Desarrollo de un modelo de Machine Learning para la clasificación de tipos de dengue de acuerdo a su nivel de severidad: Un estudio de caso de Bucaramanga, Colombia

Juan David Rojas Sánchez

Universidad El Bosque
Facultad de Ciencias
Departamento de Matématicas
Programa de Estadística
Bogotá D.C, Colombia
2023

Desarrollo de un modelo de Machine Learning para la clasificación de tipos de dengue de acuerdo a su nivel de severidad: Un estudio de caso de Bucaramanga, Colombia

Juan David Rojas Sánchez

Tesis como requisito parcial para optar al título de:
Estadístico

Director:
M.Sc. Jesús D. Ramos M.

Universidad El Bosque
Facultad de Ciencias
Departamento de Matemáticas
Programa de Estadística o Matemáticas
Bogotá D.C, Colombia
2023

Agradecimientos

Después de este largo proceso que se ha presentado como un carrusel de emociones, me gustaría agradecer a mi familia: mis padres y mi hermana, por acompañarme desde que inició mi proceso académico, por sus oraciones y frases de apoyo.

A mi novia por ser mi bastón, mi constante apoyo y mi polo a tierra en los momentos más difíciles.

A mi tutor Jesús David Ramos por la confianza, la paciencia, todos los consejos y, por supuesto todo el aporte que hizo a mi desarrollo personal y profesional.

A la Universidad, a los docentes que me acompañaron en este camino y cada una de las personas que he tenido el privilegio de haberme cruzado.

Resumen

El dengue en Colombia y en la región representa una importante problemática de salud pública, por las condiciones geográficas y sociales que hay en el país, se presentan focos cíclicos de contagio. Los avances en machine learning (ML) y ciencia de datos para la clasificación de pacientes puede representar una reducción de esfuerzos médicos, económicos y humanos para el tratamiento de la enfermedad. El diagnóstico temprano, ofrece conocimiento y seguimiento de la enfermedad. Los datos obtenidos provienen del municipio de Bucaramanga, Santander, uno de los departamentos más afectados por los brotes de dengue. Para lograr el objetivo de construir un clasificador de tipos de dengue se construyen 4 modelos ML: Regresión Logística Regularizada (RL), Random Forest (RF), Maquina de Soporte Vectorial para Clasificación (SVC) y una propuesta de ensamble de estos tres modelos que toma como meta-clasificador al algoritmo de XGBoost. Los resultados muestran como mejor modelo al modelo ensamblado (AUC = 0.9386, Accuracy = 0.936, F1-Score = 0.947), seguido de la Regresión Logística regularizada por norma ℓ_2 (AUC = 0.95, Accuracy = 0.871, F1-Score = 0.895), la Máquina de Soporte de Vectorial - Kernel Radial (AUC = 0.984, Accuracy = 0.857, F1-Score = 0.867) y por último, el *Random Forest* (AUC = 0.94, Accuracy = 0.833, F1-Score = 0.865). Además se encontró que factores como antecedentes familiares por dengue, dolor abdominal, vomito y diarrea presentan una relación causal con el presentar dengue con signos de alarma.

Palabras clave: Bioestadística, Machine Learning, Dengue, Clasificación.

Abstract

Dengue in Colombia and in the region represents a major public health problem, due to the geographical and social conditions in the country, there are cyclical outbreaks of contagion. Advances in machine learning (ML) and data science for the classification of patients may represent a reduction of medical, economic and human efforts for the treatment of the disease. Early diagnosis offers knowledge and monitoring of the disease. The data obtained come from the municipality of Bucaramanga, Santander, one of the departments most affected by dengue outbreaks. To achieve the objective of building a classifier of dengue types, 4 ML models are built: Regularized Logistic Regression (RL), Random Forest (RF), Support Vector Classification Machine (SVC) and a proposal for the assembly of these three models that takes the XGBoost algorithm as meta-classifier. The results show that the best model is the ensemble model (AUC = 0.9386, Accuracy = 0.936, F1-Score = 0.947), followed by the Logistic Regression regularized by norm ℓ_2 (AUC = 0.95, Accuracy = 0. 871, F1-Score = 0.895), the Support Vector-Radial Kernel Machine (AUC = 0.984, Accuracy = 0.857, F1-Score = 0.867) and lastly, the Random Forest (AUC = 0.94, Accuracy = 0.833, F1-Score = 0.865). It was also found that factors such as family history of dengue, abdominal pain, vomiting and diarrhea had a causal relationship with the presentation of dengue with alarm signs.

Keywords: Biostatistics, Machine Learning, Dengue, Classification.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto

El dengue es una enfermedad de tipo viral, de carácter endémico-epidémico, la cual es trasmitida por mosquitos del género Aedes (Torres, 2008), principalmente por Aedes aegypti, dicha enfermedad es causada por los cuatro serotipos del virus del dengue (DENV 1-4), los serotipos 2 y 3 han sido asociados a la mayor cantidad de casos graves y fallecimientos.

El dengue se ha convertido en un importante problema de salud pública con serias repercusiones tanto sociales como económicas, debido a, como exponen Guzman y Harris (2015), su repentino aumento de casos, la extensión geográfica que ha presentado y la gravedad misma de la enfermedad. Las autoras también detallan que el dengue es endémico en más de 100 países entre el sudeste asiático, América, el pacífico occidental, África y el mediterráneo oriental, lugares en donde su incidencia se ha multiplicado por 30 en los últimos 50 años (OMS), siendo la enfermedad vírica trasmitida por mosquitos, que más rápido se extiende.

Algunas de las expresiones clínicas que presenta el dengue son: fiebre aguda, fiebre con cefalea, malestar general, dolores osteomioarticulares, exantemas, leucopenia, sangrados, hemorragias en aparato digestivo, trombocitopenia moderada o intensa, entre otras expresiones.

La OMS estima que por año se producen entre 50 y 100 millones de infecciones por dengue. Cerca de la mitad de la población mundial vive en países en donde el dengue es endémico y actualmente, cerca del 75 % de la población expuesta al dengue se encuentra en la región de Asia en el pacífico.

La investigación de Castrillón y cols. (2015) muestra que el comportamiento de los brotes de dengue en Colombia es de carácter cíclico y los casos son reportados principalmente en los departamentos de Norte de Santander, Santander, Huila, Tolima, Valle del Cauca y Antioquia. Se sospecha que el cambio climático, el índice de desarrollo y las condiciones de vida han influido en el drástico aumento de casos desde que se describió el primer caso en la nación, en diciembre de 1989 en Puerto Berrío (Antioquia).

Es importante el poder realizar diagnósticos tempranos o preliminares de dicha enfermedad, pues como explica la OMS, las epidemias por dengue provocan sufrimiento humano, sobrecarga en los servicios sanitarios, además de fuertes pérdidas económicas. Los aumentos inesperados de casos suponen un reto para los sistemas sanitarios, que es el poder clasificar grandes volúmenes de casos, a parte del poder identificar que casos graves requerirán de servicios adicionales. Como explican Gutiérrez-Ruiz y cols. (2012), el diagnóstico de dengue inicial es presuntivo, es decir, basado en las manifestaciones clínicas del paciente, el diagnóstico confirmatorio se realiza con base en pruebas de laboratorio, ya sean celulares o inmunológicas (Guzmán y Kourí, 2004). Este diagnóstico

presuntivo sirve como acercamiento inicial al abordaje terapéutico de la enfermedad; si este diagnóstico inicial es adecuado y oportuno se puede proceder a trazar una ruta de tratamiento más rápida para el paciente.

El propósito de este trabajo es la construcción de una herramienta que sirva de apoyo en el diagnostico inicial del dengue según la gravedad de los síntomas presentados por los pacientes, apoyado en técnicas robustas de *machine learning*. Esta herramienta permitirá la clasificación de tipo de dengue según su nivel de severidad.

1.2. Antecedentes

Actualmente, se cuenta con el importante desarrollo de técnicas y algoritmos más específicos que se han implementado en el área de la medicina, los cuales han permitido una mayor eficacia en labores de diagnosis, que a su vez permite tratamientos más oportunos, que pueden acarrear costos menores. Esto es de vital importancia para enfermedades tan incidentes como el dengue y demás infecciones víricas de rápida transmisión. Pues, como concluye [@HOYOS2021], aplicaciones de **machine learning** para el desarrollo de modelos causa-efecto permiten mejoras en diagnósticos y entendimiento de la enfermedad; además de la utilidad que estas herramientas brindan en ocasiones en que la calidad de la información es baja durante la atención sanitaria.

A continuación, se presentan algunos trabajos a nivel global, que buscan el ajuste y construcción de modelos para el diagnóstico de casos de dengue.

- Rodríguez y Correa (2009) proponen un análisis de la dinámica de la epidemia del dengue en Colombia desde 1990 hasta 2006 como una caminata al azar probabilista. Con el fin de predecir el número de pacientes infectados para el año 2007. El valor real de infecciones en ese año, correspondió al 90.4 % del valor predicho por los investigadores.
- Guo y cols. (2017) usan algoritmos de aprendizaje automático (para regresión en este caso) para pacientes de la ciudad de Guangdong como, Regresión de Vector de Soporte (SVR), regresión lineal, arboles de regresión potenciados por gradiente (GBM), regresión binomial negativa, regresión LASSO y modelos aditivos generalizados (GAM). El rendimiento de dichos modelos se evaluó utilizando medidas de error cuadrático medio (RMSE) y R-cuadrado. En este estudio se concluye que el modelo propuesto por SVR fue el que mejor rendimiento de predicción presentó en comparación con las otras técnicas propuestas.
- Gambhir y cols. (2018) comparan tres técnicas de clasificación para casos de dengue de la ciudad de Delhi, India, estas son, Redes Neuronales Artificiales (ANN), árboles de decisión (DT) y el algoritmo Naive Bayes (NB), el rendimiento de estos modelos fue evaluado con las métricas de Precisión, Sensibilidad, Especificidad y Tasa de Mala Clasificación (TMC) por validación cruzada e 10 pliegues, siendo el modelo por ANN el que mejor rendimiento presentó en ya mencionados pacientes.
- Caicedo-Torres y cols. (2017) aplican modelos basados en funciones kernel o funciones núcleo, sobre datos proporcionados de incidencias semanales por el Insti-

tuto Nacional de Salud de Colombia. Los modelos propuestos fueron Regresión Kernel Ridge y Procesos Gaussianos, para el pronóstico del número de casos de dengue y chikunguña para una y cuatro semanas. Se evaluaron los modelos usando validación cruzada de origen rodante y métricas como el porcentaje medio de errores absolutos (MAPE), error medio absoluto (MAE), R-cuadrado y los porcentajes de varianza explicada por cada modelo. Siendo la regresión Kernel Ridge superior en la tarea de pronóstico.

- Iqbal y Islam (2019) para casos de dengue en India, evalúan las técnicas de k-Vecinos más próximos (kNN), Máquina de Soporte Vectorial (SVM), ANN, NB, DT, regresión logística y LogitBoost. Las métricas usadas para evaluar el desempeño fueron la precisión, sensibilidad, especificidad y TMC. Donde en conclusión, el mejor modelo fue el modelo de ensamble LogitBoost, pues este reportó las mejores métricas.
- Sarma y cols. (2020) comparan las técnicas de DT y Random Forest (RF) en datos de casos de dengue de Bangladesh. En este caso, el modelo basado en árboles de decisión fue el que mejor rendimiento predictivo arrojó. Este estudio comparte similitudes con el presente proyecto, ya que los autores desarrollan el modelo para clasificar 3 tipos de fiebre: fiebre normal, fiebre normal asociada a dengue y fiebre extrema asociada a dengue.
- Ho y cols. (2020) desarrollan un modelo bajo un estudio de casos y controles, en donde comparar DT, RL (Regresión logística) y una DNN (red neuronal profunda) en un proceso iterativo de CV. La DNN presentó los mejores resultados. Adicionalmente, realizan un análisis de asociaciones a partir de los Odds Ratios ajustados en el modelo de LR.
- Mohd Salim y cols. (2021) comparan 4 técnicas de machine learing que fueron, DT, ANN (Multilayer Perceptron), SVM y redes bayesianas (Tree Augmented Naïve Bayes). Donde la SVM (kernel lineal) presentó el mejor rendimiento en cuanto a predicciones.
- Hoyos et al. Hoyos y cols. (2021) realizan una revisión sistemática de la literatura sobre cuales han sido aquellas técnicas más frecuentes en la predicción de la enfermedad del dengue. Siendo la regresión logística la técnica de modelización más frecuente.
- M y cols. (2022) aplican la técnica de ensamble Stacking, en comparación con otros algoritmos para la clasificación de dengue en Andrapradesh, India. En conclusión, el modelo de ensamble mejora las precisiones en las predicciones realizadas por los demás algoritmos por separado.

2. Planteamiento del problema

Como ya se detalló apoyado en la evidencia científica y médica, el dengue representa un problema para la salud pública en países de regiones tropicales y subtropicales. A demás de ser la enfermedad vírica transmitida por mosquitos de más rápida propagación (World Health Organization, 2012). También la OMS, explica en "Global strategy for dengue prevention and control 2012-2020"que la morbilidad del dengue se puede reducir si se aplica una mejor predicción y detección de brotes de dengue a partir de la vigilancia epidemiológica.

Para esto, se debe tener en cuenta un concepto clave para el desarrollo de este trabajo: la clasificación revisada del dengue, propuesta por la OMS en el año 2009 (World Health Organization, 2009). En esta se establecen dos formas de la enfermedad, dengue y dengue grave.

El grupo de pacientes con dengue no grave se ha dividido en dos subgrupos: con y sin signos de alarma, cada clase se define por determinadas manifestaciones clínicas en los pacientes. Estas se muestras a detalle en la 2.1.

2.1. Pregunta Problema

Como consecuencia a la gravedad de la enfermedad y todos los aspectos socio económicos que conlleva el tratamiento de esta, sin mencionar la alta incidencia que tiene, es importante el desarrollar herramientas que permitan diagnósticos oportunos, esto permitiría el reducir el coste humano y monetario de los tratamientos, además de todas las vidas que se podrían salvar.

Teniendo en cuenta el desarrollo a diario de algoritmos y técnicas de aprendizaje automático ¿es posible desarrollar una herramienta de *machine learning* que permita el clasificar tipos de dengue a partir de los síntomas propios del paciente de manera oportuna en pacientes colombianos?

2.2. Justificación del proyecto

Teniendo en cuenta lo mencionado en la sección anterior, la prevención y tratamiento adecuado del evento en salud en cuestión permiten reducir los esfuerzos humanos, monetarios y sociales que este conlleva.

Además que la inserción de herramientas tecnológicas, apoyadas sobre fundamentos matemáticos y estadísticos, mejoraría la calidad del servicio prestado, promoviendo un avance tremendo en términos de investigación y desarrollo en el marco de la salud nacional.

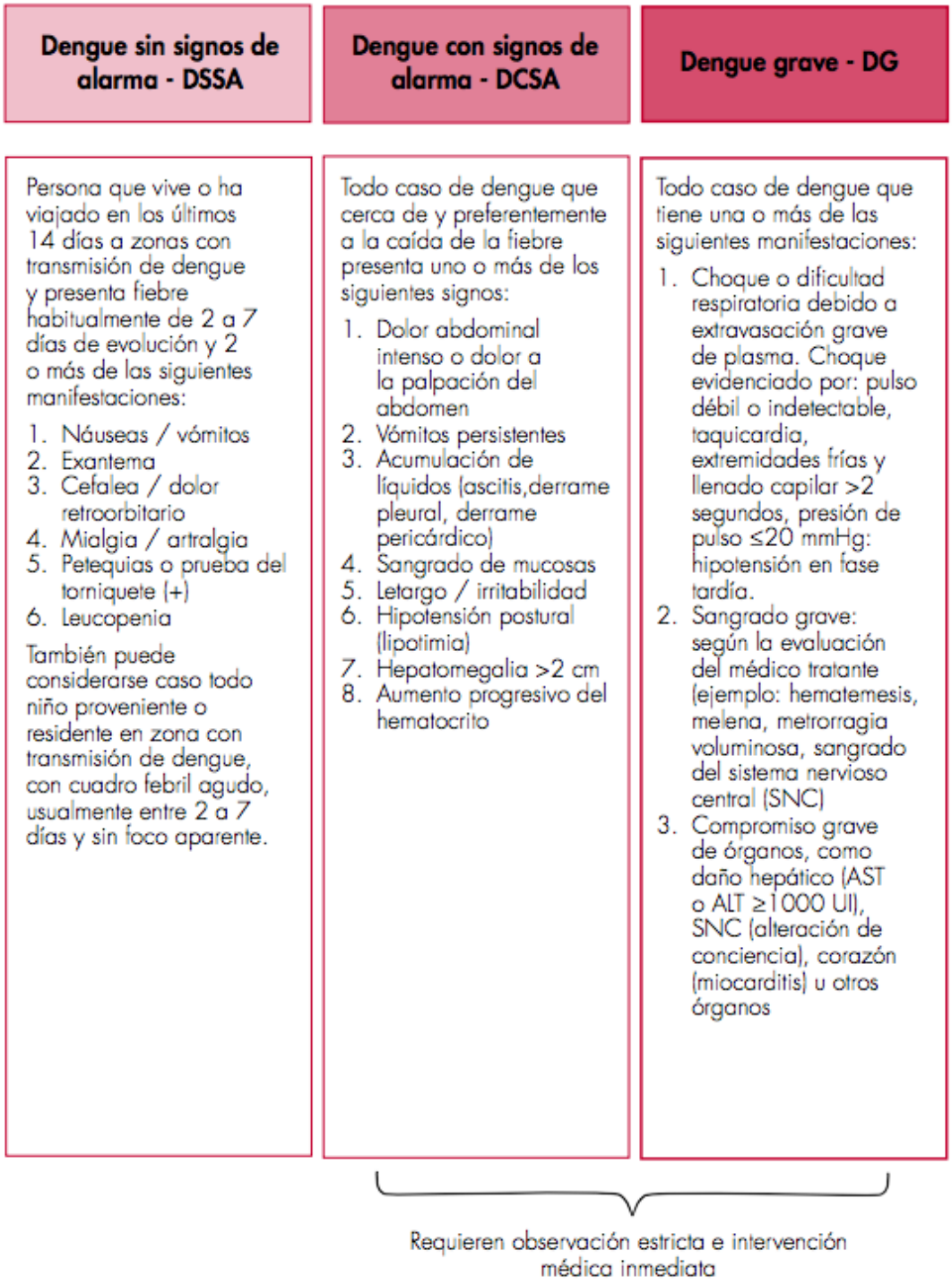

Figura 2.1: Clasificación Revisada de Dengue. Fuente: OMS/OPS World Health Organization (2009)

En la ciudad de Bucaramanga los conocimientos de la comunidad respecto al dengue son pobres, las prácticas en cuanto a prevención no son adecuadas (Cáceres-Manrique y cols., 2009); por tanto, una herramienta novedosa como un modelo de pronóstico temprano facilitaría a la comunidad el conocer a más detalle la enfermedad. Esto generaría que los afectados accedan prontamente a los servicios de salud.

Villar y cols. (2015) menciona que el comprender mejor los factores que determinan la expresión de la enfermedad permite un mejor control y gestión de esta misma. Una herramienta basada en términos como **machine learning, deep learning** ofrecería esta identificación de factores, que conlleva al poder atacar de una forma más clara el desarrollo y distribución de la enfermedad.

2.3. Objetivos

2.3.1. Objetivo General

Desarrollar un clasificador de diagnóstico temprano de nivel de severidad de dengue a partir del ensamble de algoritmos y modelos de *Machine Learning*, que permita la identificación de factores de riesgos asociados a las clasificación realizada.

2.3.2. Objetivos Específicos

- Desarrollar una etapa de preprocesamiento aplicando técnicas efectivas con el fin de obtener un conjunto de datos limpio como insumo para los modelos.
- Identificar y describir patrones de la enfermedad en los pacientes incluidos en el seguimiento.
- Entrenar de manera independiente 3 modelos de ML: Regresión Logística, *Support Vectorial Machine* y *Random Forest*, para la clasificación de tipos de dengue de acuerdo al nivel de severidad, junto a un modelo ensamblado de estos tres últimos mediante un meta-clasificador tipo *XGBoost*.
- Validar y seleccionar el modelo ML con la mayor potencia de predicción mediante a criterios apropiados.
- Identificar posibles factores de riesgo asociados al nivel de severidad sobre las clases predichas

3. Marco Teórico

A continuación, se presenta la teoría que hay detrás de los modelos y técnicas que se aplicarán en el presente trabajo, que como ya se ha mencionado tiene como propósito el identificar un buen modelo para clasificar los tipos de dengue.

3.1. *Machine Learning*

El *Machine Learning* (ML) o aprendizaje supervisado consiste en la programación de ordenadores para así optimizar un criterio de rendimiento sobre datos de entrenamiento o datos de experiencias pasadas (Alpaydin, 2014). En general, se tiene un modelo definido con sus respectivos parámetros; el ML se basa en la ejecución de un programa que optimice estos parámetros, este modelo puede ser de tipo predictivo o descriptivo que permite realizar inferencia.

Autores como Clark (2013) describen el ML (Aprendizaje automático, o aprendizaje estadístico) como una forma de estadística, que toma técnicas conocidas, aunque con un enfoque distinto al de la práctica analítica tradicional. Clark expone que la noción clave del ML es que usa patrones flexibles y automáticos para detectar patrones dentro de los datos, donde el principal enfoque es hacer predicciones sobre datos en el futuro.

Del aprendizaje automático usualmente se divide en dos tipos de aprendizaje: supervisado y no supervisado.

3.1.1. Aprendizaje Supervisado

En el enfoque de aprendizaje supervisado o predictivo, el objetivo es aprender correspondencia entre las entradas x y las salidas y, dado un conjunto de pares etiquetados de entrada-salida $D = f(x_i, y_i)g_{i=1}^{N}$, donde D es el conjunto de entrenamiento y N el número de muestras de entrenamiento. Las entradas de entrenamiento x_i es un vector D *dimensional* con las características, atributos o covariables de cada individuo de D, aunque en algunos casos x_i puede tener estructuras más complejas, como imágenes, series de tiempo, dibujos, etc. (Murphy, 2013).

Una de las principales tareas del aprendizaje supervisado es la **Clasificación**, en donde se le pide al programa que especifique a cual de las k posibles categorías pertenece la entrada. El algoritmo de aprendizaje produce una función $f : \mathbf{R}^n \; ! \; f1, ..., kg$, cuando $y = f(x)$, el modelo asigna a una entrada x una categoría identificada por un código numérico y (Goodfellow y cols., 2016).

3.1.2. Aprendizaje no Supervisado

El enfoque de aprendizaje no supervisado o descriptivo cuenta únicamente con entradas dadas por $D = \{x_i\}_{i=1}^{N}$, con el objetivo de encontrar patrones de interés en los datos. Por eso también es conocido "descubrimiento de conocimiento" (Murphy, 2013).

3.2. Árboles de decisión

Como explica Loh (2011), estos árboles son usados para construir modelos predictivos. Donde estos modelos se obtienen a partir de particiones en el espacio de los datos, ajustando modelos simples de predicción con cada partición.

- Definición: Se tiene un conjunto de entrenamiento de n observaciones con una variable de clases Y que toma los valores o clases $1, 2, ..., k$ y p variables predictoras $X_1, ..., X_p$.

Las predicciones se dan debido a estas particiones en el espacio X en k conjuntos disjuntos, $A_1, A_2, ..., A_k$ tal que el valor predicho de Y es j si X pertenece a A_j, para $j = 1, 2, ..., k$.

La construcción de estos conjuntos o espacios predictores A_j dependerá de la naturaleza de la variable X_j y si esta es cuantitativa o cualitativa.

- X_j cuantitativa

$$A_1(j, t_i) = \{X | X_j < j\} \quad y \quad A_2(j, t_i) = \{X | X_j \quad t\} \tag{3.1}$$

Siendo t un número real como punto de corte, que genera estas dos regiones.

- X_j cualitativa

$$A_1(j, c_i) = \{X | X_j = c_j\} \quad y \quad A_2(j, c_i) = \{X | X_j \not= c_i\} \tag{3.2}$$

Siendo c_i una de las categorías $i = 1, 2, ...$ propias de X_j.

La partición del espacio X a partir de estos subespacios disyuntos puede ser representado a través de árboles de decisión binarios. Los puntos donde se realizan esta particiones se denominan nodos.

Un modelo clásico que implementa árboles de decisión es CART (Classification and Regression Trees), con la diferencia que los árboles clásicos asignan una decisión a cada uno de los nodos del árbol. Mientras que CART asigna una puntuación a estos nodos, lo que permite una mejor interpretación del modelo.

Generalmente un árbol es insuficiente en la práctica, lo más común es aplicar modelos de conjunto, en donde la predicción final se define como la sumatoria de las predicciones de todos los árboles, tal que el modelo es:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F \tag{3.3}$$

en donde K es el número de árboles construidos, f_k una función en el espacio $\mathcal{F}$ y $\mathcal{F}$ el conjunto de todos los por CART posibles.

Una definición más formal para los árboles de decisión $f(x)$ viene dada por:

$$f_t(x) = w_{q(x)}, w \in R^T, \ q : R^d \rightarrow \{1, 2, ..., T\} \tag{3.4}$$

en donde w es el vector de puntuaciones dadas en las hojas del árbol y q una función que asigna cada observación o dato a su hoja correspondiente.

En estos modelos la función objetivo a optimizar se define como:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \omega(f_k) \tag{3.5}$$

en la que la función $l(.)$ es la función de pérdida de entrenamiento y $\omega(f_k)$ es la complejidad del árbol f_k.

3.2.1. *XGBoost*

Uno de los modelos más importantes para el desarrollo de este trabajo es el *XGBoost* (siglas de *Extreme Gradient Boosting"*). El término *"Gradient Boosting"* es usado por primera vez por Friedman (2001), donde se concluye que la potenciación del gradiente para árboles de decisión genera procedimiento robustos y más interpretables tanto en tareas de regresión como clasificación.

A continuación se presenta la teoria detrás del "Gradient Boostingz el "XGBoost".

- *Tree Boosting:* Un concepto clave en el desarrollo del modelo es el impulso de árboles, proceso en el cual se debe optimizar la función objetivo expuesta en la 3.5.

  La estrategia de entrenamiento para la obtención de parámetros de todos los árboles a la vez consta de un proceso tedioso. En este caso se aplica una estrategia aditiva (se agrega de a un nuevo árbol a la vez). Las predicciones del paso t se definen como $\hat{y}^{(t)}$, el entrenamiento se aplica tal que:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{3.6}$$

Teniendo en cuenta lo anterior, la función objetivo añadiendo el árbol del paso t que la optimiza quedaría como:

$$\begin{aligned} \text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t\ 1)} + f_t(x_i)) + \omega(f_t) + \text{constante} \end{aligned} \tag{3.7}$$

Por lo general, suele usarse el Error Cuadrático Medio MSE como función de pérdida. Aunque naturalmente no es la única opción. El MSE produce una forma agradable de la función objetivo, para otras opciones de funciones de pérdida que no generan estas formas, se toma la **Expansión de Taylor de la función de pérdida hasta el segundo orden**, obteniendo:

$$\text{obj}^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t\ 1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \omega(f_t) + \text{constante} \tag{3.8}$$

donde g_i y h_i son

$$\begin{aligned} g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t\ 1)}) \\ h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t\ 1)}) \end{aligned} \tag{3.9}$$

- Complejidad del Modelo: En el algoritmo de *XGBoost* la complejidad del modelo se define como

$$\omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \tag{3.10}$$

en donde γ es la reducción de pérdida mínima necesaria para realizar una partición adicional en un nodo de la hoja del árbol. T es el número de hojas y λ el parámetro de regularización para el modelo usando la norma L2.

3.2.2. *Random Forest*

Este modelo está basado en árboles de decisión, que consta en el ajuste o .ensamble"de múltiples árboles de decisión, para lograr esto, el algoritmo toma el concepto de **bagging** o muestreo vía **Bootstrap**.

- **Bagging**

Dado un conjunto de entrenamiento $X = x_1, ..., x_n$ con variable de respuesta $Y = y_1, ..., y_n$, al cual se le aplica un re-muestreo Boostrap, o muestras aleatorias con reemplazo, de donde se obtienen B muestras.

A cada una de estas B muestras, se entrena un árbol de decisión f_b, en este caso no se tiene presente algún tipo de poda respecto a sus hiperparámetros (como número de nodos, profundidad máxima, número mínimo de observaciones en el nodo, entre otros). Este número de muestras B se toma como hiperparámetro del algoritmo. Los errores de entrenamiento y prueba se suelen estabilizar dada cierta cantidad de árboles.

Para tareas de regresión, luego del entrenamiento de estos árboles se promedian las predicciones encontradas.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b \ (x) \tag{3.11}$$

En el caso de labores de clasificación la predicción será la clase predicha más frecuente.

Se diferencia del método **Bagging** clásico, ya que al realizar una división en los árboles de decisión, o en los nodos, se considera una muestra aleatoria de tamaño m de las p predictoras, de esta muestra aleatoria se toma una variable, que será la variable de la cual se realizará la partición. Generalmente se toma como tamaño de muestra $m \quad \overline{p}$.

Esto último garantiza que este conjunto de árboles no estén altamente correlacionados entre sí, lo que permite una reducción significativa de la varianza. Además que se evita que predictores muy fuertes o con mayores importancias se ubiquen cerca al nodo raíz de los árboles.

3.3. Máquinas de Soporte Vectorial

La teoría de las Máquinas de Soporte Vectorial (SVM de sus siglas en inglés **Support Vector Machines**) se basa en la idea de minimización de riesgo estructural (SRM) (Vapnik, 1995). Además de que han demostrado tener mayores desempeños que algunas herramientas de aprendizaje tradicional como lo son las redes neuronales como menciona Burges (1998).

El SVM clasifica los datos utilizando un hiperplano que separa las diferentes clases. Si los datos son linealmente separables, el hiperplano se encuentra de manera óptima en el medio de la brecha más amplia posible entre las dos clases. Si los datos no son linealmente separables, se puede utilizar un kernel para proyectar los datos en un espacio dimensional superior, donde los datos sí pueden ser separados linealmente.

3.3.1. SVC

Las *Support Vectorial Classi er* (SVC, por sus siglas en inglés) son una variante de las SVM para la clasificación de datos. El objetivo de las SVC es encontrar el hiperplano que mejor separa los datos de diferentes clases en un espacio de características de alta dimensión.

En una clasificación de dos clases, una SVC encuentra el hiperplano que separa las dos clases de forma que la distancia entre el hiperplano y los puntos más cercanos de ambas clases sea máxima. La distancia entre el hiperplano y los puntos más cercanos se conoce como la margen y es una medida de la capacidad del modelo para generalizar a nuevos datos.

En general, las SVC funcionan tal que:

Dado un conjunto de datos de entrenamiento $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, donde x_i es un vector de características y y_i es la etiqueta de clase correspondiente, que puede tomar los valores tal que y_i *2 f* $1, 1g^n$ para problemas de clasificación binaria y valores enteros para problemas de clasificación multiclase.

La SVC tiene como propósito encontrar el hiperplano de separación

$$w^T x + b = 0 \tag{3.12}$$

que maximice el margen, donde w es el vector de pesos y b es el sesgo. El margen se define como la distancia perpendicular del hiperplano a los puntos de datos más cercanos de cada clase. Los puntos de datos más cercanos se conocen como vectores de soporte.

Para encontrar el hiperplano de separación, el SVC resuelve el siguiente problema de optimización:

$$\begin{array}{c} \text{mín}_{w,b,\xi} \ \frac{1}{2}jjwjj^2 + C\sum_{i=1}^{n}\xi_i \\ sujeto\ a\ y_i(w^T x_i + b) \quad 1 \quad \xi_i, \\ \xi_i \quad 0 \end{array} \tag{3.13}$$

donde C es un parámetro de regularización que controla la importancia que se le da a la maximización del margen y la minimización de los errores de clasificación. El parámetro ξ_i es una variable de holgura que permite que algunos puntos de datos estén dentro del margen o en el lado incorrecto del hiperplano.

La función de pérdida *hinge* utilizada en el SVC está definida como:

$$L_{hinge}(y_i, f(x_i)) = \text{máx}(0, 1 \quad y_i f(x_i)) \tag{3.14}$$

donde

$$f(x_i) = sign(w^T x_i + b) = \begin{cases} 1 & y_i = 1 \\ 1 & y_i = \quad 1 \end{cases} \tag{3.15}$$

es la función de decisión que asigna una etiqueta de clase a un punto de datos.

Para datos no linealmente separables, el SVC utiliza el *kernel trick* para transformar los datos de entrada en un espacio de mayor dimensión en el que los datos son linealmente separables, esta transformación se muestra con la notación $\phi(x_i)$. Para estos casos la optimización pasa a ser un problema de programación cuadrática, que se

resuelve con la construcción de un Lagrangiano pasando de un primal a un dual. En donde finalmente la función de decisión concluye en;

$$f(x_i) = sign(w^T \phi(x_i) + b) = sign(\sum_{i2SV} y_i \alpha_i K(x_i, x) + b) \tag{3.16}$$

donde $K(x_i, x) = \phi(x_i)\ \ \phi(x)$ es el kernel, una función de similitud que calcula el producto escalar de los vectores transformados en el espacio de características de mayor dimensión. α_i son los coeficientes duales con limite superior en C. Smola y Schölkopf (2004) Betancourt (2005)

El kernel más comúnmente utilizado es el kernel radial, que está definido como:

$$K(x_i, x_j) = \exp(\quad \gamma jj x_i \quad\ x_j jj^2) \tag{3.17}$$

donde γ es un parámetro que controla la forma de la función de kernel.

3.4. Regresión Logística

La Regresión Logística (RL) (Hosmer, 2013), se aplica para predecir probabilidades en experimentos de naturaleza dicotómicos o de respuesta binaria. Pues, las entradas pasan por un proceso de transformación que determina una probabilidad entre 0 y 1.

La variable Y que se modela se define como:

$$y = \begin{cases} 1, \ éxito \\ 0, \ en\ otro\ caso \end{cases} \tag{3.18}$$

Dada una colección de p variables independientes denotadas por el vector $x^0 = (x_1, x_2, ..., x_p)$. La probabilidad condicional de que la respuesta sea exitosa es denotada por $P(Y = 1 jx) = \pi(x)$.

El modelo usa la función logística (o *logit*) para transformar las entradas x^0 en la probabilidad mencionada. La función logit del modelo de regresión logística esta dado por:

$$g(x) = \ln\left(\frac{\pi(x)}{1 \quad \pi(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p \tag{3.19}$$

Y el modelo viene dado finalmente por,

$$\pi(x) = \frac{e^{g(x)}}{1 + e^{g(x)}} = \frac{1}{1 + e^{\ g(x)}} \tag{3.20}$$

Viendo lo anterior como problema de optimización, la regresión logística con termino de regularización $r(\beta)$ minimiza la siguiente función de costo:

$$\min_{\beta} C \sum_{i=1}^{n} (\quad y_i \log(\hat{\pi}(x)) \quad (1 \quad y_i) \log(1 \quad \hat{\pi}(x))) + r(\beta) \tag{3.21}$$

En este trabajo se usarán: penalización tipo ℓ_1, que toma como término de regularización: $\|\beta\|_1 = |\beta_1| + |\beta_2| + \quad + |\beta_p|$. Tipo ℓ_2, con $r(\beta) = \frac{1}{2}\|\beta\|_2^2 = \frac{1}{2}\beta^T\beta$. Y, por último, penalización *ElasticNet*, con $r(\beta) = \frac{1}{2}\rho\beta^T\beta + \rho\|\beta\|_1$ donde ρ controla la fuerza de las regularizaciones ℓ_1 y ℓ_2.

3.5. Modelos de Ensamble

En el aprendizaje automático, los modelos de ensamble son técnicas que combinan varios modelos individuales para mejorar la precisión y la estabilidad de las predicciones. Los métodos de ensamble se utilizan comúnmente en problemas de clasificación y regresión, donde la combinación de varios modelos puede reducir el sesgo y la varianza y mejorar la generalización (Hastie y cols., 2009).

Existen múltiples tipos de modelos de ensamble, los más usuales son el *bagging* y el *boosting*. Aunque para este trabajo será también de especial interés la técnica de *Stacking*.

- Bagging: Es un método de ensamble que se utiliza para reducir la varianza de un modelo mediante la combinación de múltiples modelos entrenados de forma independiente. Cada modelo se entrena en un subconjunto aleatorio del conjunto de datos de entrenamiento, y la predicción final se realiza mediante la agregación de las predicciones individuales. Ejemplos de algoritmos de bagging incluyen Random Forest y Extra Trees (Bishop, 2007) (Breiman, 1996a).
- Boosting: Es un método de ensamble que se utiliza para mejorar la precisión de un modelo mediante la combinación de múltiples modelos débiles. En el boosting, cada modelo se entrena de forma secuencial en los errores del modelo anterior, y se da más peso a las instancias que se clasificaron incorrectamente en el modelo anterior. Ejemplos de algoritmos de boosting incluyen AdaBoost, Gradient Boosting y XGBoost (Friedman, 2001).
- Stacking: Es un método de ensamble que se utiliza para combinar múltiples modelos de forma jerárquica. En el stacking, los modelos individuales se utilizan para generar predicciones, que luego se utilizan como características de entrada para un modelo de nivel superior. Este modelo de nivel superior combina las características de entrada y genera la predicción final. El stacking se utiliza comúnmente en competiciones de aprendizaje automático, donde los participantes combinan múltiples modelos para mejorar su rendimiento (Wolpert, 1992) (Breiman, 1996b).

  El proceso de Stacking se puede resumir en los siguientes pasos:

Se divide el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba. Se entrena un conjunto de modelos base utilizando el conjunto de entrenamiento. Se utiliza cada modelo base para generar una predicción para el conjunto de prueba. Se utiliza el conjunto de predicciones generadas por los modelos base como entrada para un modelo meta o de nivel superior. El modelo meta combina las predicciones generadas por los modelos base para generar una predicción final para el conjunto de prueba.

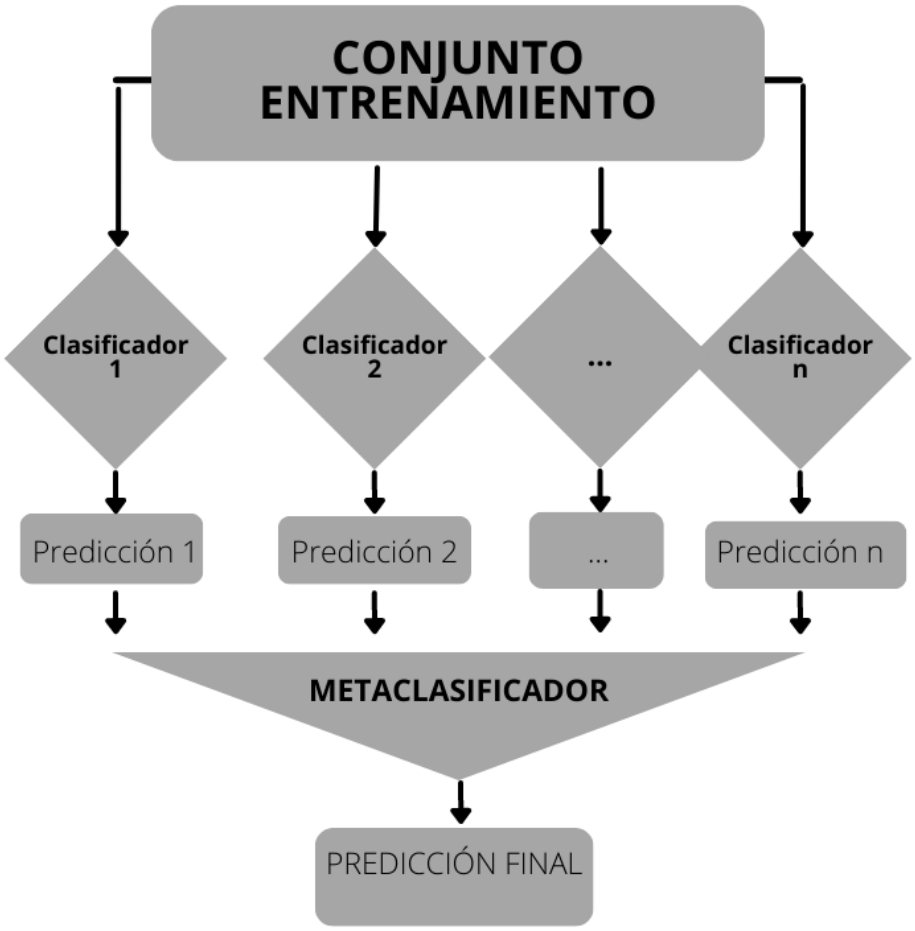

Figura 3.1: Proceso de Stacking. Elaboración Propia

Como se muestra en la Figura 3.1.

A continuación, se detallan algunas de las principales ventajas y desventajas del ensamble Stacking (Ting, 1999) (Dietterich, 2000) (Polikar, 2006) (Zhou, 2012):

Ventajas:

- Mayor precisión: El ensamble Stacking puede mejorar significativamente la preci-

	Predicción positiva	Predicción negativa
Clase positiva	True Positive (TP)	False Negative (FN)
Clase negativa	False Positive (FP)	True Negative (TN)

Tabla 3.1: *Matriz de Confusion general*

sión de la predicción en comparación con un solo modelo base. Esto se debe a que el modelo de nivel superior puede aprender a corregir los errores de los modelos base.

- Flexibilidad: El Stacking es una técnica muy flexible que puede integrar diferentes tipos de modelos base, lo que lo hace adecuado para una amplia gama de problemas de aprendizaje automático.
- Adaptabilidad: El Stacking se puede adaptar a diferentes conjuntos de datos mediante la selección de modelos base adecuados y la elección de diferentes técnicas de agregación.

Desventajas:

- Mayor complejidad: El ensamble Stacking implica la construcción de múltiples modelos base y un modelo de nivel superior, lo que aumenta la complejidad del modelo y su entrenamiento.
- Mayor costo computacional: El entrenamiento y la evaluación del modelo de Stacking pueden requerir más recursos computacionales en comparación con un solo modelo base.
- Mayor riesgo de sobreajuste: El Stacking puede ser propenso al sobreajuste si los modelos base no se seleccionan adecuadamente o si la técnica de agregación es inadecuada.

3.6. Medidas de Evaluación

En esta sección se muestran las métricas de evaluación que se tendrán en cuenta en la metodología para este trabajo. Al tratarse de una tarea de clasificación, estas medidas se basan en la matriz de confusión.

3.6.1. Matriz de confusión

La matriz de confusión es una herramienta útil para evaluar el rendimiento de un modelo de clasificación. La matriz de confusión presenta la relación entre las predicciones del modelo y las clases verdaderas. Esta matriz es una tabla que muestra el número de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) producidos por un clasificador (Geron, 2017).

En la 3.1 se muestra una generalización de las matrices de confusión y las medidas que se incluyen en esta.

3.6.2. *Accuracy*

Accuracy es una métrica que mide la proporción de predicciones correctas del modelo. Se calcula como el número de predicciones correctas ($TP + TN$) dividido por el número total de predicciones ($TP + TN + FP + FN$).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.22}$$

La precisión es útil cuando las clases están equilibradas en el conjunto de datos, es decir, cuando el número de observaciones de cada clase es similar (Raschka y cols., 2020).

3.6.3. Sensibilidad

La sensibilidad, también conocida como tasa de verdaderos positivos, es una medida de evaluación utilizada en la evaluación de modelos de clasificación binaria. Se refiere a la capacidad de un modelo para identificar correctamente las instancias positivas de la clase objetivo.

La sensibilidad se define como la proporción de verdaderos positivos (TP) sobre la suma de verdaderos positivos y falsos negativos (FN). En otras palabras, la sensibilidad mide la proporción de instancias positivas que fueron correctamente clasificadas por el modelo. La fórmula para la sensibilidad es la siguiente:

$$Sensibilidad = \frac{TP}{TP + FN} \tag{3.23}$$

3.6.4. Especificidad

La especificidad, también conocida como tasa de verdaderos negativos, es una medida de evaluación utilizada en la evaluación de modelos de clasificación binaria. Se refiere a la capacidad de un modelo para identificar correctamente las instancias negativas de la clase objetivo.

La especificidad se define como la proporción de verdaderos negativos (TN) sobre la suma de verdaderos negativos y falsos positivos (FP). En otras palabras, la especificidad mide la proporción de instancias negativas que fueron correctamente clasificadas por el modelo. La fórmula para la especificidad es la siguiente:

$$Especificidad = \frac{TN}{TN + FP} \tag{3.24}$$

3.6.5. F1-Score

El F1-score se define como la media armónica de la precisión y la exhaustividad. La precisión es la proporción de verdaderos positivos sobre la suma de verdaderos positivos y falsos positivos, mientras que la exhaustividad es la proporción de verdaderos positivos sobre la suma de verdaderos positivos y falsos negativos. La fórmula para el F1-Score es:

$$Accuracy = \frac{TP}{TP + FP} \tag{3.25}$$

3.6.6. Curvas ROC

La curva ROC (Receiver Operating Characteristic) es una representación gráfica que muestra el rendimiento de un modelo de clasificación a través de una serie de umbrales de decisión. La curva ROC se crea trazando la tasa de verdaderos positivos (TPR) en el eje y y la tasa de falsos positivos (FPR) en el eje x para cada umbral de decisión.

La TPR se define como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos ($TP/(TP + FN)$). La FPR se define como el número de falsos positivos dividido por la suma de falsos positivos y verdaderos negativos ($FP/(FP + TN)$).

La curva ROC es útil para evaluar el rendimiento de un modelo de clasificación en diferentes umbrales de decisión y para comparar diferentes modelos de clasificación (Geron, 2017) (Raschka y cols., 2020).

La 3.2 muestra un ejemplo de Curvas de Información ROC, en este caso se muestra un muy buen ajuste, pues el criterio de AUC toma un valor alto.

3.6.7. Áreas bajo la curva

El área bajo la curva (AUC) es una medida del rendimiento de un modelo de clasificación que se utiliza para evaluar la curva ROC. El AUC mide la capacidad del modelo de distinguir entre clases positivas y negativas. El AUC varía de 0 a 1, donde 1 indica un modelo perfecto y 0.5 indica un modelo que no tiene capacidad de clasificación.

El AUC se puede interpretar como la probabilidad de que el modelo clasifique una instancia positiva aleatoria más alta que una instancia negativa aleatoria (Geron, 2017).

Este valor es calculado de la siguiente forma:

$$\int_0^1 \frac{TPR}{FPR} dFPR \tag{3.26}$$

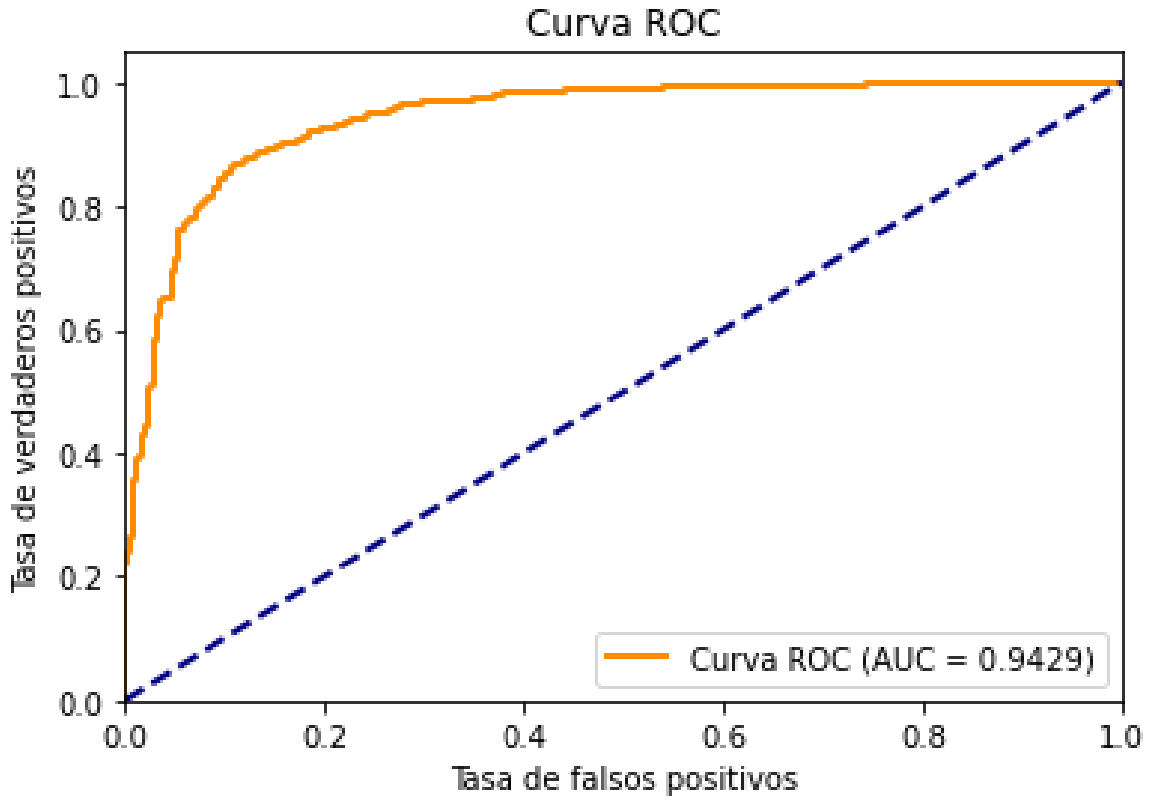

Figura 3.2: Ejemplo de Curvas ROC. Elaboración Propia

donde TPR es la tasa de verdaderos positivos y FPR es la tasa de falsos positivos (Raschka y cols., 2020).

4. Metodología

En esta sección se detalla y describe la metodología de trabajo aplicada en el presente trabajo.

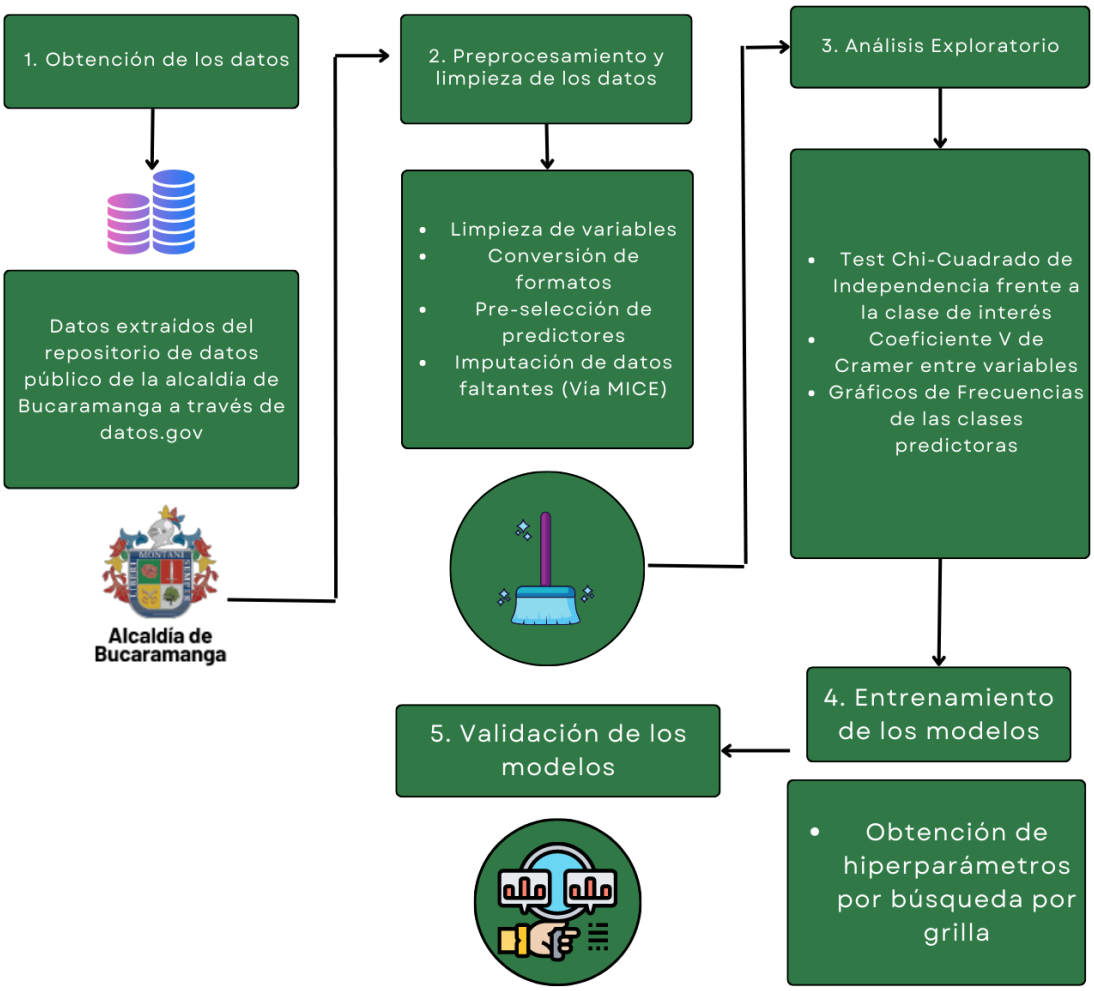

Figura 4.1: Metodología de Trabajo. Elaboración Propia

La Figura 4.1 esquematiza la metodología aplicada.

4.1. Obtención de los datos

El conjunto de datos que servirá como principal insumo para la construcción de los modelos mencionados, pertenecen a la Secretaría de Salud y Ambiente Municipal de la ciudad de Bucaramanga, Santander; del proyecto denominado "Datos abiertos observatorio digital municipal de Bucaramanga".

4.2. Pre-procesamiento y limpieza de datos

En esta etapa se busca realizar una correcta adecuación de los datos obtenidos, pues como se mención en la sección anterior se tienen 117 variables, como mediciones de tipo socio-demográfico, que para efectos del presente trabajo no serán tenidas en cuenta en su totalidad.

Esta limpieza se realizó mediante el lenguaje de programación 'R' (R Core Team, 2021), a través de las librerías *dplyr* (2021) y *DataExplorer* (2021). Estas se usarán en la limpieza de campos que no son útiles para la investigación. Muchas de estos campos son excluidos por su naturaleza, pues presentan errores de medición o codificación de la información.

Se aplicará una conversión en los formatos de las variables finalmente seleccionadas, que en este caso corresponde a su mayoría de tipo categórico, pues estos campos corresponden a síntomas indexados en valores de 1: Presentó el síntoma, 2: No presentó el síntoma.

Tras esto, se identifican valores faltantes de estos campos seleccionados previamente. Pues campos con porcentajes de pérdida superiores al 70 % no serán tenidos en cuenta en la investigación. Los campos que tengan porcentajes de pérdidas inferiores al 70 % pasarán por un proceso de imputación, en este caso serán imputados mediante la librería *mice* (2021).

Finalmente, al ser la mayoría de variables de tipo categórico, se presentan síntomas con des-balances en las clases presentadas, en casos severos de este tipo se procede a eliminar el campo des-balanceado.

4.3. Análisis Exploratorio y descriptivo

La exploración de los datos en este caso se tomará por las frecuencias mostradas en los síntomas incluidos en el conjunto de datos final, esto por la naturaleza categórica de la información.

Uno de los elementos más importantes de esta fase será el análisis de correlaciones existentes entre los predictores. Al ser estos de tipo factor, el test de independencia Chi-Cuadrado de Pearson será la herramienta aplicada. La medida del efecto de estas correlaciones se obtendrá a partir del coeficiente V de Cramer. Esto puede ser visto, como criterio de exclusión de factores que puedan aportar menos al modelo.

4.4. Entrenamiento de modelos

Las fases de construcción, entrenamiento y validación de los modelos se realizará mediante la herramienta *Python* (2020). El módulo de mayor importancia en esta tarea es *scikit-learn* (2011).

El primer paso de esta fase es la: construcción de los conjuntos de entrenamiento y validación.

Se cuenta con un volumen importante de datos, lo que permitirá al modelo un entrenamiento adecuado. Para esto se realiza una división del conjunto original en dos conjuntos; entrenamiento y validación. Las proporciones elegidas para estos son del 70 y 30 % respectivamente, del conjunto inicial, el muestreo se realizará de forma aleatoria, con el propósito de evitar sesgos de selección.

Para la optimización en la búsqueda de hiperparámetros de los modelos, se aplicará búsqueda por grilla aplicando validación cruzada. Los Scores de entrenamiento o de bondad de ajuste determinarán los hiperparámetros óptimos para cada modelo.

Con estos conjuntos construidos, se aplica la metodología *Stacking* propuesta por Wolpert (1992), en la que se ensamblan y combinan múltiples algoritmos para producir un modelo general. Estas combinaciones de modelos se realizan en múltiples etapas, en donde, las predicciones dadas por los algoritmos del nivel anterior (o nivel base) son entradas para los algoritmos de los niveles posteriores.

Esto anterior se realiza pasa un proceso iterativo por pliegues, en donde el entrenamiento de los modelos del primer nivel se realiza paso a paso.

Se obtendrá curvas de aprendizaje para cada uno de los modelos construidos, esto va a permitir la identificación de posibles sobreajustes en los modelos. Los modelos óptimos entrarán a la fase de validación.

Los resultados de esta fase serán expuestos de forma gráfica, de este modo será más amigable entender el proceso, la contracción de estas gráficas se realizará con el módulo *matplotlib* (2007), también de *Python*.

4.5. Validación de los Modelos

La validación se realizará sobre el conjunto de prueba (30 %). La evaluación de los modelos candidatos se realizará por medio del score 'AUC' (Área bajo la curva) obtenido de las curvas de información ROC, aunque se medirán también criterios de precisión, especificidad y sensibilidad de los modelos.

De la fase de entrenamiento se obtendrán 4 modelos candidatos, los criterios de AUC de estos serán comparados, para así encontrar el modelo más óptimo en la clasificación de los niveles de severidad en dengue para los pacientes del conjunto de prueba.

5. Resultados

5.1. Conjunto de Datos

El nombre dado al conjunto de datos es "13. Dengue, Dengue grave y mortalidad por dengue municipio de Bucaramanga de enero del 2015 a junio 2022 - semana 24", en el que cada fila representa un caso de dengue en el municipio.

Como se describe en el título del conjunto de datos, este último contiene casos identificados de dengue entre enero de 2015 y junio de 2022. Hasta la fecha la oficina no ha realizado nuevas actualizaciones en los datos.

El conjunto original presenta 10159 observaciones en 115 columnas. Estas como respuesta al seguimiento realizado a los pacientes, desagregando por sexo, domicilio, régimen de salud e instituciones que fueron notificando los casos.

5.2. Pre-procesamiento y limpieza de los datos

La variable de respuesta y de mayor interés de este proyecto tiene el nombre de 'clasfinal', que contiene las 3 clases presentadas por la OMS World Health Organization (2009). Las frecuencias de estas clases en el conjunto de datos original se muestran en la gráfica de barras 5.1. De esta gráfica se evidencia un fuerte des-balance respecto a la clase 'Dengue Grave', que representa el 0,26 % de los datos totales, por ese motivo, para evitar des-balances en las clasificaciones realizadas por los modelos se decide eliminar los 26 registros de esta clase.

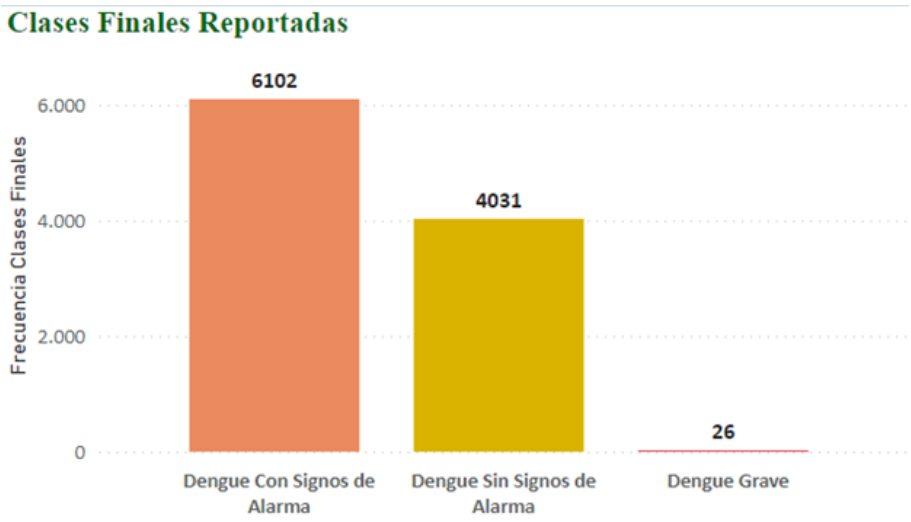

Figura 5.1: Gráfico de barras clases finales reportadas

Síntoma	Coeficiente V	OR estimado	χ^2 Calculado	Valor-P
Fiebre	0.7591	0	5836.1	0.0000
Cefalea	0.6326	0.0320	4052.2	0.0000
Dolor Retroocular	0.0442	0.4421	1978.3	0.0000
Mialgias	0.6510	0.0283	4291.8	0.0000
Artralgias	0.6039	0.0341	3692.9	0.0000
Erupción Cutánea	0.5759	0.0294	3357.7	0.0000

Tabla 5.1: *Coe cientes V de Cramer mas fuertes*

Estas clases se han codificado por facilidad en la fase de resultados como Dengue Con Signos de Alarma: 1; Dengue Sin Signos de Alarma:0. Además, los síntomas incluidos en el *dataset* final tras el pre-procesamiento se han codificado como:

- 0: Ausencia del síntoma
- 1: Presencia del síntoma

5.3. Análisis Exploratorio

Analizando los gráficos de barras de los principales síntomas incluidos en el conjunto de datos final, se evidencia que síntomas como caída de plaquetas, diarrea, dolor abdominal y vomito, se presentaron en una menor frecuencia, esto se debe a que estos síntomas se presentan en casos más severos de dengue y no corresponden a síntomas comunes de la enfermedad.

Estos gráficos de barras pueden ser consultados en el siguiente panel de visualización desarrollado en Power BI: Panel Exploratorio.

Cómo se mencionó en el capitulo de Metodología, el análisis de correlaciones se presenta gráficamente a través del Coeficiente V Cramér (1946), donde:

- $V = 0$: las variables son independientes
- $0 < V <$ 0,3: asociación débil
- 0,3 $V <$ 0,5: asociación moderada
- V 0,5: asociación fuerte

La matriz de la gráfica 5.2 contiene en la última columna los coeficientes estimados respecto a la variable de respuesta. Donde es notorio que existen asociaciones fuertes con algunos de los síntomas registrados.

La tabla 5.1 muestra los 6 coeficientes más fuertes de la gráfica 5.2. Se evidencian que las asociaciones además de fuertes son significativas. Los OR estimados a partir de las tablas de contingencias entre los tipos de dengue y los síntomas, en este caso sugieren que estos últimos tienen mayor chance de presentarse en pacientes clasificados como pacientes de dengue SIN signos de alarma.

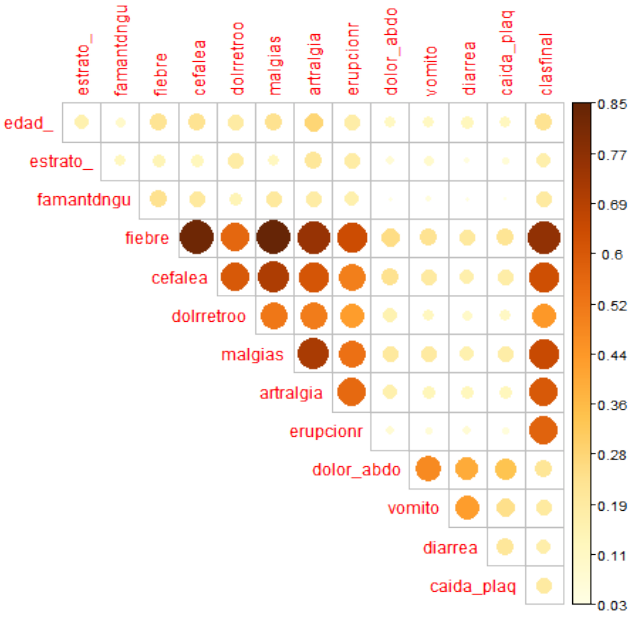

Figura 5.2: Matriz Coeficientes de Cramer. Elaboración Propia

5.3.1. Valores Faltantes

La gráfica 5.3 muestra los porcentajes de pérdida para los campos que presentan valores faltantes, se evidencias pérdidas severas en varios de los campos, así pues, campos con pérdidas superiores al 70 % son removidos del conjunto de datos, pues realizar una imputación dada esa condición puede generar problemas de sesgos para la clasificación.

Tras la limpieza de estos campos, y verificar los formatos de las demás variables que presentaron valores faltantes, se extraen columnas de fecha (por los objetivos de este trabajo no se tendrán en cuenta variables temporales), columnas que presentaban respuestas de entrada libre, pues la codificación de origen no era adecuada para el seguimiento de la ruta trazada en este proyecto y columnas que presentaban códigos de identificación del Departamento Administrativo Nacional de Estadística (DANE) debido a que todos los casos se dieron en el municipio de Bucaramanga y estos campos poseen campos repetidos que sobran en la construcción de los modelos.

Las variables que serán finalmente imputadas vía *mice*, son: $^{0}famantdngu^{0}$ (Antecedentes familiares de dengue en últimos 15 días) y $^{0}estrato\_^{0}$ (Estrato Socio-Económico).

La descripción del conjunto de datos final, tras el pre-procesamiento se muestra en la tabla A.1 disponible en la sección de Anexos.

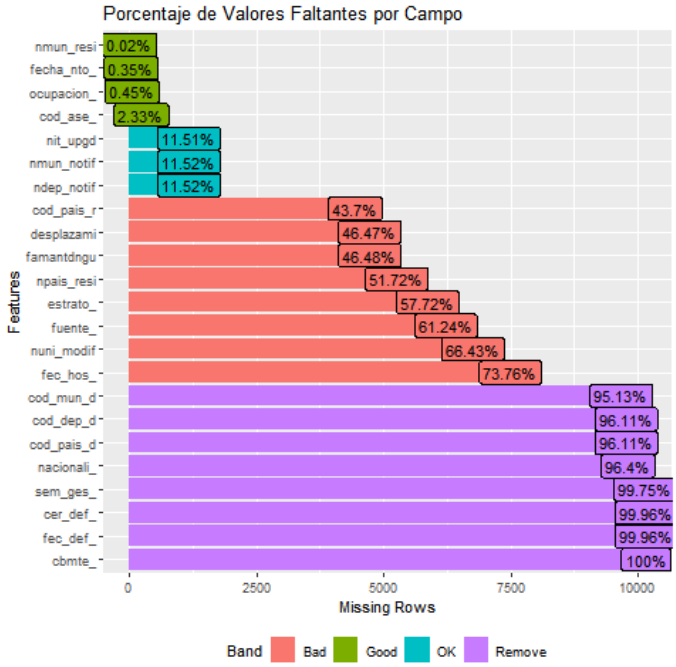

Figura 5.3: Porcentaje de Valores Faltantes. Elaboración Propia

5.4. Entrenamiento de los modelos

En la sección de la Metodología se mencionó que el entrenamiento de los modelos se realizará con una muestra aleatoria del 70 % del conjunto de datos (7093 registros).

El entrenamiento se lleva a cabo aplicando Validación Cruzada a la que se añade una búsqueda por grilla de hiperparámetros óptimos para cada uno de los modelos, esto con el fin de controlar posibles sobreajustes en los modelos.

5.4.1. Regresión Logística

En la sección del Marco Teórico se mencionan tres tipos de penalización ℓ_1, ℓ_2 y *ElasticNet*, la decisión de aplicar una RL penalizada se toma basados en los beneficios que ofrece sobre la regresión clásica, uno de ellos el manejo de multicolinealidad, también ayuda a prevenir sobreajustes pues, la penalización logra reducir la complejidad del modelo y por último, este tipo de regresión simplifica los modelos visibilizando regresores más importantes del modelo, esto se debe a que la regularización es capaz de llevar coeficientes al valor de 0 (Bühlmann y van de Geer, 2011), (González, 2018).

Los hiperparámetros que entran a variar en la grilla de búsqueda son el parámetro 'C' de regularización (Valores entre 0.0001, 0.001, 0.01, 0.02, 0.05, 0.1, 0.25, 0.5, 0.75, 1) y el tipo de regularización: (ℓ_1, ℓ_2).

Los resultados de la grilla para este modelo a través del puntaje AUC se muestran en la gráfica de entrenamiento 5.4, en ella se observa el rápido crecimiento de la métrica utilizada, llegando a un punto de estabilización en valores de C superiores a 0.01, lo que

indica que se debe tomar un valor pequeño en la regularización (recordando que, valores pequeños fortalecen la misma regularización). Además, la regularización ℓ_2, presenta un mejor compota miento que la ell_1. Como medida de reducción de sobreajuste, evidente en algunos valores de la curva, se tomarán como hiperparámetros óptimos aquellos anteriores al cambio brusco de pendiente que se detalla en la figura.

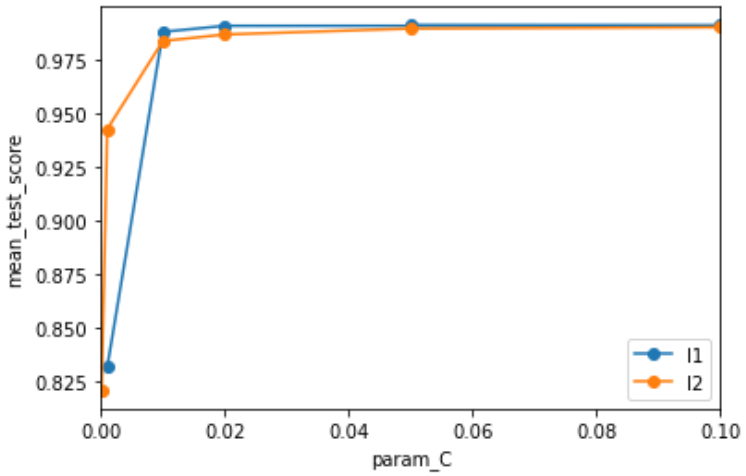

Figura 5.4: Curva de Entrenamiento RL. Elaboración Propia

Los resultados completos de la grilla de búsqueda se presentan en la tabla A.2 disponible en la sección de Anexos.

La curva de aprendizaje de este modelo se presenta en la gráfica 5.5, se evidencia que en las primeras iteraciones la precisión es cercana a 0.65 y a medida que se dan más iteraciones, la precisión aumenta y se estabiliza sobre el valor de 0.85. Este crecimiento indica la potencia del modelo a medida que se le da un mayor número de observaciones. Las brechas entre entrenamiento y validación son mínimas, lo que representa que el modelo no está generando sobreajuste.

5.4.2. *Random Forest*

En este caso la grilla de búsqueda contiene tres hiperparámetros: Número de árboles en el bosque $^{0}n\_estimators^{0}$ (Valores entre 50, 75, 100, 200 y 500), el número de características a considerar al buscar la mejor división de un nodos en los árboles *'max_features'* (Valores entre 2, 4, 6, 8, 10 y 12) y la profundidad máxima *'max_depth'*, en la que se decide variar valores enteros pequeños, para así restringir severamente el crecimiento de los árboles del bosque (Valores entre 1, 3, 5, 7, 9 y 11).

También se presenta la curva de entrenamiento para este modelo en la gráfica 5.7, en donde se observa que hay variación mínima entre el número de árboles o réplicas tipo Bootstrap, aunque sí existen variaciones respecto a la profundidad del modelo: valores de profundidad mayores o iguales a 3 muestran un fuerte cambio en la pendiente de la curva, alcanzando AUC superiores a 0.96, teniendo en cuenta esto, para evitar un modelo sobreajustado, se toma un subconjunto de hiperparámetros anterior a esta variación como conjunto óptimo.

Figura 5.5: C. de aprendizaje RL. Elaboración Propia

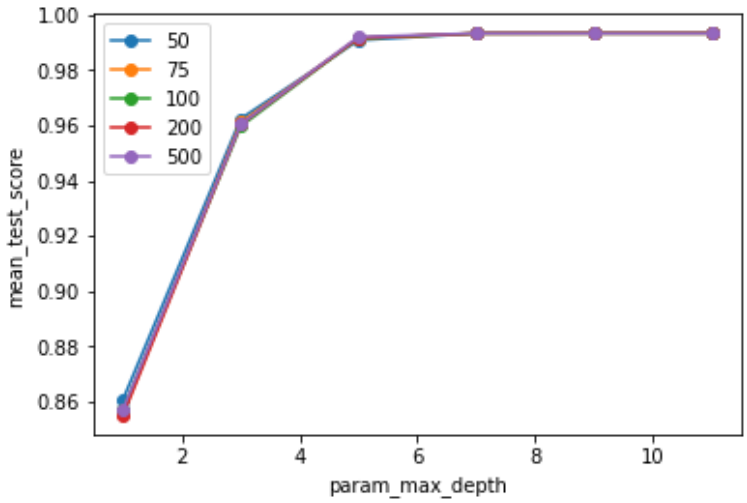

Figura 5.6: Curva de Entrenamiento RF. Elaboración Propia

Los resultados de la grilla de búsqueda también se pueden encontrar en la sección de Anexos en la tabla A.4.

La gráfica 5.7 muestra que las potencias de entrenamiento y validación tienden a estabilizarse a medida que el conjunto de entrenamiento crece, desde la primera iteración se evidencia que el modelo es potente (0.85) en cuanto a precisión, terminando en una precisión cercana a 0.83. No se presentan brechas pronunciadas de Entrenamiento vs. Validación. Aunque, en general el comportamiento es más heterogéneo a lo ya observado con la RL,

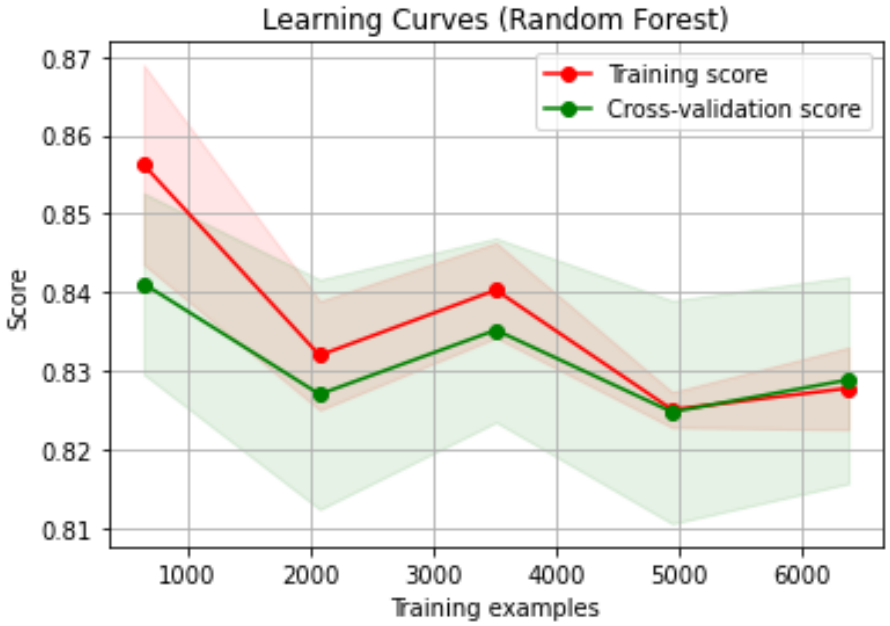

Figura 5.7: C. de aprendizaje RF. Elaboración Propia

5.4.3. *Support Vectorial Machine*

En la grilla de búsqueda para este modelo fueron incluidos tres hiperparámetros: Parámetro de Regularización 'C' (Valores entre 0.0001, 0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1), el tipo de kernel (entre Lineal y Radial 'rbf') y el parámetro γ (Valores entre 0.01, 0.1, 0.25, 0.5, 0.75 y 1, es importante mencionar que este hiperparámetro solo se tendrá en cuenta para el kernel radial).

Los scores AUC durante el entrenamiento por GridSearch para este modelo se presentan en la figura 5.9. En ella se observa un cambio de tendencia muy similar al ya observado en la RL, el kernel lineal presenta de inicio valores de AUC superiores a 0.9, que se estabilizan en valores muy cercanos a 1, este comportamiento indicaría un fuerte sobreajuste al aplicar el tipo de función Kernel, por ese motivo, se toma el kernel Radial, que presenta un crecimiento más estable, dado que inicia con un AUC aproximado a 0.6, se toma como valor de C=0.01.

La gráfica 5.9 muestra un comportamiento muy similar al de los anteriores dos modelo, con la diferencia que este modelo presentó una mayor variabilidad en cuanto a precisión pues, logra un gran pico muy cercano al 100 % de precisión y luego decrece hasta una precisión aproximada al 87 %. De igual forma, no se presentan brechas pronunciadas entre la curva de entrenamiento y la de validación, lo que tampoco daría indicios de sobreajuste.

Los resultados de la grilla de búsqueda se presentan en la tabla A.5 disponible en la sección de Anexos.

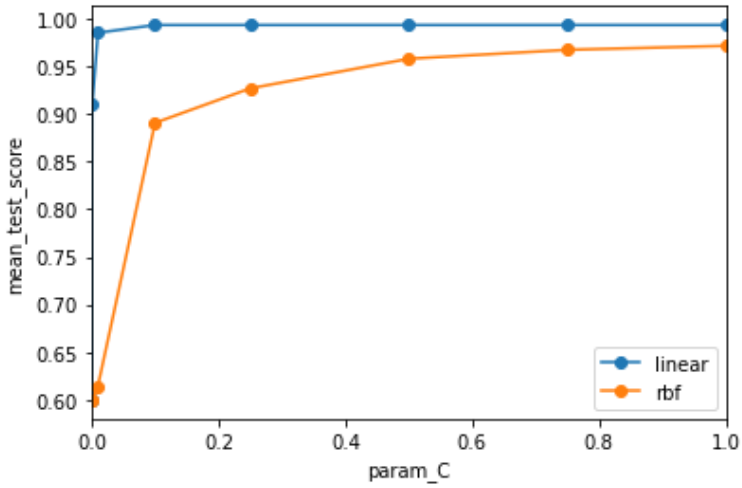

Figura 5.8: Curva de Entrenamiento SVC. Elaboración Propia

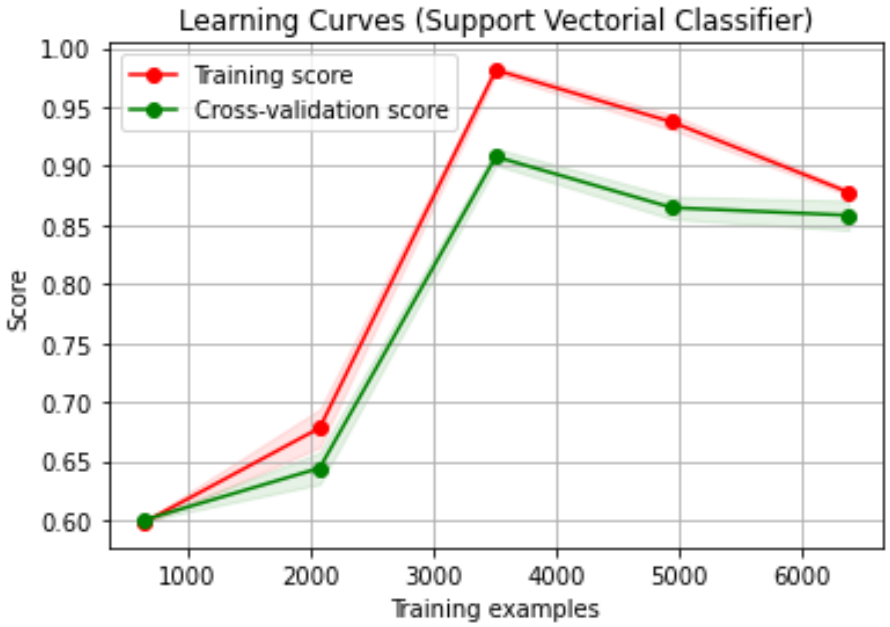

Figura 5.9: C. de aprendizaje SVC. Elaboración Propia

5.4.4. Ensamble

La propuesta de ensamble se construye con los modelos óptimos presentados anteriormente, aunque también se construye su curva de aprendizaje, ya que su entrenamiento también se da vía CV.

La gráfica 5.10 muestra que la propuesta de ensamble en el entrenamiento inicia con una precisión cercana al 0.9, creciendo y posteriormente encontrando estabilidad en una precisión del 0.98, mientras que, se observa una brecha más amplia respecto a las precisión de validación, que muestra un crecimiento constante.

Los hiperparámetros del meta-clasificador son: B = 50, LearningRate = 0.001, ProfundidadMax = 1.

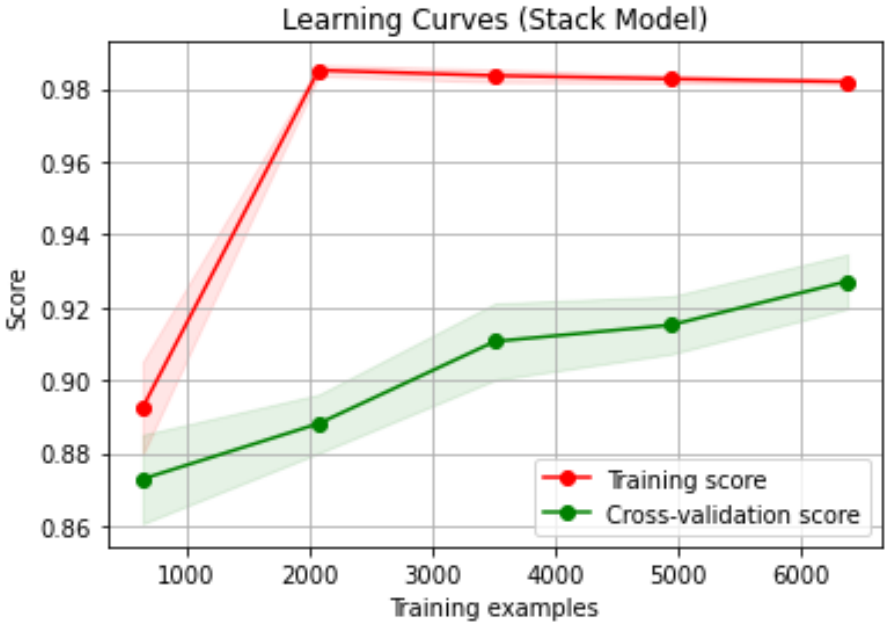

Figura 5.10: C. de aprendizaje Ensamble. Elaboración Propia

5.4.5. Modelos Óptimos

De acuerdo a lo anteriormente expuesto, se presentan los modelos de la tabla 5.2 como modelos óptimos, estos serán evaluados en la Fase de Validación.

Modelo	Hiperparámetros Óptimos	Accuracy
Regresión Logística ℓ_2	$C = 0{,}001$	0.858
Random Forest	$B = 200$; $m = 2$; ProfundidadMax = 1	0.826
SVC	$C = 0{,}1$; kernel: Radial (RBF); $\gamma = 1$	0.875
Stacking	$B = 50$; LearningRate = 0.001; ProfundidadMax = 1	0.982

Tabla 5.2: *Modelo Optimos de Entrenamiento*

5.5. Validación de los modelos

Para esta fase la validación de los modelos ya entrenados se realiza con el 30 % de los datos restantes (3040 registros).

Para esto se tendrán en cuenta las métricas mencionadas en el Marco Teórico y a partir de ellas dar una mejor conclusión respecto a un posible mejor modelo,

5.6. Curvas ROC

Como se expuso en la Fase de entrenamiento, estos modelos presentaron altas potencias de bondad de ajuste, por lo que se esperaría que al ingresar nuevos casos

(Validación) estos tengan la capacidad de emitir predicciones correctas. Las Curvas ROC, permiten evaluar los rendimientos de estos 4 modelos.

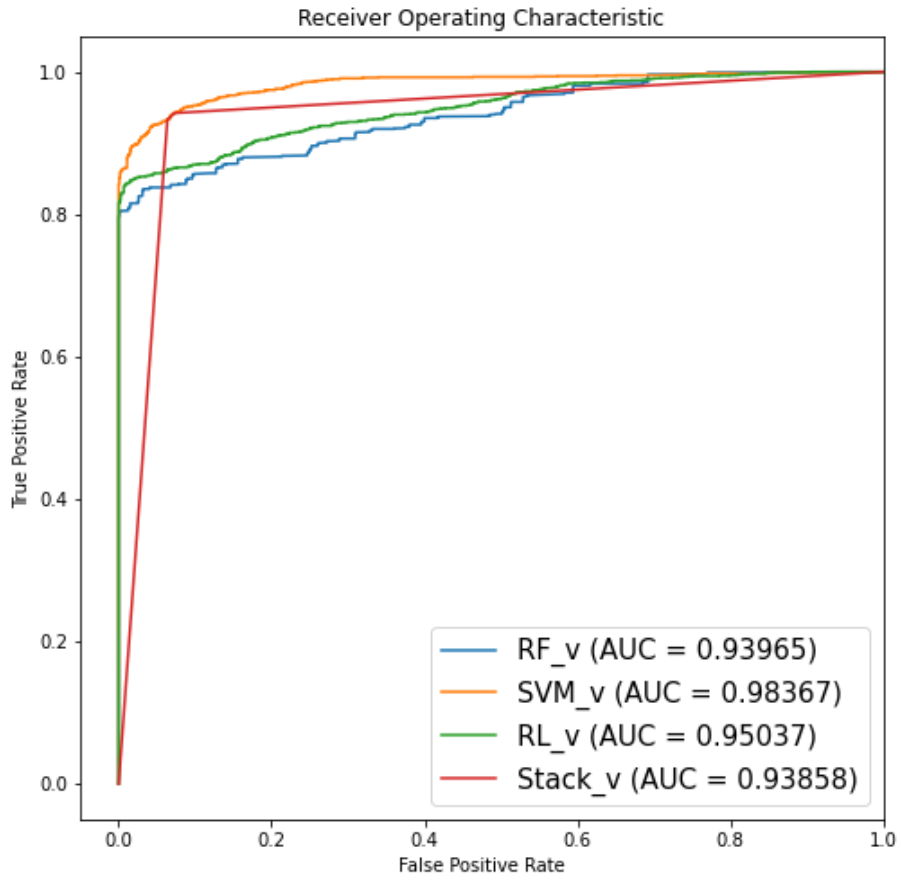

Figura 5.11: Curva ROC y AUC de los 4 modelos. Elaboración Propia

Estas curvas y los puntajes de AUC se muestran simultáneamente en la gráfica 5.11, donde los 4 modelos muestran un buen rendimiento, siendo el mejor en cuanto a este criterio la SVC, seguido de la RL, el RF y por último la propuesta de Ensamble. El criterio de AUC no parece ser determinante de los resultados finales, pues no se corresponde a los resultados del entrenamiento, por eso se procede a comparar los modelos por otros criterios como el Accuracy.

La gráfica 5.12 expone las precisiones de la Fase de Entrenamiento y la Fase de Validación, respectivamente, para los tres modelos construidos. Esta gráfica se complementa con lo expuesto en la fase de entrenamiento. Se evidencia que tomando el criterio de precisión, la propuesta de Stacking mejora los resultados respecto a sus competidores, seguido de la Regresión Logística, el SVC un eslabón inferior, que son los modelos que incluyen regularización en su construcción, estos modelos presentan rendimientos muy similares, esto refuerza la teoría de que modelos penalizados presentan mejores resultados que modelos no penalizados, ofreciendo mayor complejidad, que se traduce en menor riesgos de sobreajuste. Por último, el modelo de RF presentó los peores resultados respecto a los demás modelos.

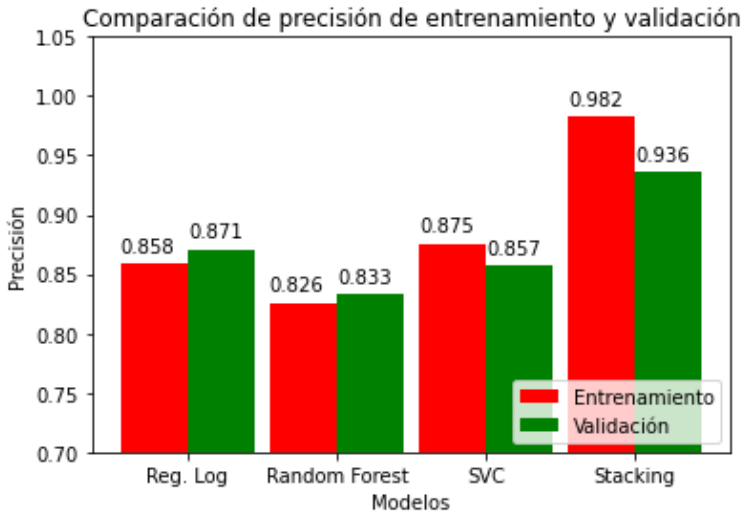

Figura 5.12: Precisión de Entrenamiento vs. Validación. Elaboración Propia

Modelo	AUC	Accuracy	F1-Score	Especificidad	Sensibilidad
Random Forest	0.9397	0.833	0.865	0.754	0.884
Regresión Logística	0.950	0.871	0.895	0.826	0.90
SVC	0.9837	0.857	0.867	1	0.764
Stacking	0.9386	0.936	0.947	0.927	0.942

Tabla 5.3: *Accuracy de Entrenamiento*

La tabla 5.3 muestra los rendimientos de los 4 clasificadores construidos. Mostrando como ganador a la novedosa propuesta de Ensamble vía Stacking presentada en este proyecto. Este modelo, a pesar de ser más costoso en procesamiento computacional, mejora la precisión de los demás modelos.

La matriz de confusión de la cual se construyen las métricas de validación del modelo ensamblado se muestra en la figura 5.13. Este modelo fue más sensible a clasificar falsos negativos sobre falsos positivos, 107 personas con dengue Con signos de alarma fueron clasificadas en dengue Sin signos de alarma por el modelo.

5.6.1. Factores de Riesgo

El modelo de RL-ℓ_2 tuvo un buen rendimiento en la fase de validación, por este motivo, además de la ventaja que proporciona al poder estimar OR de los coeficientes, se usará como identificador de factores de riesgo asociados al tipo de dengue.

La tabla 5.4 presenta un resumen de los coeficientes estimados por dicho modelo.

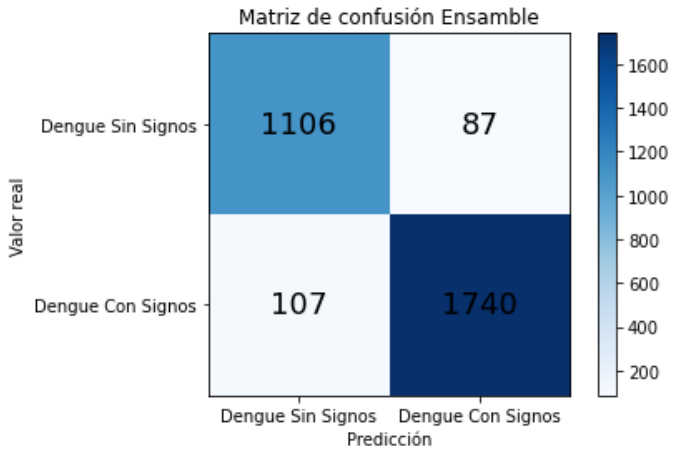

Figura 5.13: Matriz de Confusión Stacking. Elaboración Propia

	Coef.	$P > jzj$	OR	[0.025	0.975]
edad_	-0.0013	0.8457	9.987000e-01	0.9858	1.0118
estrato_	0.1070	0.3178	1.112900e+00	0.9022	1.3728
famantdngu	3.0405	0.0000	2.091590e+01	13.8435	31.6015
fiebre	-10.3274	0.0000	0.000000e+00	0.0000	0.0001
cefalea	-0.5408	0.1447	5.823000e-01	0.2815	1.2043
dolrretroo	0.3203	0.3114	1.377500e+00	0.7410	2.5609
malgias	0.6759	0.1754	1.965700e+00	0.7396	5.2247
artralgia	-0.6396	0.0517	5.275000e-01	0.2769	1.0048
erupcionr	-0.3704	0.2318	6.905000e-01	0.3762	1.2671
dolor_abdo	8.9998	0.0000	8.101471e+03	1786.8189	36732.2201
vomito	9.9111	0.0000	2.015224e+04	2469.4702	164453.3542
diarrea	9.4707	0.0000	1.297426e+04	1514.3019	111161.0284
caida_plaq	30.2851	0.9979	1.421244e+13	0.0000	inf

Tabla 5.4: *Resumen Regresion Log stica*

De esto, se evidencia que los campos significativos para este modelo fueron 'famantdngu': Antecedentes familiares en últimos 15 días, 'fiebre', 'dolo_abdo': Dolor abdominal, 'vomito', 'diarrea'y siendo algo flexibles 'artralgia'.

Para antecedentes se estimó un $\hat{OR} = 20{,}91$, que indica una fuerte asociación de la presencia de antecedentes cercanos de la familia los últimos 15 días antes del reporte del contagio con el presentar dengue con signos de alarma, síntomas como dolor abdominal, vomito y diarrea de igual forma presentan una fuerte asociación con el presentar dengue con signos de alarma, pues el coeficiente el positivo y toma valores muy altos, los $\hat{OR}$ fueron: 8102, 20152, 12973.

Mientras que la fiebre está más asociada al presentar dengue sin signos de alarma, pues el coeficiente estimado es menor que 0 y su $\hat{OR}$ 0.

Estos valores de $\hat{OR}$ tan masivos podrían dar idea de relaciones causales respecto al evento de interés Rothman y Greenland (2008), en este caso el tipo de dengue clasificado.

6. Discusión

Los resultados presentados muestran modelos de clasificación de altas efectividades para la clasificación del tipo de dengue. Los 4 modelos presentados proporcionan potencia para los datos presentados, en un contexto epidemiológico, la posibilidad de implementar herramientas potentes da un valor importante al desarrollo del Machine Learning y la Inteligencia Artificial. Los esfuerzos clínicos para reducir la carga de una enfermedad de alto impacto en salud pública serían más fructíferos.

Técnicas como la regularización de modelos, ofrecen múltiples ventajas en el desarrollo de herramientas potentes y efectivas que, además poseen equilibrio entre interpretación y complejidad, como se presentó en el desarrollo de los modelos presentados en esta investigación. Los resultados de estos modelos regularizados muestran que se logró reducir el riesgo de sobreajuste, también, estos modelos presentaron tendencias crecientes en su función de aprendizaje, lo que indica que a medida que se ingresan más datos, el algoritmo será más robusto para realizar las clasificaciones.

Uno de los resultados más importantes que se obtuvieron tras el desarrollo de esta investigación, fue la propuesta de ensamble. Ya que, en un contexto regional, esta metodología no ha sido aplicado para clasificación de dengue, por eso se planteó la hipótesis que esta metodología sería más potente que los modelos individuales, ofreciendo una importante ganancia en potencia de predicción, muy valorada en el área de las ciencias de la salud. Con los resultados presentados, se pudo confirmar la potencia de la técnica de Stacking pues, logró corregir el sesgo que presentaron los tres modelos independientes. Es importante mencionar que requiere un costo computacional alto que se traduce en alto rendimiento.

Para lograr la consolidación de resultados, fueron necesarias múltiples técnicas con el fin de disminuir riesgos de sobreajuste, como optimización de hiperparámetros, validación cruzada, re-muestreos en la construcción de los conjuntos de datos, regularizaciones, etc. Por el volumen de datos usados en este proyecto, esta tarea adquirió una mayor complejidad.

Es evidente, que la obtención de datos en salud, ya sean públicos o privados, supone un fuerte obstáculo en la investigación científica; cuando se logra obtener datos, muchas veces estos vienen cargados de errores, como sesgos de medición, errores en codificación de los datos, campos con observaciones faltantes, entre otras situaciones.

Con esto, se finaliza mencionando el aporte del aprendizaje automático en problemas de salud colombiana, estas técnicas de clasificación de dengue son novedosas en el país, se han aplicado pocas veces, es decir, que este trabajo puede servir de impulso para desarrollar herramientas de ML, no solo en enfermedades como el dengue, sino en enfermedades de transmisión viral. Con un mayor acceso a la información, se puede mejorar la metodología aplicada y obtener resultados más óptimos.

La investigación en temas de salud que afectan el desarrollo integral de territorios, no solo generaría avances en políticas de salud publica, sino que sería causal de mayor

desarrollo en el campo de la estadística en la nación y en la universidad. Logrando un apoyo constante de las facultades de medicina, el aporte sería reciproco en pro de, lograr atacar de una manera más efectiva problemáticas médicas, como la producción de nuevas generaciones de científicos que den luz al crecimiento de la ciencia de datos, la estadística y las matemáticas como herramientas de beneficio social.

7. Conclusiones

Se obtuvieron 4 herramientas potentes en la clasificación de los tipos de dengue. Son herramientas efectivas con métricas fuertes de evaluación. Donde, el mejor modelo presentado fue la propuesta de ensamble vía Stacking (AUC = 0.9386, Accuracy = 0.936, F1-Score = 0.947), seguido de la Regresión Logística regularizada por norma ℓ_2 (AUC = 0.95, Accuracy = 0.871, F1-Score = 0.895), la Máquina de Soporte de Vectorial - Kernel Radial (AUC = 0.984, Accuracy = 0.857, F1-Score = 0.867) y por último, el *Random Forest* (AUC = 0.94, Accuracy = 0.833, F1-Score = 0.865).

La propuesta de Stacking, es novedosa en cuanto a aplicaciones en datos de dengue en el país y en general en la región. Este modelo confirma ser un modelo robusto y potente, mejora los rendimientos tanto en el entrenamiento, como en la validación. Como posible trabajo futuro en pro de la mejoría en el modelo, se tomará un conjunto de variables predictoras mayor, donde los modelos de primer nivel tendrán como entrada un subgrupo del total de variables, para así dar mayor variabilidad a las predicciones, a priori se buscará dar entrada a más modelos de menor complejidad, de esta forma se reduce costo computacional y se gana variabilidad.

Las técnicas de regularización resultan ser potentes y adecuadas para la clasificación de tipo de dengue de acuerdo a su nivel de severidad, no dan indicios de sobreajuste a pesar de la cantidad de datos disponibles.

La presencia de antecedentes familiares por dengue, dolores en zona abdominal, vomito y diarrea están fuertemente relacionados con la presencia de dengue con signos de alarma. Una mejora a esta inferencia se daría con la aplicación de modelos de causalidad, como modelos de apalancamiento (*uplifting*) que permita una obtención de medidas de asociación directas a la clase dada a partir de un tratamiento realizado a los pacientes. Para esto se requiere de observaciones adecuadas para el modelo. Con esto se puede hacer una inferencia más detallada y suficiente.

Poder aplica estos resultados en otras enfermedades de transmisión viral daría mayor entendimiento del comportamiento de las infecciones. Y así poder realizar inferencia sobre las enfermedades y llevarlo a un marco nacional.

8. Referencias

Alpaydin, E. (2014). *Introduction to machine learning* (3.ª ed.). Cambridge, MA: MIT Press.

Betancourt, G. A. (2005). Las maquinas de soporte vectorial. *Scientia et Technica*, *27*(27), 67–72.

Bishop, C. M. (2007). *Pattern recognition and machine learning (information science and statistics)* (1.ª ed.). Springer. Descargado de http://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0387310738

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*(2), 123-140.

Breiman, L. (1996b). Stacked regressions. *Mach. Learn.*, *24*(1), 49-64. Descargado de http://dblp.uni-trier.de/db/journals/ml/ml24.html#Breiman96a

Bühlmann, P., y van de Geer, S. (2011). *Statistics for high-dimensional data: Methods, theory and applications*. Springer Science & Business Media.

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167. Descargado de https://doi.org/10.1023/A:1009715923555 doi: 10.1023/A:1009715923555

Cáceres-Manrique, F. d. M., Vesga-Gómez, C., Perea-Florez, X., Ruitorte, M., y Talbot, Y. (2009). Conocimientos, actitudes y prácticas sobre dengue en dos barrios de bucaramanga, colombia. *Revista de Salud Publica*, *11*, 27–38.

Caicedo-Torres, W., Montes-Grajales, D., Miranda-Castro, W., Fennix-Agudelo, M., y Agudelo-Herrera, N. (2017). Kernel-based machine learning models for the prediction of dengue and chikungunya morbidity in Colombia. *Communications in Computer and Information Science*, *735*, 472–484. doi: 10.1007/978-3-319-66562-7_34

Castrillón, J. C., Castaño, J. C., y Urcuqui, S. (2015). Dengue en Colombia: Diez años de evolución. *Revista Chilena de Infectologia*, *32*(2), 142–149. doi: 10.4067/S0716-10182015000300002

Clark, M. (2013). An Introduction to Machine Learning with Applications in R. *An Introduction to Machine Learning with Applications in R*, 1–43.

Cramér, H. (1946). *Mathematical methods of statistics*. Princeton University Press.

Cui, B. (2021). Dataexplorer: Automate data exploration and treatment [Manual de software informático]. Descargado de https://boxuancui.github.io/DataExplorer/ (R package version 1.0.0)

Dietterich, T. G. (2000). Ensemble methods in machine learning. En *Multiple classi er systems* (Vol. 1857, pp. 1–15). doi: 10.1007/3-540-45014-9_1

Foundation, P. S. (2020, octubre). *Python Language Reference, version 3.9.* Descargado de https://doi.org/10.5281/zenodo.4273455 doi: 10.5281/zenodo.4273455

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189 – 1232. Descargado de https://doi.org/10.1214/aos/1013203451 doi: 10.1214/aos/1013203451

Gambhir, S., Malik, S. K., y Kumar, Y. (2018). The Diagnosis of Dengue Disease: An Evaluation of Three Machine Learning Approaches. *International Journal of Healthcare Information Systems and Informatics*, *13*(3), 1–19. doi: 10.4018/IJHISI.2018070101

Geron, A. (2017). *Hands-on machine learning with scikit-learn and tensor ow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media.

González, F. (2018). *Regresion regularizada.* Disponible en: https://felipegonzalez.github.io/aprendizaje-maquina-verano-2018/regresion-regularizada.html.

Goodfellow, I. J., Bengio, Y., y Courville, A. (2016). *Deep learning*. Cambridge, MA, USA: MIT Press. (http://www.deeplearningbook.org)

Guo, P., Liu, T., Zhang, Q., Wang, L., Xiao, J., Zhang, Q., ... Ma, W. (2017). *Developing a dengue forecast model using machine learning: A case study in China.* doi: https://doi.org/10.1371/journal.pntd.0005973

Gutiérrez-Ruiz, L., Quintero-Gil, D., y Martínez-Gutiérrez, M. (2012). Actualización en diagnóstico del dengue: evolución de las técnicas y su aplicación real en la clínica. *La cl nica y el laboratorio*, *18*, 411–441.

Guzman, M. G., y Harris, E. (2015). Dengue. *The Lancet*, *385*(9966), 453–465. doi: 10.1016/S0140-6736(14)60572-9

Guzmán, M. G., y Kourí, G. (2004). Dengue diagnosis, advances and challenges. *International Journal of Infectious Diseases*, *8*(2), 69–80. doi: 10.1016/J.IJID.2003.03.003

Hastie, T., Tibshirani, R., y Friedman, J. (2009). *The Elements of Statistical Learning* (2.ª ed.). Nueva York: Springer. Descargado de https://hastie.su.domains/Papers/ESLII.pdf

Ho, T.-S., Weng, T.-C., Wang, J.-D., Han, H.-C., Cheng, H.-C., Yang, C.-C., ... Liu, C.-C. (2020, 11). Comparing machine learning with case-control models to identify confirmed dengue cases. *PLOS Neglected Tropical Diseases*, *14*(11), 1-21. Descargado de https://doi.org/10.1371/journal.pntd.0008843 doi: 10.1371/journal.pntd.0008843

Hosmer, D. W. (2013). *Applied logistic regression david w. hosmer, stanley lemeshow, rodney x. sturdivant.* (3rd ed. ed.). Hoboken, N.J. :: Wiley,.

Hoyos, W., Aguilar, J., y Toro, M. (2021). Dengue models based on machine learning techniques: A systematic literature review. *Arti cial Intelligence in Medicine*, *119*, 102157. Descargado de https://www.sciencedirect.com/science/article/pii/S0933365721001500 doi: https://doi.org/10.1016/j.artmed.2021.102157

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95.

Iqbal, N., y Islam, M. (2019). Machine learning for dengue outbreak prediction: A performance evaluation of different prominent classifiers. *Informatica (Slovenia)*, *43*(3), 363–371. doi: 10.31449/inf.v43i1.1548

Loh, W. Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 14–23. doi: 10.1002/widm.8

M, V. K. H., S, D. S. D., y E, D. D. P. (2022, May). Clinical dengue data analysis and prediction using multiple classifiers: An ensemble techniques. *Global Journal of Computer Science and Technology*, *22*(D2), 37–51. Descargado de https://computerresearch.org/index.php/computer/article/view/102264 doi: 10.34257/GJCSTDVOL22IS2PG37

Mohd Salim, N. A., Wah, Y. B., Reeves, C., Smith, M., Yaacob, W. F. W., Mudin, R. N., ... Haque, U. (2021). *Prediction of dengue outbreak in Selangor Malaysia using machine learning techniques* (Vol. 11) (n.º 1). doi: 10.1038/s41598-020-79193-2

Murphy, K. P. (2013). *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press. Descargado de https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2

OMS. (s.f.). Treatment, prevention and control global strategy for dengue prevention and control 2.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21–45. doi: 10.1109/MCAS.2006.1688199

R Core Team. (2021). R: A language and environment for statistical computing [Manual de software informático]. Vienna, Austria. Descargado de https://www.R-project.org/

Raschka, S., Patterson, J., y Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, *11*(4). Descargado de https://www.mdpi.com/2078-2489/11/4/193 doi: 10.3390/info11040193

Rodríguez, J., y Correa, C. (2009). Predicción Temporal de la Epidemia de Dengue en Colombia: Dinámica Probabilista de la Epidemia. *Revista de Salud Publica*, *11*(3), 443–453.

Rothman, K. J., y Greenland, S. (2008). *Modern epidemiology*. Lippincott Williams & Wilkins.

Sarma, D., Hossain, S., Mittra, T., Bhuiya, M. A. M., Saha, I., y Chakma, R. (2020). Dengue Prediction using Machine Learning Algorithms. *IEEE Region 10 Humanitarian Technology Conference, R10-HTC*, *2020-Decem*. doi: 10.1109/R10-HTC49770.2020.9357035

Smola, A. J., y Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222. doi: https://doi.org/10.1023/B:STCO.0000035301.49549.88

Ting, K. M. (1999). Issues in stacked generalization. *Journal of Arti cial Intelligence Research*, *10*, 271–289. doi: 10.1613/jair.624

Torres, E. M. (2008). Dengue. *Estudos Avancados*, *22*(64), 33–52. doi: 10.1590/s0103-40142008000300004

van Buuren, S., y Groothuis-Oudshoorn, K. (2021). Multivariate imputation by chained equations [Manual de software informático]. Descargado de https://stefvanbuuren.github.io/mice/ (R package version 3.13.0)

Vapnik, V. N. (1995). *The Nature of Statistical Theory* (2.ª ed.; M. Jordan, J. F. Lawless, S. L. Lauritzen, y V. Nair, Eds.). New York: Springer. Descargado de https://statisticalsupportandresearch.files.wordpress.com/2017/05/vladimir-vapnik-the-nature-of-statistical-learning-springer-2010.pdf

Villar, L. A., Rojas, D. P., Besada-Lombana, S., y Sarti, E. (2015, 03). Epidemiological trends of dengue disease in colombia (2000-2011): A systematic review. *PLOS Neglected Tropical Diseases*, *9*(3), 1-16. Descargado de https://doi.org/10.1371/journal.pntd.0003499 doi: 10.1371/journal.pntd.0003499

Wickham, H., François, R., Henry, L., y Müller, K. (2021). dplyr: A grammar of data manipulation [Manual de software informático]. Descargado de https://dplyr.tidyverse.org (R package version 1.0.7)

Wolpert, D. (1992). Stacked Generalization (Stacking). *Neural Networks*, *5*, 241–259.

World Health Organization. (2009). *Dengue guias para el diagnostico, tratamiento, prevencion y control: nueva edicion.* Organizacion Mundial de la Salud.

World Health Organization. (2012). *Global strategy for dengue prevention and control 2012-2020* [Publications]. World Health Organization.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC Press.

Anexos

A. Anexos

A.1. Descripción de los datos

Campo	Descripción
edad_	edad paciente
estrato_	es clasificación socio económica de los inmuebles en donde residen
famantdngu	¿Algún familiar o conviviente ha tenido sintomatología de dengue en los últimos 15 días?: 1 = Si, 0 = No , 3 = desconocido
fiebre	fiebre: 1 = Si, 0 = No
cefalea	cefalea: 1 = Si, 0 = No
dolrretroo	Dolor retroocular: 1 = Si, 0 = No
malgias	Mialgias: 1 = Si, 0 = No
artralgia	Artralgias: 1 = Si, 0 = No
erupcionr	Erupción o rash: 1 = Si, 0 = No
dolor_abdo	Dolor abdominal: 1 = Si, 0 = No
vomito	Vómito: 1 = Si, 0 = No
diarrea	Diarrea: 1 = Si, 0 = No
caida_plaq	Caída de plaquetas (< 100000): 1 = Si, 0 = No
clasfinal	Clasificación final: 1 = Dengue con signos de alarma, 2 = Dengue sin signos de alarma , 3 = Dengue grave

Tabla A.1: *Descripcion de datos tras preprocesamiento*

A.2. Resultados Grilla de Búsqueda

	param_C	param_penalty	mean_test_score	std_test_score
0	0.0001	l1	0.500	0.000
10	0.0001	l2	0.819	0.016
1	0.001	l1	0.834	0.015
11	0.001	l2	0.941	0.009
12	0.01	l2	0.985	0.003
13	0.02	l2	0.992	0.003
2	0.01	l1	0.995	0.002
14	0.05	l2	0.995	0.002
16	0.25	l2	0.996	0.002
17	0.5	l2	0.996	0.002
15	0.1	l2	0.996	0.002
18	0.75	l2	0.996	0.002
9	1	l1	0.996	0.002
19	1	l2	0.996	0.002
8	0.75	l1	0.996	0.002
7	0.5	l1	0.996	0.002
6	0.25	l1	0.996	0.002
5	0.1	l1	0.996	0.002
4	0.05	l1	0.996	0.002
3	0.02	l1	0.996	0.001

Tabla A.2: *Grilla Regresion Log stica*

	param_n_estimators	param_max_features	param_max_depth	mean_test_score	std_test_score
3	200	2	1	0.829	0.013
6	75	4	1	0.840	0.018
7	100	4	1	0.843	0.017
4	500	2	1	0.843	0.013
1	75	2	1	0.845	0.018
2	100	2	1	0.850	0.015
5	50	4	1	0.851	0.016
9	500	4	1	0.851	0.018
8	200	4	1	0.852	0.016
11	75	6	1	0.855	0.015
13	200	6	1	0.855	0.015
12	100	6	1	0.855	0.016
10	50	6	1	0.855	0.016
14	500	6	1	0.855	0.016
0	50	2	1	0.864	0.017
16	75	8	1	0.864	0.011
15	50	8	1	0.865	0.011
17	100	8	1	0.865	0.011
19	500	8	1	0.865	0.011
20	50	10	1	0.865	0.011
18	200	8	1	0.865	0.011
21	75	10	1	0.865	0.011
25	50	12	1	0.865	0.011
24	500	10	1	0.865	0.011
23	200	10	1	0.865	0.011
29	500	12	1	0.865	0.011
22	100	10	1	0.865	0.011
27	100	12	1	0.865	0.011
26	75	12	1	0.865	0.011
28	200	12	1	0.865	0.011
34	500	2	3	0.941	0.005
32	100	2	3	0.943	0.006
33	200	2	3	0.943	0.005
31	75	2	3	0.948	0.005
37	100	4	3	0.948	0.004
38	200	4	3	0.949	0.004
39	500	4	3	0.949	0.004
35	50	4	3	0.950	0.006
36	75	4	3	0.953	0.007
30	50	2	3	0.956	0.004
52	100	10	3	0.966	0.005
48	200	8	3	0.966	0.005
46	75	8	3	0.966	0.005
47	100	8	3	0.966	0.005
53	200	10	3	0.966	0.005
51	75	10	3	0.966	0.005
54	500	10	3	0.966	0.005
57	100	12	3	0.966	0.005
55	50	12	3	0.966	0.005
50	50	10	3	0.966	0.005
56	75	12	3	0.966	0.005
58	200	12	3	0.966	0.005
59	500	12	3	0.966	0.005
45	50	8	3	0.968	0.006
41	75	6	3	0.969	0.004
42	100	6	3	0.969	0.004
40	50	6	3	0.969	0.004
44	500	6	3	0.970	0.003
49	500	8	3	0.970	0.006
43	200	6	3	0.973	0.006
60	50	2	5	0.984	0.003
62	100	2	5	0.984	0.003
63	200	2	5	0.984	0.004
61	75	2	5	0.984	0.003
64	500	2	5	0.986	0.004
65	50	4	5	0.989	0.005
66	75	4	5	0.992	0.004
70	50	6	5	0.992	0.004
67	100	4	5	0.992	0.004
68	200	4	5	0.993	0.003
90	50	2	7	0.993	0.003
91	75	2	7	0.993	0.003
177	100	12	11	0.993	0.003
116	75	12	7	0.993	0.003
118	200	12	7	0.993	0.003
147	100	12	9	0.993	0.003
148	200	12	9	0.993	0.003
117	100	12	7	0.993	0.003
119	500	12	7	0.993	0.003
89	500	12	5	0.993	0.003
179	500	12	11	0.993	0.003
178	200	12	11	0.993	0.003
88	200	12	5	0.993	0.003
149	500	12	9	0.993	0.003
136	75	8	9	0.993	0.003
142	100	10	9	0.993	0.003
131	75	6	9	0.993	0.003
132	100	6	9	0.993	0.003
146	75	12	9	0.993	0.003
145	50	12	9	0.993	0.003
126	75	4	9	0.993	0.003
141	75	10	9	0.993	0.003
79	500	8	5	0.993	0.003
80	50	10	5	0.993	0.003
102	100	6	7	0.993	0.003

Tabla A.3: *Grilla Random Forest I*

	param_n_estimators	param_max_features	param_max_depth	mean_test_score	std_test_score
101	75	6	7	0.993	0.003
100	50	6	7	0.993	0.003
99	500	4	7	0.993	0.003
98	200	4	7	0.993	0.003
97	100	4	7	0.993	0.003
78	200	8	5	0.993	0.003
96	75	4	7	0.993	0.003
94	500	2	7	0.993	0.003
93	200	2	7	0.993	0.003
127	100	4	9	0.993	0.003
92	100	2	7	0.993	0.003
106	75	8	7	0.993	0.003
107	100	8	7	0.993	0.003
108	200	8	7	0.993	0.003
87	100	12	5	0.993	0.003
95	50	4	7	0.993	0.003
77	100	8	5	0.993	0.003
103	200	6	7	0.993	0.003
125	50	4	9	0.993	0.003
76	75	8	5	0.993	0.003
75	50	8	5	0.993	0.003
74	500	6	5	0.993	0.003
73	200	6	5	0.993	0.003
72	100	6	5	0.993	0.003
71	75	6	5	0.993	0.003
109	500	8	7	0.993	0.003
69	500	4	5	0.993	0.003
110	50	10	7	0.993	0.003
111	75	10	7	0.993	0.003
112	100	10	7	0.993	0.003
113	200	10	7	0.993	0.003
114	500	10	7	0.993	0.003
115	50	12	7	0.993	0.003
120	50	2	9	0.993	0.003
121	75	2	9	0.993	0.003
122	100	2	9	0.993	0.003
123	200	2	9	0.993	0.003
124	500	2	9	0.993	0.003
86	75	12	5	0.993	0.003
143	200	10	9	0.993	0.003
85	50	12	5	0.993	0.003
83	200	10	5	0.993	0.003
171	75	10	11	0.993	0.003
172	100	10	11	0.993	0.003
173	200	10	11	0.993	0.003
174	500	10	11	0.993	0.003
175	50	12	11	0.993	0.003
176	75	12	11	0.993	0.003
153	200	2	11	0.993	0.003
129	500	4	9	0.993	0.003
170	50	10	11	0.993	0.003
152	100	2	11	0.993	0.003
135	50	8	9	0.993	0.003
134	500	6	9	0.993	0.003
133	200	6	9	0.993	0.003
151	75	2	11	0.993	0.003
138	200	8	9	0.993	0.003
139	500	8	9	0.993	0.003
140	50	10	9	0.993	0.003
137	100	8	9	0.993	0.003
150	50	2	11	0.993	0.003
169	500	8	11	0.993	0.003
168	200	8	11	0.993	0.003
167	100	8	11	0.993	0.003
82	100	10	5	0.993	0.003
81	75	10	5	0.993	0.003
105	50	8	7	0.993	0.003
128	200	4	9	0.993	0.003
104	500	6	7	0.993	0.003
130	50	6	9	0.993	0.003
154	500	2	11	0.993	0.003
155	50	4	11	0.993	0.003
156	75	4	11	0.993	0.003
157	100	4	11	0.993	0.003
158	200	4	11	0.993	0.003
159	500	4	11	0.993	0.003
160	50	6	11	0.993	0.003
161	75	6	11	0.993	0.003
162	100	6	11	0.993	0.003
163	200	6	11	0.993	0.003
164	500	6	11	0.993	0.003
165	50	8	11	0.993	0.003
166	75	8	11	0.993	0.003
84	500	510	5	0.993	0.003
144	500	10	9	0.993	0.003

Tabla A.4: *Grilla Random Forest II*

	param_C	param_kernel	param_gamma	mean_test_score	std_test_score
0	0.0001	linear	NaN	0.600	0.001
8	0.0001	rbf	0.01	0.600	0.001
9	0.0001	rbf	0.1	0.600	0.001
11	0.0001	rbf	0.5	0.600	0.001
12	0.0001	rbf	0.75	0.600	0.001
13	0.0001	rbf	1	0.600	0.001
14	0.001	rbf	0.01	0.600	0.001
18	0.001	rbf	0.75	0.600	0.001
10	0.0001	rbf	0.25	0.600	0.001
17	0.001	rbf	0.5	0.600	0.001
16	0.001	rbf	0.25	0.600	0.001
25	0.01	rbf	1	0.600	0.001
24	0.01	rbf	0.75	0.600	0.001
15	0.001	rbf	0.1	0.600	0.001
23	0.01	rbf	0.5	0.600	0.001
20	0.01	rbf	0.01	0.600	0.001
19	0.001	rbf	1	0.600	0.001
22	0.01	rbf	0.25	0.631	0.006
21	0.01	rbf	0.1	0.658	0.009
31	0.1	rbf	1	0.858	0.013
30	0.1	rbf	0.75	0.871	0.011
37	0.25	rbf	1	0.881	0.011
26	0.1	rbf	0.01	0.882	0.014
29	0.1	rbf	0.5	0.901	0.008
1	0.001	linear	NaN	0.910	0.010
32	0.25	rbf	0.01	0.914	0.008
27	0.1	rbf	0.1	0.914	0.009
28	0.1	rbf	0.25	0.918	0.009
36	0.25	rbf	0.75	0.922	0.007
38	0.5	rbf	0.01	0.932	0.006
33	0.25	rbf	0.1	0.944	0.005
44	0.75	rbf	0.01	0.944	0.006
43	0.5	rbf	1	0.948	0.005
35	0.25	rbf	0.5	0.949	0.007
34	0.25	rbf	0.25	0.952	0.006
50	1	rbf	0.01	0.952	0.007
42	0.5	rbf	0.75	0.963	0.009
49	0.75	rbf	1	0.966	0.009
39	0.5	rbf	0.1	0.966	0.008
41	0.5	rbf	0.5	0.968	0.007
40	0.5	rbf	0.25	0.970	0.008
55	1	rbf	1	0.970	0.007
48	0.75	rbf	0.75	0.970	0.007
54	1	rbf	0.75	0.972	0.007
47	0.75	rbf	0.5	0.974	0.008
45	0.75	rbf	0.1	0.975	0.007
46	0.75	rbf	0.25	0.975	0.007
53	1	rbf	0.5	0.977	0.006
51	1	rbf	0.1	0.979	0.005
52	1	rbf	0.25	0.980	0.005
2	0.01	linear	NaN	0.985	0.004
7	1	linear	NaN	0.993	0.003
6	0.75	linear	NaN	0.993	0.003
5	0.5	linear	NaN	0.993	0.003
4	0.25	linear	NaN	0.993	0.003
3	0.1	linear	NaN	0.993	0.003

Tabla A.5: *Grilla Support Vectorial Machine*

A.3. Matrices de Confusión

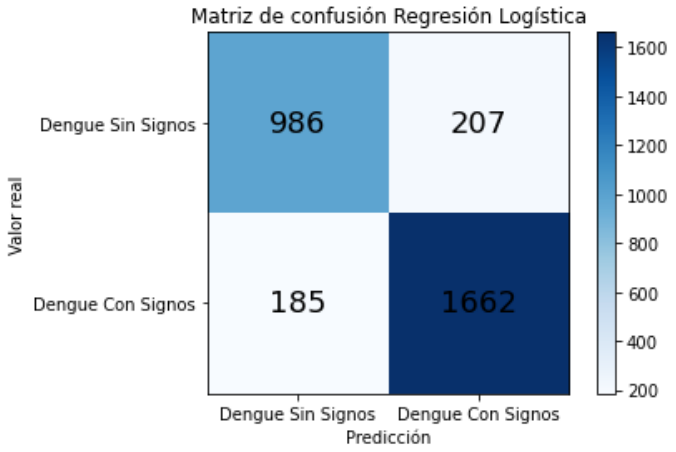

Figura A.1: Matriz de confusión Regresión Logística

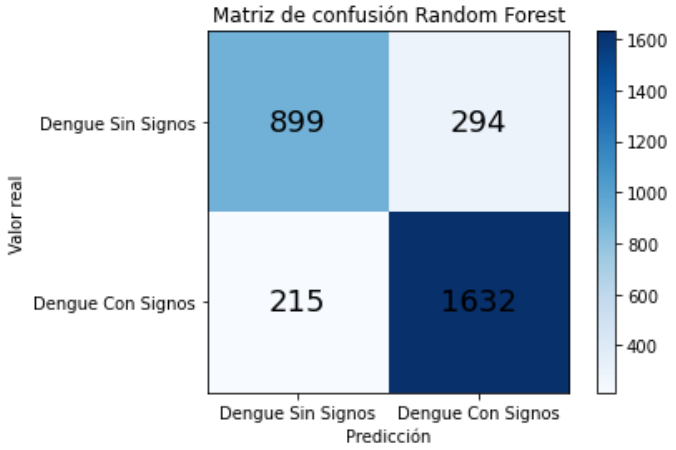

Figura A.2: Matriz de confusión *Random Forest*

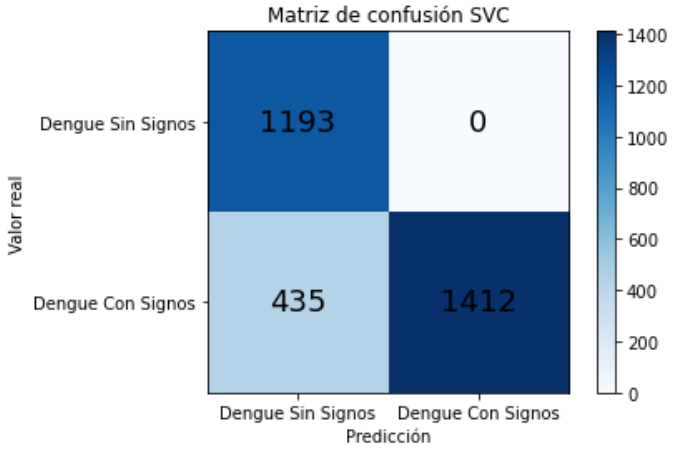

Figura A.3: Matriz de confusión *Support Vectorial Classi er*

A.4. Código del preprocesamiento en R

```
    ## Librerias Necesarias
require(tidyverse)
require(DataExplorer)
require(caret)
require(mice)
require(readxl)
library(lubridate)
library(corrplot)


## Carga de datos
dengue_act <- read.csv("G:/Mi unidad/2022II/Proyecto/Datos/
   dengueact.csv", na.strings="")

## Frecuencia en las clases observadas del dataset
table(dengue_act$clasfinal)

## Vista previa de variables con valores faltantes

plot_missing(data = dengue_act,missing_only = T, title = "
   Porcentaje de Valores Faltantes por Campo")
profile_missing(dengue_act) %>% filter(pct_missing > 0.35)
   %>% arrange(desc(pct_missing))

## Se elimina la clase dengue grave por temas de desbalance
   en las clases finales
dengue2 <- dengue_act[dengue_act$clasfinal != 'Dengue Grave',
    ]


## Eliminaci n de columnas con m s de un 70% de valores
   faltantes
mis1 <- profile_missing(dengue2) %>%
  arrange(desc(pct_missing))

## c(mis1[mis1$pct_missing>0.7,1])

dengue3 <- dengue2 %>%
  select(-c(mis1[mis1$pct_missing>0.7,1]))

## Eliminaci n de mediciones que constan de fechas, c digos
   , etc. que no son de utilidad para el modelo

dengue4 <- dengue3 %>% select(-c(orden,cod_eve, grupo, cod_
   sub,
```

```
                                    uni_med_, unid_edad, grupo_
                                       etario,
                                    nombre_nacionalidad, cod_
                                       pais_o,
                                    cod_dpto_o, cod_mun_o,
                                       ocupacion_,
                                    cod_ase_, per_etn_, fuente_,
                                    cod_pais_r, cod_mun_r, nuni_
                                       modif,
                                    fec_arc_xl, fec_aju_, nit_
                                       upgd, version,
                                    desplazami, nom_eve, ndep_
                                       proce,
                                    nmun_proce, npais_resi, nmun
                                       _resi,
                                    ndep_notif, nmun_notif)) ##
                                       edad_, uni_med_, unid_
                                       edad, ocupacion_, per_etn
                                       _
dengue4 <- dengue4 %>% mutate(fec_not = dmy(str_replace_all(
   fec_not, '/', '-')),
                              fec_con_ = as.Date(fec_con_),
                              ini_sin_ = as.Date(ini_sin_),
                              fecha_nto_ = as.Date(fecha_nto_
                                 )) %>%
  rename(anio = a  .o, danio_organ = da  .o_organ, muesrinion
      = muesri .on) %>%
  mutate(anio = factor(anio),
         clasif_edad = factor(clasif_edad),
         def_clas_edad = factor(def_clas_edad),
         sexo_ = factor(sexo_),
         area_ = factor(area_),
         tip_ss_ = factor(tip_ss_),
         estrato_ = factor(estrato_),
         gp_discapa = factor(gp_discapa),
         gp_desplaz = factor(gp_desplaz),
         gp_migrant = factor(gp_migrant),
         gp_carcela = factor(gp_carcela),
         gp_gestan = factor(gp_gestan),
         gp_indigen = factor(gp_indigen),
         gp_pobicbf = factor(gp_pobicbf),
         gp_mad_com = factor(gp_mad_com),
         gp_desmovi = factor(gp_desmovi),
         gp_psiquia = factor(gp_psiquia),
         gp_vic_vio = factor(gp_vic_vio),
         gp_otros = factor(gp_otros),
         tip_cas_ = factor(tip_cas_),
```

```
          pac_hos_ = factor(pac_hos_),
          con_fin_ = factor(con_fin_),
          ajuste_ = factor(ajuste_),
          famantdngu = factor(famantdngu),
          fiebre = factor(fiebre),
          cefalea = factor(cefalea),
          dolrretroo = factor(dolrretroo),
          malgias = factor(malgias),
          artralgia = factor(artralgia),
          erupcionr = factor(erupcionr),
          dolor_abdo = factor(dolor_abdo),
          vomito = factor(vomito),
          diarrea = factor(diarrea),
          somnolenci = factor(somnolenci),
          hipotensio = factor(hipotensio),
          hepatomeg = factor(hepatomeg),
          hem_mucosa = factor(hem_mucosa),
          hipotermia = factor(hipotermia),
          aum_hemato = factor(aum_hemato),
          caida_plaq = factor(caida_plaq),
          acum_liqui = factor(acum_liqui),
          extravasac = factor(extravasac),
          hemorr_hem = factor(hemorr_hem),
          choque = factor(choque),
          danio_organ = factor(danio_organ),
          muesttejid = factor(muesttejid),
          mueshigado = factor(mueshigado),
          muesbazo = factor(muesbazo),
          muespulmon = factor(muespulmon),
          muescerebr = factor(muescerebr),
          muesmiocar = factor(muesmiocar),
          muesmedula = factor(muesmedula),
          muesrinion = factor(muesrinion),
          conducta = factor(conducta),
          ndep_resi = factor(ndep_resi),
          clasfinal = factor(clasfinal)) %>%
  select(-c(muesttejid, mueshigado, muesbazo, muespulmon,
            muescerebr, muesmiocar, muesmedula, muesrinion,
            extravasac, hemorr_hem, choque, danio_organ))

summary(dengue4)

## Segunda eliminaci n de mediciones no  tiles

dengue4 <- dengue4 %>% select(-c(fec_not, anio, semana,
                                 cod_pre, clasif_edad,
                                 def_clas_edad,
```

```
                                    aseguradora, gp_otros,
                                    cod_dpto_r, fec_con_,
                                    ini_sin_, con_fin_,
                                    tip_cas_, ajuste_,
                                    fecha_nto_, conducta,
                                    nom_upgd, ndep_resi,
                                    COMUNA.shp, BARRIO_VER.shp))


summary(dengue4)

dengue_f <- dengue4 %>% select(-c(area_, tip_ss_, gp_discapa,
    gp_desmovi, gp_desplaz,
                                    gp_migrant, gp_carcela, gp_
                                       gestan, gp_indigen, gp_
                                       pobicbf,
                                    gp_mad_com, gp_psiquia, gp_
                                       vic_vio, pac_hos_))
## Imputaci n Via Mice

imput1 <- mice(dengue_f, m = 1)
dengue_input <- complete(imput1)
summary(dengue_input)
attach(dengue_input)

## Coeficientes de cramer entre variables para identificar
   intesidad de asociaciones a partir de
## la prueba Chi.Cuadrado de independencia

cordengue <- DescTools::PairApply(dengue_input, DescTools::
   CramerV)
corrplot(cordengue, type = "upper",
         is.corr = F, diag = F)

## Ajuste en los n veles de los factores

sapply(dengue_input, levels)

for (col in colnames(dengue_input)) {
  if (is.factor(dengue_input[[col]]) && !(col %in% c("estrato
     _", "famantdngu"))) {
    levels(dengue_input[[col]]) <- c("1", "0")
  }
}

str(dengue_input)
```

```
## Eliminaci n de campos desbalanceados

dengue_pre <- dengue_input %>% select(-c(acum_liqui,
   hipotermia, sexo_,
                                        somnolenci,
                                           hipotensio, aum_
                                           hemato,
                                        hem_mucosa,
                                           hepatomeg))
summary(dengue_pre)

writexl::write_xlsx(dengue_pre, path = "dengue_pre.xlsx")
```

A.5. Código de Entrenamiento y Validación Python

```
## Carga de Modulos

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import StackingClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import f1_score

## Carga de los datos

datosdengue = pd.read_excel("dengue_pre.xlsx",  sheet_name="
   Sheet1")
datosdengue.head()


# Seleccionamos todas las columnas que son de tipo object,
   excepto la columna 'E'
cols_to_convert = [col for col in datosdengue.columns if
   datosdengue[col].dtype == 'int64' and col != 'edad_']

# Convertimos las columnas seleccionadas a tipo category
```

```
datosdengue[cols_to_convert] = datosdengue[cols_to_convert].
   astype('category')

## Construcci  n conjuntos de entrenamiento y prueba

X = datosdengue.drop('clasfinal', axis = 1)
y = datosdengue['clasfinal']
# 30%
X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size = 0.3, random_state = 202305)



### Construcci  n de modelos

# Modelos del Primer Nivel

SVM = SVC(random_state=12)
RF = RandomForestClassifier(random_state=12, n_jobs = -1)
RL = LogisticRegression(random_state=12, n_jobs = -1,
                        solver = 'saga', max_iter=5000)

# B squeda por grilla RL

param_grid = [
    {"penalty": ["l1"], "C": [0.0001, 0.001, 0.01, 0.02,
       0.05, 0.1, 0.25, 0.5,
                        0.75, 1]},
    {"penalty": ["l2"], "C": [0.0001, 0.001, 0.01, 0.02,
       0.05, 0.1, 0.25, 0.5,
                        0.75, 1]}
]

#grid_param1 = [0.01, 0.05, 0.1, 0.25, 0.5,
#                    0.75, 1]
#grid_param2 = ['l1']
#param_grid = {'C': grid_param1,
#              'penalty': grid_param2}

grillaRL = GridSearchCV(RL, param_grid, cv = 10,
   return_train_score=True,
                        scoring='roc_auc')

grillaRL.fit(X_train, y_train)


import pandas as pd
```

```
results_df = pd.DataFrame(grillaRL.cv_results_)
results_df = results_df.sort_values(by=["rank_test_score"])

results_RL = results_df[["param_C", "param_penalty",
                         "mean_test_score", "std_test_score"
                            ]]

results_RL = results_RL.sort_values("mean_test_score")


print("Best␣parameters:", grillaRL.best_params_)
print("Best␣score:", grillaRL.best_score_)




# B squeda por grilla RF

param_grid_RF = [
    {"n_estimators": [50, 75, 100, 200, 500],
     "max_features": np.arange(2,14,2).tolist(),
     "max_depth": np.arange(1,12,2)}
]



grillaRF = GridSearchCV(RF, param_grid_RF, cv = 10)

grillaRF.fit(X_train, y_train)

###


results_df2 = pd.DataFrame(grillaRF.cv_results_)
results_df2 = results_df2.sort_values(by=["rank_test_score"])

results_RF = results_df2[["param_n_estimators", "
   param_max_features",
                         "param_max_depth",
                         "mean_test_score", "std_test_score"
                            ]]
results_RF = results_RF.sort_values("mean_test_score")


print("Best␣parameters:", grillaRF.best_params_)
print("Best␣score:", grillaRF.best_score_)
```

```
# B squeda por grilla SVC


param_grid_SVC = [
    {"C": [0.0001, 0.001, 0.01, 0.1, 0.25, 0.5,
                              0.75, 1],
     "kernel": ['linear']},
    {"C": [0.0001, 0.001, 0.01, 0.1, 0.25, 0.5,
                              0.75, 1],
     "kernel": ["rbf"],
     "gamma": [0.01, 0.1,0.25,0.5,0.75,1]}
]


grillaSVC = GridSearchCV(SVM, param_grid_SVC, cv = 10)

grillaSVC.fit(X_train, y_train)

##

results_df3 = pd.DataFrame(grillaSVC.cv_results_)
results_df3 = results_df3.sort_values(by=["rank_test_score"])


results_SVC = results_df3[["param_C", "param_kernel",
                           "param_gamma",
                           "mean_test_score", "std_test_score"
                              ]]


results_SVC = results_SVC.sort_values("mean_test_score")

print("Best␣parameters:", grillaSVC.best_params_)
print("Best␣score:", grillaSVC.best_score_)

## Scores Entrenamiento

# Grilla RL

# Agrupar los datos por param_penalty
groups = results_RL.groupby('param_penalty')

# Crear un gr fico de l neas para cada grupo
for name, group in groups:
    plt.plot(group['param_C'], group['mean_test_score'],
```

```
            label=name, marker = 'o')

# Establecer las etiquetas de los ejes y la leyenda
plt.xlabel('param_C')
plt.ylabel('mean_test_score')
plt.xlim([0,0.1])
plt.legend()

# Mostrar el gr fico
plt.show()

# Grilla RF

rf_grilla = results_RF.groupby(['param_n_estimators', '
   param_max_depth']).mean()
rf_grilla.reset_index(inplace = True)

# Agrupar los datos por param_n_estimators
groups = rf_grilla.groupby('param_n_estimators')

# Crear un gr fico de l neas para cada grupo
for name, group in groups:
    plt.plot(group['param_max_depth'], group['mean_test_score
       '],
             label=name, marker = 'o')

# Establecer las etiquetas de los ejes y la leyenda
plt.xlabel('param_max_depth')
plt.ylabel('mean_test_score')
plt.legend()

# Mostrar el gr fico
plt.show()

# Grilla SVC

svc_grilla = results_SVC.groupby(['param_C', 'param_kernel'])
   .mean()
svc_grilla.reset_index(inplace = True)

# Agrupar los datos por param_kernel
groups = svc_grilla.groupby('param_kernel')

# Crear un gr fico de l neas para cada grupo
for name, group in groups:
    plt.plot(group['param_C'], group['mean_test_score'],
             label=name, marker = 'o')
```

```
# Establecer las etiquetas de los ejes y la leyenda
plt.xlabel('param_C')
plt.ylabel('mean_test_score')
plt.legend()

# Mostrar el gr fico
plt.show()


# Modelos candidatos tras b squeda por grilla

SVC_e = SVC(random_state = 12,
            C = 0.1,
            kernel = 'rbf',
            gamma = 1,
            probability=True)
SVC_e.fit(X_train, y_train)
RF_e = RandomForestClassifier(random_state=12, n_jobs = -1,
                              max_features = 2,
                              n_estimators = 200,
                              max_depth = 1)
RF_e.fit(X_train, y_train)
RL_e = LogisticRegression(random_state=12, n_jobs = -1, C =
   0.001,
                          penalty = 'l2', solver = 'saga',
                             max_iter = 5000)
RL_e.fit(X_train, y_train)

## Construcci n Modelo Regresi n Log stica para el Resumen

import statsmodels.api as sm
import numpy as np


model_RL = sm.GLM(y_train, X_train, family=sm.families.
   Binomial())
result = model_RL.fit()
summary = result.summary2()
# obtener la tabla de coeficientes
coef_table = summary.tables[1]
# calcular los OR y agregarlos a la tabla de coeficientes
coef_table['OR'] = np.exp(coef_table['Coef.'])
# agregar los intervalos de confianza de los OR
coef_table[['[0.025', '0.975]']] = np.exp(coef_table[['[0.025
   ', '0.975]']])
print(coef_table)
```

```
##########################################################


## Curvas de aprendizaje

import numpy as np
from sklearn.model_selection import learning_curve


def plot_learning_curve(estimator, title, X, y, ylim=None, cv
   =None,
                        n_jobs=None, train_sizes=np.linspace
                           (.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training␣examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=
           train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean -
       train_scores_std,
                     train_scores_mean + train_scores_std,
                        alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean -
       test_scores_std,
                     test_scores_mean + test_scores_std,
                        alpha=0.1,
                     color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training␣score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation␣score")
    plt.legend(loc="best")
    return plt

title = "Learning␣Curves␣(Logistic␣Regression)"
```

```
cv = 10
plot_learning_curve(RL_e, title, X_train, y_train, cv=cv)
plt.show()

title = "Learning␣Curves␣(Random␣Forest)"
cv = 10
plot_learning_curve(RF_e, title, X_train, y_train, cv=cv)
plt.show()

title = "Learning␣Curves␣(Support␣Vectorial␣Classifier)"
cv = 10
plot_learning_curve(SVC_e, title, X_train, y_train, cv=cv)
plt.show()

#####################################################

## Construcci n Stacking


modelos_base = [RL_e, RF_e, SVC_e]

meta_modelo = XGBClassifier(n_jobs = -1)

param_grid_XGB = {"n_estimators": [50, 75, 100, 200, 500],
              "learning_rate": [0.001, 0.01, 0.25, 0.75, 1],
              "max_depth": np.arange(1,12,2)}

grillaXGB = GridSearchCV(meta_modelo, param_grid_XGB, cv =
   10)

X_train_meta = []
for model in modelos_base:
    y_pred = model.predict(X_train)
    X_train_meta.append(y_pred)
X_train_meta = np.column_stack(X_train_meta)

grillaXGB.fit(X_train_meta, y_train)

##

results_df4 = pd.DataFrame(grillaXGB.cv_results_)
results_df4 = results_df4.sort_values(by=["rank_test_score"])


results_XGB = results_df4[["param_n_estimators", "
   param_learning_rate",
                           "param_max_depth",
```

```
                            "mean_test_score", "std_test_score
                               "]]


results_XGB = results_XGB.sort_values("mean_test_score")


def get_stacking_model():
    level0 = [('rf', RF_e),
              ('svc', SVC_e),
              ('lr', RL_e)]
    level1 = XGBClassifier(n_estimators=50, learning_rate
       =0.001, max_depth=1)
    model = StackingClassifier(estimators=level0,
       final_estimator=level1)
    return model

stack_1 = get_stacking_model()
stack_1.fit(X_train, y_train)


title = "Learning␣Curves␣(Stack␣Model)"

plot_learning_curve(stack_1, title, X_train, y_train, cv=cv)
plt.show()

#########################################################


pred_RL2 = RL_e.predict_proba(X_test)[:,1]
pred_RF2 = RF_e.predict_proba(X_test)[:,1]
pred_SVM2 = SVC_e.predict_proba(X_test)[:,1]
pred_Stack2 = stack_1.predict_proba(X_test)[:,1]

#########################################################33



def plot_roc_auc(models, X, y):
    plt.figure(figsize=(8,8))
    for name, model in models.items():
        fpr, tpr, _ = roc_curve(y, model.predict_proba(X)
           [:,1])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label='{}␣(AUC␣=␣{:.5f})'.format(
           name, roc_auc))
    plt.xlim([-0.05, 1.0])
```

```
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc='lower right', fontsize = 15)
    plt.show()

modelos_entrenados = {'RF_v': RF_e,
           'SVM_v': SVC_e,
           'RL_v': RL_e,
           'Stack_v': stack_1}


plot_roc_auc(modelos_entrenados, X_test, y_test)




pred_RL = RL_e.predict(X_test)
pred_RF = RF_e.predict(X_test)
pred_SVM = SVC_e.predict(X_test)
pred_Stack = stack_1.predict(X_test)

from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, pred_RL2)



cmRF = confusion_matrix(y_test, pred_RF)
cmRL = confusion_matrix(y_test, pred_RL)
cmSVC = confusion_matrix(y_test, pred_SVM)
cmST = confusion_matrix(y_test, pred_Stack)

labels = ['Dengue Sin Signos',
          'Dengue Con Signos']

# Matriz de confusi n como una imagen

## RF

plt.imshow(cmRF, cmap=plt.cm.Blues)
plt.xticks(ticks=[0, 1], labels=labels)
plt.yticks(ticks=[0, 1], labels=labels)
plt.xlabel('Predicci n')
plt.ylabel('Valor real')
plt.title('Matriz de confusi n Random Forest')
for i in range(cmRF.shape[0]):
```

```
    for j in range(cmRF.shape[1]):
        plt.text(j, i, str(cmRF[i, j]), ha='center', va='
           center',
                 color = 'black', fontsize = 18)
plt.colorbar()
plt.show()

## RL

plt.imshow(cmRL, cmap=plt.cm.Blues)
plt.xticks(ticks=[0, 1], labels=labels)
plt.yticks(ticks=[0, 1], labels=labels)
plt.xlabel('Predicci n')
plt.ylabel('Valor␣real')
plt.title('Matriz␣de␣confusi n␣Regresi n␣Log stica')
for i in range(cmRL.shape[0]):
    for j in range(cmRL.shape[1]):
        plt.text(j, i, str(cmRL[i, j]), ha='center', va='
           center', fontsize = 18)
plt.colorbar()
plt.show()


## SVC

plt.imshow(cmSVC, cmap=plt.cm.Blues)
plt.xticks(ticks=[0, 1], labels=labels)
plt.yticks(ticks=[0, 1], labels=labels)
plt.xlabel('Predicci n')
plt.ylabel('Valor␣real')
plt.title('Matriz␣de␣confusi n␣SVC')
for i in range(cmSVC.shape[0]):
    for j in range(cmSVC.shape[1]):
        plt.text(j, i, str(cmSVC[i, j]), ha='center', va='
           center', fontsize = 18)
plt.colorbar()
plt.show()


## Stack

plt.imshow(cmST, cmap=plt.cm.Blues)
plt.xticks(ticks=[0, 1], labels=labels)
plt.yticks(ticks=[0, 1], labels=labels)
plt.xlabel('Predicci n')
plt.ylabel('Valor␣real')
plt.title('Matriz␣de␣confusi n␣Ensamble')
```

```
for i in range(cmST.shape[0]):
    for j in range(cmST.shape[1]):
        plt.text(j, i, str(cmST[i, j]), ha='center', va='
           center', fontsize = 18)
plt.colorbar()
plt.show()


######################

tn_rl, fp_rl, fn_rl, tp_rl = confusion_matrix(y_test, pred_RL
   ).ravel()

# Calcular la especificidad
specificity_rl = tn_rl / (tn_rl + fp_rl)

# Calcular la sensibilidad
sensitivity_rl = tp_rl / (tp_rl + fn_rl)


# Calcular Accuracy

accuracy_rl = (tp_rl+tn_rl)/(tn_rl+ fp_rl+ fn_rl+tp_rl)

print("Especificidad:", round(specificity_rl,5))
print("Sensibilidad:", round(sensitivity_rl,5))
print("Accuracy:", round(accuracy_rl,5))
print("F1-Score", round(f1_score(y_test,pred_RL),5))

########################################

tn_rf, fp_rf, fn_rf, tp_rf = confusion_matrix(y_test, pred_RF
   ).ravel()

# Calcular la especificidad
specificity_rf = tn_rf / (tn_rf + fp_rf)

# Calcular la sensibilidad
sensitivity_rf = tp_rf / (tp_rf + fn_rf)


# Calcular Accuracy

accuracy_rf = (tp_rf+tn_rf)/(tn_rf+ fp_rf+ fn_rf+tp_rf)

print("Especificidad:", round(specificity_rf,5))
print("Sensibilidad:", round(sensitivity_rf,5))
```

```
print("Accuracy:", round(accuracy_rf,5))
print("F1-Score", round(f1_score(y_test,pred_RF),5))

########################################

tn_svm, fp_svm, fn_svm, tp_svm = confusion_matrix(y_test,
   pred_SVM).ravel()

# Calcular la especificidad
specificity_svm = tn_svm / (tn_svm + fp_svm)

# Calcular la sensibilidad
sensitivity_svm = tp_svm / (tp_svm + fn_svm)

# Calcular Accuracy

accuracy_svm = (tp_svm+tn_svm)/(tn_svm+ fp_svm+ fn_svm+tp_svm
   )

print("Especificidad:", round(specificity_svm,5))
print("Sensibilidad:", round(sensitivity_svm,5))
print("Accuracy:", round(accuracy_svm,5))
print("F1-Score", round(f1_score(y_test,pred_SVM),5))

########################################

tn_st, fp_st, fn_st, tp_st = confusion_matrix(y_test,
   pred_Stack).ravel()

# Calcular la especificidad
specificity_st = tn_st / (tn_st + fp_st)

# Calcular la sensibilidad
sensitivity_st = tp_st / (tp_st + fn_st)

# Calcular Accuracy

accuracy_st = (tp_st+tn_st)/(tn_st+ fp_st+ fn_st+tp_st)

print("Especificidad:", round(specificity_st,5))
print("Sensibilidad:", round(sensitivity_st,5))
print("Accuracy:", round(accuracy_st,5))
print("F1-Score", round(f1_score(y_test,pred_Stack),5))

################################################
### Metricas entrenamiento
```

```
pred_RLe = RL_e.predict(X_train)
pred_RFe = RF_e.predict(X_train)
pred_SVMe = SVC_e.predict(X_train)
pred_Stacke = stack_1.predict(X_train)


tn_rle, fp_rle, fn_rle, tp_rle = confusion_matrix(y_train,
   pred_RLe).ravel()

# Calcular la especificidad
specificity_rle = tn_rle / (tn_rle + fp_rle)

# Calcular la sensibilidad
sensitivity_rle = tp_rle / (tp_rle + fn_rle)

# Calcular Accuracy

accuracy_rle = (tp_rle+tn_rle)/(tn_rle+ fp_rle+ fn_rle+tp_rle
   )

print("Especificidad:", round(specificity_rle,5))
print("Sensibilidad:", round(sensitivity_rle,5))
print("Accuracy:", round(accuracy_rle,5))

########################################

tn_rfe, fp_rfe, fn_rfe, tp_rfe = confusion_matrix(y_train,
   pred_RFe).ravel()

# Calcular la especificidad
specificity_rfe = tn_rfe / (tn_rfe + fp_rfe)

# Calcular la sensibilidad
sensitivity_rfe = tp_rfe / (tp_rfe + fn_rfe)

# Calcular Accuracy

accuracy_rfe = (tp_rfe+tn_rfe)/(tn_rfe+ fp_rfe+ fn_rfe+tp_rfe
   )

print("Especificidad:", round(specificity_rfe,5))
print("Sensibilidad:", round(sensitivity_rfe,5))
print("Accuracy:", round(accuracy_rfe,5))

########################################

tn_svme, fp_svme, fn_svme, tp_svme = confusion_matrix(y_train
```

```
   , pred_SVMe).ravel()

# Calcular la especificidad
specificity_svme = tn_svme / (tn_svme + fp_svme)

# Calcular la sensibilidad
sensitivity_svme = tp_svme / (tp_svme + fn_svme)

# Calcular Accuracy

accuracy_svme = (tp_svme+tn_svme)/(tn_svme+ fp_svme+ fn_svme+
   tp_svme)

print("Especificidad:", round(specificity_svme,5))
print("Sensibilidad:", round(sensitivity_svme,5))
print("Accuracy:", round(accuracy_svme,5))


########################################

tn_ste, fp_ste, fn_ste, tp_ste = confusion_matrix(y_train,
   pred_Stacke).ravel()

# Calcular la especificidad
specificity_ste = tn_ste / (tn_ste + fp_ste)

# Calcular la sensibilidad
sensitivity_ste = tp_ste / (tp_ste + fn_ste)

# Calcular Accuracy

accuracy_ste = (tp_ste+tn_ste)/(tn_ste+ fp_ste+ fn_ste+tp_ste
   )

print("Especificidad:", round(specificity_ste,5))
print("Sensibilidad:", round(sensitivity_ste,5))
print("Accuracy:", round(accuracy_ste,5))


#####################################

modelos_finales = ['Reg._Log', 'Random_Forest', 'SVC', '
   Stacking']
precision_entrenamiento = [accuracy_rle, accuracy_rfe,
   accuracy_svme, accuracy_ste]
precision_validacion = [accuracy_rl, accuracy_rf,
   accuracy_svm, accuracy_st]
```

```
ancho_barra = 0.45
barras_entrenamiento = plt.bar(np.arange(len(modelos_finales)
   ), precision_entrenamiento,
                               ancho_barra, label='
                                  Entrenamiento', color = '
                                  red')
barras_validacion = plt.bar(np.arange(len(modelos_finales)) +
    ancho_barra,
                            precision_validacion, ancho_barra
                               , label='Validaci n',
                            color = 'green')
plt.ylim([0.7, 1.05])
plt.xlabel('Modelos')
plt.ylabel('Precisi n')
plt.title('Comparaci n de precisi n de entrenamiento y
   validaci n')
plt.xticks(np.arange(len(modelos_finales)) + ancho_barra / 2,
    modelos_finales)
plt.legend(loc = "lower right")
for i, v in enumerate(precision_entrenamiento):
    plt.text(i - ancho_barra / 2, v + 0.01, str(round(v,3)),
       fontsize=10)

for i, v in enumerate(precision_validacion):
    plt.text(i + ancho_barra / 2, v + 0.01, str(round(v,3)),
       fontsize=10)

plt.show()

########################################
```