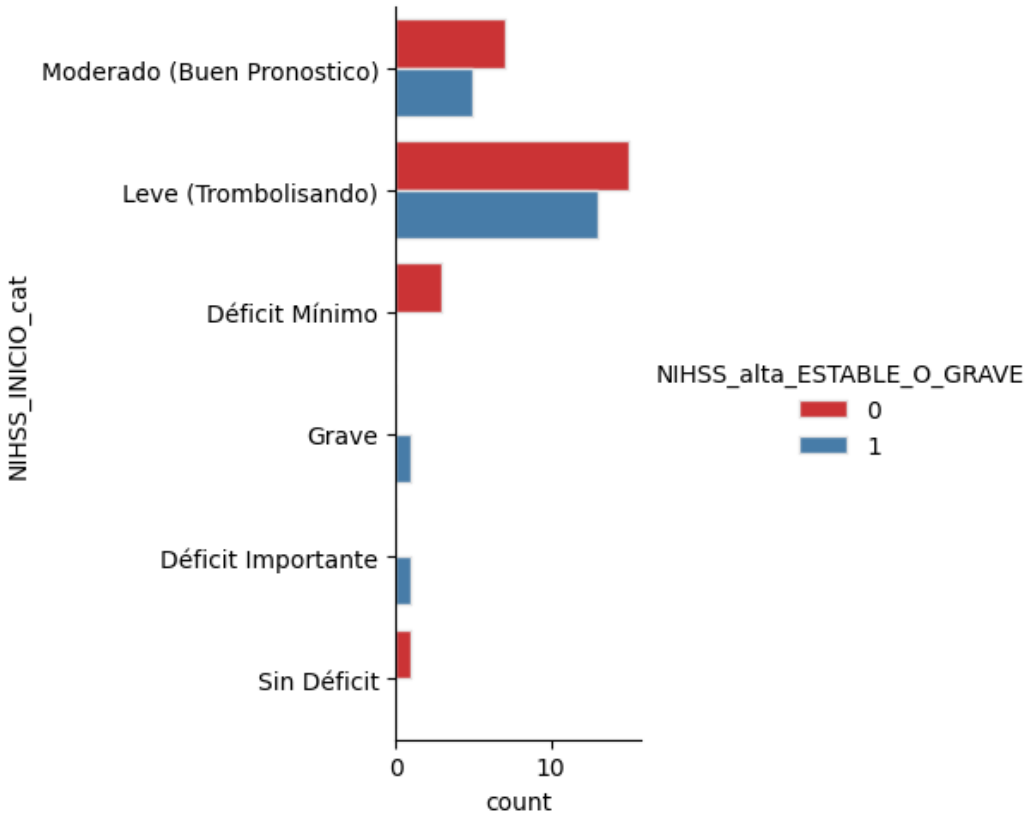

Figura 4.2: Análisis de NIHSS estable o grave en la escala NIHSS inicial

La realización de la escala fue acorde a los mismos parámetros de la variable objetivo, solo que en este caso dejamos de lado la variable binaria.

Se observa de la figura 4.2 que el valor mínimo con el que llegó un paciente fue de 0, encontrándose Sin Déficit neurológico y el máximo fue de 21 puntos con un Déficit importante. Además, se aprecia en la figura 4.2, que los pronósticos moderado y leve son los que más abundan en la muestra; asimismo se observa que esos pacientes al salir de alta, en su mayoría, tienen un buen pronóstico según la escala de NIHSS.

GLASGOW AL INICIO ACV La escala de Glasgow se divide en tres grupos puntuables de manera independiente que evalúan la apertura de ojos sobre 4 puntos, la respuesta verbal sobre 5 y la motora sobre 6, siendo la puntuación máxima y normal 15 y la mínima 3 (The status of the Glasgow Coma Scale). Se considera traumatismo craneoencefálico leve al que presenta un Glasgow de 15 a 13 puntos, moderado de 12 a 9 y grave menor o igual a 8 (Cien escalas de interés en Neurología. Prous Science,2001). A continuación, procesaremos la escala.

[20]:
```
# Análisis GLASGOW AL INICO ACV
print(f'GLASGOW AL INICO ACV Variable min:␣
 ↪{dataset["GLASGOW AL INICO ACV"].min()}')
print(f'GLASGOW AL INICO ACV Variable max:␣
 ↪{dataset["GLASGOW AL INICO ACV"].max()}')
print(f'GLASGOW AL INICO ACV Variable: {dataset["GLASGOW AL␣
 ↪INICO ACV"].nunique()}')
```

```
GLASGOW AL INICO ACV Variable min: 0.0
GLASGOW AL INICO ACV Variable max: 15.0
GLASGOW AL INICO ACV Variable: 7
```

[21]:
```
# Realizamos la clasificación de la escala
dataset['GLASGOW_cat'] = pd.cut(dataset['GLASGOW AL INICO␣
 ↪ACV'], bins=[-1, 8, 12, 15], labels=['Grave', 'Moderado',␣
 ↪'Leve'])
dataset['GLASGOW_cat'].unique()
```

[21]:
```
['Moderado', 'Leve', 'Grave']
Categories (3, object): ['Grave' < 'Moderado' < 'Leve']
```

[22]:
```
sns.catplot(y="GLASGOW_cat",␣
 ↪hue="NIHSS_alta_ESTABLE_O_GRAVE", kind="count",
            palette="Set1", edgecolor=".9",
            data=dataset);
```

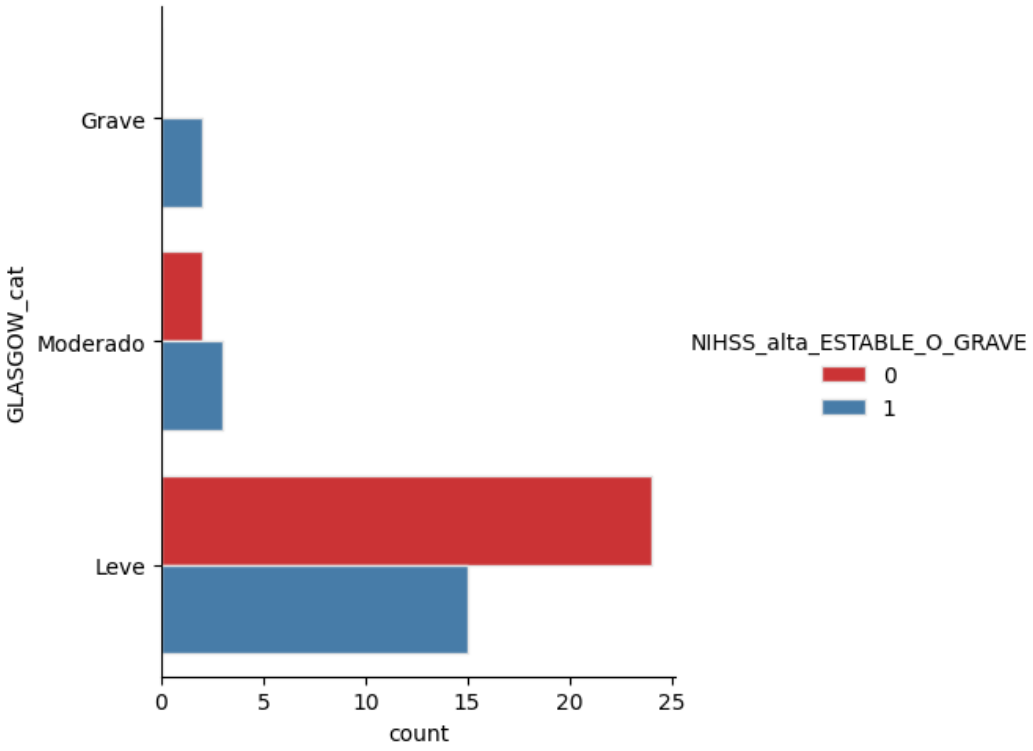

Figura 4.3: Análisis de NIHSS en la escala de Glasgow inicial

Se observa en la Figura 4.3 que el valor mínimo con el que llegó un paciente fue de 0, encontrándose grave neurológico, y el máximo con 15 puntos, encontrándose leve. Además se aprecia, en el gráfico, que el pronóstico leve es el que abunda en esta variable, asimismo se demuestra que esos pacientes al salir de alta, en su mayoría, tienen un buen pronóstico según la escala de NIHSS.

Análisis de otras variables médicas

Las siguientes clasificaciones se obtuvieron de Medline Plus [48], que es un servicio informativo de salud en líniea, el cual posee escalas para la clasificación de las variables que veremos.

Conteo de Glóbulos Blancos Los glóbulos blancos son parte del sistema inmunológico del cuerpo y ayudan a combatir infecciones y otras enfermedades. Los registraremos en 3 categorías.

[23]:
```
# Análisis CONTEO G.B.
print(f'CONTEO G.B. Variable min: {dataset["CONTEO G.B."].
 ↪min()}')
print(f'CONTEO G.B. Variable max: {dataset["CONTEO G.B."].
 ↪max()}')
print(f'CONTEO G.B. Variable: {dataset["CONTEO G.B."].
 ↪nunique()}')
```

```
CONTEO G.B. Variable min: 3.76
CONTEO G.B. Variable max: 41.9
CONTEO G.B. Variable: 38
```

[24]:
```
# Realizamos la clasificación de la escala
dataset['CONTEO G.B._cat'] = pd.cut(dataset['CONTEO G.B.'],␣
 ↪bins=[0, 4.5, 10,1000], labels=['Bajo', 'Normal', 'Alto'])
dataset['CONTEO G.B._cat'].unique()
```

[24]:
```
['Alto', 'Normal', 'Bajo']
Categories (3, object): ['Bajo' < 'Normal' < 'Alto']
```

[25]:
```
sns.catplot(y="CONTEO G.B._cat",␣
 ↪hue="NIHSS_alta_ESTABLE_O_GRAVE", kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

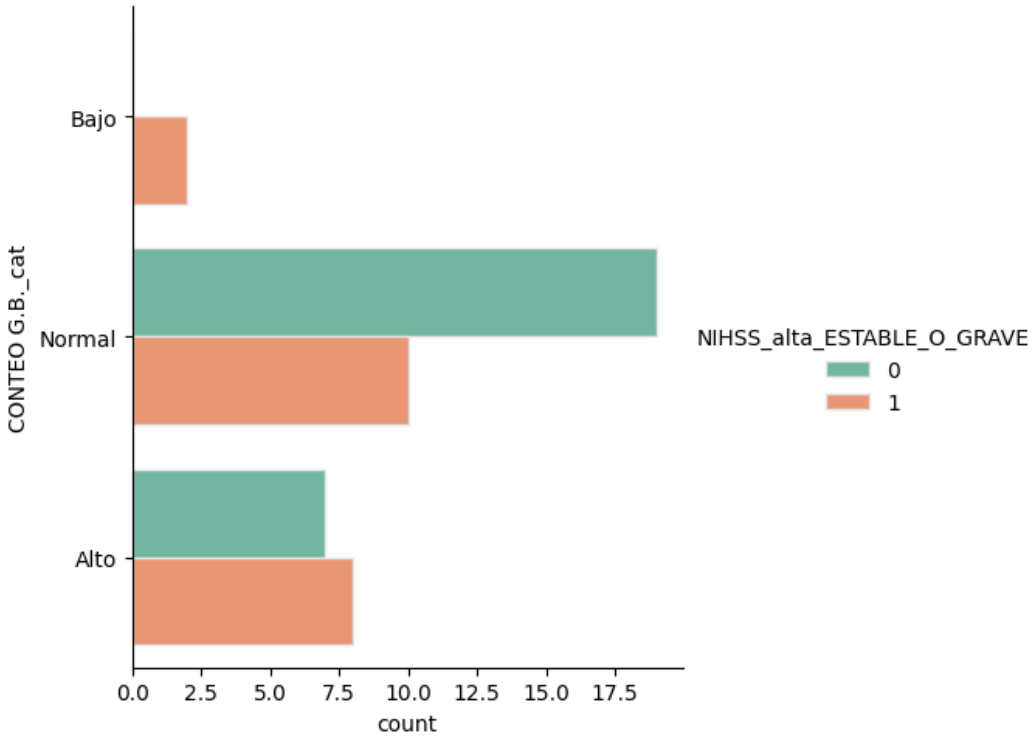

Figura 4.4: Análisis de NIHSS en el Conteo de Globulos Blancos

Se observa en la Figura 4.4 que la mayoría de los pacientes están en la categoría normal y ellos tienen, en más del 50 %, un buen pronóstico.

INR El INR es un tipo de cálculo que se basa en los resultados de las pruebas de tiempo de protrombina, que indica cuán adecuado es el proceso de coágulación de un individuo. Este examen ayuda a determinar qué tipo de medicamentos se pueden administrar o averiguar la causa de los coágulos sanguíneos anormales.

[26]:
```
# Análisis INR
print(f'INR Variable min: {dataset["INR"].min()}')
print(f'INR Variable max: {dataset["INR"].max()}')
print(f'INR Variable: {dataset["INR"].nunique()}')
```

```
INR Variable min: 0.89
INR Variable max: 3.08
INR Variable: 26
```

[27]:
```
dataset['INR_cat'] = pd.cut(dataset['INR'], bins=[0.0, 2,␣
↪4,1000], labels=['Riesgo', 'Normal', 'No Anticoagula'])
dataset['INR_cat'].unique()
```

[27]:
```
['Riesgo', 'Normal']
Categories (3, object): ['Riesgo' < 'Normal' < 'No␣
↪Anticoagula']
```

[28]:
```
sns.catplot(y="INR_cat", hue="NIHSS_alta_ESTABLE_O_GRAVE",␣
↪kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

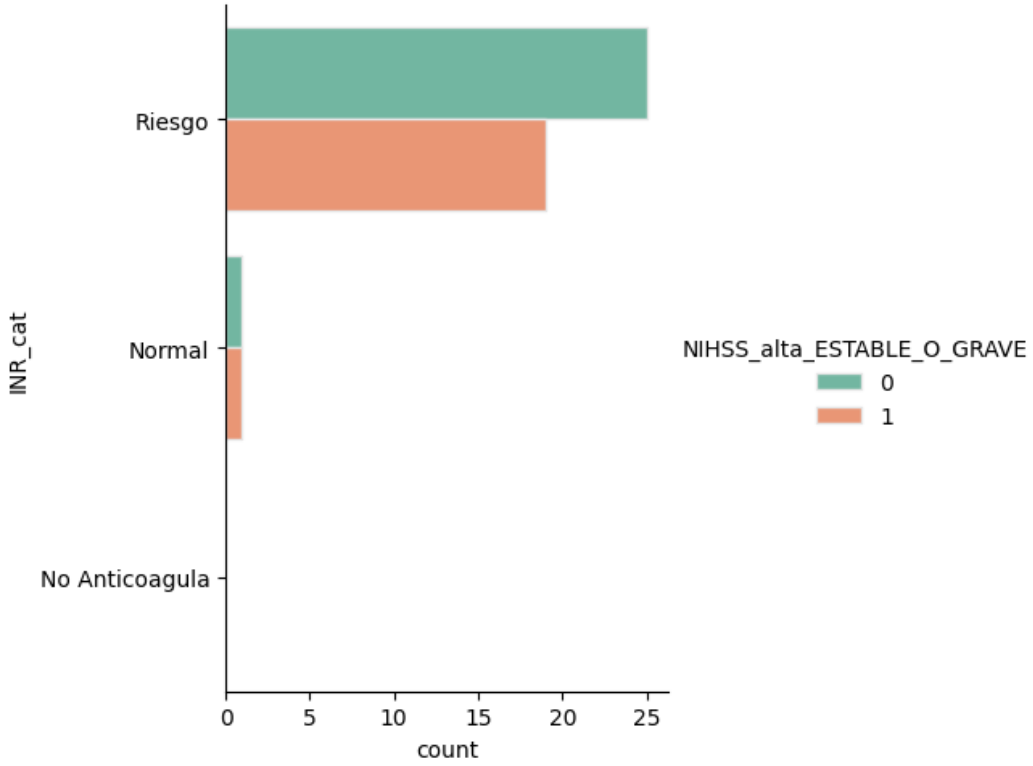

Figura 4.5: Análisis de NIHSS en el INR

Se observa en la Figura 4.5 que la mayoría de los pacientes están en la categoría de Riesgo, aunque un grupo importante tiene un pronóstico favorable. En la categoría Normal no se sabe si el paciente obtuvo un buen o mal pronóstico.

Triglicéridos Los triglicéridos son el principal tipo de grasa que se transporta en la sangre para obtener energía o se almacena en las células del cuerpo según las necesidades energéticas entre comidas.

[29]:
```
# Análisis TRIGLICERIDOS
print(f'TRIGLICERIDOS Variable min: {dataset["TRIGLICERIDOS"].min()}')
print(f'TRIGLICERIDOS Variable max: {dataset["TRIGLICERIDOS"].max()}')
print(f'TRIGLICERIDOS Variable: {dataset["TRIGLICERIDOS"].nunique()}')
```

```
TRIGLICERIDOS Variable min: 57.0
TRIGLICERIDOS Variable max: 232.0
TRIGLICERIDOS Variable: 31
```

[30]:
```
dataset['TRIGLICERIDOS_cat'] = pd.cut(dataset['TRIGLICERIDOS'], bins=[0, 150, 200, 500, 10000], labels=['Normal', 'Límite alto', 'Alto', 'Muy Alto'])
dataset['TRIGLICERIDOS_cat'].unique()
```

[30]:
```
['Normal', 'Límite alto', 'Alto']
Categories (4, object): ['Normal' < 'Límite alto' < 'Alto' < 'Muy Alto']
```

[31]:
```
sns.catplot(y="TRIGLICERIDOS_cat", hue="NIHSS_alta_ESTABLE_O_GRAVE", kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

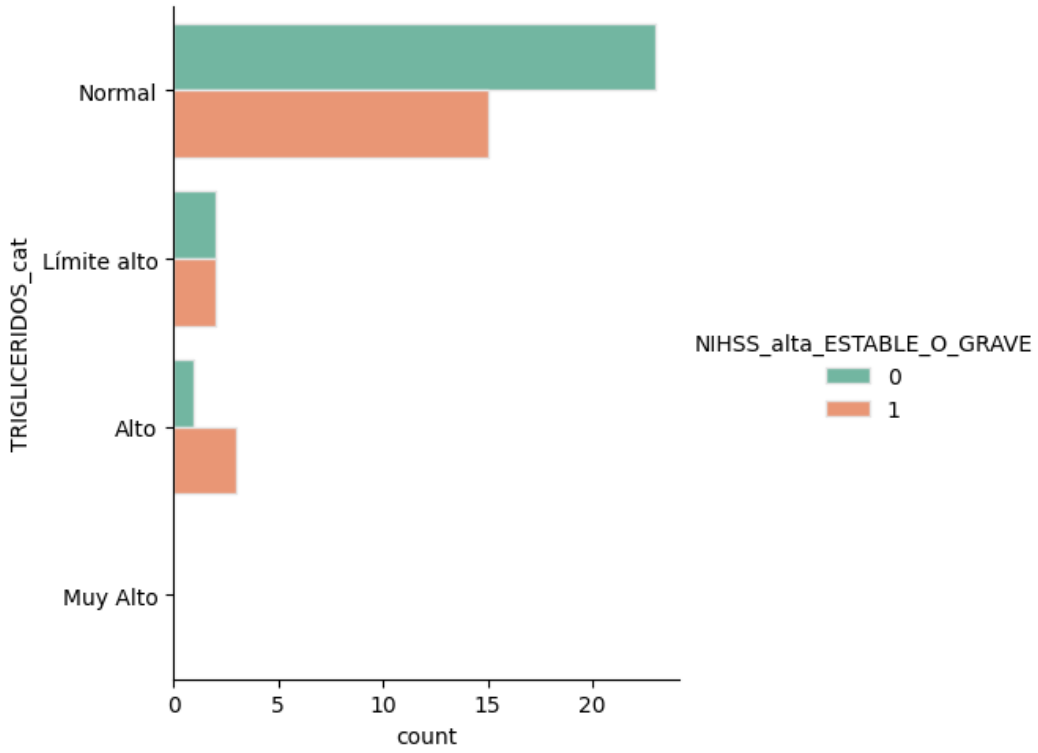

Figura 4.6: Análisis de NIHSS en los Trigliceridos

En la figura 4.6 muestra que la mayoría de los pacientes están en la categoría Normal, donde un grupo importante tiene un pronóstico favorable. En la categoría Alto un gran porcentaje tiene un mal pronóstico con el alta del ACV.

Colesterol Total El colesterol es una grasa natural que se encuentra en todas las células del cuerpo y es necesaria para que el cuerpo funcione correctamente. El colesterol se produce en el hígado en su mayoría, aunque también se puede obtener de algunos alimentos.

[32]:
```
# Análisis COL. TOTAL
print(f'COL. TOTAL Variable min: {dataset["COL. TOTAL"].
 ↪min()}')
print(f'COL. TOTAL Variable max: {dataset["COL. TOTAL"].
 ↪max()}')
print(f'COL. TOTAL Variable: {dataset["COL. TOTAL"].
 ↪nunique()}')
```

```
COL. TOTAL Variable min: 85.0
COL. TOTAL Variable max: 342.0
COL. TOTAL Variable: 34
```

[33]:
```
dataset['COL. TOTAL_cat'] = pd.cut(dataset['COL. TOTAL'],␣
↪bins=[0, 150, 400, 1000, 100000], labels=['Normal',␣
↪'Límite alto', 'Alto', 'Muy Alto'])
dataset['COL. TOTAL_cat'].unique()
```

[33]:
```
['Límite alto', 'Normal']
Categories (4, object): ['Normal' < 'Límite alto' < 'Alto'␣
↪< 'Muy Alto']
```

[34]:
```
sns.catplot(y="COL. TOTAL_cat",␣
↪hue="NIHSS_alta_ESTABLE_O_GRAVE", kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

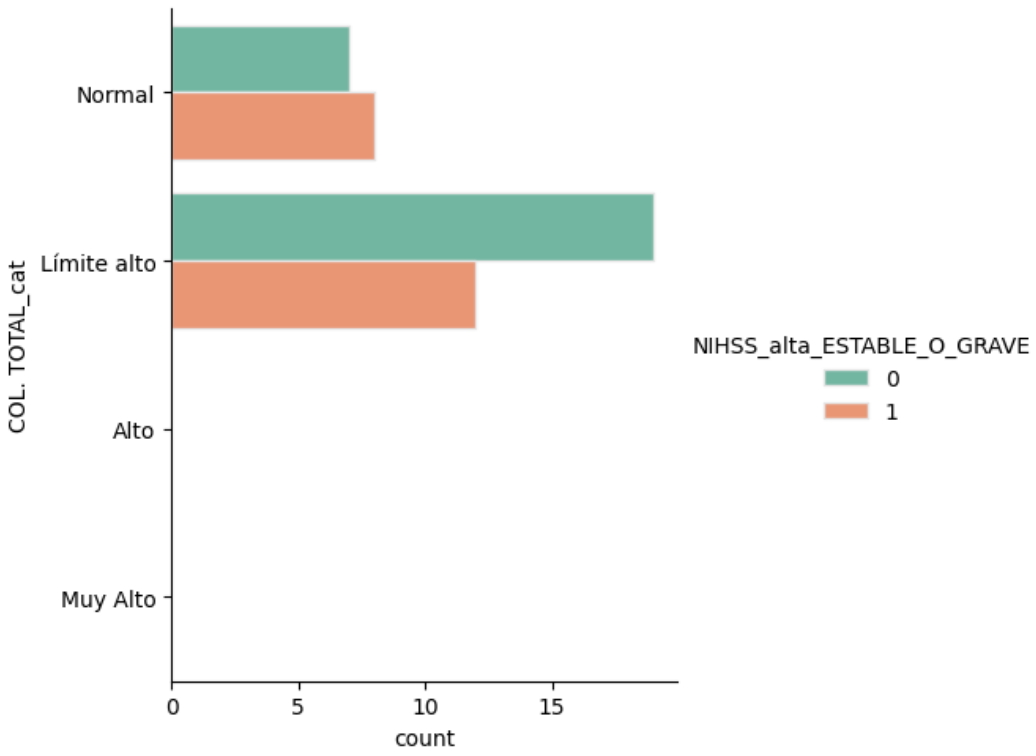

Figura 4.7: Análisis de NIHSS en el Colesterol

La figura 4.7 se muestra que la mayoría de los pacientes están en la categoría Limite alto, donde un grupo importante tiene un pronóstico favorable.

Glucosa La glucosa es la principal azúcar en la sangre. Proviene de los alimentos que se consume y es la principal fuente de energía. La sangre transporta glucosa a cada célula del cuerpo para obtener energía.

[35]:
```
# Análisis EDAD
print(f'GLUCOSA Variable min: {dataset["GLUCOSA"].min()}')
print(f'GLUCOSA Variable max: {dataset["GLUCOSA"].max()}')
print(f'GLUCOSA Variable: {dataset["GLUCOSA"].nunique()}')
```

```
GLUCOSA Variable min: 82.61
GLUCOSA Variable max: 359.42
GLUCOSA Variable: 39
```

[36]:
```
dataset['GLUCOSA_cat'] = pd.cut(dataset['GLUCOSA'],␣
↪bins=[0, 90, 160, 230, 3000], labels=['Bajo', 'Normal',␣
↪'Alto', 'Muy Alto'])
dataset['GLUCOSA_cat'].unique()
```

[36]:
```
['Normal', 'Muy Alto', 'Alto', 'Bajo']
Categories (4, object): ['Bajo' < 'Normal' < 'Alto' < 'Muy␣
↪Alto']
```

[37]:
```
sns.catplot(y="GLUCOSA_cat",␣
↪hue="NIHSS_alta_ESTABLE_O_GRAVE", kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

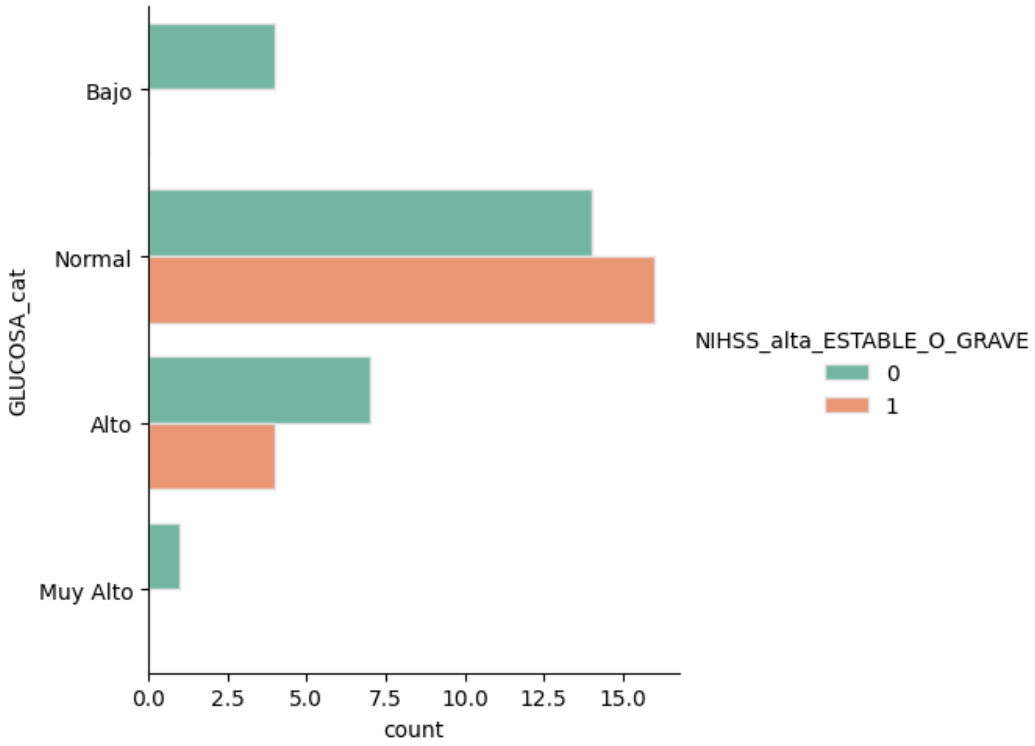

Figura 4.8: Análisis de NIHSS en la Glucosa

En la figura 4.8 muestra que la mayoría de los pacientes están en la categoría Normal, aunque un grupo importante tiene un pronóstico poco favorable en el alta. En la categoría Alta un gran porcentaje tiene un buen pronóstico con el alta del ACV.

Edad La edad juega un rol importante para determinar el estado en que se puede encontrar el cuerpo para responder a una enfermedad o tratamiento, es por eso que esta es una de las variables que debe estar presente.

[38]:
```
# Análisis EDAD
print(f'EDAD Variable min: {dataset["EDAD"].min()}')
print(f'EDAD Variable max: {dataset["EDAD"].max()}')
print(f'EDAD Variable: {dataset["EDAD"].nunique()}')
```

```
EDAD Variable min: 38.0
EDAD Variable max: 90.0
EDAD Variable: 28
```

[39]:
```
dataset['EDAD_cat'] = pd.cut(dataset['EDAD'], bins=[0, 13,␣
↪18, 45, 60, 100], labels=['Niño', 'Adolecente', 'Adulto␣
↪Joven', 'Adulto', 'Anciano'])
dataset['EDAD_cat'].unique()
```

[39]:
```
['Adulto', 'Anciano', 'Adulto Joven']
Categories (5, object): ['Niño' < 'Adolecente' < 'Adulto␣
↪Joven' < 'Adulto' <
'Anciano']
```

[40]:
```
sns.catplot(y="EDAD_cat", hue="NIHSS_alta_ESTABLE_O_GRAVE",␣
↪kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

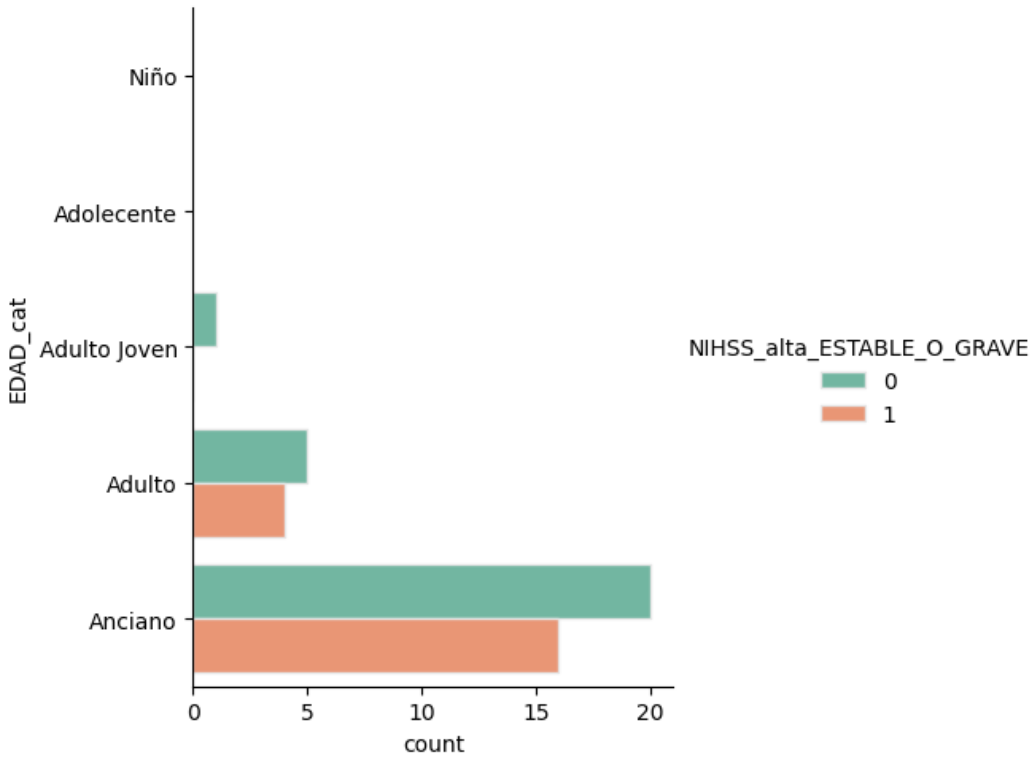

Figura 4.9: Análisis de NIHSS en EDAD

En la figura 4.9 se muestra que la mayoría de los pacientes son ancianos y que un grupo mayoritario de esa categoría tiene un pronóstico favorable en el alta. La

categoría Adulto es la segunda más grande y resultado es similar al de los ancianos en términos de pronósticos.

Diabetes La diabetes es una enfermedad crónica que afecta la forma en que el cuerpo convierte los alimentos en energía. El cuerpo descompone la mayor parte de los alimentos que ingiere en azúcar (también llamada glucosa) y la libera en la sangre. Puede existir exceso de azúcar o ausencia de azúcar en la sangre.

[41]:
```
# Análisis DIABETES


print(f'DIABETES Variable min: {dataset["DIABETES"].min()}')
print(f'DIABETES Variable max: {dataset["DIABETES"].max()}')
print(f'DIABETES Variable: {dataset["DIABETES"].nunique()}')
```

```
DIABETES Variable min: DESCONOCIDO
DIABETES Variable max: si
DIABETES Variable: 3
```

[42]:
```
sns.catplot(y="DIABETES", hue="NIHSS_alta_ESTABLE_O_GRAVE",␣
↪kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

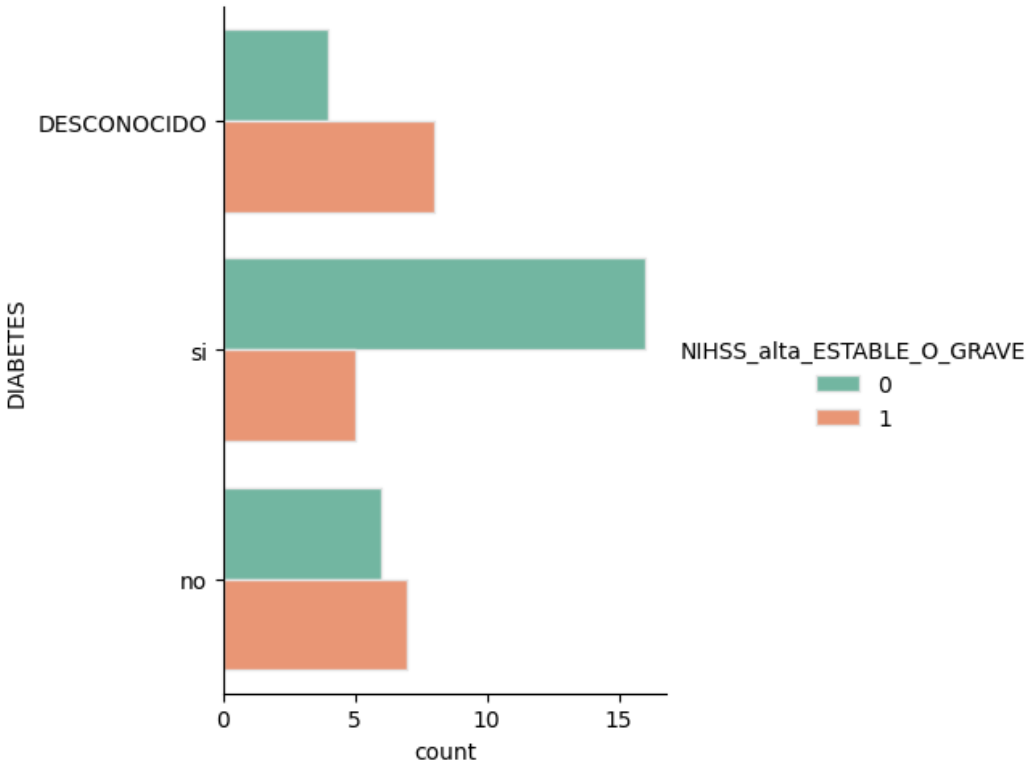

Figura 4.10: Análisis de NIHSS en pacientes en la Diabetes

Tal como se observa en la Figura 4.10, respecto de esta variable solo se indica si el paciente padecía o no diabetes. A los pacientes que no poseían este dato, se les asoció el estado "DESCONOCIDO".

La Figura 4.10 muestra que la mayoria de los pacientes que padecen diabetes tienen un buen pronóstico al alta. El estado DESCONOCIDO posee un pronóstico poco favorable al alta.

Hipertensión La presión arterial es la fuerza con que la sangre se desplaza y golpea las paredes de las arterias. La hipertensión corresponde a la presión alta y es sinónimo de enfermedades cardiacas.

[43]:
```
# Análisis DIABETES


print(f'HTA Variable min: {dataset["HTA"].min()}')
print(f'HTA Variable max: {dataset["HTA"].max()}')
print(f'HTA Variable: {dataset["HTA"].nunique()}')
```

```
HTA Variable min: DESCONOCIDO
HTA Variable max: si
HTA Variable: 3
```

[44]:
```
sns.catplot(y="HTA", hue="NIHSS_alta_ESTABLE_O_GRAVE",␣
↪kind="count",
            palette="Set2", edgecolor=".9",
            data=dataset);
```

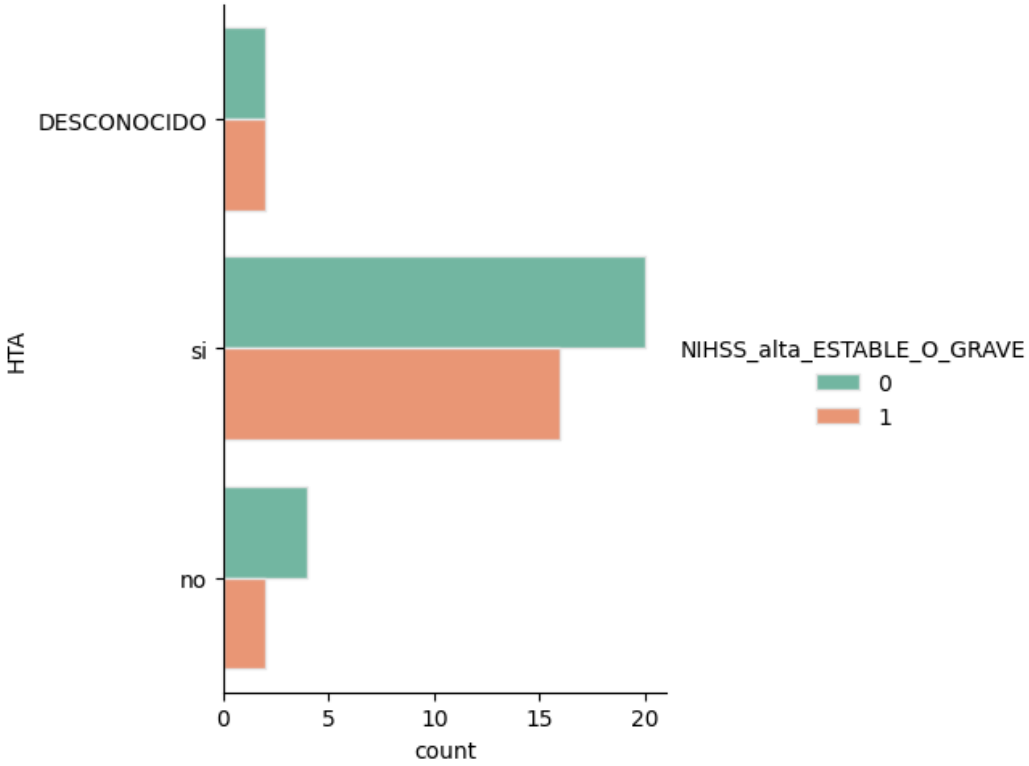

Figura 4.11: Análisis de NIHSS en la Hipertensión

En la Figura 4.11 se muestra que un número importante de pacientes poseen hipertensión y sufrieron un ACV; en su mayoría, los pacientes tienen un pronóstico favorable.

4.5.6. Binary Encoding (Codificación binaria)

La Binary Encoding es un método de codificación de variables categóricas. Se utiliza a menudo en el ML porque muchos algoritmos funcionan mejor con variables

numéricas y el Binary Encoding permite representar de manera efectiva las variables categóricas como números.

Su función es representar una variable categórica que posee dos estados con variables de 0 ó 1 (binarias)[49].

[45]:
```
binary_cols = [col for col in dataset.columns if␣
↪dataset[col].dtype not in [int, float]
               and dataset[col].nunique() == 2]

def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.
↪fit_transform(dataframe[binary_col])
    return dataframe


binary_cols
```

[45]:
```
['INR_cat', 'COL. TOTAL_cat']
```

[46]:
```
from sklearn.preprocessing import LabelEncoder,␣
↪StandardScaler, RobustScaler

for col in binary_cols:
    label_encoder(dataset, col)

dataset.head()
```

	INR_cat	COL. TOTAL_cat
0	1	0
1	1	0
2	1	0
. . .	. . .	. . .
72	1	1
73	1	0
74	1	0

Tabla 4.11: Binary Encoding con las variables que actualmente poseen dos estados activos

4.5.7. Label Encoding (Codificación de etiquetas)

El Label Encoding es un método de codificación de variables categóricas utilizado en ML, porque muchos algoritmos funcionan mejor con variables numéricas y el Label Encoding permite representar de manera efectiva las variables categóricas como números. Aunque este método puede dar lugar a una interpretación errónea en las relaciones entre categorías, en esos casos es mejor utilizar otras técnicas de clasificación como lo son la Binary Encoding.

Su función asigna a cada categoría única de una variable un número entero único y se reemplaza la variable categórica por los números correspondientes [49].

[47]:
```
labelencoder = LabelEncoder()
```

[48]:
```
dataset["NIHSS_alta_cat"] = labelencoder.
 ↪fit_transform(dataset["NIHSS_alta_cat"])
dataset["GLASGOW_cat"] = labelencoder.
 ↪fit_transform(dataset["GLASGOW_cat"])
dataset["CONTEO G.B._cat"] = labelencoder.
 ↪fit_transform(dataset["CONTEO G.B._cat"])
dataset["TRIGLICERIDOS_cat"] = labelencoder.
 ↪fit_transform(dataset["TRIGLICERIDOS_cat"])
dataset["GLUCOSA_cat"] = labelencoder.
 ↪fit_transform(dataset["GLUCOSA_cat"])
```

```
dataset["EDAD_cat"] = labelencoder.
 ↪fit_transform(dataset["EDAD_cat"])
dataset["HTA"] = labelencoder.fit_transform(dataset["HTA"])
dataset["DIABETES"] = labelencoder.
 ↪fit_transform(dataset["DIABETES"])


dataset.head(5)
```

	...	GLASGOW_cat	CONTEO G.B._cat	TRIGLICERIDOS_cat	GLUCOSA_cat	EDAD_cat
0	...	2	0	2	3	0
1	...	1	2	2	3	0
2	...	1	2	2	2	2
. . .	...	. . .	. . .	. . .	. . .	. . .
72	...	1	2	2	3	2
73	...	1	2	2	0	0
74	...	0	0	0	0	2

Tabla 4.12: Label Encoding con las variables que actualmente poseen alguna etiqueta

En las variables Diabetes e Hipertensión el estado 2 representa “si” y el “no” es representado con un estado 1.

4.5.8. One-Hot Encoding

Para las variables categóricas donde no existe tal relación ordinal, la codificación de enteros no es suficiente. De hecho, usar esta codificación y permitir que el modelo asuma un ordenamiento natural entre categorías puede dar como resultado un desempeño deficiente o resultados inesperados [50]. En este caso, se puede aplicar una codificación one-hot a la representación de enteros. Aquí es donde se elimina la variable codificada entera y se agrega una nueva variable binaria para cada valor entero único. En el ejemplo de la variable ” color “, hay 3 categorías y, por lo tanto, se necesitan 3 variables binarias. Se coloca un valor”1” en la variable binaria para el color y valores “0” para los otros colores.

[49]:
```
# One Hot Encoding
dataset = pd.get_dummies(dataset)
```

```
# Como son muchas columnas, muestro todas
pd.options.display.max_columns = 0

dataset.head(5)
```

	...	NIHSS_INICIO_cat_Déficit Importante	NIHSS_INICIO_cat_Déficit Mínimo	NIHSS_INICIO_cat_Grave	NIHSS_INICIO_cat_Leve (Trombolisando)	NIHSS_INICIO_cat_Moderado (Buen Pronóstico)	NIHSS_INICIO_cat_Sin Déficit
0	...	0	0	0	0	1	0
1	...	0	0	0	0	1	0
2	...	0	0	0	1	0	0
.	...	.	.	.	.	.	.
.		.	.	.	.	.	.
.		.	.	.	.	.	.
72	...	0	0	0	1	0	0
73	...	0	1	0	0	0	0
74	...	0	0	0	1	0	0

Tabla 4.13: One-Hot Encoding con las variables para clasificación

4.6. Análisis de datos de entrada

Esta sección corresponde al tercer paso de la metodología, el cual consiste en la construcción de gráficos que permitan, a partir de su visualización, descubrir alguna tendencia en los datos u otro antecedente relevante. Crearemos un datasets auxiliar, este tiene como fin observar los datos originales y los que incorporamos en la sección anterior.

[50]:
```
# Mostramos las variables que posee la base de datos␣
 ↪actualmente
columns_names = dataset.columns.values
print(columns_names)
```

```
['HTA' 'DIABETES' 'EDAD' 'GLUCOSA' 'COL. TOTAL'␣
 ↪'TRIGLICERIDOS' 'INR'
 'CONTEO G.B.' 'GLASGOW AL INICO ACV' 'NIHSS INICO ACV'␣
 ↪'NIHSS alta ACV'
 'NIHSS_alta_cat' 'NIHSS_alta_ESTABLE_O_GRAVE' 'GLASGOW_cat'
 'CONTEO G.B._cat' 'INR_cat' 'TRIGLICERIDOS_cat' 'COL.␣
 ↪TOTAL_cat'
```

```
 'GLUCOSA_cat' 'EDAD_cat' 'NIHSS_INICIO_cat_Déficit␣
 ↪Importante'
 'NIHSS_INICIO_cat_Déficit Mínimo' 'NIHSS_INICIO_cat_Grave'
 'NIHSS_INICIO_cat_Leve (Trombolisando)'
 'NIHSS_INICIO_cat_Moderado (Buen Pronóstico)'
 'NIHSS_INICIO_cat_Sin Déficit']
```

[51]:
```
# Variables originales
columnasOriginal = ['HTA', 'DIABETES', 'EDAD', 'GLUCOSA',␣
 ↪'COL. TOTAL', 'TRIGLICERIDOS', 'INR', 'CONTEO G.B.',␣
 ↪'GLASGOW AL INICO ACV', 'NIHSS INICO ACV', 'NIHSS alta␣
 ↪ACV']
datasetOriginal = dataset[[*columnasOriginal]]


datasetOriginal.head(5)
```

	HTA	DIABETES	EDAD	GLUCOSA	...	GLASGOW AL INICO ACV	NIHSS INICO ACV	NIHSS alta ACV
0	0	0	53	137,09	...	11	14	42
1	2	2	54	119,995	...	15	6	0
2	2	2	78	359,42	...	15	5	2
. . .	. . .	. . .	. . .	. . .	...	. . .	. . .	. . .
72	2	2	79	116,99	...	15	2	2
73	2	2	54	211,58	...	15	1	1
74	2	2	69	217,21	...	0	4	1

Tabla 4.14: Variables originales

[52]:
```
# variables agregadas
columnasAgregadas = ['NIHSS_alta_cat',␣
 ↪'NIHSS_alta_ESTABLE_O_GRAVE', 'GLASGOW_cat',
 'CONTEO G.B._cat', 'INR_cat', 'TRIGLICERIDOS_cat', 'COL.␣
 ↪TOTAL_cat',
 'GLUCOSA_cat', 'EDAD_cat', 'NIHSS_INICIO_cat_Déficit␣
 ↪Importante',
 'NIHSS_INICIO_cat_Déficit Mínimo',␣
 ↪'NIHSS_INICIO_cat_Grave',
 'NIHSS_INICIO_cat_Leve (Trombolisando)',
```

```
 'NIHSS_INICIO_cat_Moderado (Buen Pronóstico)',
 'NIHSS_INICIO_cat_Sin Déficit']
datasetAgregado = dataset[[*columnasAgregadas]]


datasetAgregado.head(5)
```

	NIHSS_alta_cat	NIHSS_alta_ESTABLE_O_GRAVE	GLASGOW_cat	...	NIHSS_INICIO_cat_Leve (Trombolisando)	NIHSS_INICIO_cat_Moderado (Buen Pronóstico)	NIHSS_INICIO_cat_Sin Déficit
0	2	1	2	...	0	1	0
1	5	0	1	...	0	1	0
2	3	0	1	...	1	0	0
. . .	. . .	. . .	. . .	...	. . .	. . .	. . .
72	3	0	1	...	1	0	0
73	1	0	1	...	0	0	0
74	1	1	0	...	1	0	0

Tabla 4.15: Variables agregadas por los métodos anteriores

Se observa que ambos datasets tienen un número de variables similares y que la última posee datos más cercanos a lo binario.

4.6.1. Análisis de densidad y estimación por variable

La densidad de datos en una variable aleatoria continua describe la probabilidad relativa según la cual dicha variable aleatoria tomará determinado valor.

[53]:
```
# Análisis de densidad y estimación en los datos originales
plt.rc('legend', fontsize=7)
datasetOriginal.plot.density();
```

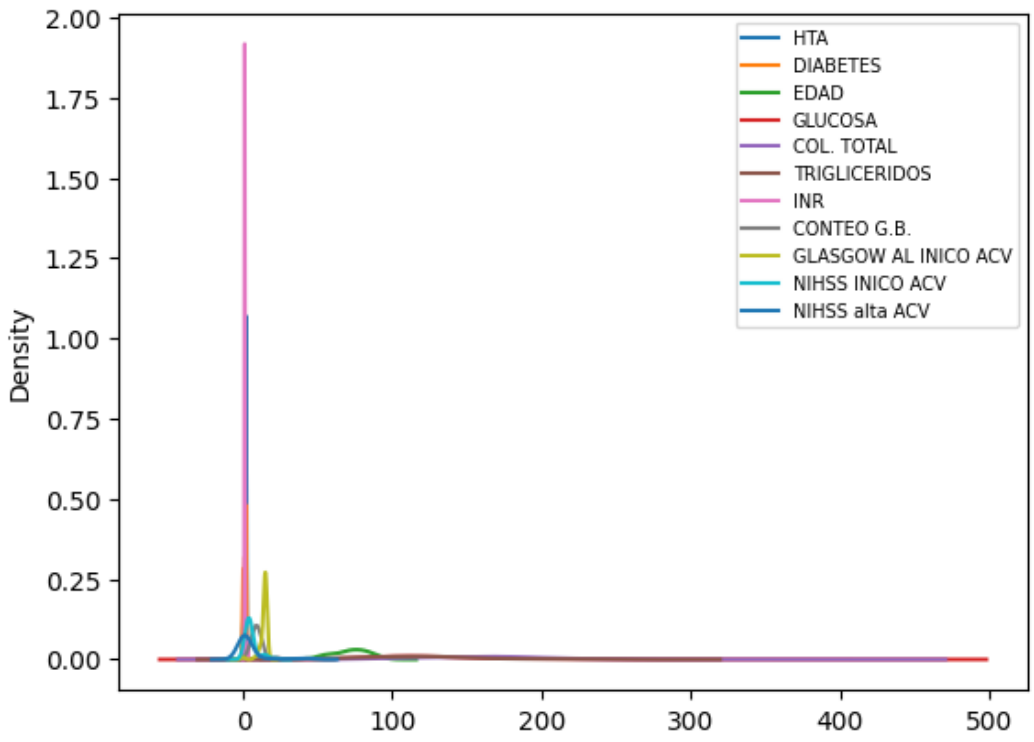

Figura 4.12: Análisis de densidad de variables originales con escalas

Como los datos son variados numéricamente, en la Figura 4.12 no se alcanza a reflejar cada variable. Si fuera necesario deberíamos analizar en este caso cada variable por separado, siendo que este no es el caso para la mayoría de las variables. A continuación, se mostrará el gráfico de densidad de algunas variables.

[54]:
```
fig,axes = plt.subplots(3, 2, figsize = (10, 10))
sns.set_style('darkgrid')
fig.suptitle("Gráfico de conteo para varias características␣
 ↪categóricas")


# Variables muy dispersas
sns.distplot(dataset['EDAD'], bins=30, color='darkred',␣
 ↪ax=axes[0, 0])
sns.distplot(dataset['GLUCOSA'], bins=30, color='darkred',␣
 ↪ax=axes[0, 1])
```

```
sns.distplot(dataset['COL. TOTAL'], bins=30,␣
 ↪color='darkred', ax=axes[1, 0])
sns.distplot(dataset['TRIGLICERIDOS'], bins=30,␣
 ↪color='darkred', ax=axes[1, 1])
sns.distplot(dataset['INR'], bins=30, color='darkred',␣
 ↪ax=axes[2, 0])
sns.distplot(dataset['CONTEO G.B.'], bins=30,␣
 ↪color='darkred', ax=axes[2, 1])

plt.show();
```

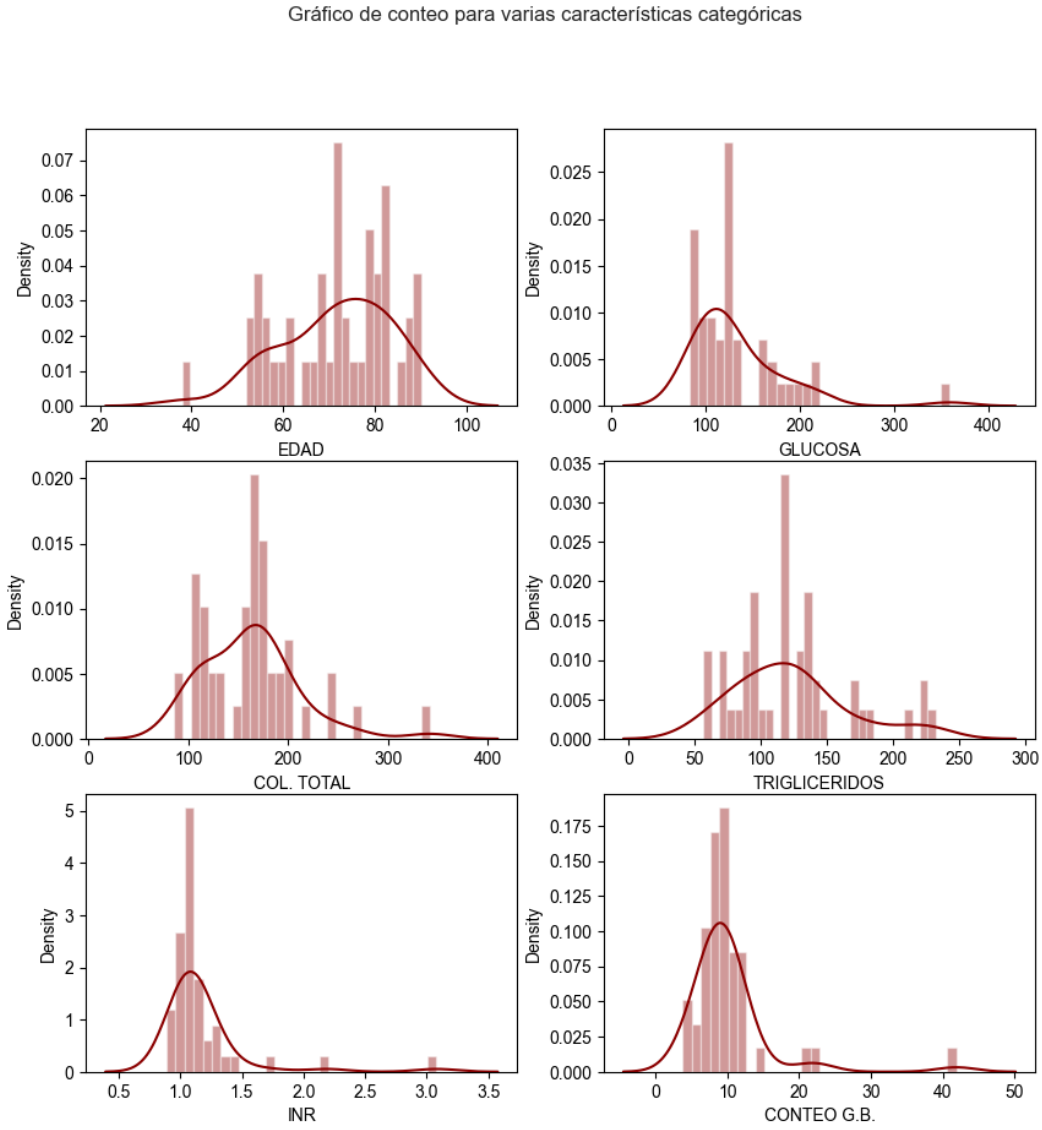

Figura 4.13: Análisis de densidad variables originales

Las tendencias de los gráficos en la Figura 4.13 están orientadas al grupo etario entre los 60 y 80 años que poseen una densidad en los datos alta.

[55]:
```
# Análisis de densidad y estimación en los datos agregados
plt.rc('legend', fontsize=7)
datasetAgregado.plot.density();
```

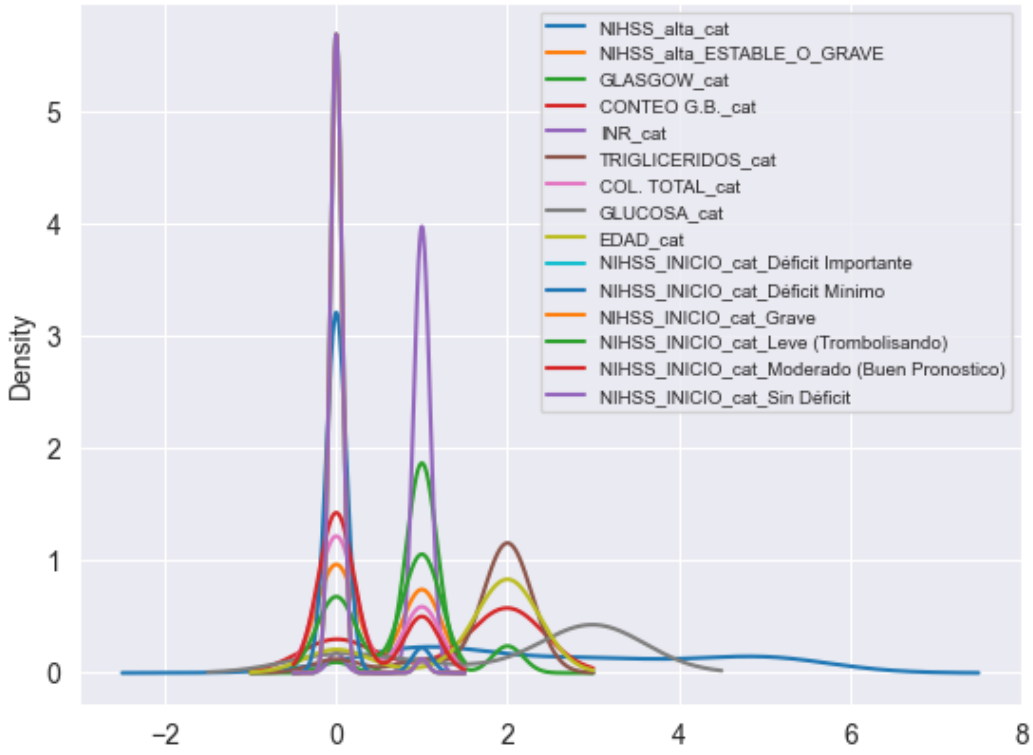

Figura 4.14: Análisis de densidad variables agregadas

La Figura 4.14 se puede interpretar de la siguiente forma:

En la variable "NIHSS_INICIO_cat_Sin Déficit" la densidad de los datos es alta en el valor 0, entendiendo que esta poseía solo dos estados, 0 para confirmar que no está presente el valor y 1 para confirmar que está presente. En otras palabras, inicialmente los pacientes sin déficit son muy pocos, según la escala NIHSS, la mayoría presenta déficit. El análisis individual no es necesario en este caso, ya que los valores están bien detallados a la vista.

4.6.2. Análisis por conteo de variables categóricas

Para mostrar la cantidad de datos presentes en algunas de las variables originales, en cada gráfico se representará con escalas de contador de cada variable.

[56]:
```
fig,axes = plt.subplots(5, figsize = (5, 16))
sns.set_style('darkgrid')
fig.suptitle("Gráfico de conteo para varias características␣
 ↪categóricas originales")

sns.countplot(ax=axes[0],data=dataset,x='HTA')
sns.countplot(ax=axes[1],data=dataset,x='DIABETES')
sns.countplot(ax=axes[2],data=dataset,x='GLASGOW AL INICO␣
 ↪ACV')
sns.countplot(ax=axes[3],data=dataset,x='NIHSS INICO ACV')
sns.countplot(ax=axes[4],data=dataset,x='NIHSS alta ACV')

plt.show()
```

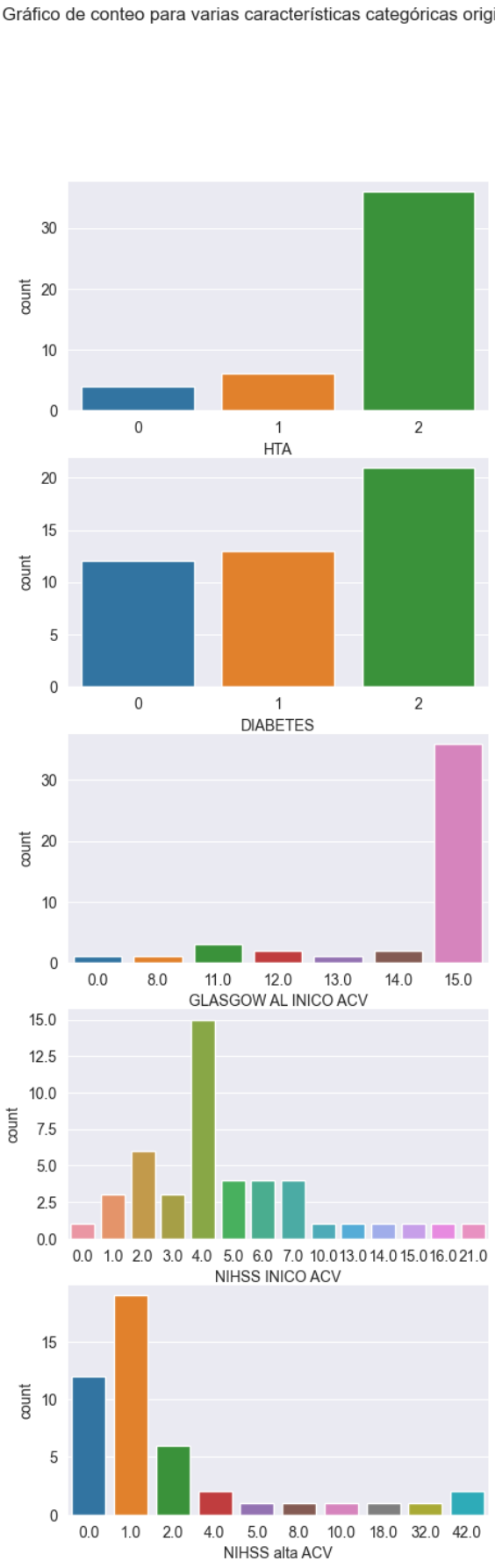

Figura 4.15: Análisis por conteo de variables categóricas

Las tasas de conteo de los pacientes en la Figura 4.15, indican una tendencia en algunos valores específicos, donde a simple vista las escalas de NIHSS y GLASGOW, en su mayoría, se encuetran con un estado con poco daño neurológico. Además, los pacientes en su mayoría, tienen Diabetes o Hipertensión.

4.6.3. Mapa de calor de variables

Los mapas de calor pueden ayudar a visualizar el impacto de muchas categorías en los valores y son una excelente opción para ver detalles cuando se trabaja con conjuntos de datos más grandes. Los mapas de calor utilizan una variedad de colores fríos y cálidos para ayudarnos a comprender qué elementos de datos generan más interés (áreas calientes) y qué elementos se ignoran (áreas frías).

[58]:
```
fig = plt.figure(figsize=(12,8))
corr = datasetOriginal.corr()
ax = sns.heatmap(corr, linewidths=.5, cmap="RdBu",␣
↪annot=True, fmt="g")
```

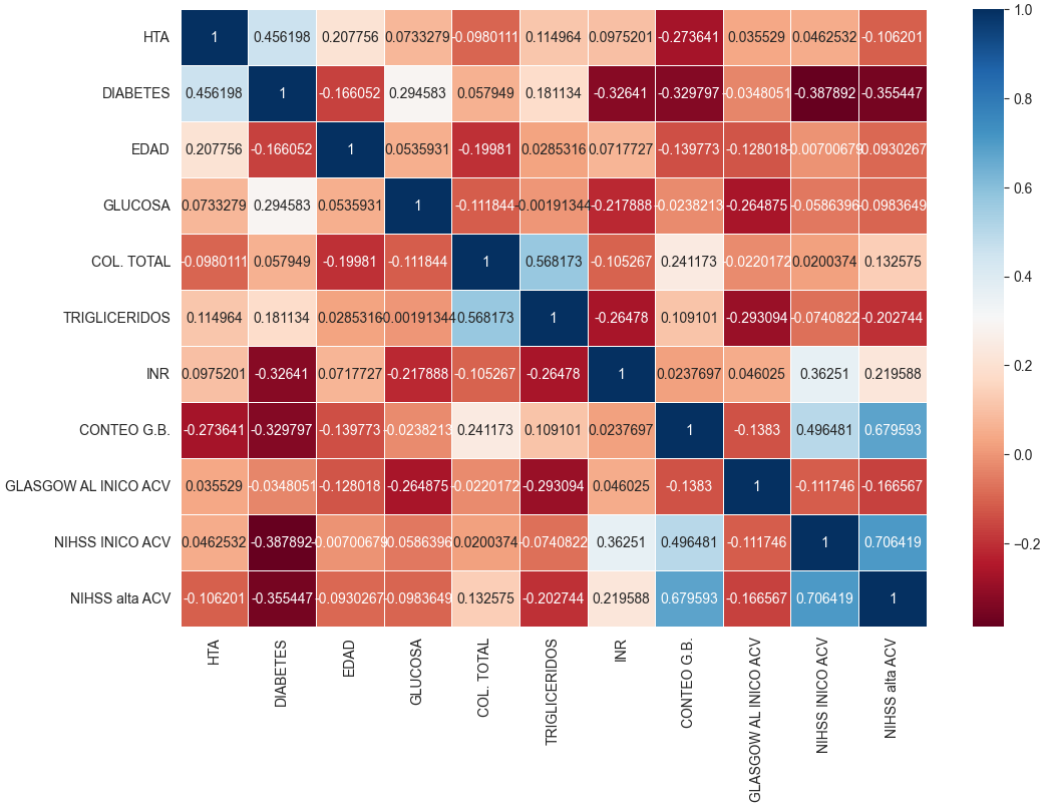

Figura 4.16: Mapa de calor de variables originales

[59]:
```
fig = plt.figure(figsize=(15,8))
corr = datasetAgregado.corr()
ax = sns.heatmap(corr, linewidths=.5, cmap="RdBu",␣
 ↪annot=True, fmt="g")
```

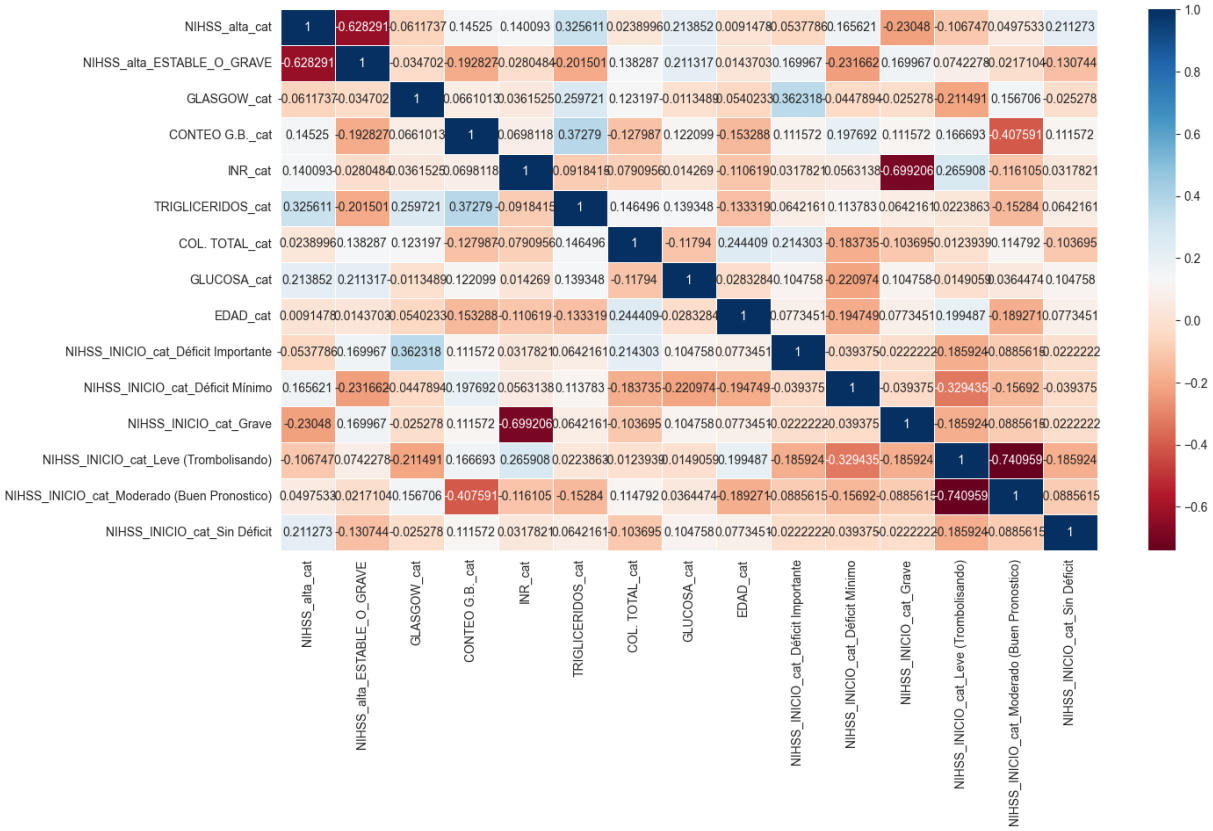

Figura 4.17: Mapa de calor de variables agregadas

El primer dataset en la Figura 4.16 muestra un alto interés en casi todas las variables tendiendo al rojizo y algunos sectores fríos, en cambio, en la Figura 4.17 el segundo dataset presenta variables más estables con estados más cercanos a la neutralidad.

4.7. Naïve Bayes - Entrenamiento del algoritmo

Como se mencionó en el Marco Teórico 2.2.5, este algoritmo es utilizado para predecir grandes volúmenes de datos. En este caso no se cuenta con un gran volumen de datos. El clasificador Naïve-Bayes aprende de los datos de entrenamiento y luego predice la clase de la instancia de prueba con la mayor probabilidad posterior. También es útil para datos dimensionales altos, ya que la probabilidad de cada atributo se estima independientemente [51]. La BDD trabajada actualmente cuenta con:

[53]:
```
print('Existen {} pacientes con {} variables.'.
 ↪format(*dataset.shape))
print("Existen", dataset.size, "elementos")
```

```
Existen 46 pacientes con 26 variables.
Existen 1196 elementos
```

4.7.1. Variable objetivo

En el paso de la preparación de los datos de entrada, propusimos la variable objetivo e independiente "NIHSS alta ACV" que podía poseer 42 valores diferentes, la cual se clasificó y se transformó en "NIHSS_alta_cat" que contenía 6 categorías las que fueron reducidas a 1 variable con dos estados. Así el paciente tendrá un buen pronóstico o no con el nombre de la variable "NIHSS_alta_ESTABLE_O_GRAVE". Las variables dependientes representan el rendimiento o conclusión que se está estudiando. Las variables independientes, además conocidas en una relación estadística como regresores, representan insumos o causas, donde se encuentran las razones potenciales de alteración.

[54]:
```
# variables objetivo e independientes:
from sklearn.model_selection import train_test_split

# X son nuestras variables independientes
X = dataset.drop('NIHSS_alta_ESTABLE_O_GRAVE', axis = 1)

# y es nuestra variable dependiente
y = dataset['NIHSS_alta_ESTABLE_O_GRAVE']

# Uso de Skicit-learn para dividir datos en conjuntos de␣
 ↪entrenamiento y prueba
# División 75% de datos para entrenamiento, 25% de datos␣
 ↪para test
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
 ↪random_state=0)
```

4.7.2. Creación del modelo y entrenamiento

Para la creación del algoritmo se utilizará la forma más estándar posible. No se realizarán ajustes antes del entrenamiento del modelo, para que sea lo más parejo posible entre los distintos algoritmos. Para ello, dividimos el dataset en dos partes; dejamos un 75 % de los datos como datos de entrenamiento (train), y reservamos el 25 % restan como datos de prueba (test). A continuación, entrenamos el modelo solo con los datos de entrenamiento.

[55]:
```
from sklearn.naive_bayes import GaussianNB


# Creamos el modelo de NB
nb = GaussianNB()
nb.fit(X_train, y_train)
```

[55]:
```
GaussianNB()
```

4.7.3. Predicciones sobre los datos de prueba y métricas de rendimiento

Para las métricas de rendimiento, se crearán variables de predicción. Las métricas de rendimiento nos ofrecerán información de cómo se comportó el algoritmo durante el entrenamiento, dando a conocer valores importantes en cada estado de la variable predictora en la precisión, exhaustividad, valor-F.

[56]:
```
# Predicción Entrenamiento
prediccionEntreno = nb.predict(X_train)


# Predicción Tests
prediccionTests = nb.predict(X_test)
```

[57]:
```
from sklearn import metrics


print("Entrenamiento - Precisión :", metrics.
 ↪accuracy_score(y_train, prediccionEntreno))
```

```
print("Entrenamiento - Reporte de clasificación:\n",␣
 ↪metrics.classification_report(y_train, prediccionEntreno))
```

```
Entrenamiento - Precisión : 0.8529411764705882
Entrenamiento - Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89        20
           1       1.00      0.64      0.78        14

    accuracy                           0.85        34
   macro avg       0.90      0.82      0.84        34
weighted avg       0.88      0.85      0.85        34
```

Con un total de 34 pacientes, el algoritmo en la fase de entrenamiento predice en forma global un 85,29 %. Por cada estado (0 y 1) la precisión de los datos de entrenamiento en el modelo tiene un valor de 80 % y 100 % para cada estado respectivo en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % y 64 % para cada estado respectivo y, finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 89 % y 78 % en los estados respectivos.

4.7.4. Matriz de Confusión

En el apartado del Marco Teórico 2.4.2 se definió los conceptos y utilización que tenía esta herramienta. Para mayor claridad, visualizaremos la matriz de confusión en forma de mapa de calor.

[58]:
```
from matplotlib import pyplot as plot
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

matriz = confusion_matrix(y_train, prediccionEntreno)
```

```
plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plot.tight_layout()
```

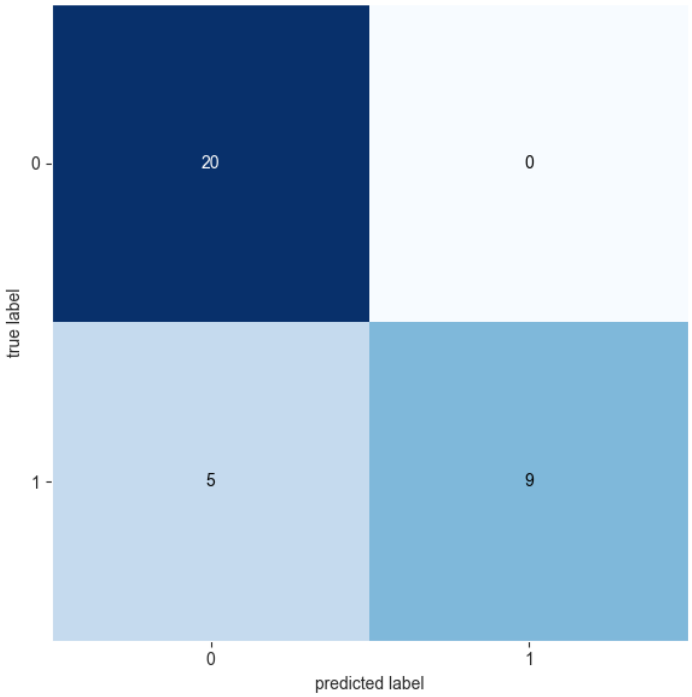

Figura 4.18: Matriz de confusión de entrenamiento Naive Bayes

En la matriz de confusión 4.18, los valores de la diagonal principal (0,0) = 20 y (1,1) = 9 corresponden con los valores estimados de forma correcta por el modelo, tanto los TN, como los TP. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.18 son (0,1) = 0 FP y (1,0) = 5 FN.

Respecto al ACV, el modelo identificó a 20 pacientes que poseen un buen pronóstico (estable) y a 9 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5. Así mismo, el modelo identifica a 5 pacientes con buen pronóstico, pero en realidad poseen mal pronóstico.

4.8. Logistic Regression - Entrenamiento del algoritmo

Como se mencionó en el Marco Teórico 2.2.6, este algoritmo de clasificación es utilizado 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 1-0, sí-no, abierto-cerrado, etc [35]. La BDD trabajada actualmente cuenta con:

[54]:
```
print('Existen {} pacientes con {} variables.'
 ↪format(*dataset.shape))
print("Existen", dataset.size, "elementos")
```

```
Existen 46 pacientes con 26 variables.
Existen 1196 elementos
```

4.8.1. Variable objetivo

Al igual que en la sub sección 4.7.1 se tomará la variable "NIHSS_alta_ESTABLE_O_GRAVE" con sus dos estados.

[55]:
```
# variables objetivo e independientes:
from sklearn.model_selection import train_test_split

# X son nuestras variables independientes
X = dataset.drop('NIHSS_alta_ESTABLE_O_GRAVE', axis = 1)

# y es nuestra variable dependiente
y = dataset['NIHSS_alta_ESTABLE_O_GRAVE']

# Uso de Skicit-learn para dividir datos en conjuntos de␣
 ↪entrenamiento y prueba
# División 75% de datos para entrenamiento, 25% de datos␣
 ↪para test
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
 ↪random_state=0)
```

4.8.2. Creación del modelo yentrenamiento

Tal cual la sub sección 4.7.2 crearemos el modelo. A continuación, entrenamos el modelo, pero sólo con los datos de entrenamiento.

[56]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split␣
↪#separa las metricas
from sklearn import metrics

# Creamos el modelo
lr = LogisticRegression(solver='lbfgs', random_state=0)
lr.fit(X_train, y_train)
```

[56]:
```
LogisticRegression(random_state=0)
```

4.8.3. Predicciones sobre los datos de prueba y métricas de rendimiento

Del mismo modo que en la sub sección 4.7.3 mostraremos las métricas del modelo en el entrenemiento.

[57]:
```
# Predicción Entrenamiento
prediccionEntreno = lr.predict(X_train)

# Predicción Tests
prediccionTests = lr.predict(X_test)
```

[58]:
```
from sklearn import metrics

print("Entrenamiento - Precisión :", metrics.
↪accuracy_score(y_train, prediccionEntreno))
print("Entrenamiento - Reporte de clasificación:\n",␣
↪metrics.classification_report(y_train, prediccionEntreno))
```

```
Entrenamiento - Precisión : 1.0
Entrenamiento - Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         3

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9
```

Con un total de 34 pacientes, el algoritmo en la fase de entrenamiento predice en forma global un 100 %. Por cada estado (0 y 1) la precisión de los datos de entrenamiento en el modelo tiene un valor de 100 % para ambos estados en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % en ambos estados. Finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 100 % en ambos estados.

4.8.4. Matriz de Confusión

En el apartado del Marco Teórico 2.4.2 se definió los conceptos y utilización que tenía esta herramienta. Para mayor claridad, visualizaremos la matriz de confusión en forma de mapa de calor.

[59]:
```
from matplotlib import pyplot as plot
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

matriz = confusion_matrix(y_train, prediccionEntreno)

plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plot.tight_layout()
```

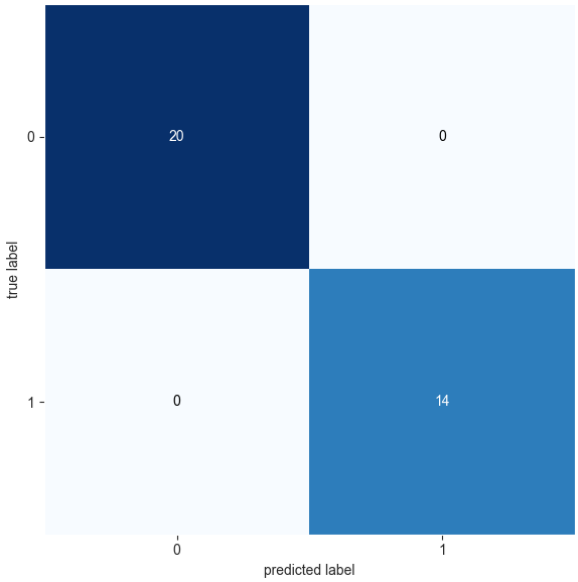

Figura 4.19: Matriz de confusión de entrenamiento Logistic Regression

En la matriz de confusión 4.19, los valores de la diagonal principal (0,0) = 20 y (1,1) = 14 corresponden con los valores estimados de forma correcta por el modelo, tanto los TP, como los TN. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.19 son (0,1) = 0 FP y (1,0) = 0 FN.

Respecto al ACV, el modelo identificó a 20 pacientes que poseen un buen pronóstico (estable) y 14 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5.

4.9. Decision Tree - Entrenamiento del algoritmo

Como se mencionó en el Marco Teórico 2.2.3, este algoritmo es utilizado para tomar decisiones, permite evaluar resultados, costos y consecuencias de una decisión compleja en grandes volúmenes datos y solventar problemas [14]. La BDD trabajada actualmente cuenta con:

[53]:
```
print('Existen {} pacientes con {} variables.'.
 ↪format(*dataset.shape))
print("Existen", dataset.size, "elementos")
```

```
Existen 46 pacientes con 26 variables.
Existen 1196 elementos
```

4.9.1. Variable objetivo

Igualmente que en la sub sección 4.7.1 se tomará la variable "NIHSS_alta_ESTABLE_O_GRAVE" con sus dos estados.

[54]:
```
# variables objetivo e independientes:
from sklearn.model_selection import train_test_split

# X son nuestras variables independientes
X = dataset.drop('NIHSS_alta_ESTABLE_O_GRAVE', axis = 1)

# y es nuestra variable dependiente
y = dataset['NIHSS_alta_ESTABLE_O_GRAVE']

# Uso de Skicit-learn para dividir datos en conjuntos de␣
↪entrenamiento y prueba
# División 75% de datos para entrenamiento, 25% de datos␣
↪para test
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
↪random_state=0)
```

4.9.2. Creación del modelo y entrenamiento

Al igual que en la sub sección 4.7.2 crearemos el modelo. A continuación, entrenamos el modelo, pero sólo con los datos de entrenamiento.

[55]:
```
from sklearn.tree import DecisionTreeClassifier

# Creamos el modelo de Arbol de Decisión (y configuramos el␣
↪número máximo de nodos-hoja)
```

```
dtc = DecisionTreeClassifier(criterion = 'gini',␣
 ↪random_state=0)
dtc.fit(X_train, y_train)
```

[55]: DecisionTreeClassifier(random_state=0)

Estructura del árbol creado

[56]:
```
# Estructura del árbol creado
from sklearn.tree import plot_tree

fig, ax = plt.subplots(figsize=(13, 6))

print(f"Profundidad del árbol: {dtc.get_depth()}")
print(f"Número de nodos terminales: {dtc.get_n_leaves()}")

plot = plot_tree(
            decision_tree = dtc,
            feature_names = feature_list,
            class_names   = 'NIHSS_alta_ESTABLE_O_GRAVE',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            ax            = ax
       )
```

```
Profundidad del árbol: 3
Número de nodos terminales: 5
```

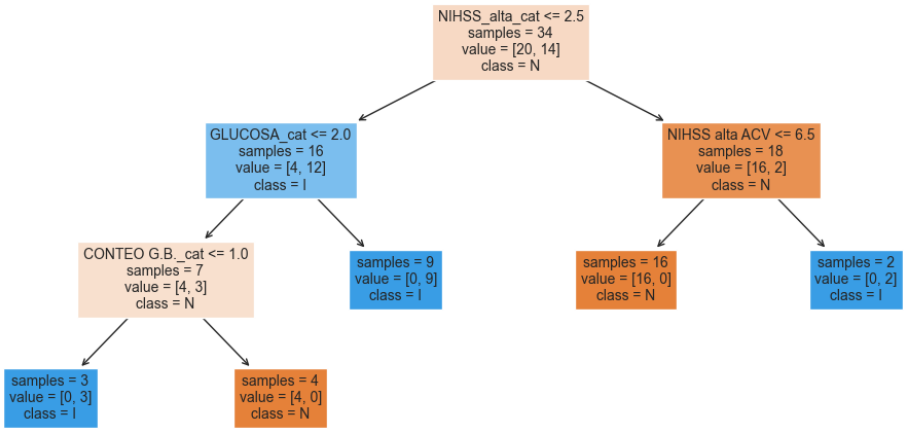

Figura 4.20: Árbol de decisión

4.9.3. Predicciones sobre los datos de prueba y métricas de rendimiento

Del mismo modo que en la sub sección 4.7.3 mostraremos las métricas del modelo en el entrenemiento.

[57]:
```
# Predicción Entrenamiento
prediccionEntreno = dtc.predict(X_train)


# Predicción Tests
prediccionTests = dtc.predict(X_test)
```

[58]:
```
from sklearn import metrics


print("Entrenamiento - Precisión :", metrics.
 ↪accuracy_score(y_train, prediccionEntreno))
print("Entrenamiento - Reporte de clasificación:\n",␣
 ↪metrics.classification_report(y_train, prediccionEntreno))
```

```
Entrenamiento - Precisión : 1.0
Entrenamiento - Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
```

```
           1       1.00      1.00      1.00        14

    accuracy                           1.00        34
   macro avg       1.00      1.00      1.00        34
weighted avg       1.00      1.00      1.00        34
```

Con un total de 34 pacientes, el algoritmo en la fase de entrenamiento predice en forma global un 100 %. Por cada estado (0 y 1) la precisión de los datos de entrenamiento en el modelo tiene un valor de 100 % para ambos estados en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % en ambos estados, finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 100 % en ambos estados.

4.9.4. Matriz de Confusión

En el apartado del Marco Teórico 2.4.2 se definió los conceptos y utilización que tenía esta herramienta. Para mayor claridad, visualizaremos la matriz de confusión en forma de mapa de calor.

[59]:
```
from matplotlib import pyplot as plot
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

matriz = confusion_matrix(y_train, prediccionEntreno)

plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plot.tight_layout()
```

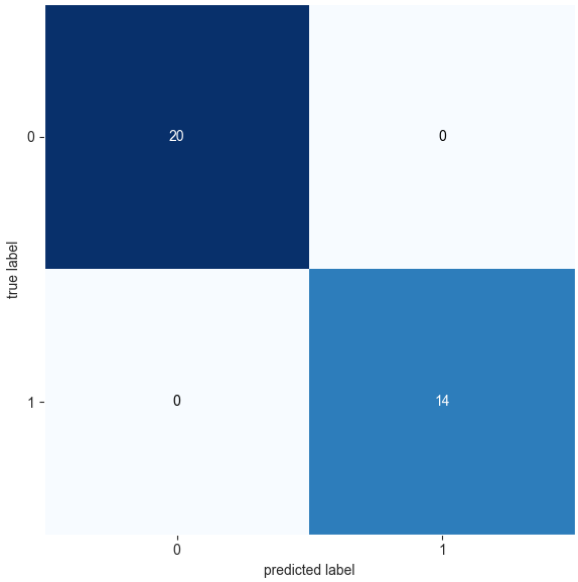

Figura 4.21: Matriz de confusión de entrenamiento Decision Tree

En la matriz de confusión 4.21, los valores de la diagonal principal (0,0) = 20 y (1,1) = 14 corresponden con los valores estimados de forma correcta por el modelo, tanto los TP, como los TN. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.19 son (0,1) = 0 FP y (1,0) = 0 FN.

Respecto al ACV, el modelo identificó a 20 pacientes que poseen un buen pronóstico (estable) y 14 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5.

4.10. Random Forest - Entrenamiento del algoritmo

Como se mencionó en el Marco Teórico 2.2.4, consta de un conjunto de árboles de decisión individuales, cada uno de los cuales se entrena mediante un procedimiento de arranque para seleccionar aleatoriamente muestras de los datos de entrenamiento originales. Esto significa que cada árbol se entrena con datos ligeramente diferentes. En cada árbol individual, las observaciones se propagan a través de bifurcaciones (nodos) que generan la estructura del árbol hasta llegar a un nodo terminal [35]. La BDD trabajada actualmente cuenta con:

[53]:
```
print('Existen {} pacientes con {} variables.'.
 ↪format(*dataset.shape))
print("Existen", dataset.size, "elementos")
```

```
Existen 46 pacientes con 26 variables.
Existen 1196 elementos
```

4.10.1. Variable objetivo

Al igual que en la sub sección 4.7.1 se tomará la variable "NIHSS_alta_ESTABLE_O_GRAVE" con sus dos estados.

[54]:
```
# variables objetivo e independientes:
from sklearn.model_selection import train_test_split

# X son nuestras variables independientes
X = dataset.drop('NIHSS_alta_ESTABLE_O_GRAVE', axis = 1)

# y es nuestra variable dependiente
y = dataset['NIHSS_alta_ESTABLE_O_GRAVE']

# Uso de Skicit-learn para dividir datos en conjuntos de␣
 ↪entrenamiento y prueba
# División 75% de datos para entrenamiento, 25% de datos␣
 ↪para test
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
 ↪test_size=0.8, random_state=0)
```

4.10.2. Creación del modelo y entrenamiento

De la misma manera que en la sub sección 4.7.2 crearemos el modelo. A continuación, entrenamos el modelo, pero sólo con los datos de entrenamiento.

[55]:
```
from sklearn.ensemble import RandomForestClassifier

# Creamos el modelo de Bosque Aleatorio (y configuramos el␣
↪número máximo de nodos-hoja)
rfc = RandomForestClassifier(criterion = 'gini',␣
↪random_state=0)
rfc.fit(X_train, y_train)
```

[55]:
```
RandomForestClassifier(random_state=0)
```

Estructura del árbol creado

[56]:
```
# Import tools needed for visualization
from sklearn.tree import export_graphviz
import pydot

# Pull out one tree from the forest
tree = rfc.estimators_[5]
print('The depth of this tree is:', tree.tree_.max_depth)

# Export the image to a dot file
export_graphviz(tree, out_file =␣
↪'randomForestClassification.dot', feature_names =␣
↪feature_list, rounded = True, precision = 1)

# Use dot file to create a graph
(graph, ) = pydot.
↪graph_from_dot_file('randomForestClassification.dot')

# Write graph to a png file
graph.write_png('randomForestClassification.png');
```

```
The depth of this tree is: 4
```

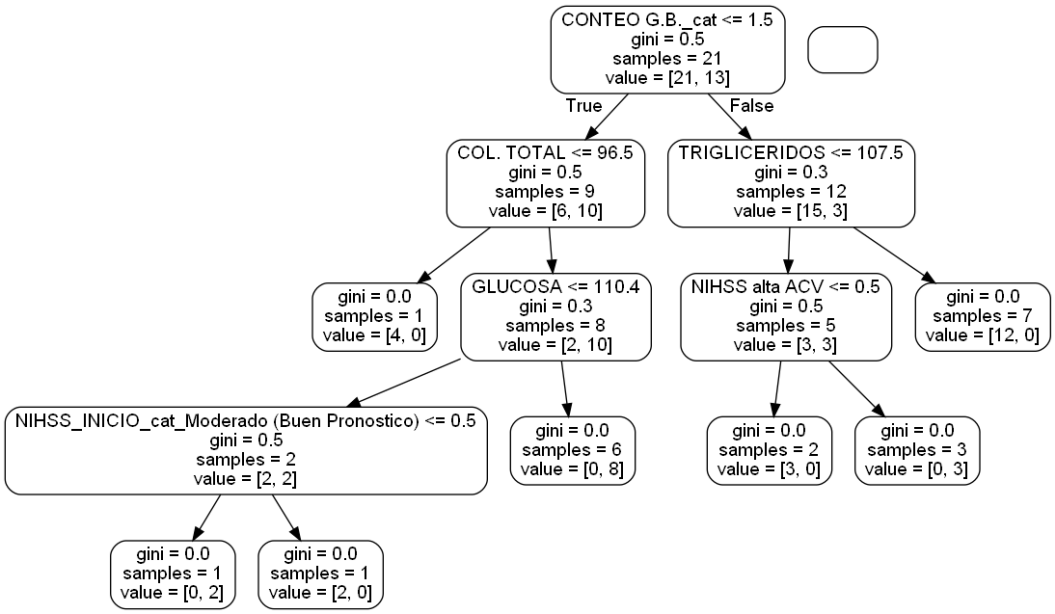

Figura 4.22: Random Forest

4.10.3. Predicciones sobre los datos de prueba y métricas de rendimiento

Del mismo modo que en la sub sección 4.7.3 mostraremos las métricas del modelo en el entrenemiento.

[57]:
```
# Predicción Entrenamiento
prediccionEntreno = rfc.predict(X_train)


# Predicción Tests
prediccionTests = rfc.predict(X_test)
```

[58]:
```
from sklearn import metrics


print("Entrenamiento - Precisión :", metrics.
 ↪accuracy_score(y_train, prediccionEntreno))
print("Entrenamiento - Reporte de clasificación:\n",␣
 ↪metrics.classification_report(y_train, prediccionEntreno))
```

```
Entrenamiento - Precisión : 1.0
Entrenamiento - Reporte de clasificación:
```

```
                precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         3

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9
```

Con un total de 34 pacientes, el algoritmo en la fase de entrenamiento predice en forma global un 100 %. Por cada estado (0 y 1) la precisión de los datos de entrenamiento en el modelo tiene un valor de 100 % para ambos estados en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % en ambos estados. Finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 100 % en ambos estados.

4.10.4. Matriz de Confusión

En el apartado del Marco Teórico 2.4.2 se definió los conceptos y utilización que tenía esta herramienta. Para mayor claridad, visualizaremos la matriz de confusión en forma de mapa de calor.

[59]:
```
from matplotlib import pyplot as plot
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

matriz = confusion_matrix(y_train, prediccionEntreno)

plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plot.tight_layout()
```

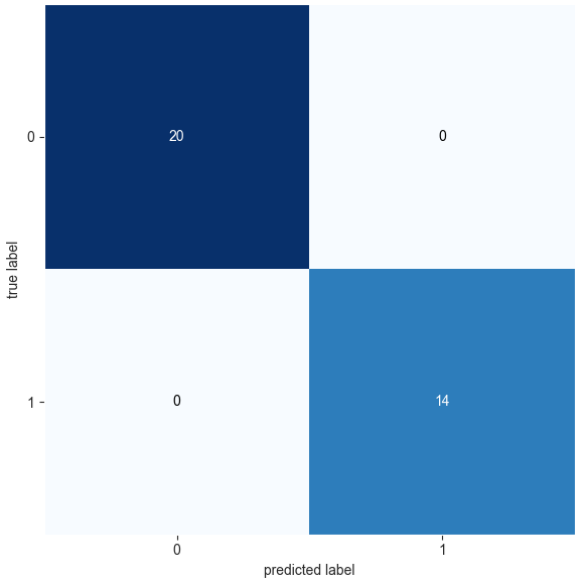

Figura 4.23: Matriz de confusión de entrenamiento Random Forest

En la matriz de confusión 4.23, los valores de la diagonal principal (0,0) = 20 y (1,1) = 14 corresponden a los valores estimados de forma correcta por el modelo, tanto los TP, como los TN. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.19 son (0,1) = 0 FP y (1,0) = 0 FN.

Respecto al ACV, el modelo identificó a 20 pacientes que poseen un buen pronóstico (estable) y 14 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5.

Capítulo 5

RESULTADOS

El presente capítulo tiene como finalidad la implementación del método utilizado. Los testing y usos del algoritmo nos ayudarán a observar y concluir los resultados, qué algoritmo funciona mejor en las pruebas métricas realizadas.

5.1. Naïve Bayes - Testeo del algoritmo

El testing tiene la finalidad de llevar a cabo la prueba si el modelo funciona correctamente, identificando riesgos o errores que se produjeron en los datos. No se realizará ajustes posteriores al testing para poder comparar los algoritmos en la sección de resultados.

5.1.1. Predicciones sobre los datos del testing y métricas de rendimiento

Ahora es momento de evaluar los datos ya entrenados con el testing. A continuación, se dará a conocer las métricas que se evaluaron en el Estudio Empírico del capítulo 4 de la sub sección Predicciones sobre los datos de prueba y métricas de rendimiento 4.7.3.

[59]:

```
print("Tests - Precisión :", metrics.accuracy_score(y_test,␣
 ↪prediccionTests))
print("Tests - Reporte de clasificación:\n", metrics.
 ↪classification_report(y_test, prediccionTests))
```

```
Tests - Precisión : 0.5833333333333334
Tests - Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.55      1.00      0.71         6
           1       1.00      0.17      0.29         6

    accuracy                           0.58        12
   macro avg       0.77      0.58      0.50        12
weighted avg       0.77      0.58      0.50        12
```

Por cada estado (0 y 1) la precisión de los datos del testing en el modelo tiene un valor de 55 % y 100 % para cada estado respectivo en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % y 17 % para cada estado respectivo y, finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 71 % y 29 % en los estados respectivos.

Lo que se busca es la precisión del modelo, por consecuencia, el algoritmo de ML Naïve Bayes tiene una precisión del 58.3 % de predicción en los 12 pacientes de muestra del testing.

5.1.2. Matriz de Confusión

Evaluaremos la matriz de confusión que se elaboró con los datos del testing.

[60]:
```
matriz = confusion_matrix(y_test, prediccionTests)
```

```
plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plt.tight_layout()
```

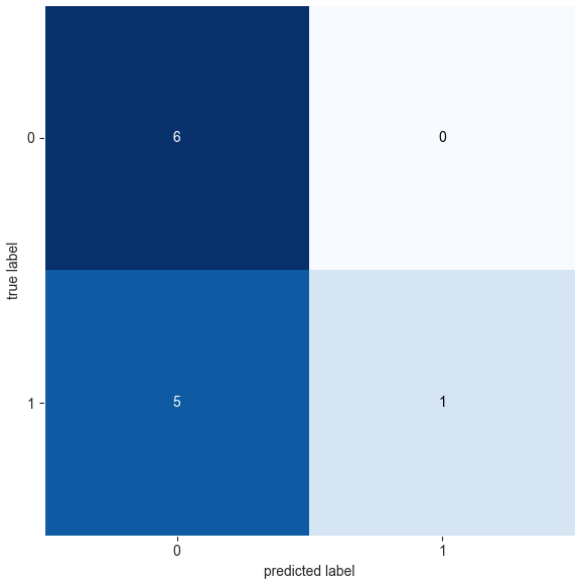

Figura 5.1: Matriz de confusión de testing Naive Bayes

En la matriz de confusión 5.1, los valores de la diagonal principal (0,0) = 6 y (1,1) = 1 corresponden con los valores estimados de forma correcta por el modelo, tanto los TN, como los TP. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.18 son (0,1) = 0 FP y (1,0) = 5 FN.

Las afirmaciones anteriores sugieren que las predicciones son altas, pero también existen errores en la predicción.

Respecto al ACV, el modelo identificó a 6 pacientes que poseen un buen pronóstico (estable) y solo 1 paciente que posee un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5. Así mismo, el modelo identifica a 5 pacientes que poseen buen pronóstico, pero en realidad poseen mal pronóstico.

5.2. Naïve Bayes - Uso del algoritmo

El último paso de la metodología es el uso del algoritmo, nosotros lo utilizaremos para desarrollar probabilidades en los predictores. No todos los modelos poseen los mismos métodos ni atributos, por ende, se tratará de realizar comparaciones con métodos similares entre sí.

5.2.1. Importancia de los predictores

Por experiencia previa y contemplando los gráficos producidos en el paso 3, sabemos que algunas características no son útiles para nuestro problema de predicción. Reducir la cantidad de funciones será la mejor alternativa, lo que acotará el tiempo de ejecución, con suerte sin comprometer significativamente el rendimiento, así podemos examinar la importancia de las funciones de nuestro modelo. La importancia de cada predictor en el modelo se calcula como la reducción total (normalizada) en el criterio de división. Si un predictor no ha sido seleccionado en ninguna división, no se ha incluido en el modelo y, por lo tanto, su importancia es 0.

[61]:
```
# Predicciones probabilísticas
# ==============================
# Con .predict_proba() se obtiene, para cada observación,␣
 ↪la probabilidad predicha
# de pertenecer a cada una de las dos clases.
predicciones = nb.predict_proba(X_test)
predicciones = pd.DataFrame(predicciones, columns = nb.
 ↪classes_)
predicciones
```

	0	1
0	0,999617	0,000383
1	0,999992	8,06E-06
2	1	9,05E-08
3	0,999997	2,93E-06
4	1	4,81E-07
5	1	3,42E-07
6	0,999105	0,000895
7	0,999973	2,69E-05
8	0,999798	0,000202
9	9E-176	1
10	1	3,4E-08
11	1	0

Tabla 5.1: Predicciones probabilísticas para cada observación Bayes

Este método acepta un solo argumento que corresponde a los datos sobre los cuales se calculan las probabilidades y devuelve una matriz de listas que contienen las probabilidades de clase para los puntos de datos de entrada.

En este caso, en la tabla 5.1 se observa los estados de la variable predictora (0 y 1) y el porcenteje por tupla o paciente a predecir. Por ejemplo la tupla 0 posee un 99 % y fracción de precisión para el estado 0 y un 3,83 % x 10^{-4} para el estado 1, en otras palabras, el paciente posee un buen pronóstico con un 99 % de probabilidad para este algoritmo.

[62]:
```
# Predicciones con clasificación final
# ==============================
# Con .predict() se obtiene, para cada observación, la␣
 ↪clasificación predicha por
# el modelo. Esta clasificación se corresponde con la clase␣
 ↪con mayor probabilidad.
predicciones = nb.predict(X_test)
predicciones = pd.DataFrame(predicciones)
predicciones
```

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	1
10	0
11	0

Tabla 5.2: Predicciones probabilísticas con clasificación final Bayes

Se observa en la tabla 5.2 un valor binario 0 o 1, donde se muestra el valor que toma la variable desarrollada en el modelo. El valor 1 demuestra que la tupla no pertenece al estado 0 de la variable predictora, y por el contrario, el estado 0 demuestra que predice el estado en esa tupla. Sin embargo, estos resultados de clasificación final, son los que el modelo clasifica, si hubiera un error, este se analiza en la Matriz de Confusión 5.1.

Recordemos que Matriz de Confusión 5.1 casilla (1,1), poseía solo 1 paciente que pertencía a que tenia un pronóstico menos favorable (estado 1), en este caso, en la tupla o paciente número 9 de la tabla 5.2, la clasificación predicha por el modelo apunta que es 1, es decir, no alcanza a ser parte de ese estado; por el contrario, las demás tuplas las reconoce que pertencen a ese estado (tienen buen pronóstico en términos de ACV).

5.3. Logistic Regression - Testeo del algoritmo

En esta sección se ralizará el Testing del algoritmo Logistic Regression como en la sección 5.1.

5.3.1. Predicciones sobre los datos del testing y métricas de rendimiento

Tal como, la sección 5.1.1 se realizará la evaluación de métricas del Testing.

[60]:
```
print("Tests - Precisión :", metrics.accuracy_score(y_test,␣
 ↪prediccionTests))
print("Tests - Reporte de clasificación:\n", metrics.
 ↪classification_report(y_test, prediccionTests))
```

```
Tests - Precisión : 0.8333333333333334
Tests - Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         6
           1       1.00      0.67      0.80         6

    accuracy                           0.83        12
   macro avg       0.88      0.83      0.83        12
weighted avg       0.88      0.83      0.83        12
```

Por cada estado (0 y 1) la precisión de los datos del testing en el modelo tiene un valor de 75 % y 100 % para cada estado respectivo en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % y 67 % para cada estado respectivo y, finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 86 % y 80 % en los estados respectivos.

Lo que se busca es la precisión del modelo, por consecuencia, el algoritmo de ML Logistic Regression tiene una precisión del 83,33 % de predicción en los 12 pacientes de muestra del testing.

5.3.2. Matriz de Confusión

Evaluaremos la matriz de confusión que se elaboró con los datos del testing.

[61]:
```
matriz = confusion_matrix(y_test, prediccionTests)


plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plt.tight_layout()
```

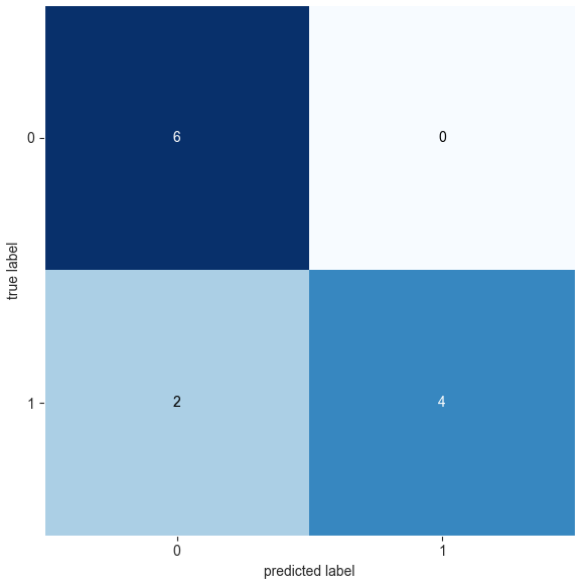

Figura 5.2: Matriz de confusión de testing Logistic Regression

En la matriz de confusión 5.2, los valores de la diagonal principal (0,0) = 6 y (1,1) = 4 corresponden con los valores estimados de forma correcta por el modelo, tanto los TN, como los TP. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.18 son (0,1) = 0 FP y (1,0) = 2 FN.

Las afirmaciones anteriores sugieren que las predicciones son altas, pero también existen algunos errores en la predicción.

Respecto al ACV, el modelo identificó a 6 pacientes que poseen un buen pronóstico (estable) y 4 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5. Así mismo, el modelo identifica a 2 pacientes que poseen buen pronóstico, pero en realidad poseen mal pronóstico.

5.4. Logistic Regression - Uso del algoritmo

Al igual que en la sección 5.2, realizaremos comparaciones de predicción y probabilidad.

5.4.1. Importancia de los predictores

Al igual que la sub sección 5.2.1, realizaremos los procedimientos.

[62]:
```
# Predicciones probabilísticas
#␣
↪==========================================================================
# Con .predict_proba() se obtiene, para cada observación,␣
↪la probabilidad predicha
# de pertenecer a cada una de las dos clases.
predicciones = lr.predict_proba(X_test)
predicciones = pd.DataFrame(predicciones, columns = lr.
↪classes_)
predicciones
```

	0	1
0	0,769731	0,230269
1	0,965470	0,034530
2	0,993616	0,006384
3	0,998082	0,001918
4	0,420635	0,579365
5	0,576335	0,423665
6	0,197545	0,802455
7	0,980097	0,019903
8	0,104532	0,895468
9	0,008064	0,991936
10	0,978317	0,021683
11	0,999660	0,000340

Tabla 5.3: Predicciones probabilísticas para cada observación Logistic Regression

De acuerdo a lo mostrado en la tabla 5.3, se observa como ejemplo, que la tupla 0 posee un 76 % y fracción de precisión para el estado 0 y un 23 % y fracción para

el estado 1, en otras palabras, el paciente posee un buen pronóstico con un 76 % de probabilidad para este algoritmo.

[63]:
```
# Predicciones con clasificación final
#␣
↪=============================================================================
# Con .predict() se obtiene, para cada observación, la␣
↪clasificación predicha por
# el modelo. Esta clasificación se corresponde con la clase␣
↪con mayor probabilidad.
predicciones = lr.predict(X_test)
predicciones = pd.DataFrame(predicciones)
predicciones
```

	0
0	0
1	0
2	0
3	0
4	1
5	0
6	1
7	0
8	1
9	1
10	0
11	0

Tabla 5.4: Predicciones probabilísticas con clasificación final Logistic Regression

Tomaremos para interpretación la tupla número 4 de la tabla 5.4. Esta tupla no pertenece al estado 0, por ende, en términos de ACV, el paciente número 4 no tiene un pronóstico favorable.

Recordamos que la Matriz de Confusión 5.2 poseía 4 pacientes que tenian un pronóstico menos favorable, ellos son los que se registran con el valor 1 en la columna 0 de la tabla.

5.5. Decision Tree - Testeo del algoritmo

En esta sección se ralizará el Testing del algoritmo Decision Tree como en la sección 5.1.

5.5.1. Predicciones sobre los datos del testing y métricas de rendimiento

Tal como, en la sección 5.1.1 se realizará la evaluación de métricas del Testing.

[60]:
```
print("Tests - Precisión :", metrics.accuracy_score(y_test,␣
↪prediccionTests))
print("Tests - Reporte de clasificación:\n", metrics.
↪classification_report(y_test, prediccionTests))
```

```
Tests - Precisión : 0.8333333333333334
Tests - Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83         6
           1       0.83      0.83      0.83         6

    accuracy                           0.83        12
   macro avg       0.83      0.83      0.83        12
weighted avg       0.83      0.83      0.83        12
```

Por cada estado (0 y 1) la precisión de los datos del testing en el modelo tiene un valor de 83 % en ambos estados en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 83 % para ambos estados. Finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 83 % en ambos estados.

Lo que se busca es la precisión del modelo, por consecuencia, el algoritmo de ML Decision Tree tiene una precisión del 83,33 % de predicción en los 12 pacientes

de muestra del testing.

5.5.2. Matriz de Confusión

Evaluaremos la matriz de confusión que se elaboró con los datos del testing.

[61]:
```
matriz = confusion_matrix(y_test, prediccionTests)


plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plt.tight_layout()
```

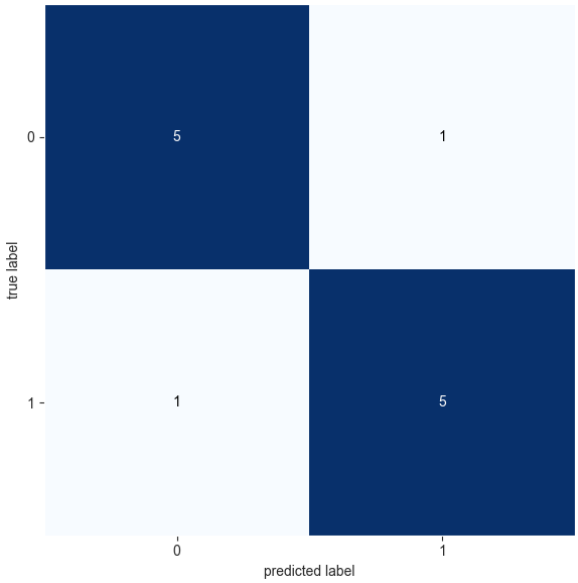

Figura 5.3: Matriz de confusión de testing Decision Tree

En la matriz de confusión 5.3, los valores de la diagonal principal (0,0) = 5 y (1,1) = 5 corresponden con los valores estimados de forma correcta por el modelo, tanto los TN, como los TP. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.18 son (0,1) = 1 FP y (1,0) = 1 FN.

Las afirmaciones anteriores sugieren que las predicciones acertadas son altas, pero también existen algunos errores en la predicción.

Respecto al ACV, el modelo identificó a 5 pacientes que poseen un buen pronóstico (estable) y 5 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5. Así mismo, el modelo identifica a 1 paciente que posee buen pronóstico, pero en realidad posee mal pronóstico, y a su vez, el modelo identifica a 1 paciente que posee mal pronóstico, pero en realidad posee buen pronóstico.

5.6. Decision Tree - Uso del algoritmo

Al igual que en la sección 5.2, realizaremos comparaciones de predicción y probabilidad. Además, se incorporará métricas especiales para árboles de su librería en esta sección.

5.6.1. Importancia de los predictores

Al igual que la sub sección 5.2.1, realizaremos los procedimientos.

[62]:
```
# Predicciones probabilísticas
# ======================================
# Con .predict_proba() se obtiene, para cada observación,␣
 ↪la probabilidad predicha
# de pertenecer a cada una de las dos clases.
predicciones = dtc.predict_proba(X_test)
predicciones = pd.DataFrame(predicciones, columns = dtc.
 ↪classes_)
predicciones
```

	0	1
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
5	0	1
6	0	1
7	1	0
8	0	1
9	0	1
10	1	0
11	1	0

Tabla 5.5: Predicciones probabilísticas para cada observación Decision Tree

De acuerdo a lo mostrado en la tabla 5.5, se observa como ejemplo, que la tupla 0 posee un 0 % de probabilidad para el estado 0 y un 100 % de probabilidad para el estado 1, en otras palabras, el paciente posee un mal pronóstico con un 100 % de probabilidad para este algoritmo.

[63]:
```
# Predicciones con clasificación final
# ======================================
# Con .predict() se obtiene, para cada observación, la␣
 ↪clasificación predicha por
# el modelo. Esta clasificación se corresponde con la clase␣
 ↪con mayor probabilidad.
predicciones = dtc.predict(X_test)
predicciones = pd.DataFrame(predicciones)
predicciones
```

	0
0	1
1	0
2	0
3	0
4	1
5	1
6	1
7	0
8	1
9	1
10	0
11	0

Tabla 5.6: Predicciones probabilísticas con clasificación final Decision Tree

En la tabla 5.6, se observa para interpretación como ejemplo la tupla número 4. Esta tupla no pertenece al estado 0, por ende, en términos de ACV, el paciente número 4 no tiene un pronóstico favorable.

Recordamos que la Matriz de Confusión 5.3 poseía 5 pacientes que tenían un pronóstico menos favorable, ellos son los que se registran con el valor 1 en la columna 0 de la tabla.

[64]:
```
importancia_predictores = pd.DataFrame(
                            {'Predictor': dataset.
 ↪drop(columns = "NIHSS_alta_ESTABLE_O_GRAVE").columns,
                             'importancia': dtc.
 ↪feature_importances_}
                            )
print("Importancia de los predictores en el modelo")
print("----------------------------------------")
importancia_predictores.sort_values('importancia',␣
 ↪ascending=False)
```

	Predictor	importancia
11	NIHSS_alta_cat	0,419841
10	NIHSS alta ACV	0,215873
13	CONTEO G.B._cat	0,208163
17	GLUCOSA_cat	0,156122
0	HTA	0
14	INR_cat	0
23	NIHSS_INICIO_cat_Moderado (Buen Pronostico)	0
22	NIHSS_INICIO_cat_Leve (Trombolisando)	0
21	NIHSS_INICIO_cat_Grave	0
20	NIHSS_INICIO_cat_Déficit Mínimo	0
19	NIHSS_INICIO_cat_Déficit Importante	0
18	EDAD_cat	0
16	COL. TOTAL_cat	0
15	TRIGLICERIDOS_cat	0
12	GLASGOW_cat	0
1	DIABETES	0
9	NIHSS INICO ACV	0
8	GLASGOW AL INICO ACV	0
7	CONTEO G.B.	0
6	INR	0
5	TRIGLICERIDOS	0
4	COL. TOTAL	0
3	GLUCOSA	0
2	EDAD	0
24	NIHSS_INICIO_cat_Sin Déficit	0

Tabla 5.7: Importancia de los predictores Decision Tree

[65]:
```
# Get numerical feature importances
importances = list(dtc.feature_importances_)
# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for␣
↪feature, importance in zip(feature_list, importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key =␣
↪lambda x: x[1], reverse = True)


# Reset style
plt.style.use('fivethirtyeight')


# lista de x ubicaciones para trazar
```

```
x_values = list(range(len(importances)))

# Gráfico de barras
plt.bar(x_values, importances, orientation = 'vertical',␣
 ↪color = 'r', edgecolor = 'k', linewidth = 1.2)

# Marque las etiquetas para el eje x
plt.xticks(x_values, feature_list, rotation='vertical')

# Etiquetas de eje y título
plt.ylabel('Importancia'); plt.xlabel('Variable'); plt.
 ↪title('Importancia de Variables');
```

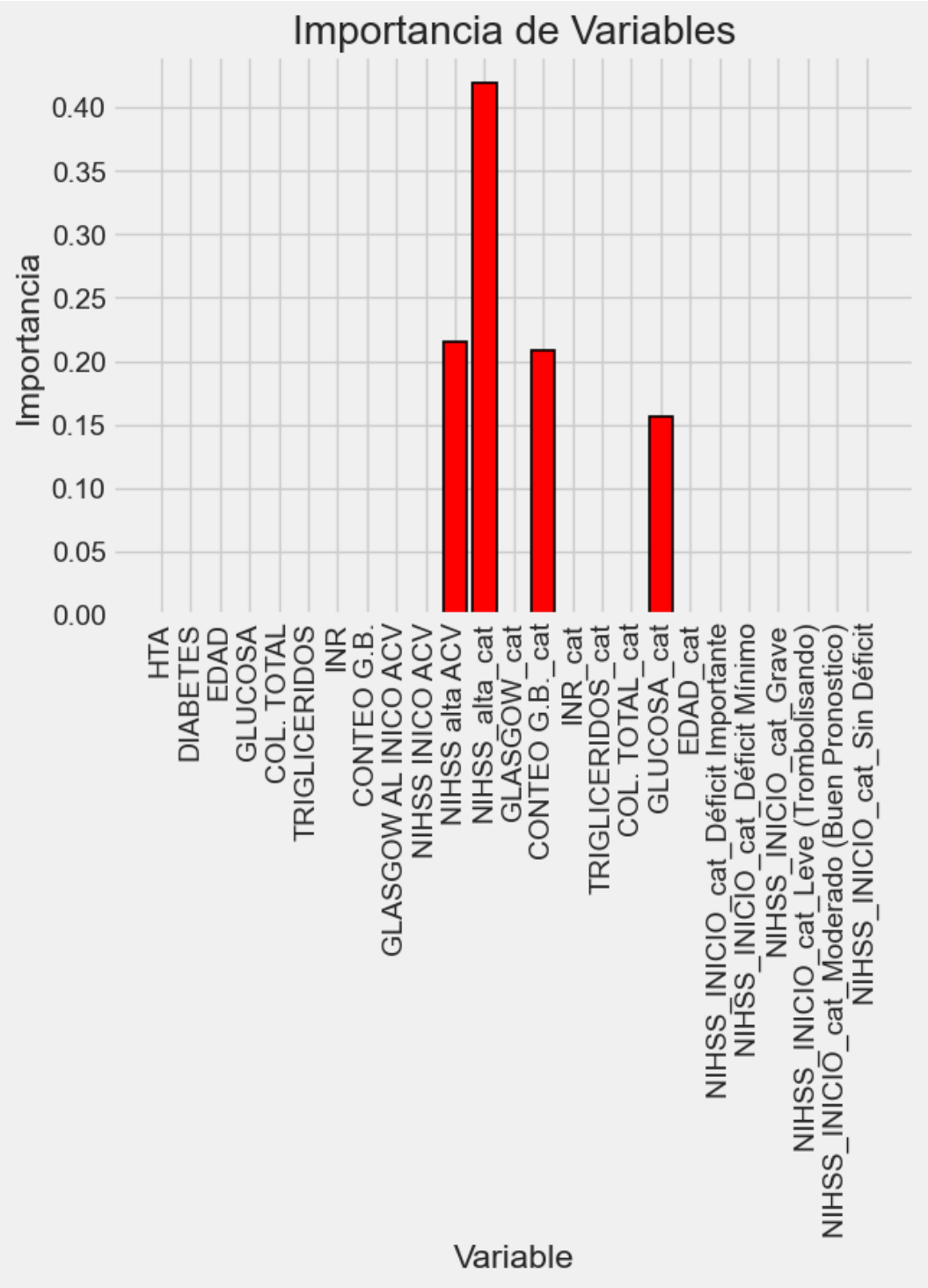

Figura 5.4: Importancia de los predictores en Decision Tree

Las variables que presentan gran importancia al momento de la predicción según la tabla 5.7 y la Figura 5.4 son “NIHSS_alta_cat”, “NIHSS alta ACV”, “CONTEO G.B._cat” y “GLUCOSA_cat” con más del 98 % de importancia en la predicción. El mayor predictor es “NIHSS_alta_cat” con un 41,98 % de importancia

para este modelo.

5.6.2. Importancia acumulada

Ahora reduciremos la cantidad de funciones en uso por el modelo a solo aquellas requeridas para representar el 95 % de la importancia. Se debe usar el mismo número de características en los conjuntos de entrenamiento y testing.

[66]:
```
# Lista de funciones ordenadas de mayor a menor importancia
sorted_importances = [importance[1] for importance in␣
 ↪feature_importances]
sorted_features = [importance[0] for importance in␣
 ↪feature_importances]

# Importancias acumulativas
cumulative_importances = np.cumsum(sorted_importances)

# Haz un gráfico de líneas
plt.plot(x_values, cumulative_importances, 'g-')

# Dibujar línea al 95% de importancia retenida
plt.hlines(y = 0.95, xmin=0, xmax=len(sorted_importances),␣
 ↪color = 'r', linestyles = 'dashed')

# Formato x ticks y etiquetas
plt.xticks(x_values, sorted_features, rotation = 'vertical')

# Etiquetas de eje y título
plt.xlabel('Variable'); plt.ylabel('Importancia acumulada');
 ↪ plt.title('Importancia acumulada');
```

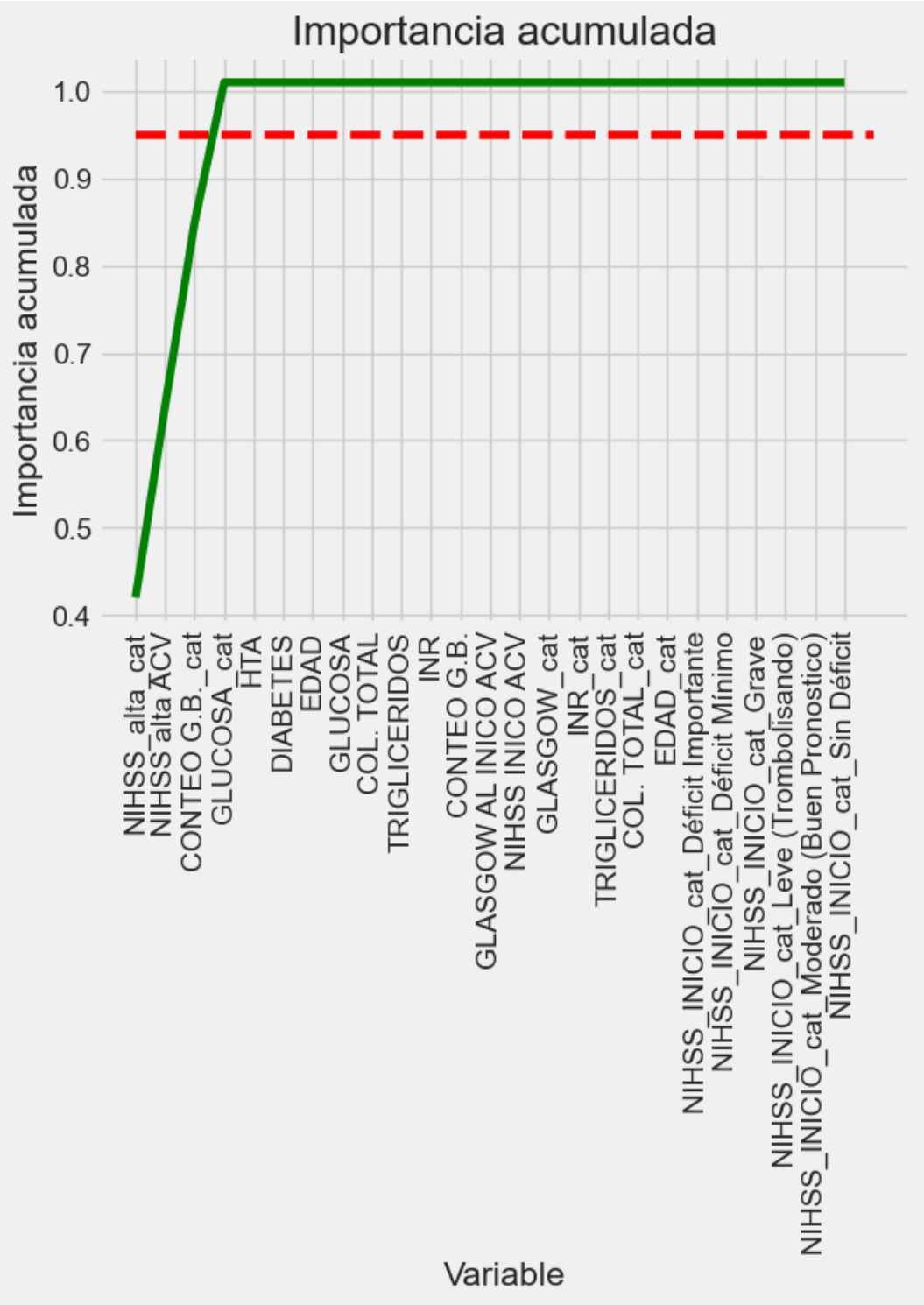

Figura 5.5: Importancia de los predictores acumulada en Decision Tree

[67]:
```
# Encuentre el número de características para una␣
 ↪importancia acumulada del 95%
# Agregue 1 porque Python está indexado a cero
```

```
print('Número de columna para el 95 % de importancia:', np.
 ↪where(cumulative_importances > 0.95)[0][0] + 1)
```

```
Número de columna para el 95 % de importancia: 4
```

En la Figura 5.5 de importancia acumulada, la curva se dispara con la "GLUCOSA_cat" , ratificándose con el resultado del 95 % de importancia medido anteriormente.

5.7. Random Forest - Testeo del algoritmo

En esta sección se ralizará el Testing del algoritmo Random Forest como en la sección 5.1.

5.7.1. Predicciones sobre los datos del testing y métricas de rendimiento

Tal como la sección 5.1.1, se realizará la evaluación de métricas del Testing.

[60]:
```
print("Tests - Precisión :", metrics.accuracy_score(y_test,␣
 ↪prediccionTests))
print("Tests - Reporte de clasificación:\n", metrics.
 ↪classification_report(y_test, prediccionTests))
```

```
Tests - Precisión : 0.6666666666666666
Tests - Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.60      1.00      0.75         6
           1       1.00      0.33      0.50         6
```

```
    accuracy                           0.67        12
   macro avg       0.80      0.67      0.62        12
weighted avg       0.80      0.67      0.62        12
```

Por cada estado (0 y 1) la precisión de los datos del testing en el modelo tiene un valor de 60 % y 100 % para cada estado respectivo en predicción. La exhaustividad informa la cantidad de datos capaz de identificar y, en este caso, es de un 100 % y 33 % para cada estado respectivo y, finalmente, el F1 combina los valores de precisión y exhaustividad obteniéndose un 75 % y 50 % en los estados respectivos.

Lo que se busca es la precisión del modelo, por consecuencia, el algoritmo de ML Logistic Regression tiene una precisión del 66,66 % de predicción en los 12 pacientes de muestra del testing.

5.7.2. Matriz de Confusión

Evaluaremos la matriz de confusión que se elaboró con los datos del testing.

[61]:
```
matriz = confusion_matrix(y_test, prediccionTests)


plot_confusion_matrix(conf_mat=matriz, figsize=(6,6),␣
 ↪show_normed=False)
plt.tight_layout()
```

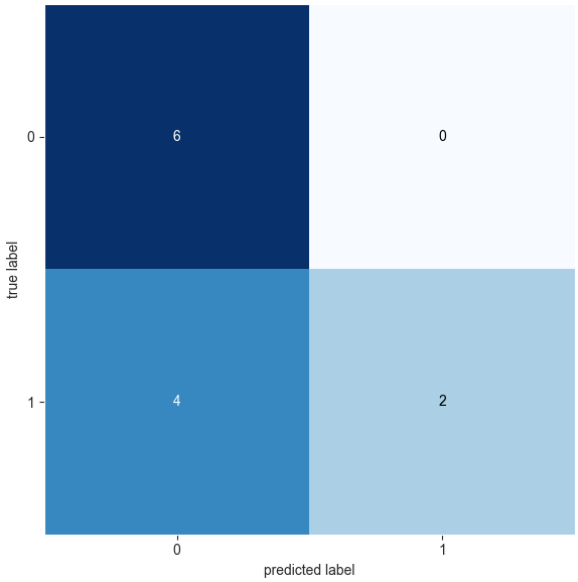

Figura 5.6: Matriz de confusión de testing Random Forest

En la matriz de confusión 5.6, los valores de la diagonal principal (0,0) = 6 y (1,1) = 2 corresponden a los valores estimados de forma correcta por el modelo, tanto los TN, como los TP. La otra diagonal, representa los casos en los que el modelo *"se ha equivocado"*, según la matriz de confusión 4.18 son (0,1) = 0 FP y (1,0) = 4 FN.

Las afirmaciones anteriores sugieren que las predicciones correctas son altas, pero también existen algunos errores en la predicción.

Respecto al ACV, el modelo identificó a 6 pacientes que poseen un buen pronóstico (estable) y 2 pacientes que poseen un pronóstico no tan favorable, según la variable objetivo detallada en 4.5.5. Así mismo, el modelo identifica a 4 pacientes que poseen buen pronóstico, pero en realidad poseen mal pronóstico.

5.8. Random Forest - Uso del algoritmo

Al igual que en la sección 5.2, realizaremos comparaciones de predicción y probabilidad. Además, se incorporará métricas especiales para árboles de su librería en esta sección.

5.8.1. Importancia de los predictores

Al igual que la sub sección 5.2.1, realizaremos los procedimientos.

[62]:
```
# Predicciones probabilísticas
# ==============================
# Con .predict_proba() se obtiene, para cada observación,␣
 ↪la probabilidad predicha
# de pertenecer a cada una de las dos clases.
predicciones = rfc.predict_proba(X_test)
predicciones = pd.DataFrame(predicciones, columns = rfc.
 ↪classes_)
predicciones
```

	0	1
0	0,53	0,47
1	0,81	0,19
2	0,86	0,14
3	0,85	0,15
4	0,60	0,40
5	0,52	0,48
6	0,29	0,71
7	0,62	0,38
8	0,51	0,49
9	0,28	0,72
10	0,97	0,03
11	0,95	0,05

Tabla 5.8: Predicciones probabilísticas para cada observación Random Forest

De acuerdo a lo mostrado en la tabla 5.8, se observa que la tupla 0 posee un 53 % de probabilidad para el estado 0 y un 47 % de probabilidad para el estado 1, en otras palabras, el paciente posee un buen pronóstico con un 53 % de probabilidad para este algoritmo.

[63]:
```
# Predicciones con clasificación final
#␣
 ↪==============================================================
```

```
# Con .predict() se obtiene, para cada observación, la
 ↪clasificación predicha por
# el modelo. Esta clasificación se corresponde con la clase
 ↪con mayor probabilidad.
predicciones = rfc.predict(X_test)
predicciones = pd.DataFrame(predicciones)
predicciones
```

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	0
8	0
9	1
10	0
11	0
12	0

Tabla 5.9: Predicciones probabilísticas con clasificación final Random Forest

En la tabla 5.9, se observa la tupla número 6. Esta tupla no pertenece al estado 0, por ende, en términos de ACV, el paciente número 6 no tiene un pronóstico favorable.

Recordamos que la Matriz de Confusión 5.6 poseía 2 pacientes que tenían un pronóstico menos favorable, ellos son los que se registran con el valor 1 en la columna 0 de la tabla.

[64]:
```
importancia_predictores = pd.DataFrame(
                                {'Predictor': dataset.
 ↪drop(columns = "NIHSS_alta_ESTABLE_O_GRAVE").columns,
                                 'importancia': rfc.
 ↪feature_importances_}
                                )
print("Importancia de los predictores en el modelo")
```

```
print("-----------------------------------------")
importancia_predictores.sort_values('importancia',␣
↪ascending=False)
```

	Predictor	importancia
11	NIHSS_alta_cat	0,159261
2	EDAD	0,103086
9	NIHSS INICO ACV	0,080756
5	TRIGLICERIDOS	0,078785
10	NIHSS alta ACV	0,077216
7	CONTEO G.B.	0,066683
3	GLUCOSA	0,056563
8	GLASGOW AL INICO ACV	0,056080
6	INR	0,055534
1	DIABETES	0,054373
4	COL. TOTAL	0,053273
17	GLUCOSA_cat	0,051421
15	TRIGLICERIDOS_cat	0,032218
13	CONTEO G.B._cat	0,023646
12	GLASGOW_cat	0,010939
22	NIHSS_INICIO_cat_Leve (Trombolisando)	0,008097
23	NIHSS_INICIO_cat_Moderado (Buen Pronostico)	0,007339
21	NIHSS_INICIO_cat_Grave	0,004856
16	COL. TOTAL_cat	0,004865
0	HTA	0,004678
14	INR_cat	0,003802
20	NIHSS_INICIO_cat_Déficit Mínimo	0,002603
18	EDAD_cat	0,002603
19	NIHSS_INICIO_cat_Déficit Importante	0,001010
24	NIHSS_INICIO_cat_Sin Déficit	0,000000

Tabla 5.10: Importancia de los predictores Random Forest

[65]:
```
# Get numerical feature importances
importances = list(rfc.feature_importances_)
# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for␣
↪feature, importance in zip(feature_list, importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key =␣
↪lambda x: x[1], reverse = True)
```

```
# Reset style
plt.style.use('fivethirtyeight')

# lista de x ubicaciones para trazar
x_values = list(range(len(importances)))

# Gráfico de barras
plt.bar(x_values, importances, orientation = 'vertical',␣
 ↪color = 'r', edgecolor = 'k', linewidth = 1.2)

# Marque las etiquetas para el eje x
plt.xticks(x_values, feature_list, rotation='vertical')

# Etiquetas de eje y título
plt.ylabel('Importancia'); plt.xlabel('Variable'); plt.
 ↪title('Importancia de Variables');
```

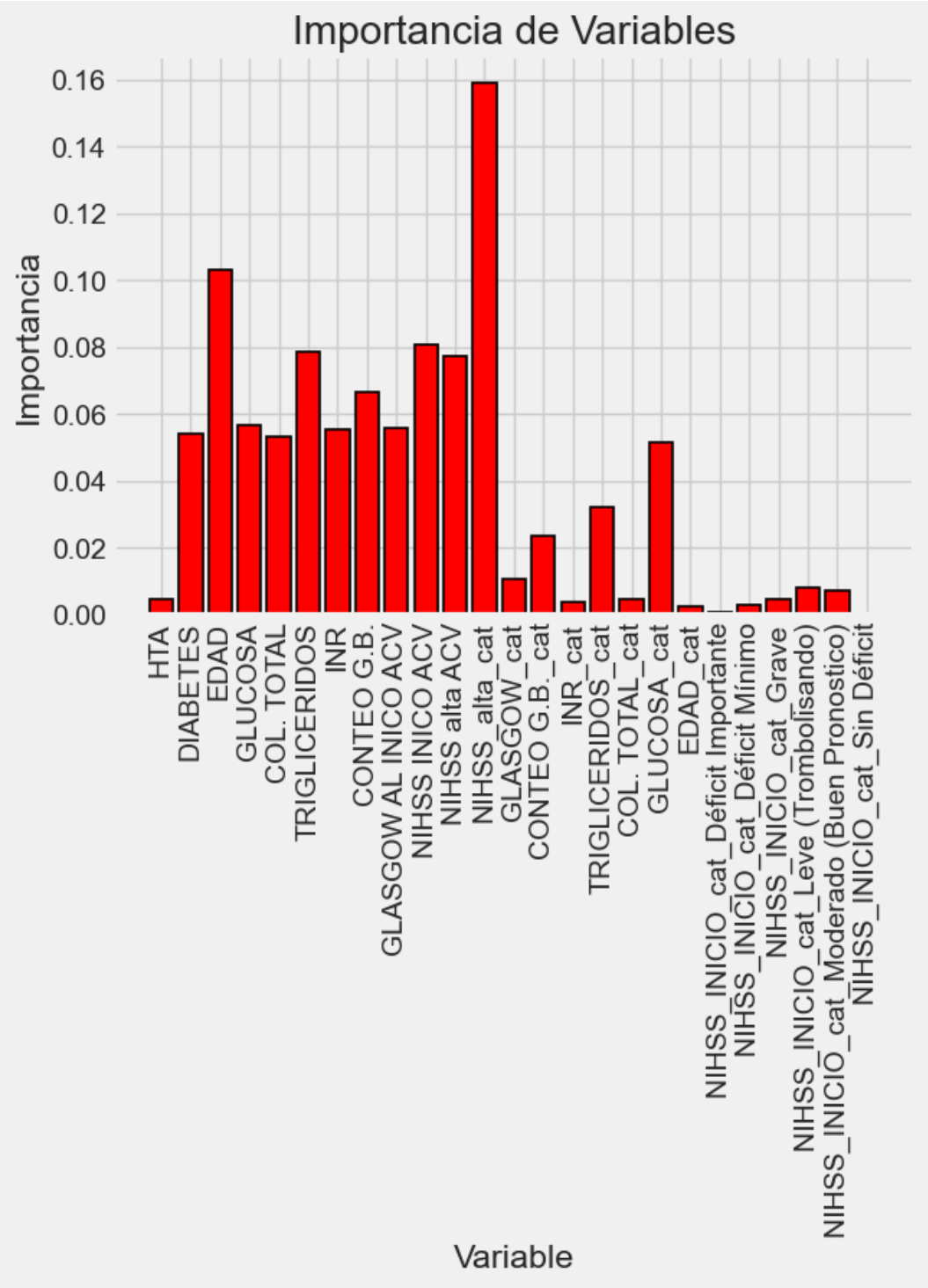

Figura 5.7: Importancia de los predictores en Random Forest

Las variables que presentan gran importancia al momento de la predicción según la tabla 5.10 y la Figura 5.7 son “NIHSS_alta_cat”, ”EDAD” y “NIHSS alta ACV”, “CONTEO G.B._cat” con más del 34 % de importancia en la predicción. El mayor predictor es “NIHSS_alta_cat” con un 15,92 % de importancia para este

modelo.

5.8.2. Importancia acumulada

Al igual que, en la sub sección 5.6.2, reduciremos al 95 % de los datos.

[66]:
```
# Lista de funciones ordenadas de mayor a menor importancia
sorted_importances = [importance[1] for importance in␣
 ↪feature_importances]
sorted_features = [importance[0] for importance in␣
 ↪feature_importances]

# Importancias acumulativas
cumulative_importances = np.cumsum(sorted_importances)

# Haz un gráfico de líneas
plt.plot(x_values, cumulative_importances, 'g-')

# Dibujar línea al 95% de importancia retenida
plt.hlines(y = 0.95, xmin=0, xmax=len(sorted_importances),␣
 ↪color = 'r', linestyles = 'dashed')

# Formato x ticks y etiquetas
plt.xticks(x_values, sorted_features, rotation = 'vertical')

# Etiquetas de eje y título
plt.xlabel('Variable'); plt.ylabel('Importancia acumulada');
 ↪ plt.title('Importancia acumulada');
```

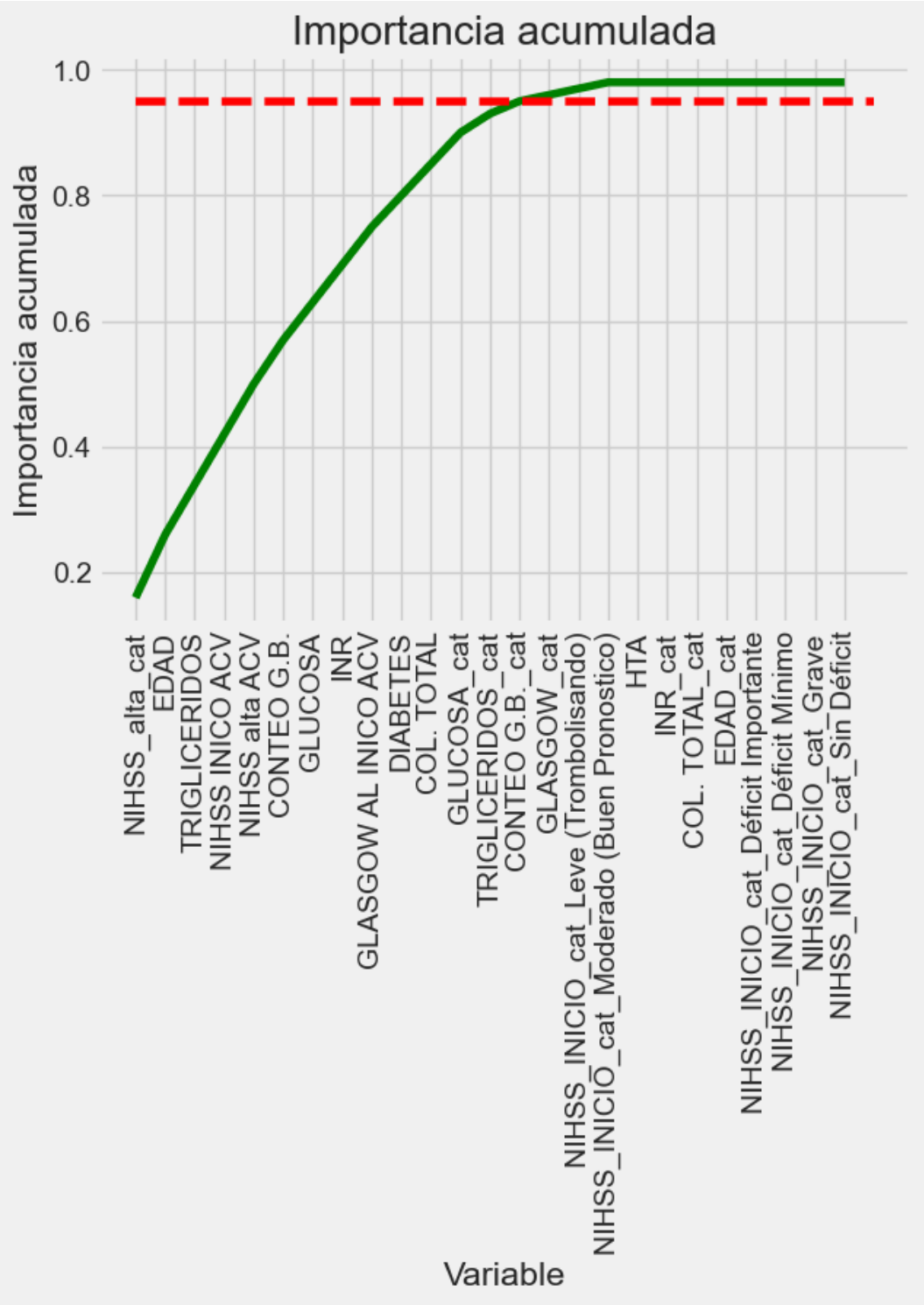

Figura 5.8: Importancia de los predictores acumulada en Random Forest

[67]:
```
# Encuentre el número de características para una␣
 ↪importancia acumulada del 95%
# Agregue 1 porque Python está indexado a cero
```

```
print('Número de columna para el 95 % de importancia:', np.
 ↪where(cumulative_importances > 0.95)[0][0] + 1)
```

```
Número de columna para el 95 % de importancia: 14
```

En la Figura 5.8 de importancia acumulada, la curva se dispara con la "CONTEO G.B._cat", lo que da con el resultado del 95 % de importancia medido anteriormente.

5.9. Comparativa de pronósticos

Para la investigación, como se dijo en párrafos anteriores, es de vital importancia conocer si los pacientes post ACV presentarán un buen pronóstico después de la evaluación con la escala NIHSS de alta. Como nuestra variable predictora está en un formato binario, es útil la interpretación de los resultados, asi podremos entender si los pacientes del hospital Herminda Martin tienen un porcentaje predictivo con un buen pronóstico o mal pronóstico, de esta manera, la variable nos dirá el futuro del paciente con un estado 0 de buen pronóstco o 1 de mal pronóstico. La exhaustividad, precisión y el F1 son los que serán evaluados para decidir cuál algoritmo es más compatible con los datos procesados del tests.

5.9.1. Pronóstico favorable post ACV

En la sub sección 4.5.5 pudimos visualizar cómo asociamos las clasificaciones de la escala NIHSS a la variable binaria actual. A continuación, analizaremos el estado 0 de la variable binaria, en donde los pacientes tienen un pronóstico favorable post ACV.

[7]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
data = pd.DataFrame({'Precisión' : [55, 75, 83, 60],
                     'Exahustividad': [100, 100, 83, 100],
                     'F1': [71, 86, 83, 75]},
                    index=('Naïve Bayes', 'Logistic␣
↪Regression', 'Decision Tree', 'Random Forest'))
total = data.sum(axis=1)
fig, ax = plt.subplots()
ax.set_ylabel('Cantidad de Puntos')
ax.set_title('Suma de métricas para variable favorable')

rects1 = ax.bar(x - width/2, total, width)

def autolabel(rects):
    """Funcion para agregar una etiqueta con el valor en␣
↪cada barra"""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() /␣
↪2, height),
                    xytext=(0, 3),  # 3 points vertical␣
↪offset
                    textcoords="offset points",
                    ha='center', va='bottom')

#Añadimos las etiquetas para cada barra
autolabel(rects1)
plt.bar(total.index, total)
plt.show()
```

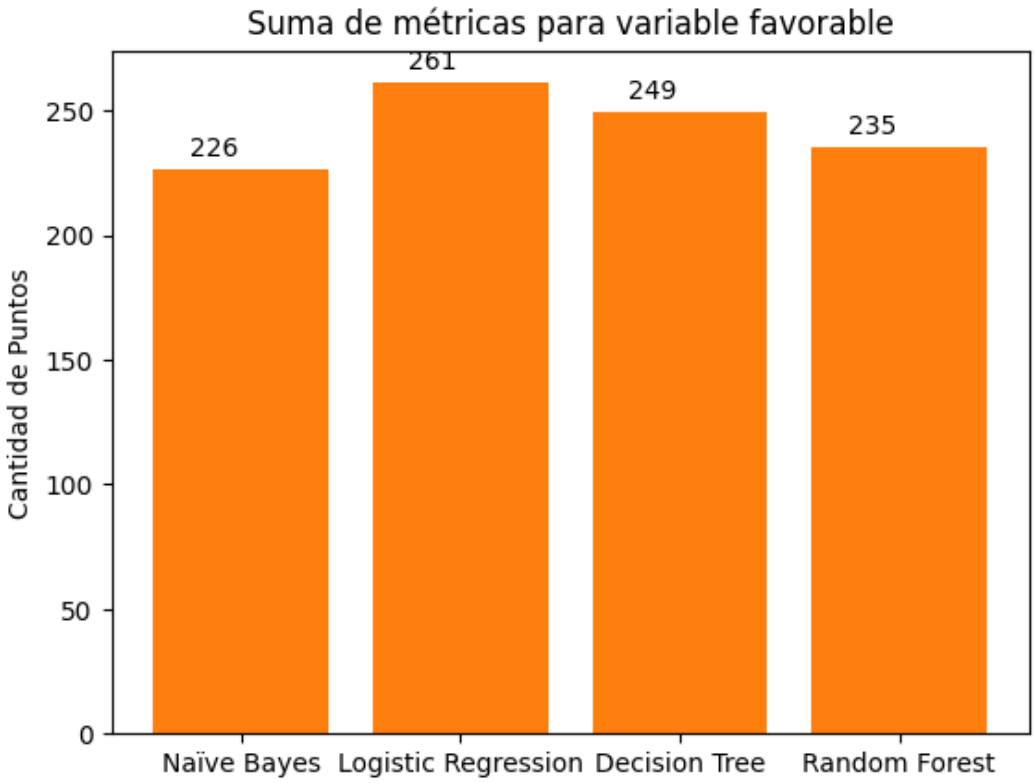

Figura 5.9: Sumatoria de métricas en pronóstico favorable

Como se demuestra en el Figura 5.9, la suma de los valores de las métricas de predicción en porcentaje supera los 225, pero no alcanza los 300 que es el máximo para todos los valores de las métricas. El peor algoritmo en la suma de todas sus métricas es Naïve Bayes con 226 puntos, en cambio, el mejor algoritmo con un total de 261 puntos es Logistic Regression.

[71]:
```
n = len(data.index)
x = np.arange(n)
width = 0.25


fig, ax = plt.subplots()
#Añadimos las etiquetas de identificacion de valores en el␣
 ↪gráfico
ax.set_title('Comparativa de rendimiento en la predicción␣
 ↪del Pronóstico')
```

```
plt.bar(x - width, data.Precisión, width=width,␣
 ↪label='Precisión')
plt.bar(x, data.Exahustividad, width=width,␣
 ↪label='Exahustividad')
plt.bar(x + width, data.F1, width=width, label='F1')
plt.xticks(x, data.index)
plt.legend(loc='best')
plt.show()
```

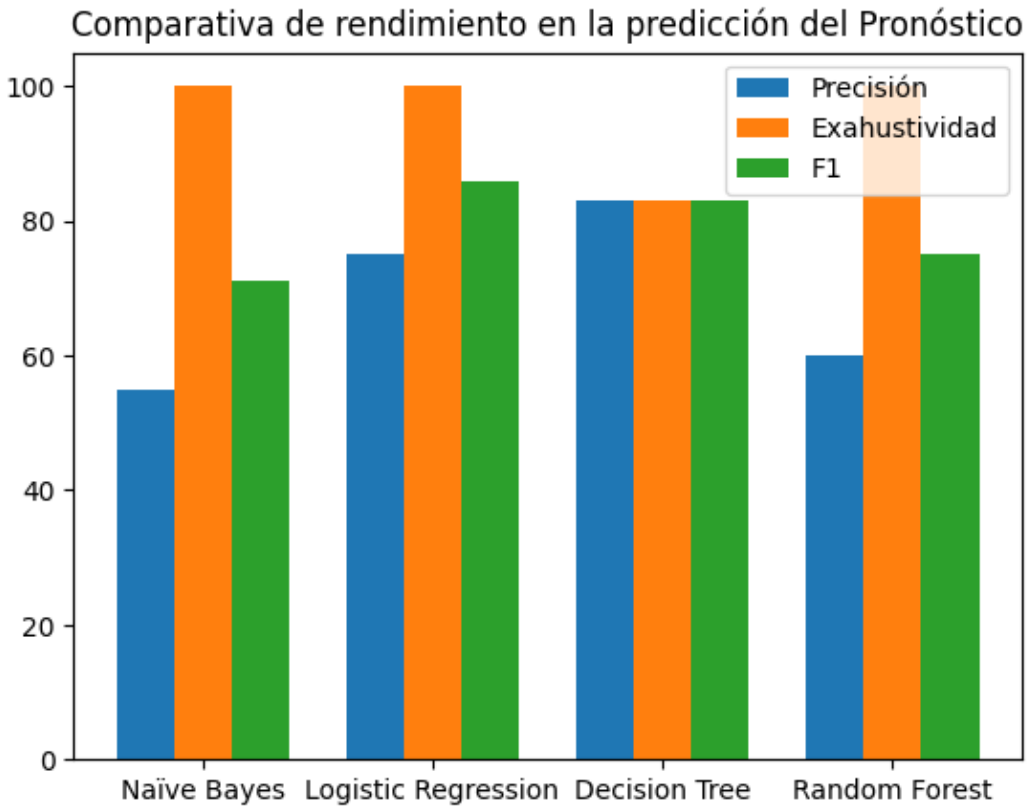

Figura 5.10: Comparativa de métrica en pronóstico favorable

La Figura 5.10 muestra los valores de Predicción, Exhaustividad y F1 en cada uno de los algoritmos. Se demuestra con el estado 0 que el algoritmo más preciso fue Decision Tree con un 83 % y el menos preciso fue Naïve Bayes con un 55 %. En la Exhaustividad el peor algoritmo fue Decision Tree con un 83 % y los demás quedaron empatados con un 100 % de exhaustividad. El F1 peor fue para Naïve

Bayes y el mejor fue Logistic Regression. Como resultado final, en la comparación del pronóstico favorable del paciente, el mejor algoritmo por una difencia mínima es para Logistic Regression. El resultado se tomó por la suma de su métricas, aunque el algoritmo más estable fue Decisión Tree, siendo destacable en su desempeño.

5.9.2. Pronóstico menos favorable post ACV

En la sub sección 4.5.5 pudimos visualizar como asociamos las clasificaciones de la escala NIHSS a la variable binaria actual. A continuación, analizaremos el estado 1 de la variable binaria, en donde los pacientes tienen un pronóstico menos favorable post ACV.

[9]:
```
data = pd.DataFrame({'Precisión' : [100, 100, 83, 100],
                     'Exahustividad': [17, 67, 83, 33],
                     'F1': [29, 80, 83, 50]},
                    index=('Logistic Regression', 'Naïve␣
 ↪Bayes', 'Decision Tree', 'Random Forest'))

total = data.sum(axis=1)
rects1 = ax.bar(x - width/2, total, width)
fig, ax = plt.subplots()
ax.set_ylabel('Cantidad de Puntos')
ax.set_title('Suma de métricas para variable menos␣
 ↪favorable')
autolabel(rects1)
plt.bar(total.index, total)
plt.show()
```

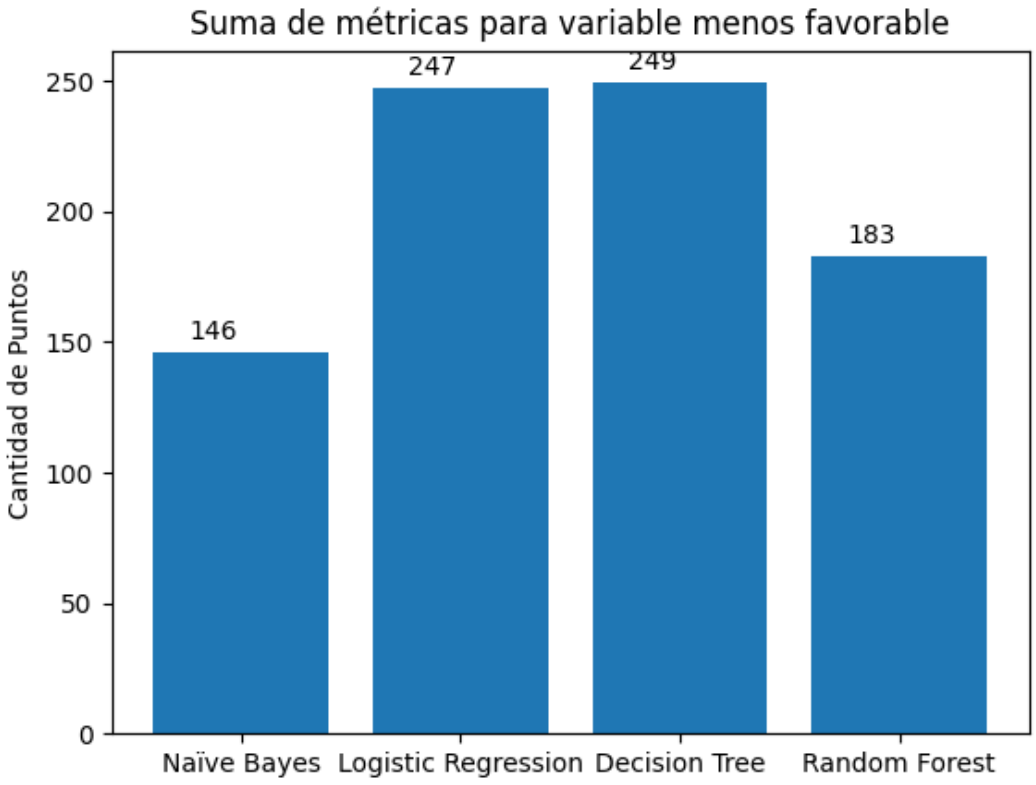

Figura 5.11: Sumatoria de métricas en pronóstico menos favorable

Como se demuestra en la Figura 5.11, la suma de los valores del estado 1 de las métricas de predicción en porcentaje supera los 145, pero no alcanzan los 300 que es el máximo para todos los valores de las métricas. El peor algoritmo en la suma de todas sus métricas es la Naïve Bayes con 146 puntos, en cambio el mejor algoritmo con un total de 249 puntos es Decision Tree. En este caso los algoritmos en su mayoría fueron menos efectivos al momento de predecir en contraparte con el otro estado.

[83]:
```
n = len(data.index)
x = np.arange(n)
width = 0.25


fig, ax = plt.subplots()
#Añadimos las etiquetas de identificacion de valores en el␣
 ↪gráfico
```

```
ax.set_title('Comparativa de rendimiento en la predicción␣
 ↪del Pronóstico')
plt.bar(x - width, data.Precisión, width=width,␣
 ↪label='Precisión')
plt.bar(x, data.Exahustividad, width=width,␣
 ↪label='Exahustividad')
plt.bar(x + width, data.F1, width=width, label='F1')
plt.xticks(x, data.index)
plt.legend(loc='best')
plt.show()
```

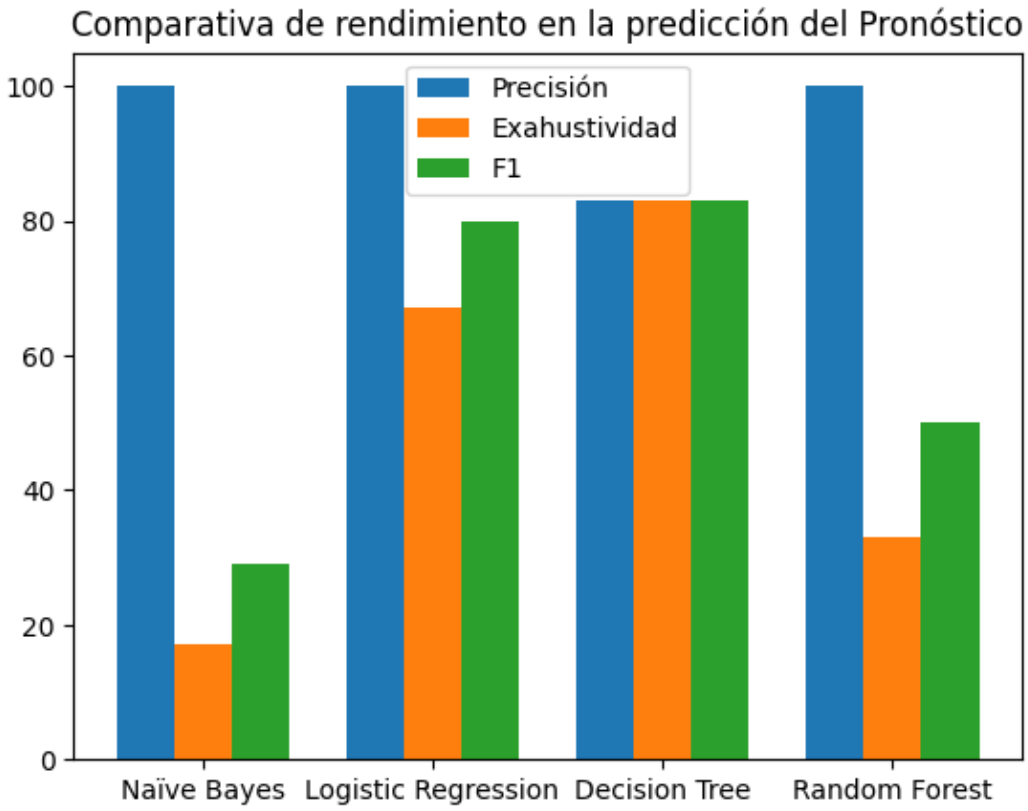

Figura 5.12: Comparativa de métricas en pronóstico menos favorable

La Figura 5.12 muestra los valores de Predicción, Exhaustividad y F1 en cada uno de los algoritmos. Se demuestra con el estado 1 que el algoritmo más preciso fue un triple empate con un 100 %, solo dejando al menos preciso Decision Tree con

un 83 %. En la Exhaustividad el peor algoritmo fue Naïve Bayes con un 17 % y el mejor fue Decision Tree con un 83 %. El F1 peor fue para Naïve Bayes con un 29 % y el mejor fue Decision Tree. Como resultado final, en la comparación del pronóstico menos favorable del paciente, el mejor algoritmo para el estado 1 es Decision Tree.

5.10. Comparación Final

En la comparación final se evalúa el porcentaje de predicción acumulada para la variable predictora en sus dos estados. Se hace hincapié que el resultado muestra el reflejo de los datos obtenidos y que los resultados pueden cambiar dependiendo de la BDD.

[82]:
```
precisiones = [59.4, 58.3, 83.3, 62.1]


total = data.sum(axis=1)
rects1 = ax.bar(x - width/2, precisiones, width)
fig, ax = plt.subplots()
ax.set_ylabel('Cantidad de Puntos')
ax.set_title('Predicción Acumulada')
autolabel(rects1)
plt.bar(data.index, precisiones)
plt.show()
```

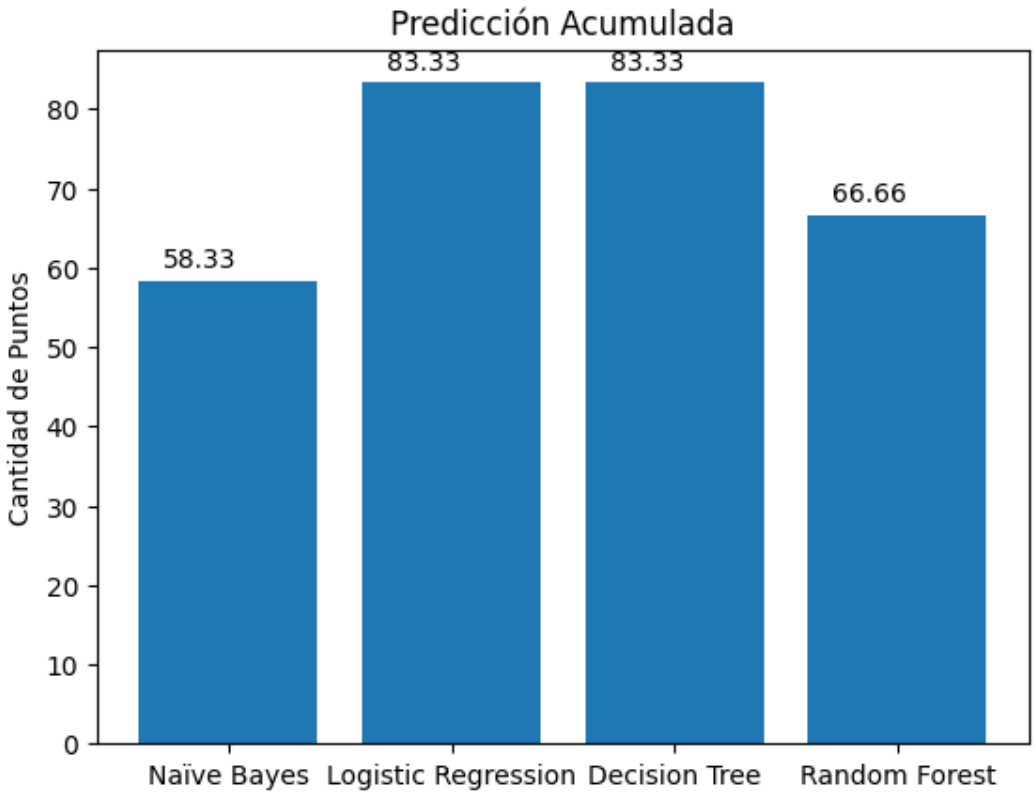

Figura 5.13: Predicción Acumulada

Los resultados arrojados por el gráfico 5.13 fueron los siguientes:

- El peor algoritmo con la tasa de predicción más baja es Naïve Bayes, con un 58.33 %.
- El antepenúltimo puesto es para Random Forest, con un 66,66 % de predicción.
- El segundo puesto lo ocupa Logistic Regression, con un 83,33 % de predicción.
- El mejor algoritmo es Decision Tree, con un 83 % de predicción.

Decision Tree y Logistic Regression quedaron empatados en las métricas de precisión general, la diferencia se produjo en el estado de la variable dicotómica. En el estado de pronóstico favorable, Logistic Regression fue superior por poco, aunque Decision Tree fue más estable. Por el contrario, en el estado de pronóstico menos favorable fue Decision Tree y el algoritmo siguió manteniendo la estabilidad presentada en el estado anterior de la variable.

Decision Tree logró alcanzar estabilidad en sus métricas y lo hicieron el algoritmo más preciso para este sistema de clasificación.

Capítulo 6

DISCUSIONES, CONCLUSIONES Y RECOMENDACIONES

En la investigación se realizó un estudio sobre exámenes o resultados de pacientes post Accidente Cerebrovascular del Hospital Herminda Martin de Chillán por medio de técnicas de ML. La recopilación de datos de los pacientes fue una colaboración entre la investigadora e Informática Dra. Carolina Figueroa con el médico e investigador Dr. Carlos Escudero.

Por medio del trabajo, se concluye que existe relación entre el estado de alta del paciente y los datos que están en la base de datos, lo cual ha sido confirmado por los modelos de ML. Asimismo, los resultados son menores en relación con investigaciones anteriores, las cuales fueron expuestas en los trabajos relacionados, bordeando el promedio de predicción en un 90 % para las otras investigaciones y el mejor modelo elaborado en este trabajo un 83 %. Este resultado prueba la validez de la hipótesis de la investigación, asimismo, permite determinar la técnica de ML más precisa, con el fin de clasificar según el tipo de secuela a los pacientes que han sufrido ACV Isquémico del Hospital Herminda Martín. En otras palabras, la clasificación para conocer si un paciente tendrá un pronóstico de secuelas que le permitan tener un buen diario vivir o no, a través del modelo más confiable fue Decision Tree con un 83 % de probabilidades de precisión.

Durante la investigación de las técnicas clásicas de ML se lograron estudiar varios modelos que fueron expuestos en el capítulo 2 (Marco Teórico) y el capítulo 3 (Trabajos Relacionados). En las técnicas con algoritmos supervisados, existían más trabajos orientados al ámbito de salud. Debe señalarse que el DL era la mejor opción, pero no se podía aplicar por la escasez de datos de la BDD.

En resumidas cuentas, las técnicas eran muy variadas y su utilización estaba optimizada en muchos casos con hiperparametros, aunque los modelos que más aparecieron en la literatura fueron Logistic Regression, Naive Bayes, Decision Tree y Random Forest. Cabe destacar que estos algoritmos se desenvuelven bien en problemáticas de salud y algunos actúan mejor con variables dicotómicas para una mejor predicción.

Se estableció que el mecanismo para que el modelo llegará a poder predecir debía ser sacado de la literatura e investigaciones en salud de ML de código libre, en este caso particular, se guió la investigación por el libro "Machine Learning in Action" en su metodología y varios procesos de clasificación de datos.

Se determinó que las variables de Hipertensión y Diabetes eran significativas para la investigación, descrito en la sub sección 4.6.2, su densidad y cantidad orientan a patrones marcados, dando como conclusión, que los pacientes con esas enfermedades pueden sufrir con más tendencia un ACV. Se demostró que las variables creadas en el proceso tenían una similitud parecida en importancia. Dependiendo el modelo hay variables que poseían mayor importancia para la predicción, como es el ejemplo de Decision Tree que contaba con la importancia más alta en la variable creada y procesada de "NIHSS_alta_cat" con un 41,98 %, la seguía "NIHSS alta ACV" con un 21,58 %. En cambio, Random Forest su variable más importante era "NIHSS_alta_cat" con un 15,92 % y lo seguía la "EDAD" con un 10,3 %.

En cuanto a la comparativa de las técnicas de ML, el estado de Pronóstico Favorable, el ganador fue "Logistic Regression" con 261 puntos de 300 totales,

destacando que los resultados estuvieron parejos en este estado. En cambio, el estado de Pronóstico menos Favorable el ganador fue "Decision Tree", arrasando ante sus competidores con 249 puntos sobre 300. En la comparación final, se tomo en cuenta la precisión global y estabilidad del algoritmo con sus demás métricas, como lo detalla la sección 5.10 de comparativa final de algoritmos, el cual arrojo el primer lugar "Decision Tree" con un 83,33 % sobre su competidor más cercano Logistic Regression con un 83,33 % de predicción de los datos, pero en estabilidad "Decision Tree" fue muy lejano.

En el Procesamiento de los datos se utilizaron técnicas de clasificación de Missing Data, clasificaciones propias de las variables, Label Encoding, entre otras, que proponen disminuir la perdida de datos, darle etiquetas a los valores y las etiquetas asignarles valores numéricos, respectivamente. El One-Hot Encoding es una técnica que en la mayoría de las modelos de ML de clasificación se usa como paso anterior al entrenamiento de los datos en el modelo, eliminando la variable codificada entera (dos o más valores de esa variable) y se agrega una nueva variable binaria para cada valor entero único, ayudando a que las variables pasen a un estado binario. Cabe concluir, que este trabajo se enfocó en representar a los pacientes de una forma dicotómica en una esperanza favorable o menos favorable según la escala NIHSS, contemplando el daño neurológico en los pacientes post ACV Isquémico, dando como resultado general que la mayoría de los pacientes post ACV Isquémico tendrán un pronóstico favorable cuando están de alta del Hospital Herminda Martín, enfatizamos que el alta no indica que el paciente se encuentre bien de salud, ya que el daño neurológico corresponde a perdida cognoscitiva o motor. Señalamos que aunque la muestra fue pequeña, se pueden obtener resultados en las posibles variables de importancia para nuevos trabajos investigativos en el campo de la medicina del ACV y ML.

Finalmente, los aportes de la IA a la medicina pueden ser un cambio que favorecerá para la respuesta más instantánea de la toma de decisiones de los médicos, pero debe haber una cooperación entre ambos campos de manera abierta, con inversiones para la salud como para la IA, contando con equipos de trabajos y bases de

datos gigantescas, bien clasificadas como se menciona en los trabajos relacionados, esto llevará a poder dar un mejor pronóstico teniendo en cuenta que los equipos multidisciplinarios realizan grandes avances e involucran más aspectos que son necesarios para evaluar el sistema completo que compone al ser humano.

Bibliografía

[1] C. F. Flores, *Visual Saliency for Object Recognition, and Object Recognition for Visual Saliency*. 2 2021.

[2] I. Ortiz-Galeano, N. E. F. Balmaceda, and A. Flores, "Cardiovascular risk factors in patients with stroke," *Revista Virtual de la Sociedad Paraguaya de Medicina Interna*, vol. 7, pp. 50–55, 3 2020.

[3] C. M. de Salud, "Ataque cerebrovascular - ministerio de salud - gobierno de chile," 2022.

[4] J. Gaudiano, D. Graña, M. Goñi, V. Colina, A. Cosentino, R. Pensado, V. Ruglio, M. Scaron, L. Vidart, J. Gaudiano, D. Graña, M. Goñi, V. Colina, A. Cosentino, R. Pensado, V. Ruglio, M. Scaron, and L. Vidart, "Epidemiológica del ataque cerebro vascular en un hospital universitario," *Revista Uruguaya de Medicina Interna*, vol. 4, pp. 24–31, 7 2019.

[5] E. C. Rodríguez, A. S. Fernández, A. G. González, M. G. Camejo, and Y. F. Viltres, "Características de los pacientes con enfermedad cerebrovascular," *Multimed*, 2020.

[6] S. de Salud de Ñuble, "El 80 % de los casos de ataque cerebrovascular se pueden prevenir," 10 2018.

[7] W. H. Curioso and M. J. Brunette, "Artificial intelligence and innovation to optimize the tuberculosis diagnostic process," *Revista Peruana de Medicina Experimental y Salud Publica*, vol. 37, pp. 554–558, 7 2020.

[8] S. Grazia and T. Vizoso, "Reflexiones sobre la inteligencia artificial y la bibliotecología reflections on artificial intelligence and librarianship," 2022.

[9] M. A. Moreno Herrera and M. G. Reyes Maldonado, *Diseño de una aplicación de data science para la predicción del rendimiento académico de los estudiantes de la Unidad Educativa "José Elías Altamirano", año 2021*. PhD thesis, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas . . . , 2021.

[10] A. L. Samuel, "Machine learning," *The Technology Review*, vol. 62, no. 1, pp. 42–45, 1959.

[11] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1. McGraw-hill New York, 1997.

[12] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Interpretable machine learning: definitions, methods, and applications," *arXiv preprint arXiv:1901.04592*, 2019.

[13] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques third edition," *University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University*, 2012.

[14] P. Harrington, *Machine Learning in Action*. manning publications ed., 2012.

[15] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[16] F. Cánovas-García, F. Alonso-Sarría, F. Gomariz-Castillo, and F. Oñate-Valdivieso, "Modification of the random forest algorithm to avoid statistical dependence problems when classifying remote sensing imagery," *Computers & Geosciences*, vol. 103, pp. 1–11, 2017.

[17] K. Vembandasamy, R. Sasipriya, E. D. I. J. of, and undefined 2015, "Heart diseases detection using naive bayes algorithm," *ijiset.com*, vol. 2, 2015.

[18] J. Bell, *Machine Learning: Hands-On for Developers and Technical Professionals*. Indianapolis, IN: Wiley, 2015.

[19] J. C. Stoltzfus, "Logistic regression: A brief primer," *Academic Emergency Medicine*, vol. 18, pp. 1099–1104, 10 2011.

[20] D. Dantas, M. De, C. Nunes, S. Terra, L. Paulo, B. Schorr, and N. Calegario, "Machine learning for carbon stock prediction in a tropical forest in southeastern brazil," *BOSQUE*, vol. 42, pp. 131–140, 2021.

[21] R. Salas, "Redes neuronales artificiales," *Universidad de Valparaıso. Departamento de Computación*, vol. 1, pp. 1–7, 2004.

[22] J. F. P. BARRERA, I. Mecatrónica, D. A. Hurtado, and R. J. Moreno, "Prediction system of erythemas for phototypes i and ii, using deep-learning sistema de predicciÓn de eritema para fototipo i y ii, utilizando deep-learning," vol. 22, pp. 188–196, 2015.

[23] C. Arana, "Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales," tech. rep., Serie Documentos de Trabajo, 2021.

[24] J. A. Peña-Torres, R. E. Gutiérrez, V. A. Bucheli, and F. A. González, "How to adapt deep learning models to a new domain: The case of biomedical relation extraction cómo adaptar un modelo de aprendizaje profundo a un nuevo dominio: el caso de la extracción de relaciones biomédicas how to adapt deep learning models to a new domain: The case of biomedical relation extraction," vol. 22, pp. 49–62, 2019.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[27] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," 2006.

[28] V. Astudillo and D. Revelo, "Apoyo al diagnóstico de neumonía y detección de opacidades pulmonares usando segmentación de instancias semánticas en imágenes de rayos x de tórax," vol. 39, pp. 259–274, 2021.

[29] F. Provost and T. Fawcett, *Data Science for Business: What you need to know about data mining and data-analytic thinking*. .º'Reilly Media, Inc.", 2013.

[30] C. G. Alfonso, A. E. M. Reyes, V. García, A. R. Fajardo, I. Torres, and J. C. Casas, "Actualización en diagnóstico y tratamiento del ataque cerebrovascular isquémico agudo," *Universitas Médica*, vol. 60, pp. 1–17, 6 2019.

[31] H. P. Adams, B. H. Bendixen, L. J. Kappelle, J. Biller, B. B. Love, D. L. Gordon, and E. E. Marsh, "Classification of subtype of acute ischemic stroke. definitions for use in a multicenter clinical trial. toast. trial of org 10172 in acute stroke treatment.," *Stroke*, vol. 24, pp. 35–41, 1 1993.

[32] J. M. Seguin, A. B. Vena, L. C. Campàs, I. Benalbdelhak, and F. P. García, "Revisión sistemática de las características y pronóstico de los sujetos que sufren un ictus criptogénico no lacunar de mecanismo embólico," *Revista de Neurología*, vol. 66, p. 325, 2018.

[33] R. A. Radu, E. O. Terecoasă, O. A. Băjenaru, and C. Tiu, "Etiologic classification of ischemic stroke: Where do we stand?," *Clinical Neurology and Neurosurgery*, vol. 159, pp. 93–106, 8 2017.

[34] M. Wintermark, P. C. Sanelli, G. W. Albers, J. Bello, C. Derdeyn, S. W. Hetts, M. H. Johnson, C. Kidwell, M. H. Lev, D. S. Liebeskind, H. Rowley, P. W. Schaefer, J. L. Sunshine, G. Zaharchuk, and C. C. Meltzer, "Imaging recommendations for acute stroke and transient ischemic attack patients: A joint statement by the american society of neuroradiology, the american college of radiology, and the society of neurointerventional surgery," 2013.

[35] F. B. Pareja, J. D. Guzmán, and J. Porta-Etessam, "Cien escalas de interés en neurología clínica," 2001.

[36] J. Montaner and J. Álvarez-Sabín, "La escala de ictus del national institute of health (nihss) y su adaptación al español," *Neurología (Barc., Ed. impr.)*, pp. 192–202, 2006.

[37] A. W. Heinemann, R. L. Harvey, J. R. McGuire, D. Ingberman, L. Lovell, P. Semik, and E. J. Roth, "Measurement properties of the nih stroke scale during acute rehabilitation," *Stroke*, vol. 28, no. 6, pp. 1174–1180, 1997.

[38] M. R. Frankel, L. Morgenstern, T. Kwiatkowski, M. Lu, B. Tilley, J. Broderick, R. Libman, S. Levine, T. Brott, *et al.*, "Predicting prognosis after stroke: a placebo group analysis from the national institute of neurological disorders and stroke rt-pa stroke trial," *Neurology*, vol. 55, no. 7, pp. 952–959, 2000.

[39] M. Nagendran, Y. Chen, C. A. Lovejoy, A. C. Gordon, M. Komorowski, H. Harvey, E. J. Topol, J. P. A. Ioannidis, G. S. Collins, and M. Maruthappu, "Artificial intelligence versus clinicians: systematic review of design, reporting standards, and claims of deep learning studies," *BMJ*, p. m689, 3 2020.

[40] J. N. Heo, J. G. Yoon, H. Park, Y. D. Kim, H. S. Nam, and J. H. Heo, "Machine learning–based model for prediction of outcomes in acute stroke," *Stroke*, vol. 50, pp. 1263–1265, 5 2019.

[41] D. Scrutinio, C. Ricciardi, L. Donisi, E. Losavio, P. Battista, P. Guida, M. Cesarelli, G. Pagano, and G. D'Addio, "Machine learning to predict mortality after rehabilitation among patients with severe stroke," *Scientific Reports 2020 10:1*, vol. 10, pp. 1–10, 11 2020.

[42] J. Yu, S. Park, H. Lee, C. S. Pyo, and Y. S. Lee, "An elderly health monitoring system using machine learning and in-depth analysis techniques on the nih stroke scale," *Mathematics 2020, Vol. 8, Page 1115*, vol. 8, p. 1115, 7 2020.

[43] Y. Xie, B. Jiang, E. Gong, Y. Li, G. Zhu, P. Michel, M. Wintermark, and G. Zaharchuk, "Use of gradient boosting machine learning to predict patient outcome in acute ischemic stroke on the basis of imaging, demographic, and clinical information," *https://doi.org/10.2214/AJR.18.20260*, vol. 212, pp. 44–51, 10 2018.

[44] S. Pang, Z. Yu, and M. A. Orgun, "A novel end-to-end classifier using domain transferred deep convolutional neural networks for biomedical images," *Computer methods and programs in biomedicine*, vol. 140, pp. 283–293, 3 2017.

[45] R. Bioética, H. D. C. Nunes, R. M. C. Guimarães, and L. Dadalto, "Desafios bioéticos do uso da inteligência artificial em hospitais," *Revista Bioética*, vol. 30, pp. 82–93, 5 2022.

[46] C. S. Shao, Z. X. Xiong, C. D. Dong, C. Dong, and J. L. Li, "An improved deep learning model for traffic crash prediction," 2018.

[47] H. Bar, "Missing data — mechanisms and possible solutions / datos ausentes: mecanismos y posibles soluciones," *https://doi.org/10.1080/11356405.2017.1365426*, vol. 29, pp. 492–525, 2017.

[48] "Conteo de glóbulos blancos - serie—resultados: Medlineplus enciclopedia médica."

[49] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *Journal of Big Data*, vol. 7, pp. 1–41, 12 2020.

[50] R. E. Bank and C. C. Douglas, "Sparse matrix multiplication package (smmp)," *Advances in Computational Mathematics*, vol. 1, pp. 127–137, 2 1993.

[51] R. Mosquera, O. D. Castrillón, L. Parra, R. Mosquera, O. D. Castrillón, and L. Parra, "Máquinas de soporte vectorial, clasificador naïve bayes y algoritmos genéticos para la predicción de riesgos psicosociales en docentes de colegios públicos colombianos," *Información tecnológica*, vol. 29, pp. 153–162, 12 2018.