Predicción de probabilidad de diabetes basada en atributos relacionados con los hábitos de alimentación y condición de salud de los pacientes

Ana Estefanía Henao Restrepo
Juan José Gil Hoyos

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor
Angelower Santana Velásquez, Magíster (MSc)

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2023

Cita	(Henao Restrepo & Gil Hoyos, 2023)
Referencia **Estilo APA 7 (2020)**	Henao Restrepo, A.E., & Gil Hoyos, J. J. (2023). *Predicción de probabilidad de diabetes basada en atributos relacionados con los hábitos de alimentación y condición de salud de los pacientes* Trabajo de grado especialización]. Universidad de Antioquia, Medellín, Colombia.

Especialización en Analítica y Ciencia de Datos, Cohorte V.

Centro de Investigación Ambientales y de Ingeniería (CIA).

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.
Decano: Julio Cesar Saldarriaga Molina
Jefe departamento: Diego José Luis Botia Valderrama

PREDICCIÓN DE PROBABILIDAD DE DIABETES BASADA EN ATRIBUTOS RELACIONADOS CON LOS HÁBITOS DE ALIMENTACIÓN Y CONDICIÓN DE SALUD DE LOS PACIENTES

Dedicatoria

Queremos dedicar este hito educativo a todas las personas que de una u otra manera, nos ayudaron a refinar detalles para hoy podernos convertir en especialistas en Analítica y Ciencia de Datos. Yo, Juan José, quiero darle un agradecimiento muy especial a Ana, no solo por ser la mejor compañera de equipo que pude haber encontrado, sino también por ser mi confidente y mi compañera de vida. Yo, Ana Estefanía, quiero agradecer a Juan por haber sido el mejor compañero de equipo, por todo su apoyo y paciencia y por ser mi compañero de vida.

Agradecimientos

Primero que todo, queremos agradecer a Dios todo poderoso por permitirnos finalizar este programa de especialización exitosamente. También queremos agradecer a nuestras familias por inculcarnos esas ganas insaciables de aprender todos los días desde pequeños. Finalmente, queremos agradecer a nuestro asesor durante este proyecto de monografía, el cual se encargó de revisar nuestro documento final en reiteradas ocasiones y nos hizo las correcciones del caso para poder hacer entrega de esta que es la última versión.

"La educación es el pasaporte hacia el futuro, el mañana pertenece a aquellos que se preparan para él en el día de hoy"

Malcom X

(1925-1965)

Tabla de contenido

Lista de tablas

Lista de figuras

Siglas, acrónimos y abreviaturas

Adaboost	Adaptive Boosting
AHA	American Heart Association
ANN	Artificial Neural Network
APA	American Psychological Association
API	Application Programming Interface
BMI	Body Mass Index
CDC	Centers for Disease Control and Prevention
CSR	Certificate Signing Request
FN	False Negative
FP	False Positive
GDPR	General Data Protection Regulation
HDL	High-Density Lipoprotein
HIPAA	Health Insurance Portability and Accountability Act
KNN	K-Nearest Neighbors
LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LDL	Low-Density Lipoprotein
LIME	Local Interpretable Model-agnostic Explanations
LOF	Local Outlier Factor
MB	Megabyte
ML	Machine Learning
MLP	Multilayer Perceptron
ROC	Receiver Operating Characteristic
SVC	Support Vector Classifier
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
XGBoost	Extreme Gradient Boosting

Resumen

El conjunto de datos *"Diabetes Prediction Competition (Diabetes Prediction Competition (TFUG CHD Nov 2022) | Kaggle, s. f.)"* se utilizó para optimizar los hiperparámetros de un modelo destinado a contribuir con la determinación de la probabilidad de diabetes en pacientes por parte de médicos tratantes. El *dataset* consta de 17 variables asociadas con el estado de salud general del paciente y una variable de respuesta de dos clases: paciente con diabetes (1) y sin diabetes (0). La cantidad de registros corresponde a la información de 80.692 pacientes.

Se evaluaron nueve modelos de aprendizaje supervisado, los cuales fueron: Regresión Logística, *KNeighbors Classifier, Naive Bayes Classifier, Random Forest Classifier, Support Vector Classifier, XGBoost Classifier, Adaboost Classifier*, una red neuronal *MLP Classifier* y otra desarrollada directamente usando la librería *Keras.*

Se enfrentaron desafíos como el alto costo computacional del *MLP Classifier*, con una ejecución de más de ocho horas, un tiempo de ejecución del algoritmo de validación cruzada de más de siete horas y la "maldición" de la dimensionalidad para el modelo KNN cuando se utilizaba la métrica de distancia *cosine*.

Entre los resultados más notables, se determinó que la presencia de datos atípicos no influyó significativamente en el rendimiento de los modelos. Finalmente, el modelo óptimo fue la red neuronal artificial desarrollada con *Keras* con un arreglo de capa inicial, cinco capas ocultas, seis capas de *dropout* y una capa final, 64 neuronas por cada capa oculta, función de activación tangente hiperbólica, optimizador Adam y tasa de *dropout* de 0.2.

Repositorio GitHub Ana Estefanía Henao Restrepo: https://rb.gy/hid3y
Repositorio GitHub Juan José Gil Hoyos: https://rb.gy/9i8qt

Palabras clave: Aprendizaje, supervisado, modelo, *Keras y F1-score*.

Abstract

The dataset "Diabetes Prediction Competition (*Diabetes Prediction Competition (TFUG CHD Nov 2022) | Kaggle, s. f.)"* was used to optimize the hyper parameters of a model intended to contribute to the determination of the probability of diabetes in patients by treating physicians. The dataset consists of 17 variables associated with the patient's general health status and a response variable of two classes: patient with diabetes (1) and without diabetes (0). The number of records corresponds to the information of 80,692 patients.

Nine supervised learning models were evaluated, which were: Logistic Regression, KNeighbors Classifier, Naive Bayes Classifier, Random Forest Classifier, Support Vector Classifier, XGBoost Classifier, Adaboost Classifier, an MLP Classifier neural network and another developed directly using the Keras library.

Challenges such as the high computational cost of the MLP Classifier, with an execution of more than eight hours, a cross-validation algorithm execution time of more than seven hours, and the "curse" of dimensionality for the KNN model when using the cosine distance metric were faced.

Among the most remarkable results, it was determined that the presence of outliers did not significantly influence the performance of the models. Finally, the optimal model was the artificial neural network developed with Keras with an initial layer array, five hidden layers, six drop out layers and a final layer, 64 neurons for each hidden layer, hyperbolic tangent activation function, Adam optimizer and drop out rate of 0.2.

GitHub repository Ana Estefanía Henao Restrepo: https://rb.gy/hid3y
GitHub Repository Juan José Gil Hoyos: https://rb.gy/9i8qt

Keywords: Learning, supervised, model, Keras and F1-score.

1. Descripción del problema

La diabetes se está convirtiendo en una enfermedad cada vez más común. Existe evidencia científica que establece que los casos de diabetes están incrementando dramáticamente en el mundo. Tanto así, que se la ha catalogado como una epidemia para la humanidad por parte del foro económico mundial durante el año 2020 (La diabetes y su corazón, 2021).

Factores como el consumo excesivo de alcohol, una dieta desbalanceada, poca actividad física y fumar, tienen una relación estrecha con la probabilidad de contraer diabetes tipo 2 (Olabiyi & Oguntibeju, 2013).

¿Qué implicaciones tiene para la salud en general la diabetes?

- Si tiene diabetes es mucho más probable que desarrolle una enfermedad cardiaca o derrame cerebral que aquellas que no la padecen (Story, 2023).
- La asociación americana de enfermedades cardiovasculares, (AHA por sus siglas en inglés), establece que la diabetes disminuye los niveles de colesterol HDL (también conocido como colesterol bueno), aumenta los niveles de triglicéridos y los niveles de colesterol LDL (también conocido como colesterol malo). Estos últimos incrementan el riesgo de padecer enfermedades cardiovasculares y derrames cerebrales (Síntomas y causas de la diabetes - NIDDK, s. f.).

Pero la diabetes no solo ataca la salud física sino también tiene grandes consecuencias sobre la salud mental:

- Hay medicinas que pueden alterar las células beta o alterar el funcionamiento de la insulina, como los medicamentos para enfermedades psiquiátricas (Sociedad Iberoamericana de Información Científica (SIIC), s. f.).
- Los enfermos con esquizofrenia tienen de 2 a 4 veces más riesgo de presentar diabetes tipo 2. Los antecedentes familiares de diabetes son más comunes en pacientes con esquizofrenia (La diabetes y la salud mental, 2023).
- Las personas con diabetes tienen entre 2 y 3 veces más probabilidades de presentar depresión que las personas sin diabetes (La diabetes y la salud mental, 2023).

Se pretende hacer uso del *dataset* extraído de la plataforma *Kaggle* para calcular la probabilidad de que un paciente padezca o no diabetes, basados en atributos relacionados con hábitos de alimentación y condición de salud.

1.1.Problema de negocio

En los hospitales se hace necesario contar con un modelo capaz de determinar la probabilidad de que un paciente contraiga diabetes según su condición de salud actual. A través de este proyecto de monografía, se propone una alternativa a las ya existentes que puede ser empleada por los doctores de manera muy intuitiva y cuyos resultados son interpretables tanto por el profesional en salud como por el directamente involucrado dentro del estudio.

1.2.Aproximación desde la analítica de datos

Se pretende brindar un modelo que sirva como herramienta capaz de apoyar a los médicos tratantes en la determinación de la probabilidad de contraer diabetes en pacientes a partir de ciertas características de su estado de salud.

1.3.Origen de los datos

Esta fuente de datos proviene de la limpieza de los resultados de una encuesta realizada por *Centers for Desease Control and Prevention (CDC)* por sus siglas en inglés, cuyo nombre fue *Behavioral Risk Factor Surveillance System*, la cual inicialmente contenía 441.456 registros y fue llevada a cabo durante el año 2015 (CDC - 2015 BRFSS Survey Data and documentation, s. f.). Esta información se adaptó para utilizarse en una de las competiciones que se generan dentro de la herramienta *Kaggle*. Su lanzamiento dentro de esta plataforma se produjo en noviembre de 2022.

La carpeta comprimida que contiene los archivos cargados durante la creación del reto consta de tres fuentes de datos en formato .csv. Sin embargo, solamente se utilizará el primero de estos como insumo durante la etapa de entrenamiento y validación de los modelos de clasificación considerados en este proyecto. Finalmente, se realizará una prueba definitiva con los datos remanentes del *dataset* que no se utilizaron ni en el entrenamiento ni en la validación para evaluar el desempeño del modelo que haya sido seleccionado como el óptimo durante la etapa de validación.

1.4.Métricas de desempeño

Como métricas de desempeño, para este caso de estudio se seleccionan el *Accuracy, Recall y el F1-score*. Sin embargo, la métrica que determinará la validez del modelo será el *F1-score*, ya que éste es una medida que involucra a las otras 2 métricas seleccionadas. Es necesario también analizar la matriz de confusión para determinar gráficamente los resultados en las predicciones de los modelos. Se selecciona un *F1-score* mayor o igual al 75% para considerar el modelo como viable.

Como métrica de negocio, se propone hacer uso del modelo de aprendizaje supervisado que se desarrolle, para que, a partir de un conjunto de datos recolectados de un paciente sea posible ofrecerle al médico tratante una herramienta de apoyo adicional a las ya existentes para determinar la probabilidad de que un paciente pueda padecer de diabetes en un tiempo determinado.

2. Objetivos

2.1.Objetivo general

Diseñar e implementar un modelo de aprendizaje automático supervisado utilizando una base de datos que contiene información demográfica y de hábitos de vida del paciente para establecer la probabilidad de que pueda padecer de diabetes.

2.2.Objetivos específicos

- Identificar patrones, relaciones y posibles pasos de preprocesamiento necesarios para optimizar el rendimiento de los modelos de aprendizaje automático supervisado a evaluar.

- Acondicionar el conjunto de datos utilizando diferentes técnicas de normalización y estandarización de variables.

- Encontrar los mejores hiperparámetros que permitan la generalización de los modelos, garantizando su confiabilidad en la determinación de la probabilidad de la aparición de diabetes en pacientes.

- Determinar el modelo óptimo utilizando una base de datos nueva a través de la aplicación de validación cruzada.

3. Datos

3.1. Datos originales

Dentro del *dataset*, existen 18 columnas, las cuales se dividen en 17 variables características y una sola variable de respuesta. Éstas se describen a continuación:

3.1.1. Variables categóricas

- ***Age*:** Esta variable está relacionada con los rangos etarios de los pacientes que diligenciaron la encuesta. La codificación para este atributo de la base de datos es la siguiente:

Tabla 1 Codificación de la variable *Age*

Codificación	Rango representativo de edad
1	18-24
2	25-29
3	30-34
4	35-39
5	40-44
6	45-49
7	50-54
8	55-59
9	60-64
10	65-69
11	70-74
12	75-79
13	80 o más

- ***GenHlth*:** Almacena la respuesta a la siguiente pregunta en una escala de uno a cinco: diría usted que en general su salud es: 1: excelente, 2: muy buena, 3: buena, 4: regular y 5: mala.

3.1.2. Variables categóricas binarizadas

- ***HighChol*:** Hace referencia a los niveles de colesterol en la sangre del paciente. 0: Dentro del límite normal y 1: Mayor al límite o fuera del rango.
- ***Sex:*** Corresponde con el género del paciente. 0: femenino y 1: masculino.
- ***CholCheck:*** Determina si el paciente ha tenido un examen para medir sus niveles de colesterol en la sangre en los últimos cinco años. 0: No y 1: Sí.
- ***Smoker*:** Almacena la respuesta a la pregunta: ¿Ha fumado al menos 100 cigarrillos en toda su vida? 0: No y 1: Sí.
- ***HeartDiseaseorAttack*:** Hace referencia a la respuesta de la pregunta: ¿Sufre enfermedades coronarias o ha tenido infarto del miocardio? 0: No y 1: Sí.
- ***PhysActivity*:** Establece si el paciente ha practicado actividad física durante los últimos 30 días sin incluir la actividad inherente al trabajo. 0: No y 1: Sí.
- ***Fruits*:** Permite conocer si el paciente consume fruta una o más veces al día. 0: No y 1: Sí.
- ***Veggies*:** Define si el paciente consume vegetales una o más veces al día. 0: No y 1: Sí.
- ***HvyAlcoholConsump*:** Hace referencia a la siguiente pregunta dependiendo del género del paciente: para un adulto hombre, se pregunta si toma más de 14 bebidas alcohólicas por semana. Para una mujer adulta, se pregunta si toma más de siete bebidas alcohólicas por semana. 0: No y 1: Sí.
- ***DiffWalk*:** Almacena la respuesta de la siguiente pregunta: ¿tiene usted dificultades serias para caminar o subir escaleras? 0: No y 1: Sí.
- ***Hypertension*:** Esta variable hace referencia a si el paciente padece de hipertensión. 0: No y 1: Sí.
- ***Stroke*:** Establece si el paciente ha sufrido un derrame cerebral. 0: No y 1: Sí.

3.1.3. Variables numéricas y de respuesta

- ***MentHlth*:** Variable numérica discreta que contabiliza los días de una baja salud mental en una escala de uno a 30 días.
- ***PhysHlth*:** Variable numérica discreta que establece los días de lesión o enfermedad que ha sufrido el paciente en una escala de uno a 30 días.

- **BMI:** Variable numérica continua, hace referencia al índice de masa corporal, da información de los niveles de obesidad que tiene el paciente.
- **Diabetes:** Esta variable categórica es la variable de respuesta dentro del conjunto de datos y hace referencia a si el paciente padece diabetes o no. 0: No y 1: Sí.

El archivo que permitirá entrenar los modelos tiene un total de 80.692 registros con un tamaño en memoria de 11.1 MB.

El objetivo principal de la variable de respuesta es proporcionar una etiqueta que permita establecer la probabilidad de que un paciente padezca de diabetes a partir de su estado de salud. El enlace para consultar la fuente de datos es el siguiente: https://acortar.link/c7DNzC (*Diabetes Prediction Competition (TFUG CHD Nov 2022) | Kaggle, s. f.).*

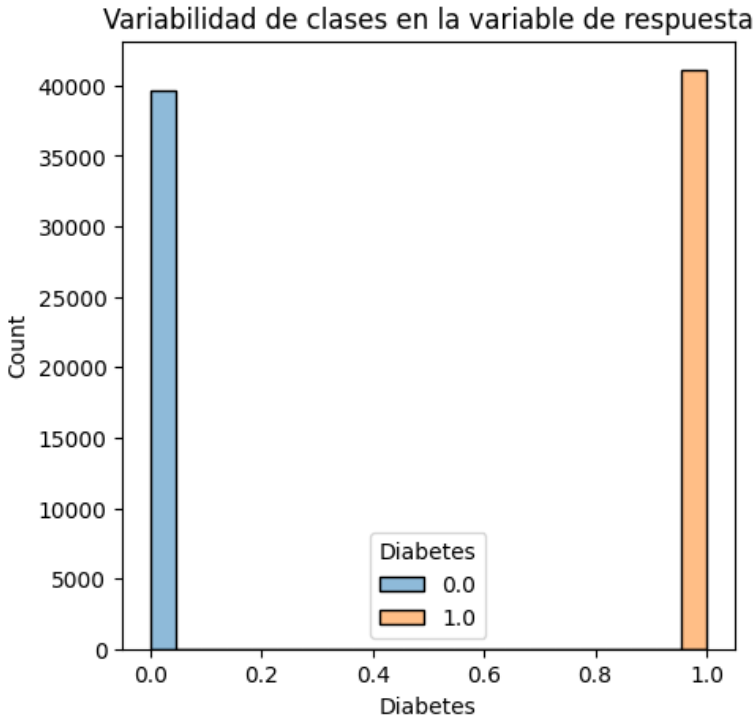
3.2. *Datasets*

Si bien las variables contaban con una codificación *label encoding*, la misma puede conducir a un sesgo de los modelos al considerar que la mayor clase dentro de cada variable es más influyente que las demás. Esta es una dificultad que se resuelve a través del método *get_dummies* de la librería pandas en *Python*. Se realizó una codificación de las variables categóricas *Age* y *GenHlth* a través de la metodología *One-Hot-Encoding*. Esta forma de representación de las variables aumenta la dimensionalidad del *dataset* desde las 17 columnas que se tenían inicialmente hasta un total de 34. Pese a este incremento en el número de características, implementar esta codificación evita que existan relaciones de ordinalidad entre los valores de las diferentes columnas y a su vez se impide el *bias* al que pueden incurrir los modelos de aprendizaje.

Haciendo uso del *dataset* codificado a través del método *get_dummies*, se emplea el algoritmo *train_test_split* para particionar el *dataset* en una proporción de 80% para los datos de entrenamiento, y un 20% restante para los datos que se emplearán durante las etapas de validación y prueba del modelo seleccionado como óptimo. Es importante que se establezca un *random_state* igual a 42 para que así los resultados sean replicables cada vez que se ejecuten las líneas de código asociadas con la partición del conjunto de datos.

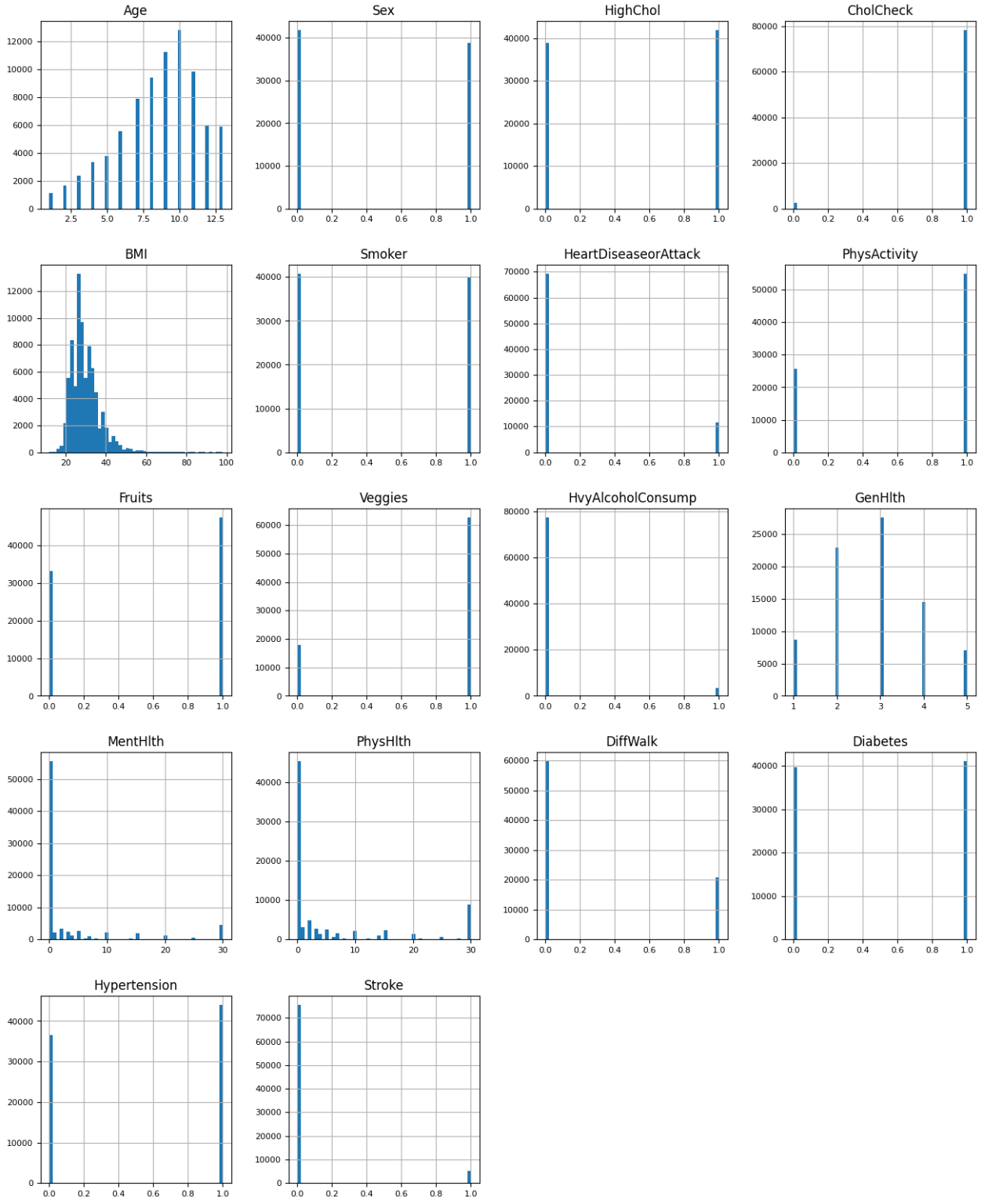
3.3.Analítica descriptiva

Figura 1 Distribución de la variable de respuesta Diabetes



Como puede apreciarse en la **Figura 1**, las clases de la variable de respuesta están balanceadas, esto debido a que no hay una diferencia significativa entre los pacientes que padecen diabetes con respecto a los que no.

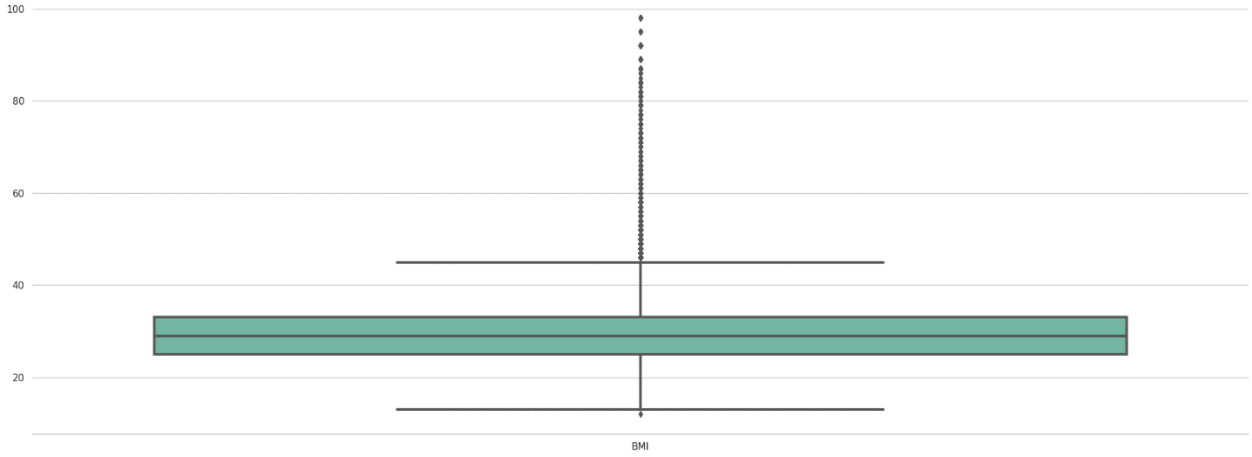
Figura 2 Matriz de histogramas para las variables del *dataset*



Como puede visualizarse en la **Figura 2**, únicamente la variable BMI posee una distribución continua de los datos. Si bien las variables *MentHlth* y *PhysHlth* son discretas en

comparación con la variable BMI, estas se consideraron también como numéricas debido a que ambas tienen un rango de valores que va de uno a 30 días. Este es más amplio con respecto a los rangos que pueden tomar las demás variables del *dataset*. Por esta razón solamente las variables *BMI*, *MentHlth* y *PhysHlth* se consideraron numéricas y las demás categóricas.

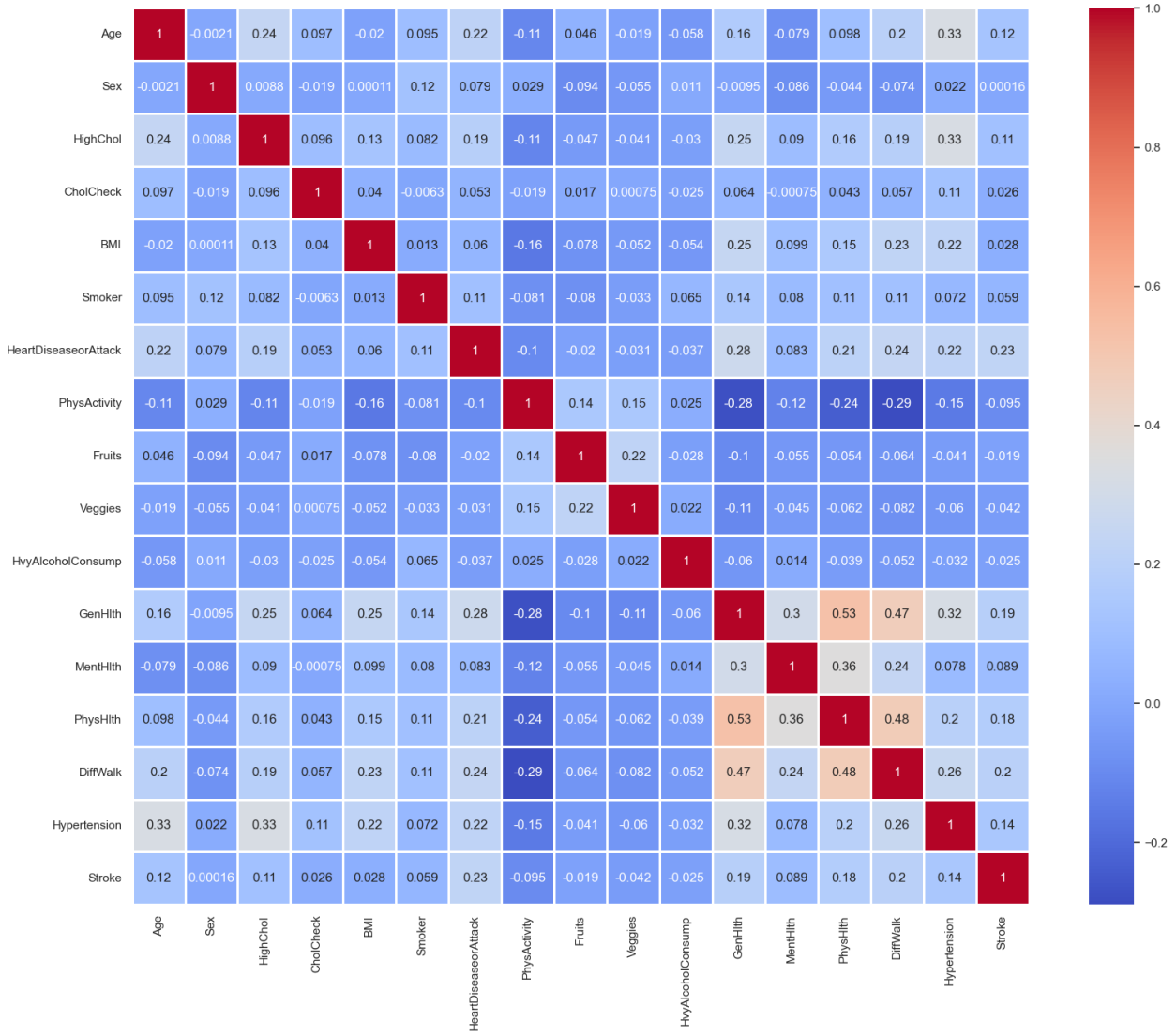
Figura 3 Gráfico de caja y bigotes para la variable numérica continua del *dataset (*BMI*)*



Como puede apreciarse en el gráfico de caja y bigotes de la **Figura 3**, el valor mínimo de la variable numérica continua dentro de la base de datos, (el cual a su vez coincide con el bigote inferior), se sitúa en 12 $\frac{kg}{m^2}$. El percentil 25% en 25 $\frac{kg}{m^2}$, el percentil 50% (o mediana) de la variable, se alcanza en un valor de 29 $\frac{kg}{m^2}$, el percentil 75 % se sitúa en un valor de 33 $\frac{kg}{m^2}$, mientras que el máximo valor de esta columna, (o bigote superior), es de 98 $\frac{kg}{m^2}$.

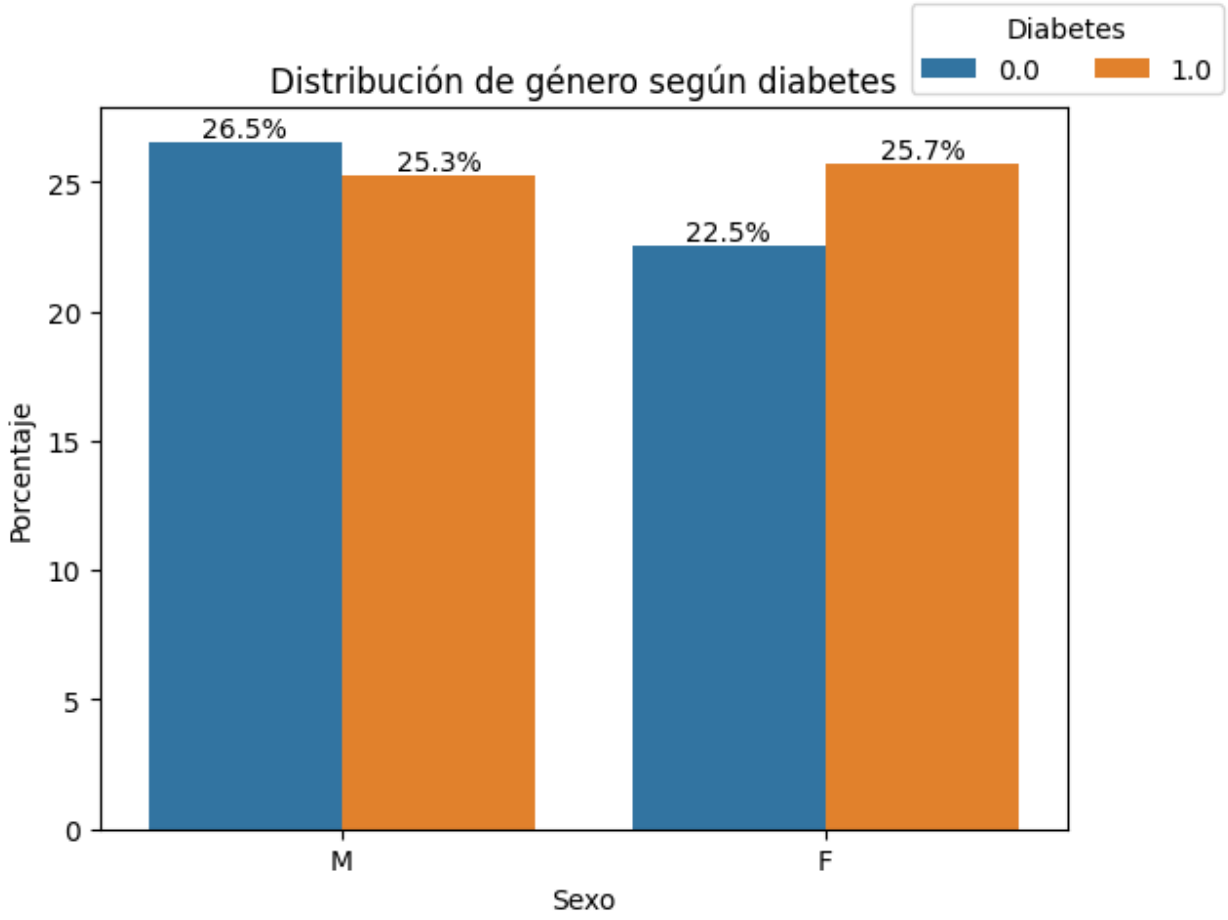
Desde un enfoque estadístico, los valores que se encuentran por encima del bigote superior en un diagrama de caja y bigotes se consideran como atípicos dentro de la distribución de la variable. Sin embargo, es necesario mencionar que uno de los factores que influye en la aparición de diabetes es la obesidad, y el BMI mide o da cuenta de esta condición de salud. Por esta razón, no se debe pensar en que los datos por encima del bigote superior son valores atípicos. Este es un claro ejemplo de cómo el conocimiento del negocio impide la toma de decisiones que, si bien estadísticamente están avaladas, sería incurrir en un error que se podría generalizar cuando se emplee la base de datos en modelos posteriores.

Figura 4 Matriz de correlación entre los diferentes pares de variables de la base de datos



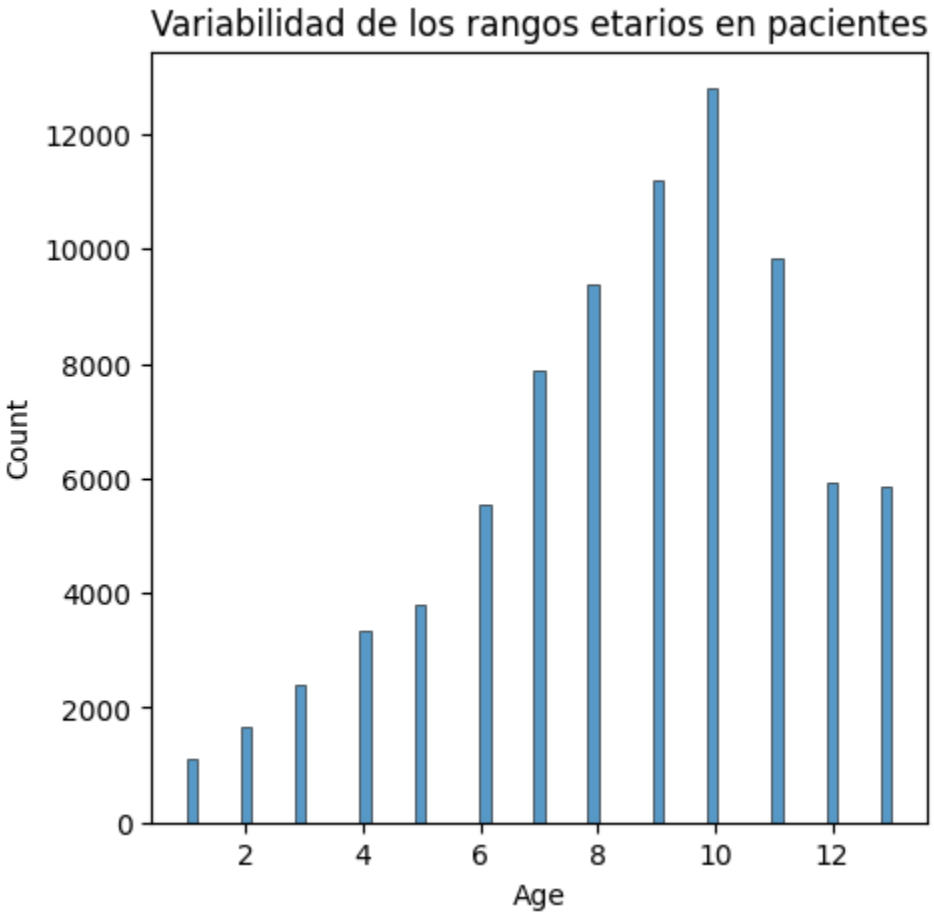
Como puede apreciarse en la **Figura 4**, no existen pares de variables que estén altamente correlacionadas entre sí. Esta conclusión se fundamenta en el hecho de que ningún valor de la matriz está cercano a la unidad, con excepción de la diagonal principal. El máximo valor positivo que existe, lo logran las variables *PhysHlth* y *GenHlth*. Sin embargo, la correlación es de apenas el 53%. Adicionalmente, el máximo valor negativo alcanzado dentro de la matriz de correlación se obtiene por parte del par de variables *PhysActivity* y *DiffWalk*. No obstante, este valor es de apenas el -29%.

Figura 5 Distribución del género del paciente según la variable de respuesta



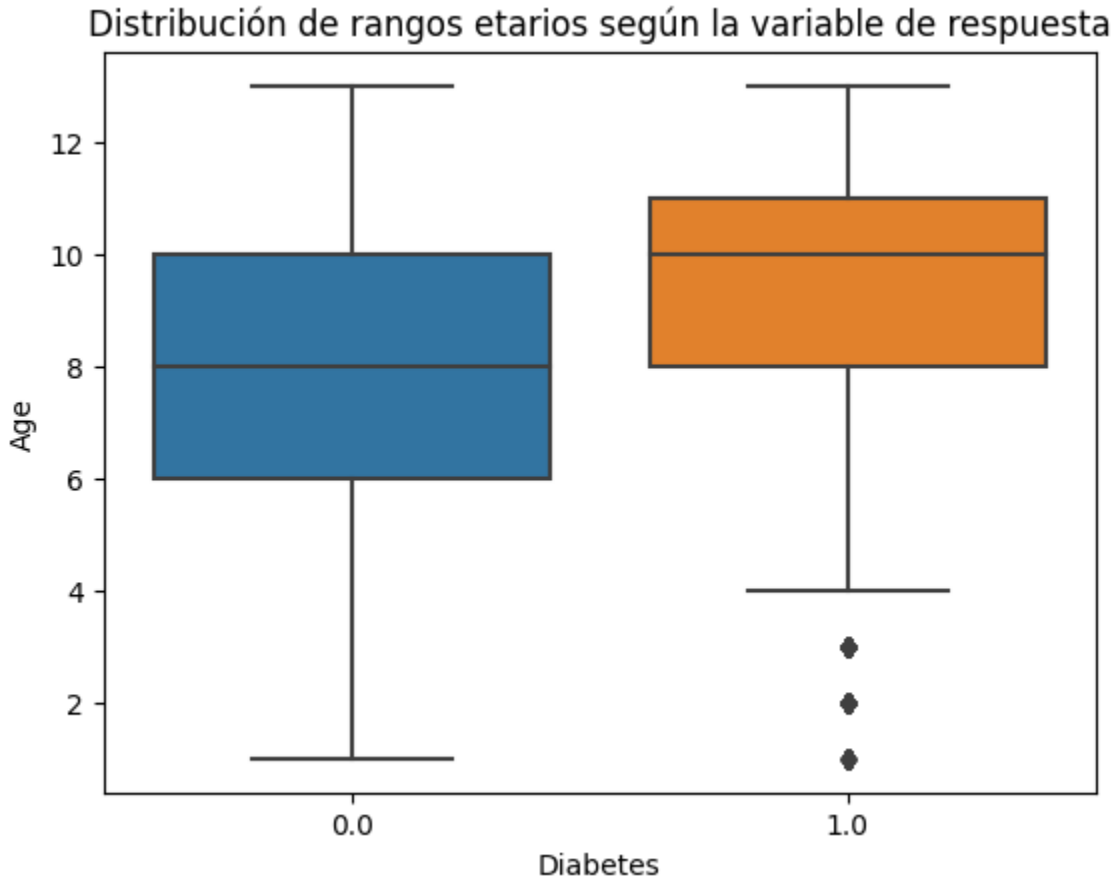
Como puede apreciarse en la **Figura 5**, si se realiza una evaluación del balanceo de las clases de la variable de respuesta como función de una de las características del *dataset*, (como el sexo, por ejemplo), se reafirma lo anteriormente dicho: las clases de la variable de respuesta están balanceadas, ya que no hay una prevalencia de una de las etiquetas por encima de la otra si se toma en consideración el género del paciente.

Figura 6 Distribución de la variable que da cuenta de la edad de los pacientes



Como puede observarse en la **Figura 6**, la variable *Age* que da cuenta del rango etario de los pacientes, tiene una distribución cuasi normal desplazada hacia la izquierda con su valor de media μ en 10, y una desviación estándar σ entre cero y uno.

Figura 7 Diagrama de caja y bigotes para los rangos etarios



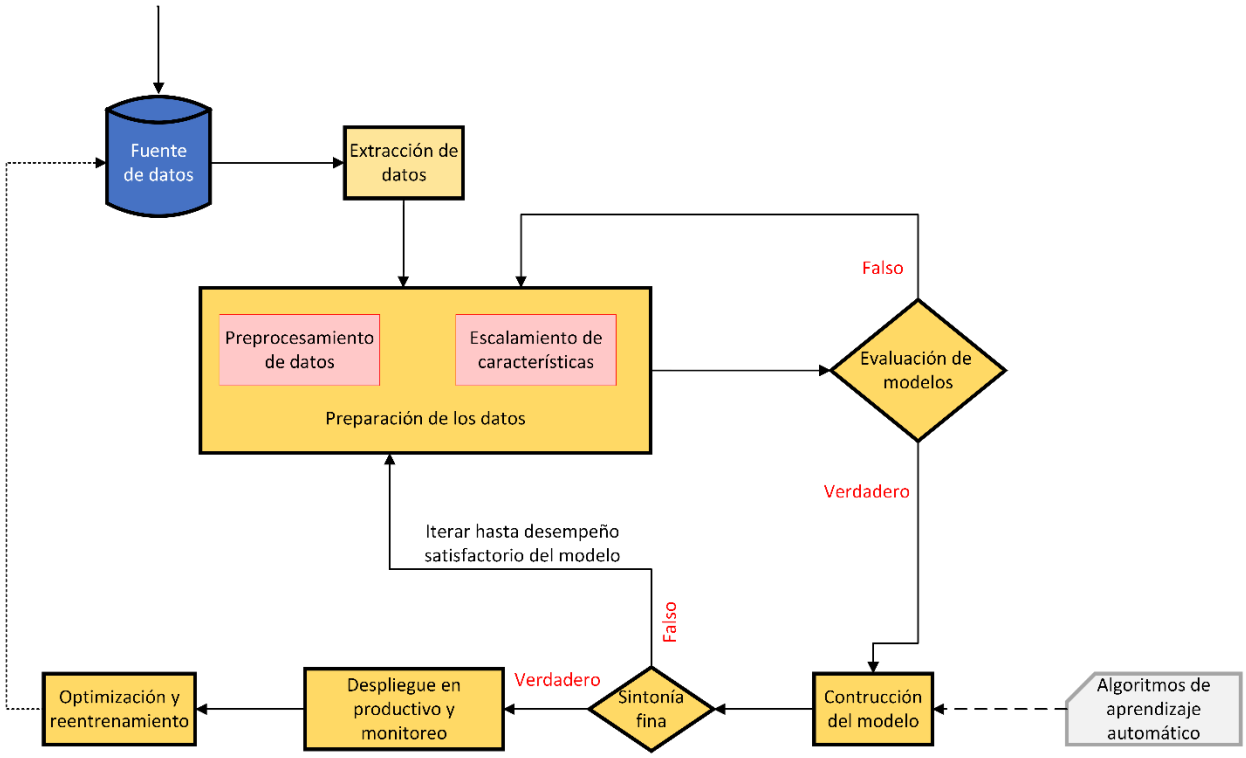
Como se aprecia en la **Figura 7**, cuando un paciente tiene una alta probabilidad de padecer diabetes (clase 1), se presenta un corrimiento de la mediana en el eje vertical superior. Esto permite concluir que, a mayor rango etario, la probabilidad de padecer diabetes se incrementa.

4. Proceso de analítica

4.1. *Pipeline* principal

A continuación, se representa el *pipeline* que muestra cuáles fueron las etapas que permitieron adecuar la fuente de datos para una posterior evaluación de los modelos de aprendizaje automático supervisado y posteriormente aplicar sintonía fina con el objetivo de encontrar los mejores hiperparámetros del modelo seleccionado:

Figura 8 *Pipeline* bases de datos



4.2.Preprocesamiento

Como se puede apreciar en la **Figura 1**, las clases de la variable de respuesta se encuentran balanceadas. Es por esta razón que no fue necesario hacer un submuestreo o un sobre muestreo del *dataset* para equilibrar las clases.

La siguiente etapa en el preprocesamiento de los datos consistió en clasificar a las variables como categóricas o numéricas. Como ya se mencionó, solamente una de las características cumple

con el hecho de ser continua y es el BMI. Las demás variables son discretas debido a que únicamente pueden tomar un número determinado de valores en comparación con los que puede adquirir el BMI. Ante este escenario, se optó por realizar una codificación *One-Hot-Encoding* de las variables discretas *Age* y *GenHlth*. Esto para evitar que los modelos de aprendizaje automático supervisado terminen por darle más importancia a una de las clases por encima de otras, y además para que las relaciones de ordinalidad que existen entre las clases no terminen por sesgar los modelos. Para las variables que tuvieran solo dos clases, se optó por no aplicar ningún tipo de codificación, ya que estas ya se encuentran cifradas como lo haría el método *get_dummies* con características que solamente tienen dos clases.

Finalmente, para evitar que modelos como la red neuronal perceptrón multicapa le asignen un peso mayor a aquella variable con los valores más altos dentro del *dataset*, se realizó un escalamiento *min max* de los datos. Esta metodología de normalización le asigna un valor entre cero y uno a todos los registros dentro de una columna.

4.3.Modelos

Todos los modelos que se consideraron durante este proyecto son de aprendizaje automático supervisado y se seleccionaron por exhibir un buen desempeño en bases de datos de clasificación binaria. Los hiperparámetros óptimos para todos los algoritmos fueron seleccionados luego de haber aplicado una búsqueda en malla o *grid search*.

Dentro de las experimentaciones que se realizaron haciendo uso de los modelos de Regresión Logística Binomial, *KNeighbors classifier, Random Forest* y *XGBoost*, se utilizó el hiperparámetro *n_jobs* con un valor de -1 con el objetivo de poder paralelizar su ejecución al hacer uso de todos los núcleos del procesador que está integrado con el ordenador donde se compilen los códigos. Adicionalmente, para los algoritmos de Regresión Logística Binomial, *Random Forest, XGBoost, Adaboost, Support Vector Classifier* y *MLPClassifier* se utilizó el hiperparámetro *random state* como una constante para que la aleatoriedad del modelo no sea un factor influyente a la hora de replicar los resultados. A continuación, se realiza una descripción de cada uno de ellos:

1. **Regresión logística binomial (*LogisticRegression*):** Este algoritmo implementa regresión logística regularizada utilizando los solucionadores *'liblinear', 'newton-cg', 'sag',* '*saga'* y

'*lbfgs*'. Es aplicable para el manejo de bases de datos densas y dispersas. Utiliza matrices ordenadas en C o matrices CSR que contengan valores flotantes de 64 *bits* para un rendimiento óptimo.

Los solucionadores *'newton-cg', 'sag'* y *'lbfgs'* solo admiten la regularización L2. El solucionador *'liblinear'* admite la regularización L1 y L2, con una formulación dual solo para la penalización L2. La regularización *Elastic-Net* solo está soportada por el solucionador de 'saga'.

Finalmente, se establecen los rangos sobre los cuales fueron evaluados los hiperparámetros para este modelo a través de la siguiente tabla:

Tabla 2 Hiperparámetros por ajustar regresión logística

Hiperparámetro	Descripción	Rango
Penalty	Es el hiperparámetro que permite establecer bajo qué norma se calcularán la longitud y el tamaño de los vectores del modelo.	[L1, L2, *Elasticnet*]
Solver	Es aquel criterio que permite definir bajo qué metodología se resolverá el problema de optimización de la regresión.	[*lbfgs, liblinear, newton-cg, sag, saga, netwon-cholesky*]
C	Esta cantidad representa el inverso de la fuerza de regularización del modelo y debe ser un valor flotante positivo. Tal como en las máquinas de soporte vectorial, valores más pequeños de C permiten especificar una regularización más fuerte (*Sklearn.linear_model.LogisticRegression,* s. f.)	[0.01, 0.1, 1, 10, 100]

2. **Voto de K vecinos más cercanos (*KNeighborsClassifier*):** Este algoritmo implementa el aprendizaje basado en *k* vecinos más cercanos de cada punto de consulta, donde *k* es un valor entero especificado por el usuario. La elección óptima del valor *k* depende en gran medida de los datos: en general, un mayor *k* suprime los efectos del ruido, pero hace que

los límites de clasificación sean menos claros (*Sklearn.neighbors.KNeighborsClassifier*, s. f.).

Los hiperparámetros que fueron optimizados para este modelo, así como el rango de valores que podían tomar, se establecen a continuación:

Tabla 3 Hiperparámetros por ajustar *KNeighborsClassifier*

Hiperparámetro	Descripción	Rango
k	Es el hiperparámetro que permite establecer el número de vecinos a utilizar de forma predeterminada.	[21,93,171,255]
Metric	Es aquel criterio que permite calcular la distancia entre puntos del algoritmo.	[*euclidean, manhattan, chebyshev, cosine*]

3. **Clasificador *Naive Bayes* para modelos *Bernoulli* multivariados (*BernoulliNB Classifier*):** Este modelo funciona como un clasificador binomial de *Naive Bayes* para modelos multivariable de *Bernoulli*. Al igual que el modelo *MultinomialNB*, este algoritmo es adecuado para bases de datos que cuentan con variables categóricas o discretas. La diferencia es que mientras *MultinomialNB* funciona con recuentos de ocurrencias, *BernoulliNB* está diseñado para características binarias o booleanas (*Sklearn.naive_bayes.BernoulliNB*, s. f.).

El papel que tiene cada uno de los hiperparámetros de este modelo que fueron evaluados en la etapa de entrenamiento y validación se describen a continuación:

Tabla 4 Hiperparámetros por ajustar *BernoulliNB Classifier*

Hiperparámetro	Descripción	Rango
Binarize	Este hiperparámetro mide el umbral de binarización (mapeo a booleanos) de características de muestra. Si no hay ninguno, se asume que la entrada ya consta de vectores binarios.	[0, 0.35, 0.5, 0.75]

Alpha	Parámetro aditivo de suavización de *Laplace/Lidstone*. Para que no se utilice la suavización dentro del modelo, se debe igualar *Alpha* a cero y el hiperparámetro *force_alpha* en verdadero.	[0, 0.25, 0.5, 0.75, 1]
Force_alpha	Controla la cantidad de suavizado aplicado a los datos. El suavizado es una técnica utilizada para evitar que el clasificador haga *overfit* sobre los datos de entrenamiento. Funciona agregando una pequeña cantidad de ruido, lo que ayuda al clasificador a generalizar mejor a los nuevos datos.	[*True, False*]

4. **Clasificador *Random Forest*:** Este algoritmo es un meta estimador que ajusta una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos. Este modelo consiste en una colección de clasificadores estructurados conocidos como árboles, los cuales pueden representarse a través del siguiente conjunto $\{h(\boldsymbol{x}, \Theta_k), k = 1\}$ donde los términos $\{\Theta_k\}$ son vectores aleatorios que están independiente e idénticamente distribuidos en cada árbol con el objetivo de que cada uno de estos genere un voto para la clase que más se presente en la entrada x. El tamaño de la submuestra se controla con el parámetro *max_samples* si *Bootstrap* es igual a *True*. De lo contrario se utiliza todo el conjunto de datos para construir cada árbol (Breiman, 2001).

   Dentro de las experimentaciones que se realizaron haciendo uso de este modelo de clasificación, el hiperparámetro *class_weight* se tomó como una constante e igual a "*balanced*" ya que las clases de la variable de respuesta del problema se encuentran equilibradas.

   En la siguiente tabla, se pueden apreciar los hiperparámetros que fueron evaluados para este modelo:

Tabla 5 Hiperparámetros por ajustar *RandomForestClassifier*

Hiperparámetro	Descripción	Rango
N_estimators	Este hiperparámetro permite establecer la cantidad de árboles de decisión.	[100,500,1000]
Max_depth	Parámetro que define la profundidad máxima del árbol.	[1, 3, 5]
Criterion	Este hiperparámetro mide la calidad de una división.	[*gini, entropy, log_loss*]

5. **Algoritmo de *Extreme Gradient Boosting Classifier*:** Algoritmo de aumento de gradiente que construye un conjunto de árboles de decisión secuencialmente, optimizando una función objetivo que combina las predicciones de estos. Utiliza regularización, muestras ponderadas y "poda" los árboles para crear un modelo predictivo robusto y potente. La versatilidad y el rendimiento de *XGBoost* lo han convertido en una opción popular para una amplia gama de tareas de aprendizaje automático.

   La función objetivo específica para cuantificar el rendimiento del modelo y definir cómo se deben construir los árboles en tareas de clasificación, utiliza la pérdida de registro (también conocida como *cross entropy*)**.**

   Los hiperparámetros considerados para este modelo de clasificación se establecen a continuación, así como los valores que podían tomar:

Tabla 6 Hiperparámetros por ajustar *XGBoost Classifier*

Hiperparámetro	Descripción	Rango
Learning_rate	Juega un papel crucial en el control del tamaño del paso o la velocidad a la que el modelo aprende durante el entrenamiento. Es un valor entre cero y uno que determina cuánto se actualizan los parámetros (por ejemplo, pesos o coeficientes) en cada iteración de la optimización.	[0.03, 0.06, 0.1, 0.3]

N_estimators	Determina el número de modelos base (árboles) del conjunto. Afecta a la complejidad, la generalización y el tiempo de ejecución del modelo.	[100, 500, 1000]
Max_depth	Esta cantidad establece la profundidad máxima de un árbol de decisión. Aumentarlo hará que el modelo sea más complejo y propenso a sobre ajustarse. Es necesario tener en cuenta que *XGBoost* presenta un alto consumo de memoria al entrenar un árbol muy profundo.	[1, 3, 5]

6. **Clasificador *Adaboost*:** Este algoritmo es un meta estimador que comienza ajustando un clasificador en el conjunto de datos original y luego ajusta copias adicionales del clasificador en el mismo conjunto de datos, pero donde los pesos de las instancias clasificadas incorrectamente se ajustan de modo que los clasificadores posteriores se centren más en casos difíciles.

   La definición de los hiperparámetros seleccionados para la evaluación de este modelo de clasificación se muestra a continuación:

Tabla 7 Hiperparámetros por ajustar *AdaboostClassifier*

Hiperparámetro	Descripción	Rango
Learning_rate	Juega un papel crucial en el control del tamaño del paso o la velocidad a la que el modelo aprende durante el entrenamiento. Es un valor entre cero y uno que determina cuánto se actualizan los parámetros (por ejemplo, pesos o coeficientes) en cada iteración de la optimización.	[0.03, 0.06, 0.1, 0.3]
N_estimators	Determina el número de modelos base (árboles) del conjunto. Afecta a la complejidad, la generalización y el tiempo de ejecución del modelo.	[100, 500, 1000]

Algorithm	Este parámetro controla el algoritmo base utilizado para construir los clasificadores débiles (también conocidos como estimadores bases) en el ensemble *AdaBoost.*	[*SAMME, SAMME.R]*

7. Máquina de soporte vectorial de clasificación (*Support Vector Classifier*): La implementación se basa en *libsvm*, el cual hace referencia a un modelo fácil de usar y eficiente para la clasificación y regresión de *Support Vector Machine*. Resuelve la clasificación *C-SVM*, la clasificación *nu-SVM*, la *SVM* de una clase, la regresión *épsilon-SVM* y la regresión *nu-SVM*. También proporciona una herramienta de selección automática de modelos para la clasificación *C-SVM*.

Todos los algoritmos relacionados con las máquinas de vectores de soporte requieren de un tiempo de ajuste que se escala al menos cuadráticamente con el número de muestras y puede ser poco práctico más allá de decenas de miles de registros (*Sklearn.svm.SVC*, s. f.).

En la siguiente tabla, se pueden apreciar los rangos sobre los cuales fueron evaluados los hiperparámetros de este modelo:

Tabla 8 Hiperparámetros por ajustar *Support Vector Classifier*

Hiperparámetro	Descripción	Rango
Kernel	Establece el tipo de *solver* que se utilizará en el algoritmo. Si no se especifica, se utiliza *'rbf*' por defecto.	[*linear, poly, rbf, sigmoid*]
C	Esta cantidad contabiliza el parámetro de regularización. La influencia o el efecto de la regularización es inversamente proporcional a C y este debe ser positivo. La penalización es una penalización cuadrática L2.	[0.01, 0.1, 1, 10]
Gamma	Esta cantidad representa el coeficiente que se utilizará dentro del *Kernel*. si se usa *gamma* igual a *'scale*' (predeterminado), entonces el coeficiente se calcula	[*scale, auto*]

	como *1 / (n_features * X.var*()). En caso de que gamma sea 'auto', el coeficiente se calcula como *1 / n_features.*

8. Red Neuronal Perceptrón Multicapa de Clasificación (*MLPClassifier*): Este modelo optimiza la función de pérdida logarítmica utilizando *LBFGS* o descenso de gradiente estocástico.

Dentro de las experimentaciones que se realizaron haciendo uso de este modelo de clasificación, el hiperparámetro *max_iter* se tomó como una constante igual a 200 para tener un valor realista de cuantas veces se debe ejecutar este modelo que es tan robusto.

Cada hiperparámetro evaluado para este modelo se menciona en la siguiente tabla:

Tabla 9 Hiperparámetros por ajustar *MLPClassifier*

Hiperparámetro	Descripción	Rango
hidden_layer_sizes	Representa el número de neuronas en la i-ésima capa oculta.	[(50,), (100,), (50, 50), (100, 50)]
activation	Establece la función de activación de la capa oculta.	[*relu, logistic*]
solver	Selecciona el solucionador para la optimización de los pesos.	[*lbfgs,adam, sgd*]
learning_rate	Esta cantidad representa las tasas de aprendizaje con las cuales se actualiza el peso dentro de cada neurona.	[*constant, invscaling, adaptive*]
alpha	Mide la fuerza de la regularización L2. El término de regularización L2 se divide por el tamaño de la muestra cuando se suma a la pérdida.	[0.0001, 0.001, 0.01]

9. Red Neuronal Artificial de Clasificación Multicapa desarrollada con *Keras*: Una Red Neuronal Artificial (ANN, por sus siglas en inglés) desarrollada con *Keras* es un modelo computacional inspirado en la estructura y función de las redes neuronales biológicas,

utilizado para diversas tareas de aprendizaje automático y aprendizaje profundo. *Keras* es una API de redes neuronales de alto nivel que se ejecuta sobre bibliotecas de aprendizaje profundo de nivel inferior como *TensorFlow* y *Theano,* lo que facilita la definición, el entrenamiento y la evaluación de modelos de redes neuronales.

Dentro de las experimentaciones que se realizaron haciendo uso de este modelo de clasificación, fue necesario crear una clase en *Python* que permitiera implementar los paquetes que se tienen dentro de la librería *Keras* con la búsqueda en malla, la cual pertenece a la librería *scikit-learn*. Esta clase se conoce como un *wrapper,* el cual se refiere a un fragmento de código o un componente de software que encapsula un modelo *Keras* y proporciona funcionalidad adicional u opciones de personalización. Los *wrappers* se utilizan a menudo para agilizar el proceso de construcción, entrenamiento y uso de redes neuronales al proporcionar una interfaz de nivel superior o al agregar funcionalidad específica a los modelos.

Los cinco hiperparámetros optimizados durante la evaluación de este modelo, son los que se definen a continuación:

Tabla 10 Hiperparámetros por ajustar Red Neuronal Artificial *Keras*

Hiperparámetro	Descripción	Rango
hidden_units	Se refiere al número de neuronas o unidades en cada capa oculta.	[16, 32, 64, 128]
activation	Especifica la función de activación utilizada en cada neurona de las capas ocultas.	[*relu, tanh*]
optimizer	Determina el algoritmo de optimización utilizado para actualizar los pesos de la red neuronal durante el entrenamiento.	[*adam, sgd, rmsprop*]
dropout_rate	Es una técnica de regularización utilizada para evitar el sobreajuste en redes neuronales que consiste en desactivar un porcentaje de neuronas de las capas ocultas de la red.	[0.2, 0.5]

4.4.Métricas

4.4.1. Métricas de desempeño

Las métricas de desempeño de clasificación *accuracy*, *recall* y *F1-score* se importan desde la librería *sklearn*. Durante la etapa de entrenamiento y validación de los modelos, estas se calcularon utilizando el hiperparámetro *scoring* tanto en el método *GridSearchCV*, como en el algoritmo *cross_validate* que pertenecen a la librería *scikit-learn* de *Python*. Para esto, se declaró una variable tipo lista con las siguientes entradas*: scoring = [‘accuracy’, ‘recall’,’F1’*]

4.4.2. Métrica de negocio

La métrica de negocio de este proyecto es el modelo de aprendizaje supervisado que luego de haber pasado por las etapas de entrenamiento, validación y prueba, exhibió el mejor desempeño en comparación con los otros modelos que fueron considerados. El procedimiento para determinar cuál es el modelo óptimo fue el siguiente:

- Los nueve modelos seleccionados fueron entrenados con múltiples combinaciones de hiperparámetros a través de una búsqueda en malla o *GridSearchCV*, utilizando un tamaño de datos de entrenamiento equivalente al 80% del *dataset* original.
- Luego de la aplicación de la búsqueda en malla, se obtienen las hiperparámetros óptimos para cada modelo de aprendizaje supervisado. Los nueve modelos son nuevamente puestos a prueba en una etapa de validación a través del algoritmo *cross_validate* haciendo ahora uso de un 10% adicional del *dataset* y fijando sus hiperparámetros en los valores que resultaron ser los mejores luego del *GridSearchCV*.
- Finalmente, el modelo que condujo al mayor valor de *F1-score* en la etapa de validación, fue seleccionado como el mejor modelo. El modelo que se escogió como óptimo es puesto a prueba haciendo uso del 10% del *dataset* remanente que no fue empleado ni en la etapa de entrenamiento ni en la etapa de validación, razón por la cual es desconocido para el modelo seleccionado.

5. Metodología

5.1. *Baseline*

5.1.1. Análisis exploratorio de datos

La primera iteración de este proyecto de monografía inició con un análisis exploratorio de las variables de la base de datos. El primer reto que se presentó fue clasificar las variables dependiendo de si eran numéricas o categóricas. Para las características binarias, fue claro que debían tratarse como categóricas, debido a que los valores que podían tomar eran únicamente 0 o 1. El índice de masa corporal del paciente (BMI) fue claramente identificada como una variable numérica ya que representa una cantidad que puede asumir una amplia gama de valores numéricos basados en el peso y la altura del paciente. Para las variables *Age*, *GenHlth*, *MentHlth* y *PhysHlth*, se optó por considerarlas numéricas puesto que su frecuencia de valores posibles se asemejaba más a la del BMI que a las binarias.

Posteriormente, se optó por realizar un análisis de detección de datos atípicos a través del algoritmo *Local Outlier Factor* (LOF). El grado de contaminación definida para el algoritmo fue del 10%. Se utilizó como parámetros de vecinos más cercanos el valor cinco y la distancia euclidiana. El siguiente reto que se presentó fue determinar qué tan efectiva había resultado ser la eliminación controlada de datos atípicos, con el fin de evitar una pérdida considerable de información valiosa dentro del *dataset*. Para esto, se realizó el cálculo de la entropía de *Shannon*, la cual arrojó que la cantidad de información original no se vio afectada en gran medida, ya que los valores de entropía de tres de las variables numéricas permanecieron prácticamente constantes, como se muestra a continuación:

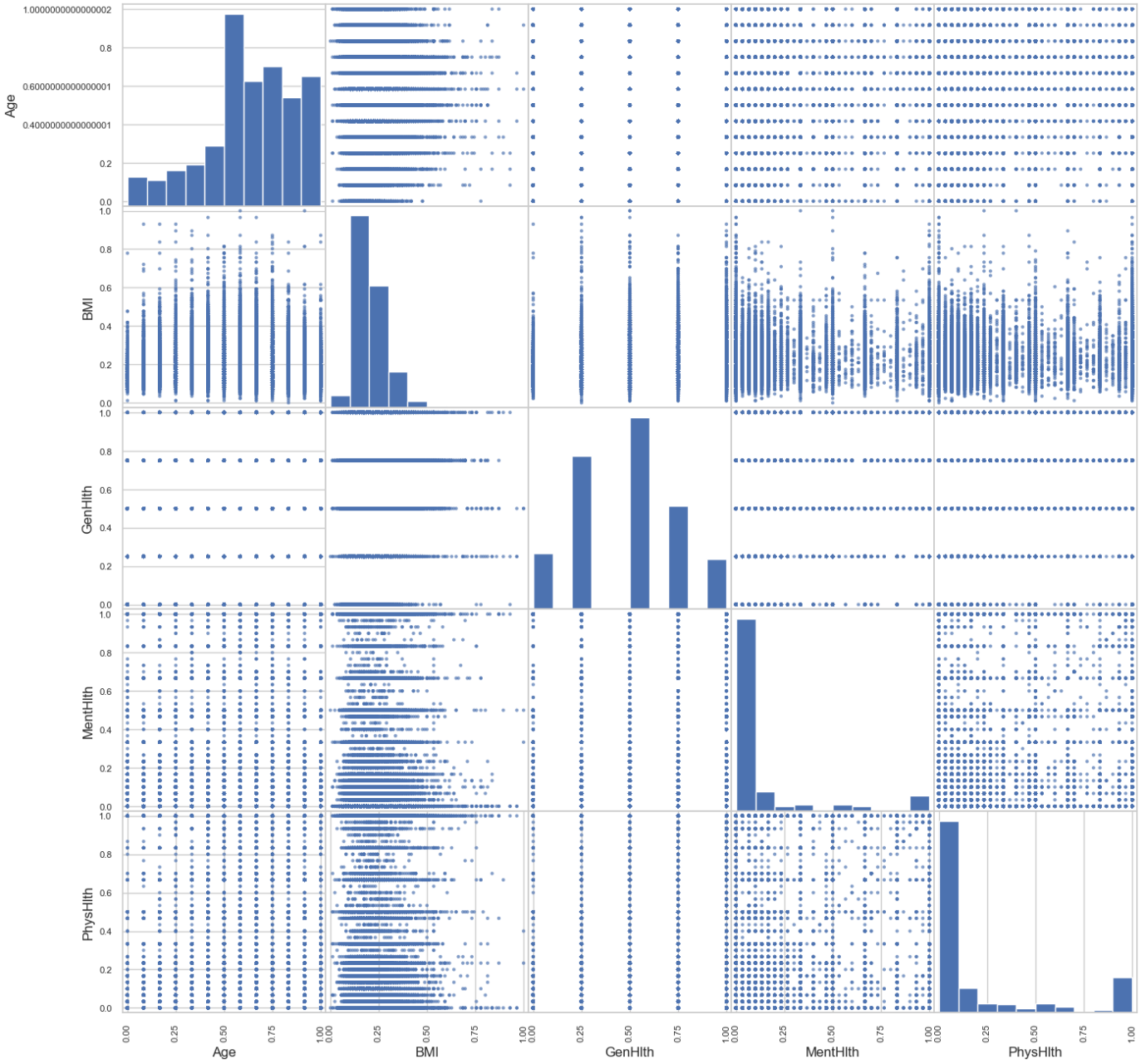
Tabla 11 Comparativo de las entropías de *Shannon*

Variable	Entropía Original	Entropía LOF
BMI	0.311079	0.312066
MentHlth	0.065087	0.063935
PhysHlth	0.082470	0.081624

Posterior a la eliminación de datos atípicos, se realizó un análisis de correlación entre pares de variables numéricas a través de la construcción de una matriz de gráficos de dispersión. Ésta no

arrojó ningún comportamiento que permitiera establecer que una o más variables estuvieran altamente correlacionadas con otras dentro del conjunto de datos:

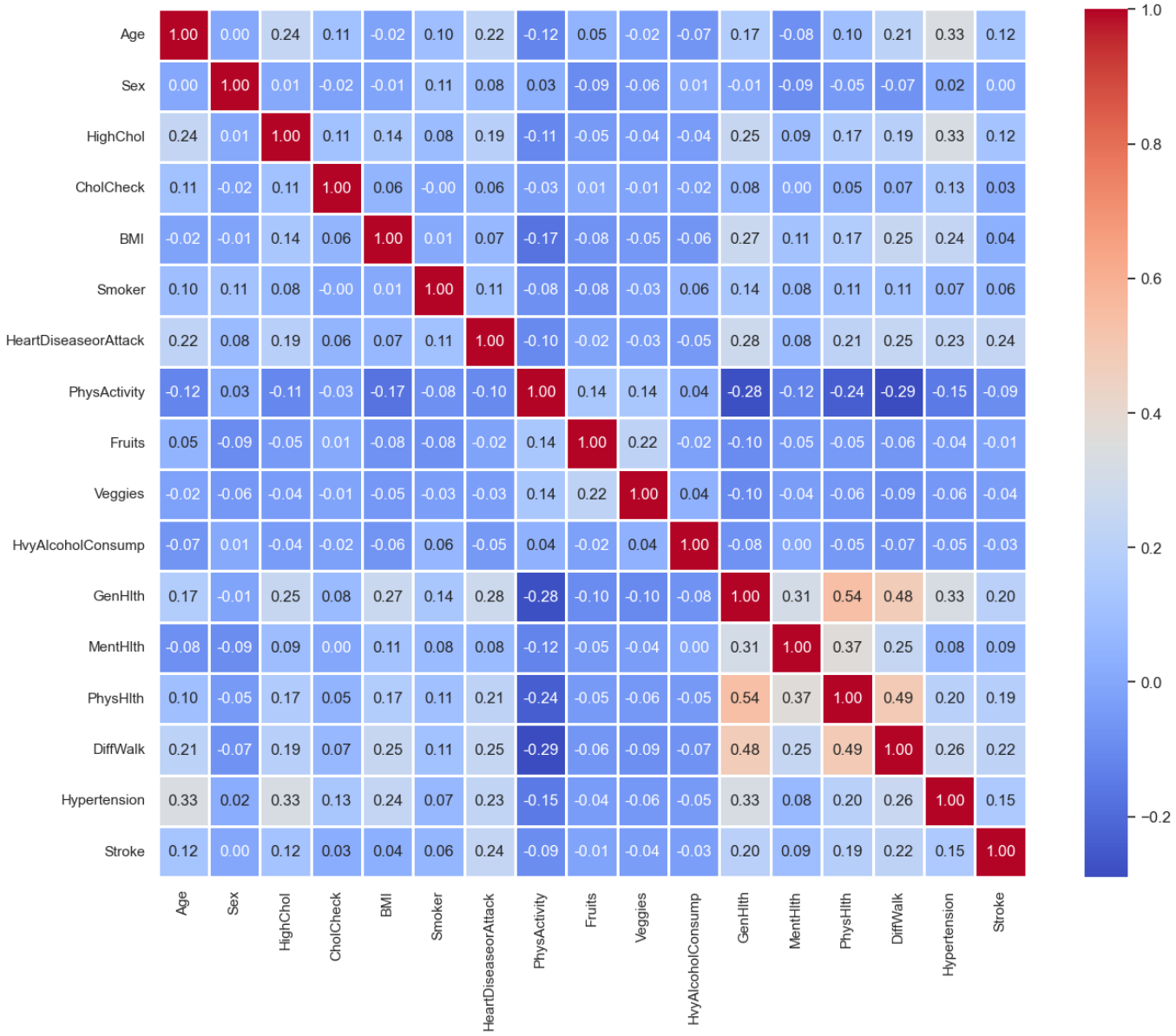
Figura 9 Matriz de gráficos de dispersión de las variables numéricas de la primera iteración



Finalmente, el análisis exploratorio de datos concluyó con la construcción de una matriz de correlación que incluyera todas las variables del *dataset* luego de haber llevado a cabo la eliminación de datos atípicos. Al igual que en el caso de la **Figura 9**, la matriz de correlación entre

pares de variables de todo el conjunto de datos permitió determinar que no había presencia de variables altamente correlacionadas:

Figura 10 Matriz de correlación de pares de variables con eliminación de datos atípicos



5.1.2. Etapa de entrenamiento, validación y prueba de modelos de aprendizaje supervisado

Para la determinación del modelo de clasificación más adecuado en el cálculo de la probabilidad de diabetes en pacientes, se evaluó el desempeño de siete modelos diferentes. Estos fueron: regresión logística, *KNeighbors Classifier*, clasificador de *Naive Bayes* para modelos

Bernoulli multivariable, *Random Forest, Support Vector Classifier, XGBoost Classifier* y *Adaptive Boost Classifier.*

Inicialmente, se utilizó el algoritmo de búsqueda en malla, (*Grid search*), para determinar los mejores hiperparámetros de cada uno de los modelos mencionados. Durante este proceso, se hizo uso del 80% de la base datos de entrenamiento con y sin eliminación de datos atípicos a través del algoritmo LOF para los modelos de: Regresión Logística, KNN, *BernoulliNB* y *Random Forest*. Para los modelos faltantes, (SVC, *XGBoost* y *Adaboost*), debido a su alto costo computacional, se optó por aplicar el *Grid search* con 4000, 8000 y 40000 registros del total de la base datos, los cuales se seleccionaron a través de la implementación del algoritmo *train_test_split*, utilizando un *random_state* igual a 42 con el fin de que cuando se ejecute nuevamente el código, los resultados de esta partición del conjunto de datos pueda ser replicable con y sin eliminación de datos atípicos.

La búsqueda en malla realizada arrojó los siguientes valores de hiperparámetros óptimos para cada uno de los modelos, así como la métrica *F1-score* de prueba más alta:

Tabla 12 Hiperparámetros óptimos y métrica de desempeño en la primera iteración

Modelo	**Hiperparámetros**	**Valores óptimos**		**Métrica de desempeño evaluada** *mean_test_F1-score*	
		Sin eliminación de datos atípicos	**Con eliminación de datos atípicos**	**Sin eliminación de datos atípicos**	**Con eliminación de datos atípicos**
Regresión logística	**C**	0.1	0.1	75.75%	76.76%
	Penalty	l1	l1		
	Solver	*Saga*	*liblinear*		
KNN	***n_neighbors***	93	21	75.24%	76.12%
	metric	*manhattan*	*manhattan*		
Bernoulli Naive Bayes	***Binarize***	0,35	0,35	72.82%	73.43%
	Alpha	0	0,5		
	Force_alpha	*True*	*False*		
Random Forest	***n_estimators***	500	1000	75.27%	75.96%
	max_depth	5	5		
	criterion	*Gini*	*Gini*		
Support Vector Classifier	**Registros del *dataset* utilizados durante el *Grid Search* y el *cross validation***	40.000	40.000	76.10%	77.33%

	C	1	1		
	Kernel	*rbf*	*rbf*		
	gamma	*Auto*	*auto*		
XGBoost Classifier	**Registros del *dataset* utilizados durante el *Grid Search* y el *cross validation***	40.000	40.000	76.08%	77.38%
	learning_rate	0.06	0.06		
	n_estimators	500	500		
	max_depth	3	3		
Adaboost Classifier	**Registros del *dataset* utilizados durante el *Grid Search* y el *cross validation***	40.000	40.000	75.68%	76.81%
	learning_rate	0.3	0.1		
	n_estimators	500	500		
	algorithm	*SAMME.R*	*SAMME.R*		

Posteriormente, se implementó un modelo de validación cruzada haciendo uso de todos los modelos evaluados con sus respectivos hiperparámetros óptimos, y tomando un 10% adicional de los datos del *dataset* que no se utilizaron durante la búsqueda en malla. Para el caso del escenario sin eliminación de datos atípicos, el mayor valor de la métrica *F1-score* de prueba es del 77.35% y se alcanzó con el modelo de SVC. Por otra parte, para el escenario con eliminación de datos atípicos, el máximo valor de *F1-score* de prueba es del 78.58% y se obtuvo con el modelo de *XGBoost.*

Finalmente, se utilizó el 10% de los datos del *dataset* que no fueron implementados ni en la búsqueda en malla ni en la validación cruzada para una prueba definitiva haciendo uso de los modelos de SVC para el escenario sin eliminación de datos atípicos y el modelo *XGBoost* para el caso de eliminación de datos atípicos. Para el caso del modelo SVC, el valor de *F1-score* que se obtuvo durante la predicción de la clase 0 en la variable de respuesta, (la cual representa la probabilidad de que un paciente no padezca de diabetes), fue de 72%, mientras que para la clase 1, (la cual representa la probabilidad de que un paciente sí padezca de diabetes), fue del 76%. Por otra parte, para el modelo *XGBoost*, el valor de *F1-score* que se obtuvo para la predicción en la clase 0 fue del 74%, mientras que para la predicción en la clase 1 fue del 77%. Estos resultados permiten concluir que la eliminación de datos atípicos a través del algoritmo LOF permitió alcanzar el valor

mínimo de la métrica que se propuso alcanzar (75% en promedio). Sin embargo, no existe una diferencia significativa entre el valor que se alcanzó luego de implementar este mecanismo de eliminación controlada de datos atípicos con respecto a cuando no se utilizó.

5.2.Validación

Durante la primera iteración de este proyecto, se realizó una partición del *dataset*. Para el caso particular de la regresión logística, KNN, *BernoulliNB* y *Random Forest*, se utilizó el 80% de la base datos en la etapa de entrenamiento en los escenarios con y sin eliminación de datos atípicos. Para los modelos *Support Vector Classifier, XGBoost Classifier* y *Adaptive Boost Classifier*, como se mencionó anteriormente, durante la etapa de entrenamiento se truncaron los datos a 4000, 8000 y 40000, lo que significa que se utilizó un porcentaje del *dataset* en la etapa de entrenamiento inferior al 80%, tanto en el escenario con eliminación como sin eliminación de datos atípicos.

En la etapa de validación cruzada se utilizó un 10% adicional para evaluar los modelos con sus mejores hiperparámetros, es decir, que el 90% de la base de datos se empleó en la etapa de validación cruzada. Finalmente, el 10% restante se empleó en la etapa de prueba definitiva donde se evaluó el desempeño del modelo que alcanzó el mayor valor de *F1-score* en la etapa de validación cruzada.

5.3.Iteraciones y evolución

Se realizaron seis iteraciones posteriores a la primera iteración en las cuales se entrenaron todos los modelos considerados con el 80% de la base de datos original. Este es un hecho a destacar ya que, en la primera iteración, para los modelos de *Support Vector Classifier, XGBoost Classifier* y *Adaptive Boost Classifier* se truncó la base de datos como se mencionó en el numeral 5.2. Esta decisión se había tomado teniendo en cuenta que, si no se reducía el tamaño del *dataset* de entrenamiento, los algoritmos tardaban más de tres horas en finalizar su ejecución. Sin embargo, es necesario recordar que una vez se ejecuta la etapa de entrenamiento con el 80% del *dataset* original, el resultado puede ser almacenado y no es necesario volver a ejecutar el código previamente desarrollado. En pocas palabras, solamente es necesario esperar las horas de ejecución una vez y almacenar el resultado para evitar volver a ejecutar el código.

En la segunda iteración del proyecto, se utilizó nuevamente el algoritmo LOF para la eliminación de datos atípicos. La métrica de desempeño que se obtuvo en esta iteración para el escenario sin eliminación de datos atípicos fue de 77.35%, mientras que para el escenario con eliminación de datos atípicos fue de 78.50%. La diferencia entre ambos valores es de 1.15%, la cual no es material para el análisis de este proyecto. Esta fue una de las razones por las cuales no se continuó implementando eliminación de datos atípicos en las demás iteraciones. Otra razón que justifica el por qué no utilizar eliminación de datos atípicos es que, desde el punto de vista médico, una persona con un índice de masa corporal (BMI) superior a 30 kg/m^2 no debería omitirse del análisis ya que se trata de una persona con obesidad, no de una anomalía de los datos (*Cómo evaluar su peso,* 2023).

Los resultados de la etapa de entrenamiento para la segunda iteración se muestran a continuación:

Tabla 13 Hiperparámetros óptimos y métrica de desempeño en la segunda iteración

Modelo	**Hiperparámetros**	**Valores óptimos**		**Métrica de desempeño evaluada**	
				mean_test_f1_score	
		Sin eliminación de datos atípicos	**Con eliminación de datos atípicos**	**Sin eliminación de datos atípicos**	**Con eliminación de datos atípicos**
Regresión Logística	**C**	0,1	0,1	75,75%	76,76%
	Penalty	l1	l1		
	Solver	*saga*	*liblinear*		
KNN	***n_neighbors***	93	21	75,24%	76,12%
	metric	*manhattan*	*manhattan*		
Bernoulli Naive Bayes	***Binarize***	0,35	0,35	72,82%	73,43%
	Alpha	0	0,5		
	Force_alpha	*True*	*False*		
Random Forest	***n_estimators***	500	1000	75,27%	75,96%
	max_depth	5	5		
	criterion	*Gini*	*Gini*		
Support Vector Classifier	**C**	1	1	76,50%	77,47%
	Kernel	*rbf*	*rbf*		
	gamma	*auto*	*auto*		
XGBoost Classifier	***learning_rate***	0,1	0,06	76,49%	77,65%
	n_estimators	500	100		

	max_depth	3	5		
Adaboost Classifier	***learning_rate***	0,03	0,06	76,05%	77,11%
	n_estimators	1000	500		
	algorithm	*SAMME.R*	*SAMME.R*		

Para el caso de la tercera iteración, se buscó evaluar la influencia que tiene el escalamiento de los datos de entrenamiento previo a la ejecución de la búsqueda en malla. Para los algoritmos de regresión logística, *BernoulliNB* y R*andom Forest*, el costo computacional permaneció aproximadamente constante, ya que el tiempo de ejecución de los algoritmos solamente aumentó unos segundos. Para el caso particular del modelo KNN, el costo computacional se duplicó cuando no se escalaron los datos.

Para el caso del *Support Vector Classifier*, se evidenció que el escalamiento de los datos es una etapa previa primordial para la disminución en el costo computacional de este algoritmo. Cuando no se escalaron los datos de entrenamiento, transcurrieron más de 15 horas y el algoritmo no había terminado de ejecutarse. Cuando los datos se escalaron, el algoritmo finalizó su ejecución luego de una hora y 56 min. Los resultados de esta iteración se muestran a continuación:

Tabla 14 Hiperparámetros óptimos y métrica de desempeño en la tercera iteración

Modelo	**Hiperparámetros**	**Valores óptimos**	**Métrica de desempeño evaluada**
			mean_test_f1_score
Regresión logística	**C**	0,1	75,75%
	Penalty	*l1*	
	Solver	*liblinear*	
KNN	***n_neighbors***	93	75,22%
	metric	*manhattan*	
Bernoulli Naive Bayes	***Binarize***	0,35	71,39%
	Alpha	0,5	
	Force_alpha	False	
Random Forest	***n_estimators***	500	75,27%
	max_depth	5	
	criterion	*gini*	
Support Vector Classifier	**C**	1	76,50%
	Kernel	*rbf*	
	gamma	*auto*	

XGBoost Classifier	***learning_rate***	0,1	76,49%
	n_estimators	500	
	max_depth	3	
Adaboost Classifier	***learning_rate***	0,03	76,05%
	n_estimators	1000	
	algorithm	*SAMME.R*	

En la iteración cuatro se implementó la técnica de codificación *One-Hot-Encoding* sobre todas las variables categóricas del *dataset*. Se decidió implementar este método ya que es necesario recordar que los algoritmos de aprendizaje automático le dan más relevancia a aquellas variables que tienen un valor superior a otras y esta manera de representar las variables es una forma de facilitarle a los modelos la interpretación de la importancia que tiene cada característica dentro del *dataset* de entrenamiento. Un gran reto que se presentó luego de implementar esta codificación fue el hecho de que el *dataset* pasó de tener 17 columnas a 89 columnas, lo cual representa un aumento considerable en la dimensionalidad. Este incremento a su vez representó un aumento en el tiempo de ejecución de todos los modelos con respecto a la iteración 3, siendo más evidente en el algoritmo *Support Vector Classifier*, el cual pasó a ejecutarse en ocho horas 16 min durante la etapa de entrenamiento. Los resultados se muestran a continuación:

Tabla 15 Hiperparámetros óptimos y métrica de desempeño en la cuarta iteración

Modelo	**Hiperparámetros**	**Valores óptimos**	**Métrica de desempeño evaluada**
			mean_test_f1_score
Regresión logística	**C**	0,1	75,99%
	Penalty	*l1*	
	Solver	*liblinear*	
KNN	***n_neighbors***	171	76,03%
	metric	*cosine*	
Bernoulli Naive Bayes	***Binarize***	0,35	72,69%
	Alpha	0,5	
	Force_alpha	*False*	
Random Forest	***n_estimators***	500	74,48%
	max_depth	5	
	criterion	*gini*	
Support Vector Classifier	**C**	0,1	76,21%
	Kernel	*linear*	

	gamma	*scale*	
XGBoost Classifier	***learning_rate***	0,06	76,21%
	n_estimators	500	
	max_depth	3	
Adaboost Classifier	***learning_rate***	0,1	75,89%
	n_estimators	1000	
	algorithm	*SAMME.R*	

Ante la alta dimensionalidad que se utilizó en la iteración cuatro, en la quinta iteración no se implementó la codificación *One-Hot-Encoding* sobre todas las variables categóricas del *dataset*, sino únicamente en las variables categóricas binarias. Esto ocasionó que el número de columnas del *dataset* de entrenamiento pasara de ser 89 a 17 nuevamente.

Adicionalmente, dentro de esta iteración se incluyó un nuevo modelo de aprendizaje supervisado, el cual fue la Red Neuronal Artificial proveniente de la librería *Sklearn*. Se decidió incluir este modelo, ya que, si se remiten a los resultados de las iteraciones anteriores, el valor de la métrica de desempeño durante la etapa de entrenamiento no superó el 80%, y se pretendía aumentar este valor con dicho modelo que resulta ser más robusto que los considerados anteriormente. Los resultados se muestran a continuación:

Tabla 16 Hiperparámetros óptimos y métrica de desempeño en la quinta iteración

Modelo	**Hiperparámetros**	**Valores óptimos**	**Métrica de desempeño evaluada**
			mean_test_f1_score
Regresión logística	**C**	0,1	75,75%
	Penalty	*l1*	
	Solver	*saga*	
KNN	***n_neighbors***	93	75,25%
	metric	*manhattan*	
Bernoulli Naive Bayes	***Binarize***	0,35	72,82%
	Alpha	0	
	Force_alpha	*True*	
Random Forest	***n_estimators***	1000	75,31%
	max_depth	5	
	criterion	*gini*	
Support Vector Classifier	**C**	1	76,50%

	Kernel	*rbf*	
	gamma	*auto*	
XGBoost Classifier	***learning_rate***	0,1	76,49%
	n_estimators	500	
	max_depth	3	
Adaboost Classifier	***learning_rate***	0,03	76,05%
	n_estimators	1000	
	algorithm	*SAMME.R*	
Multilayer Perceptron Classifier	***activation***	*logistic*	76,90%
	alpha	0,0001	
	hidden_layer_sizes	(100,)	
	learning_rate	*invscaling*	
	solver	*adam*	

Dentro de la iteración seis, las variables *MentHlth* y *PhysHlth* pasaron de considerarse categóricas a considerarse numéricas. Esto debido a que el rango de valores que pueden tomar dichas variables está entre uno y 30 días. Adicionalmente, se ejecutó una codificación *One-Hot-Encoding* únicamente sobre las variables *Age* y *GenHlth*, debido a que las variables categóricas binarias ya están codificadas, razón por la cual, implementar algoritmos para codificarlas sería redundante. Los resultados fueron:

Tabla 17 Hiperparámetros óptimos y métrica de desempeño en la sexta iteración

Modelo	**Hiperparámetros**	**Valores óptimos**	**Métrica de desempeño evaluada**
			mean_test_f1_score
Regresión logística	**C**	1	76,03%
	Penalty	*l2*	
	Solver	*lbfgs*	
KNN	***n_neighbors***	255	75,90%
	metric	*cosine*	
Bernoulli Naive Bayes	***Binarize***	0	73,52%
	Alpha	0,5	
	Force_alpha	*True*	
Random Forest	***n_estimators***	100	74,86%
	max_depth	5	
	criterion	*log_loss*	
Support Vector Classifier	**C**	10	76,28%

	Kernel	*linear*	
	gamma	*auto*	
XGBoost Classifier	***learning_rate***	0,03	76,41%
	n_estimators	500	
	max_depth	5	
Adaboost Classifier	***learning_rate***	0,1	76,01%
	n_estimators	1000	
	algorithm	*SAMME.R*	
Multilayer Perceptron Classifier	***activation***	*logistic*	76,74%
	alpha	0,0001	
	hidden_layer_sizes	(50,)	
	learning_rate	*adaptive*	
	solver	*adam*	

La séptima y última iteración implementó un noveno modelo de aprendizaje automático supervisado con el objetivo de alcanzar una métrica de desempeño superior al 80% durante la etapa de entrenamiento. Dicho algoritmo consiste en una Red Neuronal Artificial Perceptrón Multicapa parametrizada haciendo uso de la librería *Keras*. Esta librería está especializada en el desarrollo de este tipo de algoritmos de aprendizaje profundo y una de las razones por la que se decidió utilizar este modelo es por el hecho de que puede introducirse el hiperparámetro de *dropout* para controlar el *overfitting*. Los resultados obtenidos con la arquitectura de la red neuronal seleccionada son los que se muestran a continuación:

Tabla 18 Hiperparámetros óptimos y métrica de desempeño en la séptima iteración

Red Neuronal Keras	**Arreglo de capas**	Capa inicial, cinco capas ocultas, seis capas de *dropout* y una capa final	76,75%
	hidden_units	64	
	activation	*tanh*	
	optimizer	*adam*	
	dropout_rate	0,2	

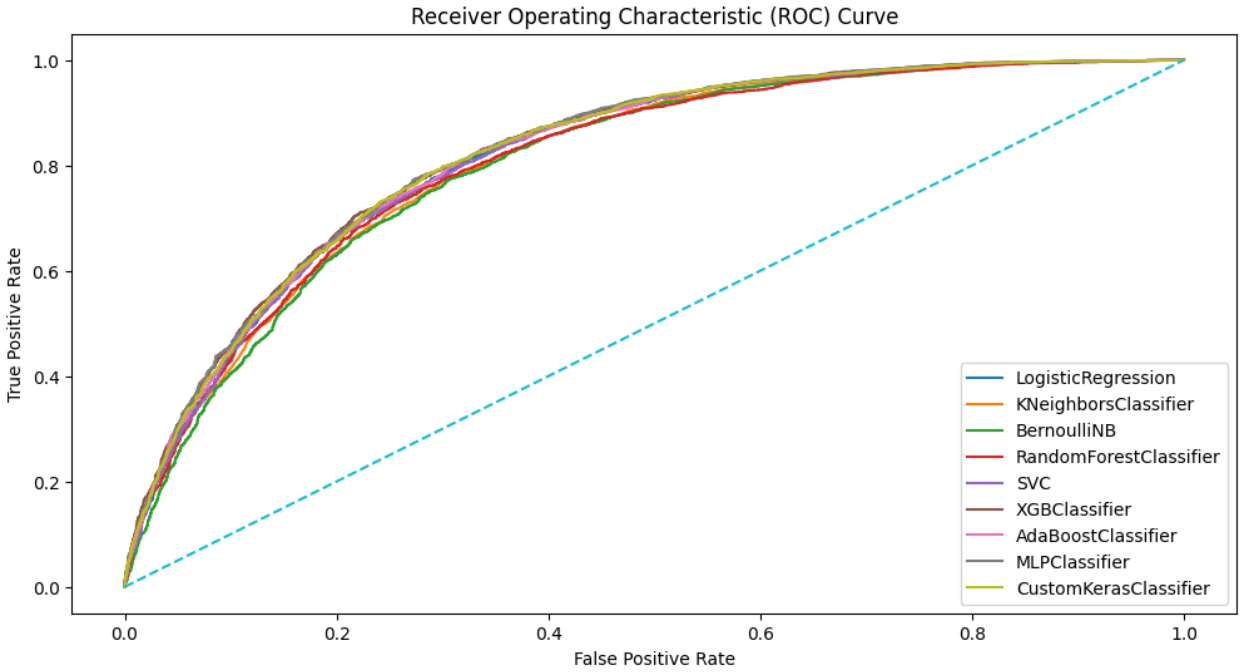
5.4 Herramientas

Las herramientas que se utilizaron durante este proyecto fueron: los entornos de desarrollo *Google Colab* y *Visual Studio Code, Microsoft Excel, OneDrive, Google Drive, Sharepoint, Chat GPT, Google Bard, Github, Github Copilot, Microsoft Word, Microsoft Teams, Google Meet* y *WhatsApp*. El lenguaje de programación empleado fue *Python*.

6. Resultados y discusión

Luego de realizar siete iteraciones con diferentes algoritmos de aprendizaje supervisado, se logró obtener un valor máximo en la métrica *F1-score* de 76,75% con el modelo de una Red Neuronal Perceptrón Multicapa cuya arquitectura se define en la **Tabla *18***.

Figura 11 Curva ROC para los modelos considerados en la séptima iteración



Un hecho que se desprende de la **Figura 11** es que las curvas de características operativas del receptor (ROC, por sus siglas en inglés), se encuentran muy cercanas entre sí para todos los algoritmos considerados. La proximidad de las curvas ROC para los nueve modelos indica un alto grado de similitud en su capacidad para distinguir entre clases, mostrando un nivel uniforme de rendimiento predictivo de probabilidad. Sin embargo, es necesario recordar que esto no implica que los modelos estén desempeñándose bien; simplemente sugiere que están funcionando de manera similar. Es necesaria una evaluación adicional, como un análisis de las métricas de desempeño y de negocio para obtener una comprensión completa de su rendimiento general.

6.1. Métricas

Comparando los resultados de la **Tabla 12** y la **Tabla 13**, puede apreciarse que, durante la etapa de entrenamiento, cuando no se truncaron los datos del *dataset*, los modelos *XGBoost Classifier* y *Adaboost Classifier* tuvieron un cambio en los mejores hiperparámetros con respecto a cuando sí se truncaron los valores del *dataset*. Esto permite establecer que solamente estos dos modelos resultaron ser sensibles a la proporción de datos con los que debe entrenarse el modelo.

Remitiéndose a la **Tabla *14***, puede apreciarse que el modelo que tuvo un mayor impacto en cuanto a su desempeño cuando no se escalaron los datos previos al entrenamiento fue el *Bernoulli Naive Bayer Classifier*. Esto debido a que obtuvo el menor valor de *F1-score* promedio, el cual fue de 71,39%. Sin embargo, esta métrica no se encuentra muy alejada del valor máximo obtenido en esta iteración, el cual fue de 76,50% y lo alcanzó el modelo *Support Vector Classifier*.

Por otra parte, si se comparan los valores de *F1-score* alcanzados durante la tercera iteración (**Tabla *14***) con respecto a la segunda iteración (**Tabla *13***), no se evidencia que el escalamiento de datos previo al entrenamiento, tenga influencia sobre los modelos de Regresión logística, KNN, *Random Forest, Support Vector Classifier, XGBoost Classifier* y *Adaboost Classifer*, ya que la métrica alcanza los mismos valores tanto en el escenario de escalamiento de datos como sin escalamiento.

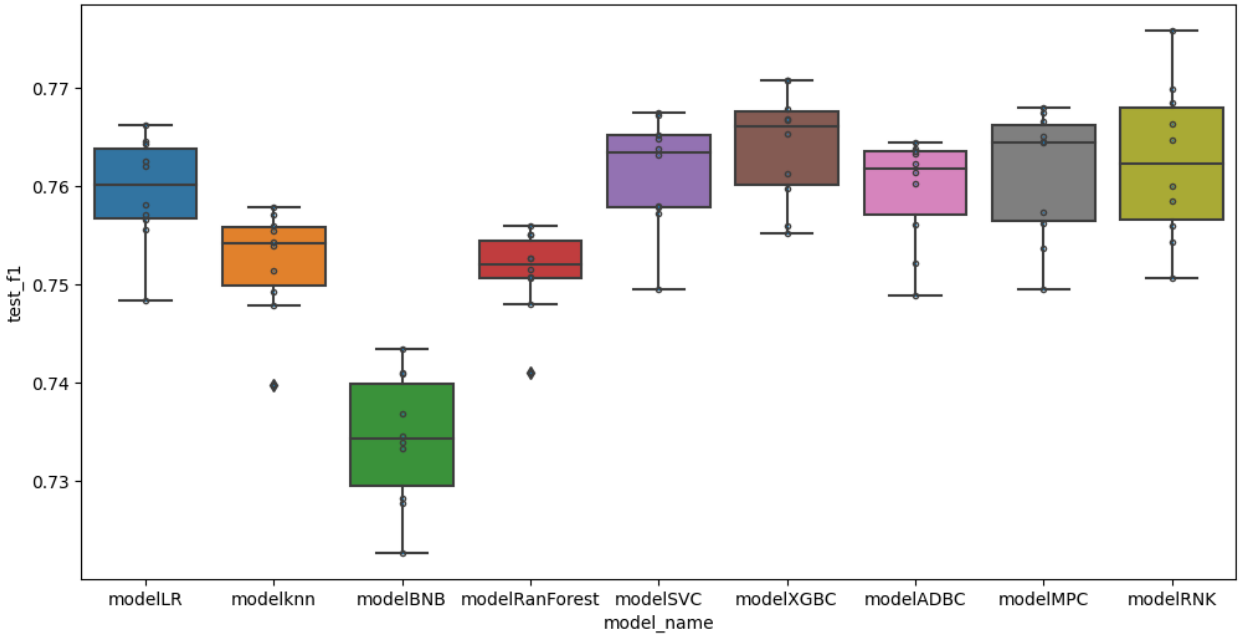
Teniendo en cuenta los resultados de la **Tabla *15***, puede evidenciarse que la codificación *One-Hot-Encoding* sobre todas las variables categóricas del *dataset* de entrenamiento, contribuyó en el cambio de los mejores hiperparámetros para los modelos KNN, *Support Vector Classifier, XGBoost Classifier* y *Adaboost Classifier*, con respecto a los obtenidos en la **Tabla *14***. Adicionalmente, el valor de la métrica de desempeño mejoró levemente para los modelos de regresión logística, KNN y *Bernoulli Naive Bayes*. Para el caso de los demás modelos, esta codificación influyó en el desempeño de manera negativa. Los anteriores resultados permiten establecer que, los modelos de aprendizaje automático supervisado más robustos empleados en este proyecto de monografía, tienen cierta sensibilidad a la codificación *One-Hot-Encoding*, ocasionando que su desempeño se vea disminuido.

Para el caso de los resultados consignados en la **Tabla *16*** y comparándolos con los de la **Tabla *15***, al menos uno de los hiperparámetros de los modelos que se consideraron, sufrió un cambio en sus valores óptimos. Adicionalmente, en esta iteración fue la primera vez que se implementó una red neuronal perceptrón multicapa que permitió alcanzar el mejor valor de métrica de desempeño, (hasta ese momento), para el escenario en el que no se implementó una eliminación controlada de datos atípicos, con un valor de 76,90%.

Posteriormente, comparando los resultados de la **Tabla *17*** con respecto a los de la **Tabla *16***, puede apreciarse que todos los modelos sufren un cambio sobre al menos uno de sus mejores hiperparámetros. Adicionalmente, se alcanza un resultado que ya se había obtenido durante la comparación de **Tabla *14*** y la **Tabla *15*** y es el hecho de que la codificación *One-Hot-Encoding* resulta influyente sobre el desempeño de los modelos de aprendizaje supervisado más robustos para este conjunto de datos en particular. Esto debido a que se presentó una disminución de la métrica de desempeño para los modelos: *Random Forest*, *Support Vector Classifier*, *XGBoost Classifier*, *Adaboost Classifer* y *Multilayer Perceptron Classifier.*

Finalmente, en la última iteración se alcanzaron los mismos resultados que en la **Tabla *17*** en cuanto a los mejores hiperparámetros y métrica de desempeño para los modelos considerados hasta ese momento. En la **Tabla *18***, se resume la arquitectura y la métrica de desempeño alcanzada haciendo uso de una red neuronal perceptrón multicapa desarrollada a través de la librería *Keras*. Este resultó ser el mejor modelo del proyecto, ya que alcanzó el mayor valor de métrica de desempeño durante la última iteración del proyecto. Esto sumado a que su costo computacional es menos elevado que el que requieren modelos como el *Support Vector Classifier* y el *Multilayer Perceptron Classifier* de la librería *scikit-learn.*

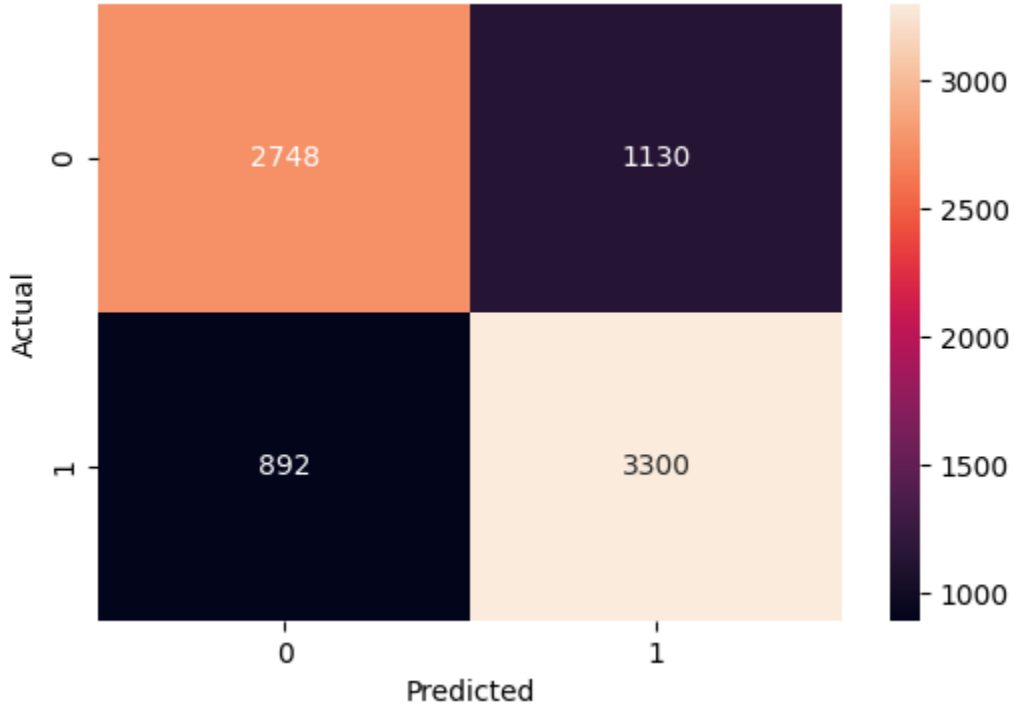
Figura 12 Gráfico de caja y bigotes para la métrica de desempeño de la última iteración.



Como puede apreciarse en la **Figura 12**, el modelo de Red Neuronal Perceptrón Multicapa desarrollado a través de la librería *Keras* alcanzó la distribución más uniforme en cuanto a los valores de métrica de desempeño, esto debido a que no hay presencia de datos atípicos sobre su distribución, ya que no hay valores por debajo del bigote inferior ni por encima del bigote superior.

Los modelos *Multilayer Perceptron Classifier, Adaboost Classifier, XGBoost Classifier, Support Vector Classifier, Bernoulli Naive Bayes* y regresión logística también alcanzaron una distribución uniforme, pero con valores inferiores a los alcanzados por la Red Neuronal desarrollada con *Keras*. Los modelos KNN y *Random Forest Classifier* por su parte fueron los únicos modelos en presentar datos atípicos en su distribución, ya que ambos modelos exhiben valores inferiores al bigote inferior.

Figura 13 Matriz de confusión construida a partir de la Red Neuronal empleando *Keras*



La matriz de confusión dispuesta en la **Figura 13**, es una matriz de 2x2 en la que se comparan los resultados de la predicción de probabilidad de diabetes en pacientes alcanzada, haciendo uso de la arquitectura de la Red Neuronal Perceptrón Multicapa, con respecto a los valores reales de la variable de respuesta para el 10% remanente de la base de datos que no se empleó ni en la etapa de entrenamiento ni en la etapa de validación cruzada. Es decir, una proporción de la base de datos hasta ahora no explorada por ninguno de los modelos de aprendizaje automático supervisado.

Al haber dos filas y dos columnas en esta matriz, significa que hay dos clases o etiquetas en la variable de respuesta: 0 (paciente sin diabetes) y 1 (paciente con diabetes). Las filas de la matriz representan las etiquetas de la clase para cada muestra, es decir los valores reales de salida y las columnas representan las etiquetas resultantes de la predicción hecha por el modelo para cada muestra evaluada.

Los principales resultados dentro de la matriz de confusión son:

- **Verdaderos positivos (TP):** Corresponden con el número de instancias que se predijeron correctamente como positivas o 1. En este caso, hay 3300 verdaderos positivos.

- **Falsos positivos (FP):** Equivalen al número de instancias que se predijeron incorrectamente como positivas o 1, es decir, las muestras son negativas (0) pero el modelo las clasificó como positivas (1). En este caso, hay 1130 falsos positivos.
- **Verdaderos negativos (TN):** Representan el número de instancias que se predijeron correctamente como negativas o 0. En este caso, se tienen 2748 verdaderos negativos.
- **Falsos negativos (FN):** Esta cantidad contiene el número de instancias que se predijeron incorrectamente como negativas o 0, en otras palabras, las muestras son positivas (1), pero el modelo las clasificó como negativas (0). En este caso, son 892 falsos negativos.

Con base en los anteriores resultados para esta matriz de confusión, es posible calcular las siguientes métricas de desempeño:

- ***Accuracy*:** Corresponde con el porcentaje de instancias que se predijeron correctamente. La ecuación que permite conocer esta métrica es la siguiente:

$$Accuracy = \frac{TP+TN}{TN+FP+TP+FN}$$ **Ecuación 1**

En este caso, el *accuracy* es (3300 + 2748) / (2748 + 1130 + 3300 + 892) = 83,2%.

- ***Precision*:** Equivale al porcentaje de casos positivos pronosticados que fueron realmente positivos, la cual está dada por:

$$Precision = \frac{TP}{TP+FP}$$ **Ecuación 2**

En este caso, la precisión es 3300 / (3300+1130) = 74,5%.

- ***Recall*:** Esta cantidad es igual al porcentaje de casos positivos reales que se predijo que serían positivos. Su ecuación es la siguiente:

$$Recall = \frac{TP}{TP+FN}$$ **Ecuación 3**

En este caso, el *recall* es 3300 / (3300 + 892) = 78,7%.

- ***F1-score:*** Esta cantidad representa un promedio armónico de *precision* y *recall*. Un promedio armónico es un tipo de promedio que se calcula tomando el recíproco del

promedio de los recíprocos de los valores de los datos. Es un promedio más conservador que la media aritmética y es menos sensible a los valores atípicos (B, 2021). Su ecuación está dada por:

$$F1_{Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$ **Ecuación 4**

En este caso, el F1-Score es 76,5%.

6.2. Evaluación cualitativa

Un resultado que vale la pena destacar de las iteraciones de este proyecto, es que en ninguno de los experimentos se presentó el fenómeno de *overfitting*. Esto debido a que la métrica de desempeño tanto en la etapa de entrenamiento como de validación alcanzó valores similares y no excedieron el 80% de *F1-score*, lo que implica que todos los modelos fueron capaces de generalizar los resultados de la etapa de entrenamiento y no "memorizaron" los datos. Tampoco se presentaron casos de *underfitting* en ninguna de las iteraciones, ya que los modelos eran lo suficientemente robustos y complejos como para capturar los principales patrones entre los datos de entrenamiento, y así poder llevar a cabo predicciones de probabilidad de diabetes en pacientes de manera acertada.

Otro resultado que se desprende de esta serie de experimentaciones y que debe tenerse en cuenta en futuros proyectos que puedan surgir de esta investigación, es el hecho de que la métrica de distancia del algoritmo KNN es realmente muy importante para poder obtener resultados luego de la aplicación de este algoritmo. Cuando se empleó un número de estimadores mayor a 170 pero menor a 260 con la métrica de distancia *cosine*, la ejecución del código terminaba por congelar todos los procesos que en ese momento se estuvieran ejecutando en el ordenador que se empleó durante este proyecto, y por lo tanto el computador dejaba de responder. Sin embargo, cuando se empleaba la métrica de distancia *Manhattan*, los algoritmos sí finalizaron su ejecución. Las iteraciones para las cuales la métrica de distancia *cosine* resultó ser el mejor hiperparámetro para el algoritmo KNN, fueron aquellas en las cuales se utilizó la codificación *One-Hot-Encoding*. El cálculo de la similitud del coseno puede ser costoso desde el punto de vista computacional, especialmente para grandes conjuntos de datos o en casos en los que la dimensionalidad de los datos es alta.

Comparando los resultados que se obtienen para la precisión y el *recall*, se puede establecer que el modelo seleccionado tiene un mejor desempeño en la predicción de la probabilidad de la clase negativa o cero, en comparación con la predicción de la probabilidad de la clase positiva o uno. Esto se debe a que, de acuerdo con la **Ecuación 3**, el *recall* es la relación entre el número de resultados positivos verdaderos y el número de todas las muestras relevantes (es decir, la suma de los positivos verdaderos y los falsos negativos). En pocas palabras, este mide la capacidad del clasificador para encontrar todos los casos relevantes dentro de un conjunto de datos. Un valor alto de *recall*, (78,7% en este caso), indica que el clasificador tiene una tasa baja de falsos negativos. Por otra parte, la **Ecuación 2** permite evidenciar que la precisión es la relación entre el número de resultados positivos verdaderos y el número de todos los resultados positivos devueltos por el clasificador. Es decir, es la medida de la precisión de las predicciones positivas realizadas por el clasificador. Una puntuación de precisión alta, (74,5% en este caso), indica que el clasificador tiene una tasa baja de falsos positivos.

Finalmente, es necesario mencionar que la métrica de desempeño de este proyecto nunca superó el umbral del 78%. Adicionalmente, una correlación que existe entre esta métrica (*F1-score*), con la métrica de negocio (el modelo de aprendizaje supervisado que se ofrecerá al médico tratante como una herramienta de apoyo adicional a las ya existentes para determinar la probabilidad de que un paciente padezca de diabetes), es el hecho de que entre más complejo sea el modelo de aprendizaje supervisado, el valor de *F1-score* aumenta levemente, pero no logra superar el 78% en ninguna de las etapas evaluadas durante este proyecto, las cuales fueron entrenamiento, validación y prueba.

6.3.Consideraciones de producción

Al considerar la puesta en producción de un modelo de red neuronal desarrollado con *Keras*, es crucial tener en cuenta diversas consideraciones técnicas para garantizar un despliegue efectivo y eficiente. Algunas de estas son:

- **Reentrenamiento constante del modelo:** Una vez el modelo se encuentre desplegado en un entorno productivo, se deberán definir plazos prudentes de recolección de datos y reentrenamiento del modelo que se seleccionó como óptimo en este proyecto de

monografía. Esta etapa es vital para obtener una predicción de probabilidad de diabetes en pacientes en productivo ya que el comportamiento de la *data* original de entrenamiento cambia y se generan nuevos patrones en los datos que pueden mejorar o empeorar el rendimiento con cada paciente nuevo que ingresa al hospital.

- **Integración con *streams* de datos:** Para un despliegue en productivo del modelo que exhibió un mejor desempeño durante este proyecto, será necesario establecer una integración fluida con los repositorios de datos para asegurar que el modelo se mantenga vigente. Esto puede implicar el diseño de un sistema de ingestión de datos que permita la actualización continua del modelo en función de la información más reciente disponible, es decir, con cada paciente que ingresa al hospital.
- **Implementación de *APIs*:** Será necesario desarrollar *APIs* que permitan el acceso al modelo desde otros sistemas y aplicaciones. Esto facilita la integración del modelo en aplicaciones web o móviles y permite a los usuarios acceder a las funcionalidades del modelo de forma sencilla y segura.
- **Ciberseguridad y privacidad de los datos:** Es primordial implementar protocolos de seguridad robustos para proteger los datos sensibles utilizados en el entrenamiento y la validación del modelo. Esto puede implicar el uso de técnicas de encriptación, autenticación y autorización, así como el cumplimiento de regulaciones de privacidad de datos, como *GDPR* o *HIPAA*, según corresponda.
- **Versionado y control de la configuración:** Finalmente, será necesario establecer un sistema de control de versiones para el código fuente y los prototipos del modelo, lo que facilita el seguimiento de cambios y la reproducción de resultados. Además, implementar prácticas de gestión de la configuración para garantizar la consistencia y reproducibilidad del entorno de producción.

7. Conclusiones

El análisis exploratorio de datos inicial que se realizó sobre el *dataset* original permitió identificar que no existen correlaciones fuertes entre ningún par de variables. Esto debido a que los valores de los coeficientes de correlación de la matriz que se encuentra en la **Figura 4**, no sobrepasan el 53%, el cual está alejado de la unidad.

El análisis *LIME* permitió identificar que el balanceo de las clases en la variable de respuesta se mantuvo incluso cuando se grafica ésta como función de una variable de entrada del *dataset*. Este resultado se puede evidenciar en la **Figura *5***, donde al representar el porcentaje de hombres y mujeres como función de las clases 0 y 1 de la variable diabetes, los porcentajes no varían en gran proporción entre un género y otro.

De todas las variables del *dataset*, únicamente en 2 de ellas se implementó la codificación O*ne-Hot-Encoding*. Éstas fueron las variables asociadas con la edad y con el estado de salud en general del paciente. Únicamente se implementó la codificación O*ne-Hot-Encoding* sobre estas dos variables debido a que, en su gran mayoría, las variables eran de tipo binaria, las cuales no requieren de una codificación O*ne-Hot-Encoding* previa o porque la distribución de la variable era continua en comparación con estas dos características.

Para el modelo SVC, en todos los experimentos fue necesario llevar a cabo un escalamiento de los datos con los cuales éste se entrenó. Si no se hacía el escalamiento previo al entrenamiento, después de 15 horas, el algoritmo no finalizaba su ejecución.

La metodología que se utilizó para encontrar los mejores hiperparámetros de cada uno de los modelos evaluados corresponde con el *pipeline* que debe seguirse a la hora de emplear algoritmos de aprendizaje automático: se realizó un análisis exploratorio de los datos con el objetivo de identificar correlaciones entre pares de variables de entrada, al ser un proyecto de clasificación binaria, se evaluó qué tan balanceadas estaban las etiquetas de la variable de salida o de respuesta. Se empleó el algoritmo de búsqueda en malla con un porcentaje del 80% de la base de datos original para encontrar la combinación de hiperparámetros de cada modelo que condujera al mayor *F1-score* de entrenamiento. Con este resultado, se empleó el algoritmo de validación cruzada haciendo uso de los mejores hiperparámetros encontrados en la etapa de búsqueda en

malla, esta vez con un 10% adicional de la base de datos original. Finalmente, con los resultados del mejor modelo con sus mejores hiperparámetros, el cual proviene de la validación cruzada, se llevó a cabo la etapa de pruebas con el 10% restante de la base de datos que era desconocida para este, el cual resultó ser una Red Neuronal Perceptrón Multicapa cuya arquitectura fue definida haciendo uso de la librería *Keras*, tal como lo muestra la **Tabla *18***. Este modelo fue seleccionado como el óptimo para este proyecto de monografía, dado que fue el que demostró un rendimiento superior en comparación con los otros modelos de aprendizaje automático considerados, además, tiene un bajo costo computacional. Esto sugiere que la estructura y la capacidad de aprendizaje de la red neuronal son capaces de capturar la complejidad de los datos y proporcionar resultados precisos.

Finalmente, es necesario considerar que la métrica de desempeño seleccionada para este proyecto de monografía no superó el umbral del 78% pese a los diferentes modelos de aprendizaje automático supervisado que fueron utilizados. No se le atribuye el resultado a la metodología de las experimentaciones, ya que estas incluyeron modelos con una amplia gama de parámetros y que además van desde lo más simple, (como una regresión logística binomial), hasta lo más complejo (como una máquina de soporte vectorial). Por lo tanto, se concluye que fue la base de datos original la que impidió obtener mejores resultados en cuanto a la predicción de probabilidad de diabetes en pacientes.

8. Recomendaciones

Las investigaciones futuras que se desprenden de este proyecto de monografía deberán enfocarse en implementar una etapa previa al análisis exploratorio de los datos, la cual consiste en enriquecer el *dataset* original. Este procedimiento permite preparar el terreno para la implementación exitosa de modelos de aprendizaje automático supervisado al mejorar la calidad, cantidad y relevancia de los datos, lo que a su vez conduce a mejores resultados predictivos y una comprensión más profunda del problema.

Es necesario mencionar que durante la ejecución del algoritmo KNN, se presentó lo que se conoce como la "maldición" de la dimensionalidad, que para efectos de este trabajo es cuando se implementa un número de estimadores mayor a 170 pero menor a 260 junto con una métrica de distancia *cosine*. Se recomienda no optar por llevar a cabo predicciones de probabilidad de diabetes en pacientes con este modelo para bases de datos tan robustas como la que se empleó durante este proyecto de monografía. Adicionalmente, a la hora de ejecutar un algoritmo de máquina de soporte vectorial con una base de datos cuyo número de registros sea similar al de la base de datos que se implementó durante este proyecto, se requiere hacer una normalización previa sobre las variables de entrada. Este procedimiento previo disminuye considerablemente el costo computacional del modelo durante la etapa de entrenamiento, ya que el algoritmo pasará de compilarse durante 15 horas y no finalizar su ejecución, a finalizar su ejecución en tan solo una hora y 56 min.

Finalmente, se recomienda que las capacidades internas del ordenador donde se lleven a cabo las ejecuciones del código de este proyecto de monografía o de investigaciones similares, cuenten como mínimo con una unidad de procesamiento gráfica adicional a la que viene con el procesador del equipo de cómputo por defecto, si es que las líneas de código se ejecutarán en el entorno local. Adicionalmente, se debe contar con un procesador de mínimo ocho núcleos. Estos requerimientos de hardware tan específicos se deben al hecho de que los modelos utilizados son muy robustos, razón por la cual su máxima capacidad se optimiza con este tipo de unidades de procesamiento y utilizando el hiperparámetro *n_jobs* con un valor de -1 cuando sea posible.

Referencias

B, H. N. (2021, 12 diciembre). Confusion matrix, accuracy, precision, recall, F1 score. Medium. https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd#:~:text=F1%20Score%20becomes%201%20only,0.857%20%2B%200.75)%20%3D%200.799.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32. https://doi.org/10.1023/a:1010933404324

CDC - 2015 BRFSS Survey Data and documentation. (s. f.). https://www.cdc.gov/brfss/annual_data/annual_2015.html

Cómo evaluar su peso. (2023, 21 julio). Centers for Disease Control and Prevention. https://www.cdc.gov/healthyweight/spanish/assessing/index.html#:~:text=Si%20su%20IMC%20es%20entre%2025.0%20y%2029.9%2C%20se%20encuentra,dentro%20del%20rango%20de%20obesidad

Diabetes Prediction Competition (TFUG CHD Nov 2022) | Kaggle. (s. f.). https://www.kaggle.com/competitions/diabetes-prediction-competitiontfug-chd-nov-2022/data

La diabetes y la salud mental. (2023, 15 mayo). Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/spanish/living/mental-health.html

La diabetes y su corazón. (2021, 27 abril). Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/spanish/resources/features/diabetes-and-heart.html#:~:text=Modo%20en%20la%20diabetes%20afecta%20el%20coraz%C3%B3n&text=La%20presi%C3%B3n%20arterial%20alta%20aumenta,el%20riesgo%20de%20enfermedad%20cardiaca

Síntomas y causas de la diabetes - NIDDK. (s. f.). National Institute of Diabetes and Digestive and Kidney Diseases. https://www.niddk.nih.gov/health-information/informacion-de-la-salud/diabetes/informacion-general/sintomas-causas

Sklearn.linear_model.LogisticRegression. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn.naive_bayes.BernoulliNB. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html

Sklearn.neighbors.KNeighborsClassifier. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Sklearn.svm.SVC. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Sociedad Iberoamericana de Información Científica (SIIC). (s. f.). *DIABETES, TRASTORNOS PSIQUIÁTRICOS e INTERACCIONES ENTRE AMBAS ENTIDADES.* https://www.siicsalud.com/dato/resiiccompleto.php/130807#:~:text=Los%20enfermos%20con%20esquizofrenia%20tienen,en%20los%20pacientes%20con%20esquizofrenia

Story, C. M. (2023, 1 junio). *A guide to living with diabetes and high cholesterol.* Healthline. https://www.healthline.com/health/high-cholesterol/treating-with-statins/guide-to-diabetes-and-high-cholesterol

Anexos

Anexo 1. *Notebooks* con los experimentos realizados durante el proyecto de monografía

- Exp 1 Eliminación de atípicos, datos truncados y escalamiento minmax antes del Split
- Exp 2 Variable GenHlth categórica, sin truncar datos y escalamiento minmax antes del Split
- Exp 3 Solo BMI numérica, sin truncar datos, sin eliminar atípicos y sin escalamiento minmax antes del Split
- Exp 4 Variables categóricas dummies salvo diabetes, drop_first = True y sin escalamiento minmax antes del Split
- Exp 5 Variables categóricas dummies salvo diabetes, drop_first = True y datos escalados minmax antes del Split
- Exp 6. Simulación final sin red de keras
- Exp 7. Simulación final

Estos anexos pueden encontrarse en los siguientes repositorios de GitHub:

Repositorio GitHub Ana Estefanía Henao Restrepo: https://rb.gy/hid3y

Repositorio GitHub Juan José Gil Hoyos: https://rb.gy/9i8qt