Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y computación.

Dr. Hernán Camilo Rocha Niño
Decano de la Facultad de Ingeniería

Dr. Gerardo Mauricio Sarria
Director Carrera Ingeniería de Sistemas y Computación

Dra. Gloria Inés Álvarez
Directora Trabajo

Dr. Diego Luis Linares
Director Trabajo

ING. Jorge Finke
Jurado 1

ING. Juan Carlos Martínez
Jurado 2

Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería de sistemas y computación

Fecha: 21 de febrero de 2022

Autores: Katherine Camacho Calderon

Nombre del Proyecto de Grado: Técnicas de Ensamble Aplicadas a un Conjunto de Datos Perteneciente a Pacientes de Leishmaniasis Cutánea para Predecir la Efectividad del Tratamiento Glucantime

Directores: Gloria Inés Álvarez y Diego Luis Linares

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Firma de directora
del Proyecto de Grado

Escriba el texto aquí

Firma de director
del Proyecto de Grado

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Ingeniería de Sistemas y Computación.
Proyecto de Grado.

Técnicas de Ensamble Aplicadas a un Conjunto de Datos Perteneciente a Pacientes de Leishmaniasis Cutánea para Predecir la Efectividad del Tratamiento Glucantime

Katherine Camacho Calderon

Directora: Dra. Gloria Inés Álvarez
Codirector: Dr. Diego Luis Linares

23-11-2021

Santiago de Cali, 23-11-2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente nos permitimos informarle que la estudiante de Ingeniería de Sistemas y Computación Katherine Camacho Calderon (cod: 8934985) trabaja bajo nuestra dirección en el proyecto de grado titulado "Técnicas de Ensamble Aplicadas a un Conjunto de Datos Perteneciente a Pacientes de Leishmaniasis Cutánea para Predecir la Efectividad del Tratamiento Glucantime".

Atentamente,

Dra. Gloria Inés Álvarez

Dr. Diego Luis Linares

Santiago de Cali, 23-11-2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el anteproyecto de grado titulado "Técnicas de Ensamble Aplicadas a un Conjunto de Datos Perteneciente a Pacientes de Leishmaniasis Cutánea para Predecir la Efectividad del Tratamiento Glucantime" con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

Katherine Camacho

Katherine Camacho Calderon
Código: 8934985

Resumen

La leishmaniasis es una enfermedad re-emergente en Colombia. Catalogada por la Organización Mundial de la Salud como "enfermedad olvidada", debido a que está asociada directamente con las personas que viven en regiones muy pobres. Por lo tanto, existe poco interés en invertir en el desarrollo de medicamentos y estrategias para controlar esta enfermedad.

Con este proyecto se obtuvieron cuatro modelos de aprendizaje automático con el fin de que permitan predecir cuándo el tratamiento glucantime contra la leishmaniasis cutánea funcionará para un paciente y cuándo no. Inicialmente se realizó un proceso de preparación de los datos, y luego con ayuda de las técnicas de ensamble se construyeron dichos modelos.

Se realizaron evaluaciones de los distintos modelos construidos y se permitió inferir que ningún modelo presentó un desempeño que permita confiar en sus predicciones. Sin embargo, la técnica que permitió obtener el modelo con el mejor desempeño indica que probablemente con una cantidad un poco mayor de datos se puedan obtener mejores predicciones.

CIDEIM (Centro Internacional de Entrenamiento e Investigaciones Médicas) fue el ente que proporcionó el conjunto de datos con el cual se construyeron los modelos.

Palabras Clave: Leishmaniasis cutánea, CIDEIM, técnicas de ensamble, aprendizaje automático.

Índice general

Introducción

La leishmaniasis cutánea es una enfermedad infecciosa que causa cicatrices de por vida y discapacidad grave. La Organización Mundial de la Salud(OMS) estima que aproximadamente tres millones de personas sufren de esta enfermedad y 350 millones viven en zonas de riesgo.[1]

En Colombia, la leishmaniasis cutánea se presenta en forma endémica en focos localizados en áreas rurales, muchos de ellos alejados de los centros de atención de salud de las zonas urbanas[1]. Dichos focos se encuentran en una gran diversidad de regiones naturales que incluyen zonas selváticas de la Costa Pacífica y del Amazonas, áreas de bosque seco tropical como la Costa Caribe, la región andina, áreas de llanos y desiertos localizadas en región interandina, oriente del país y península de la Guajira. Además, se ha demostrado que en Colombia existe la presencia de seis especies de Leishmania, esta diversidad de especies hace más compleja la caracterización eco epidemiológica y el control de esta enfermedad en el país.

La falta de efectividad del tratamiento y el poco interés hacia el estudio de esta enfermedad conlleva a que el personal de la salud de las zonas endémicas tenga muy poca experiencia sobre los diagnósticos.

Este proyecto tiene como objetivo construir un modelo de aprendizaje automático que permita predecir cuándo un tratamiento de leishmaniasis cutánea para un paciente funcionará y cuándo no, dada ciertas características socio-demográficas que serán obtenidas a partir del conjunto de datos.

El conjunto de datos que se obtendrá para el estudio sobre la falla terapéutica de la leishmaniasis cutánea será dado por CIDEIM, el cual es un centro de investigación, desarrollo tecnológico y formación de recurso humano en el campo de la salud. Este conjunto de datos contiene muy pocos registros y gran cantidad de atributos, por lo que el reto será seleccionar los atributos y las técnicas adecuadas para poder suministrar una predicción útil. De tal forma que la predicción obtenida permita a los terapeutas tomar decisiones sobre el tratamiento a seguir teniendo una fuente más de información.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

La leishmaniasis es una enfermedad de tipo infeccioso que es causada por un parásito protozoo del género *leishmania*, y se transmite por una picadura de un mosquito infectado de la familia flebótomos. Este mosquito se localiza en zonas rurales con clima mediterráneo, subtropical y tropical[2].

Esta enfermedad se manifiesta desde lesiones cutáneas en la zona de la picadura del insecto hasta daños viscerales más graves, que si no se tratan a tiempo pueden llegar a causar la muerte[2].

Actualmente esta enfermedad ha adquirido gran importancia como enfermedad re-emergente en el mundo y en Colombia, debido al aumento de casos. La Organización Mundial de la Salud declaró que esta enfermedad tiene el carácter de "enfermedad olvidada", ya que con frecuencia afecta a las regiones más pobres del planeta, debido a que está asociada a la malnutrición, desplazamientos de población, malas condiciones de vivienda, falta de recursos[3]. Razón por la cual hay muy poco interés en invertir en el desarrollo de medicamentos y estrategias para controlar esta enfermedad.

CIDEIM es un centro de investigación, desarrollo tecnológico y formación de recurso humano en el campo de la salud. Este centro busca disminuir el impacto negativo y los costos de las enfermedades infecciosas[4]. Una de las enfermedades que se estudia en CIDEIM es la leishmaniasis cutánea, la investigación está dirigida a conocer y mitigar los factores que determinan el desarrollo de la enfermedad y la falla terapéutica[5].

Uno de los objetivos que tiene CIDEIM es identificar terapias alternativas y regímenes terapéuticos más seguros y de administración ajustada a los individuos afectados[5].Glucantime que es uno de los tratamientos para la leishmaniasis cutánea que estudia este centro de investigación, no cumple con los requisitos de efectividad, existen casos donde el tratamiento funciona y otros donde no. Es aquí donde las herramientas computacionales juegan un papel importante. Los algoritmos de aprendizaje automático, que son aquellos algoritmos que permiten establecer o detectar patrones para realizar predicciones o clasificar información, han permitido predecir con cierta probabilidad, cuándo el tratamiento funciona y cuándo no, utilizando un conjunto de datos con información demográfica de diferentes pacientes.

El conjunto de datos con el cuál se ha trabajado cuenta con muy pocos registros, por lo cual se utilizó las técnicas de ensamble, las cuales se comportan bien cuando los modelos no son muy robustos

y el conjunto de datos no es muy grande.

1.1.1. Formulación

¿Cómo predecir la respuesta de un paciente al tratamiento de glucantime a partir de técnicas de ensamble utilizando un conjunto de datos pequeño?

1.1.2. Sistematización

- ¿Cómo representar la información para facilitar el buen desempeño de las técnicas que harán la predicción?
- ¿Cómo estimar los parámetros de las técnicas a usar, para lograr modelos con el mejor desempeño posible?
- ¿Cómo construir los modelos usando las técnicas seleccionadas?
- ¿Cómo evaluar los modelos obtenidos a partir de las diferentes técnicas de ensamble para identificar cuál es el modelo que permite hacer la mejor predicción?

1.2. Objetivos

1.2.1. Objetivo General

Predecir la respuesta de un paciente al tratamiento glucantime a partir de técnicas de ensamble utilizando un conjunto de datos pequeño.

1.2.2. Objetivos Específicos

- Seleccionar la representación de la información para facilitar el buen desempeño de las técnicas que harán la predicción.
- Estimar los parámetros de las técnicas a usar, para lograr modelos con el mejor desempeño posible.
- Construir los modelos usando las técnicas seleccionadas.
- Evaluar los modelos obtenidos a partir de las diferentes técnicas de ensamble para identificar cuál es el modelo que permite hacer la mejor predicción.

Capítulo 2

Marco de referencia

2.1. Áreas Temáticas

Este proyecto se enfoca en las siguientes definiciones propias de las disciplinas como punto de partida en la contextualización:

- Modelos de aprendizaje automático.
- Técnicas de ensamble.
- Técnica de bagging.
- Técnica de boosting.
- Técnica de stacking.
- Leishmaniasis cutánea.

2.2. Marco Teórico

- **Modelos de aprendizaje automático:** Son la salida de información que se genera cuando se entrena un algoritmo de aprendizaje automático con datos. Este modelo permite reconocer determinados tipos de patrones[8]. Con un modelo entrenado se pueden empezar a realizar predicciones sobre estos.
- **Técnicas de ensamble:** Son técnicas que permiten la construcción de modelos a través de múltiples algoritmos de aprendizaje. Cada algoritmo produce una predicción distinta, dichas predicciones se combinan para obtener una única predicción[9]. La ventaja que se obtiene al combinar modelos diferentes es que cada modelo funciona de forma diferente, lo que permite entonces que sus errores tiendan a compensarse dado que estos modelos se protegen mutuamente de sus errores individuales. Algunas de las técnicas de ensamble son:

  - **Técnica de bagging:** Es una técnica que combina varios modelos de machine learning[9]. La forma de conseguir que los errores se compensen entre sí, es que cada modelo se entrena por separado con subconjuntos del conjunto de entrenamiento. Estos subconjuntos se forman eligiendo muestras aleatorias con repetición del conjunto de entrenamiento que están definidas a su vez por un subconjunto aleatorio de atributos. Los resultados se combinan. Para los

problemas de clasificación lo que se hace es usar el voto suave, que consiste en darle más importancia a los resultados en los que algún modelo esté muy seguro, esto para los modelos que den probabilidades. Para los problemas de regresión, se utiliza la media aritmética.

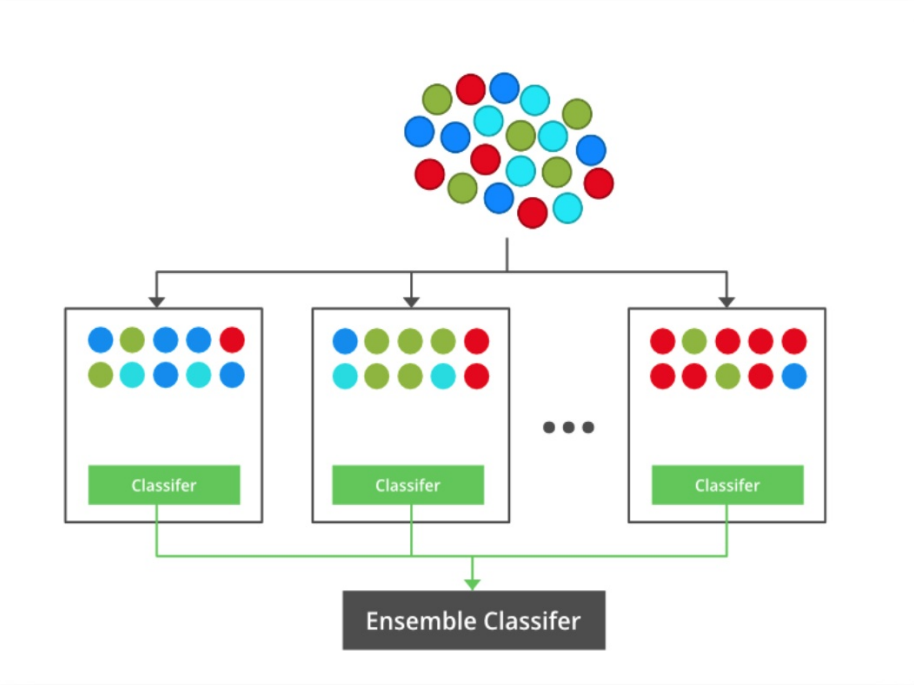

Figura 2.1: Técnica de bagging[10]

- **Técnica de boosting**: En esta técnica la idea es que cada modelo intente arreglar los errores de los modelos anteriores[9]. En el caso de clasificación, el primer modelo una vez entrenado seguramente cometerá algunos errores, así que el segundo modelo intentará reducir dichos errores. De tal forma, que de forma secuencial se combinen los algoritmos para mejorar la predicción final. Para los problemas de regresión, las predicciones con un mayor error cuadrático medio es el que tendrá más peso para el siguiente modelo. Los algoritmos simples que utiliza boosting se entrenan con todas las muestras del conjunto de entrenamiento definidas a su vez por un subconjunto de atributos. Los siguientes son algoritmos de boosting conocidos por tener éxito en problemas de clasificación binaria:

Gradient Boosting Classifier:

Un modelo Gradient Boosting utiliza el algoritmo de árboles de decisión para construir los estimadores base. Es decir, utiliza árboles de decisión de forma secuencial para producir predicciones precisas. Para determinar los parámetros que usará cada árbol de decisión se usa el

algoritmo de descenso de gradiente que a través de derivadas tiene como objetivo minimizar la función de pérdida. Es así que se agregan árboles con distintos parámetros, que buscan que los modelos subsiguientes sean agregados y corrijan los errores, con la intención de que al final la combinación de dichos árboles pueda minimizar la pérdida del modelo.

AdaBoost Classifier:

Un modelo de AdaBoost Classifier funciona eligiendo un algoritmo base, generalmente se usan árboles de decisión y mejorando las predicciones al aplicar el estimador base n veces al conjunto de entrenamiento, de tal forma que en cada iteración tome en cuenta aquellos casos que dicho algoritmo base ha clasificado incorrectamente, aumentando los pesos de los ejemplos mal clasificados.

Figura 2.2: Técnica de boosting[10]

- **Técnica de stacking**: La idea de esta técnica es apilar modelos, cuando se apilan modelos, lo que en realidad se está haciendo es usar la salida de varios modelos como la entrada de otros modelos[9]. Se comienza entrenando varios aprendices o algoritmos base sobre los datos de entrenamiento, y se termina entrenando el modelo final sobre los datos originales considerando como características adicionales las predicciones de los primeros. Esta técnica usa validación

cruzada para entrenar y evaluar cada estimador base, con el fin de evitar un sobreajuste de estos. Es así como cada estimador base se entrena con un subcojunto de datos definidos a su vez por un subconjunto de atributos.

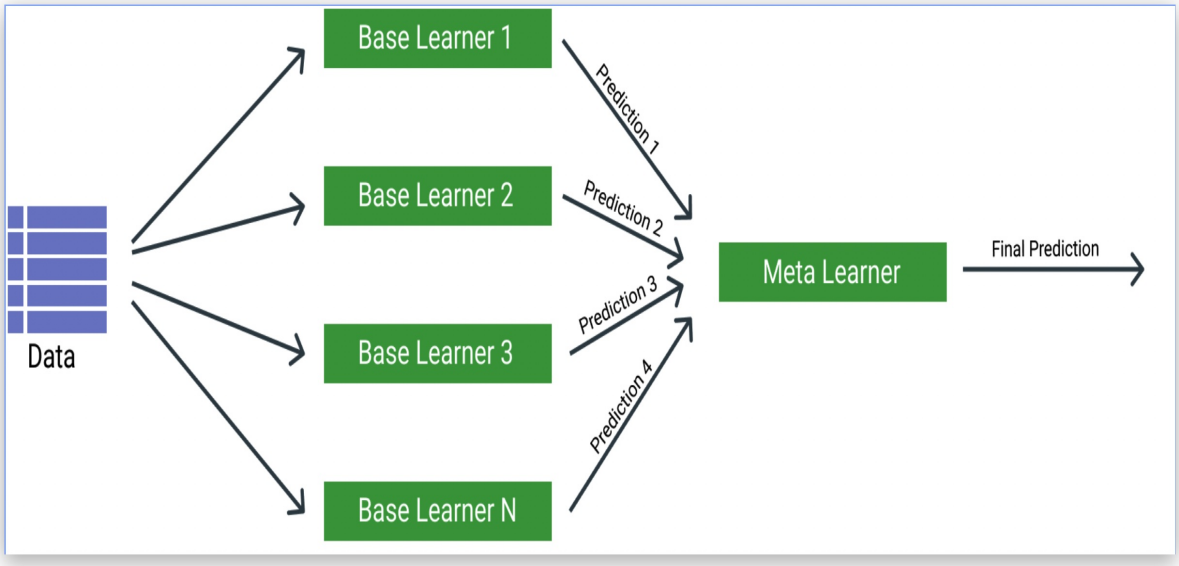

Figura 2.3: Técnica de stacking[11]

- **Leishmaniasis cutánea:**

Es una enfermedad cutánea causada por un parásito del género leishmania. Los síntomas incluyen lesiones en la piel, que se desarrollan durante varias semanas o meses después de la exposición. Estas lesiones pueden llegar a ser muy grandes y dar lugar a llagas con un borde elevado, que puede estar cubierto de escamas o de una costra.[12]

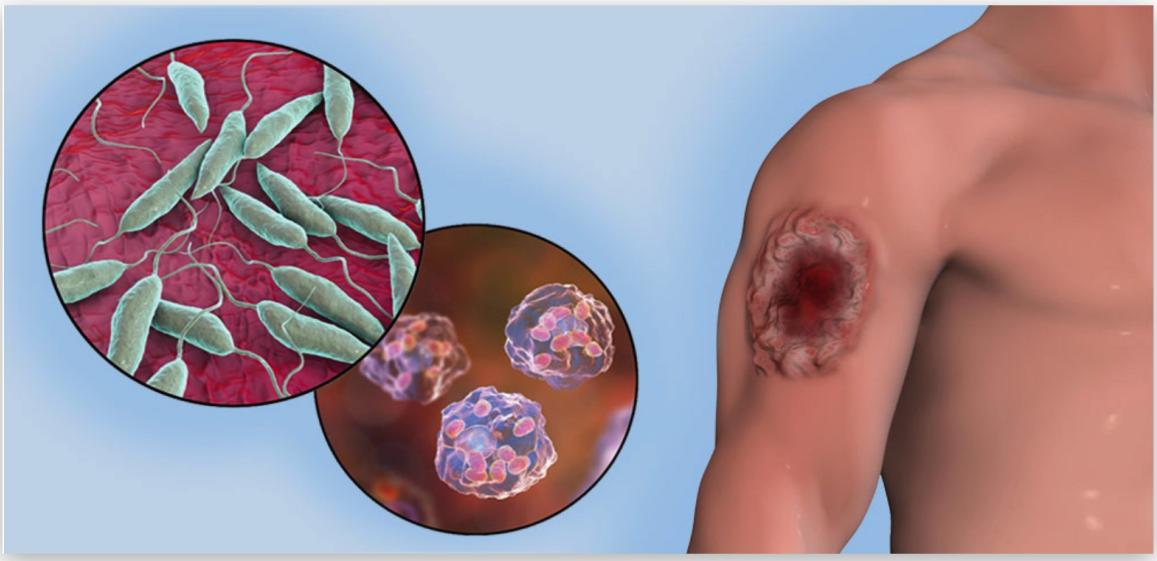

Figura 2.4: Leishmaniasis cutánea[13]

2.3. Trabajos Relacionados

En el artículo **Identificación de nuevos compuestos con potencial actividad antileishmaniásica mediante estudios in silico** [14] se expone que la quimioterapia empleada para tratar la leishmaniasis presenta una alta frecuencia de efectos citotóxinos serios, reacciones adversas e incidencia creciente a la resistencia, por lo que se hace necesaria la identificación rápida de nuevas alternativas terapéuticas. Los resultados de esta investigación permitieron demostrar que a través de experimentos y análisis realizados mediante algoritmos de simulación y predicción computacional es posible disminuir los altos costos de procesos de síntesis y bioensayos, al mismo tiempo que identificaron compuestos efectivos contra diversas fallas terapéuticas.

Durante el proceso se utilizaon técnicas de inteligencia artificial empleando software como WEKA que permitió el aprendizaje automático y la minería de datos, ChemAxon que permitió la realización de tareas quimioinformáticas y JChem que permitió el manejo de estructuras y capacidades de visualización de dentro de un entorno de Microsoft Office. Se usó la base de datos DrugBank la cual es una fuente única tanto bioinformática como quimioinformática, y que proporciona datos sobre fármacos.

Se obtuvieron ocho modelos con las técnicas k vecinos más cercanos, árboles de clasificación, perceptrón multicapa y máquina vectorial de soporte, las cuales alcanzaron porcentajes de clasificación entre 80 y 92 % , el valor de la predicción se constató a través de procedimientos de validación externa. También se usaron técnicas de ensamble en la predicción de compuestos y mostraron mejores resultados de exactitud 91,33 y 96,77 %, permitiendo la identificación de forma óptima de 8

compuestos con potencial actividad contra Leishmania amazonensis.

En el artículo **Empleo de técnicas de inteligencia artificial en la identificación de nuevos compuestos con potencial actividad contra Leishmania major**[15] se expone la idea de que el tratamiento de la leishmaniasis no es sencillo, está limitado a unos cuantos medicamentos que no siempre son eficaces y se asocian a importantes efectos adversos. Además expone que en la actualidad no existen fármacos totalmente efectivos para el tratamiento de las formas de leishmaniasis causadas por Leishmania major por lo que entonces se plantea como objetivo la identificación de nuevos compuestos con potencial actividad contra este tipo de Leishmania mediante métodos QSAR(Relaciones Cuantitativas Estructura-Actividad/Propiedad) basados en técnicas de inteligencia artifical.

Durante la investigación las técnicas usadas fueron: Técnicas k vecinos más cercanos, árboles de clasificación, redes neuronales y máquina vectorial de soporte, las cuales alcanzaron porcentajes de clasificación entre 87 y 94 %. También se hizo uso de técnicas de ensamble, las cuales en conjunto con las técnicas antes mencionados permitieron la dentificación de 187 nuevos compuestos con potencial actividad contra Leishmania major.

En el artículo **Minería in silico del proteoma de Leishmania aplicada para identificar el potencial objetivo del fármaco que se utilizará para tratar la leishmaniasis visceral y tegumentaria**[16] expone que son necesarias nuevas estrategias terapéuticas contra la leishmaniasis debido a la toxicidad, el alto costo y/o la resistencia parasitaria, por lo cual se hace vital encontrar nuevos compuestos antileishmania. En este estudio se presenta una estrategia que permite seleccionar nuevos fármacos con baja homología con proteínas humanas pero que son relevantes para el parásito. Como resultado se identificó una estrategia de minería de proteomas con ayuda de herramientas bioinformáticas que finalmente mostraron un posible candidato, una proteína que presentó el comportamiento deseado.

Capítulo 3

Descripción y preparación del conjunto de datos

3.1. Descripción del conjunto de datos inicial

El conjunto de datos que entrega CIDEIM presenta información sociodemográfica de pacientes con leishmaniasis cutánea, y algunas características de la lesión causada por esta enfermedad.

Algunos datos a tener en cuenta, son los siguientes:

1. Contiene 25 registros y 18 atributos.
2. Los atributos son los siguientes:
   - **EB_LC_SEXO:** Indica el sexo del paciente.
   - **EB_LC_ETNIA:** Indica la etnia del paciente.
   - **EDAD:** Indica la edad del paciente.
   - **EB_LC_DPTO_INFECCION:** Indica el departamento donde probablemente el paciente contrajo la infección.
   - **EB_LC_PESO:** Indica el peso en kilogramos del paciente.
   - **EB_LC_ESTATURA:** Indica la estatura en centímetros del paciente.
   - **EB_LC_TTO_MCTO_PRESCRITO:** Indica el tratamiento prescrito (medicamento glucantime/miltefosine).
   - **EB_LC_DX_PREVIO:** Indica si previamente ya el paciente ha sido diagnosticado con leishmaniasis.
   - **EB_LC_NUMERO_LC_ACTIVAS:** Indica el número de lesiones cutáneas activas que tiene el paciente.
   - **EB_LC_TIEMPO_EVOLUCION:** Indica el tiempo de evolución en días de la primera lesión, por la que consulta actualmente.
   - **EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1:** Descripción de la longitud en milímetros del eje horizontal de la úlcera.
   - **EB_LC_ULCERA_EJE_VERTICAL_LESION_1:** Descripción de la longitud en milímetros del eje vertical de la úlcera.

- **EB_ULCERA_AREA_1:**Indica el área total de la úlcera.
- **EB_LC_LESION_EJE_HORIZONTAL_LESION_1:** Descripción de la longitud en milímetros del eje horizontal de la lesión.
- **EB_LC_LESION_EJE_VERTICAL_LESION_1:** Descripción de la longitud en milímetros del eje vertical de la lesión.
- **EB_LESION_AREA_1:** Indica el área total de la lesión.
- **EB_LC_TTO_MCTO_GLUCANTIME_DOSIS:**Indica la dosis diaria de glucantime en mililitros.
- **EF_LC_ESTADO_FINAL_ESTUDIO:**Estado final del paciente, si curó o no lo hizo.

3. La variable objetivo es **EF_LC_ESTADO_FINAL_ESTUDIO**.

4. Los tipos de datos que contiene el conjunto son:

Tipo de dato	**Descripción**
Cadena	Indica que los valores de las columnas son cadenas de texto.
Entero	Indica que los valores de las columnas son números enteros.
Decimal	Indica que los valores de las columnas son números decimales.

Cuadro 3.1: Tipo de datos que contiene el cojunto

5. El tipo de dato de cada atributo es:

Nombre del atributo	Tipo de dato
EB_LC_SEXO	Entero
EB_LC_ETNIA	Entero
EDAD	Entero
EB_LC_DPTO_INFECCION	Cadena
EB_LC_PESO	Decimal
EB_LC_ESTATURA	Entero
EB_LC_TTO_MCTO_PRESCRITO	Entero
EB_LC_DX_PREVIO	Entero
EB_LC_NUMERO_LC_ACTIVAS	Entero
EB_LC_TIEMPO_EVOLUCION	Entero
EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1	Decimal
EB_LC_ULCERA_EJE_VERTICAL_LESION_1	Decimal
EB_ULCERA_AREA_1	Decimal
EB_LC_LESION_EJE_HORIZONTAL_LESION_1	Decimal
EB_LC_LESION_EJE_VERTICAL_LESION_1	Decimal
EB_LESION_AREA_1	Decimal
EB_LC_TTO_MCTO_GLUCANTIME_DOSIS	Decimal
EF_LC_ESTADO_FINAL_ESTUDIO	Decimal

Cuadro 3.2: Tipo de dato de cada atributo

6. Los valores que pueden tomar los atributos categóricos son:

Nombre del atributo	Valor 1	Valor 2	Valor 3	Valor 4
EB_LC_SEXO	1=Masculino	2=Femenino		
EB_LC_ETNIA	1=Afrocolombiana	2=Indígena	3=Mestiza	
EB_LC_DPTO_INFECCION	Narino	Choco	Tolima	Cauca
EB_LC_TTO_MCTO_PRESCRITO	1=Glucantime	2=Miltefosine		
EB_LC_DX_PREVIO	0=No	1=Si		
EF_LC_ESTADO_FINAL_ESTUDIO	0=Curación definitiva	1=Falla terapéutica		

Cuadro 3.3: Valores de los atributos categóricos

7. Medidas de centralidad y desviación para atributos númericos:

	Atributo	Mínimo	Máximo	Media	Desviación estándar
0	EDAD	18.0	51.0	30.76000	8.776294
1	EB_LC_PESO	53.9	105.0	73.79200	13.334727
2	EB_LC_ESTATURA	150.0	183.0	166.72000	9.162969
3	EB_LC_NUMERO_LC_ACTIVAS	1.0	8.0	2.44000	1.872610
4	EB_LC_TIEMPO_EVOLUCION	2.0	21.0	6.92000	5.163332
5	EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1	0.0	75.4	20.19600	19.988673
6	EB_LC_ULCERA_EJE_VERTICAL_LESION_1	0.0	56.4	16.85600	15.863666
7	EB_ULCERA_AREA_1	0.0	9638.9	1897.13200	2895.299592
8	EB_LC_LESION_EJE_HORIZONTAL_LESION_1	2.0	87.1	29.27200	24.753443
9	EB_LC_LESION_EJE_VERTICAL_LESION_1	2.0	72.2	26.24400	21.119246
10	EB_LESION_AREA_1	12.6	19756.3	3825.88400	5157.502096
11	EB_LC_TTO_MCTO_GLUCANTIME_DOSIS	13.0	20.0	17.23913	2.670842

Figura 3.1: Medidas de centralidad y desviación para atributos númericos

8. Medidas de centralidad para atributos categóricos:

	Atributo	Moda
0	EB_LC_SEXO	1
1	EB_LC_ETNIA	1
2	EB_LC_DPTO_INFECCION	Narino
3	EB_LC_TTO_MCTO_PRESCRITO	1
4	EB_LC_DX_PREVIO	0
5	EF_LC_ESTADO_FINAL_ESTUDIO	0.0

Figura 3.2: Medidas de centralidad y desviación para atributos categóricos

Análisis:
Dentro del conjunto de datos existe:

*Una mayor cantidad de personas de sexo masculino.

*Una mayor cantidad de personas que pertenecen a la etnia afrocolombiana.

*Una mayor cantidad de personas que contrajeron la enfermedad en el departamento de Narino.

*Una mayor cantidad de personas que reciben como tratamiento contra la leishmaniasis cutánea el medicamento glucantime.

*Dado que se conoce que el valor de todos los registros para el atributo EB_LC_DX_PREVIO es el mismo, entonces se puede decir que el conjunto de datos pertenece a personas que no se les había detectado antes leishmaniasis cutánea.

*Una mayor cantidad de personas que curaron definitivamente después de someterse al tratamiento contra la leishmaniasis cutánea.

9. Columnas altamente correlacionadas:

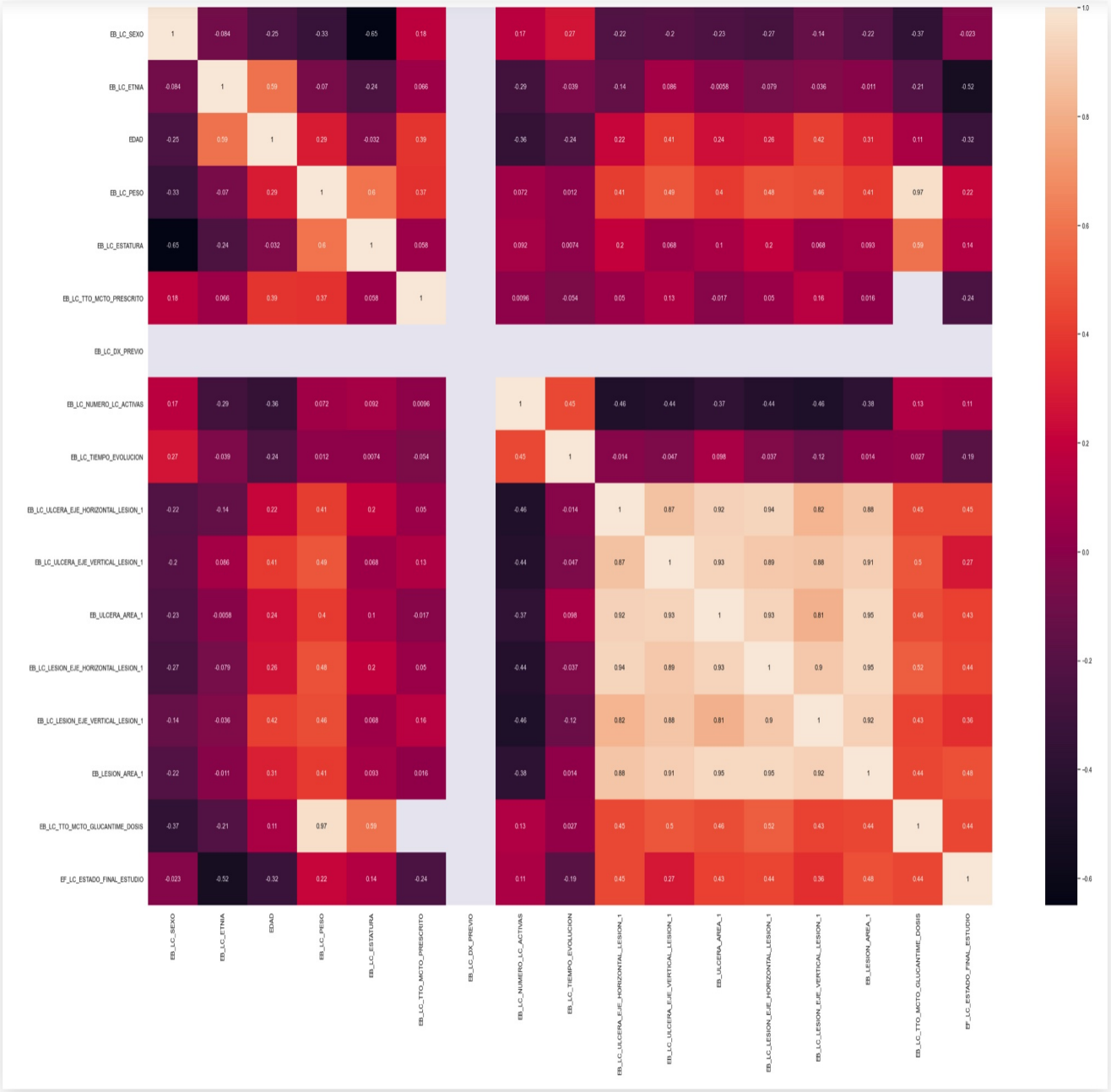

Figura 3.3: Matriz de correlación. Los colores pueden ser utilizados para identificar el nivel de correlación de los atributos en la matriz.

Aquellos atributos que presentaron una correlación mayor o igual a 0.85 son:

	Columna 1	Columna 2	Correlación
0	EB_LC_ULCERA_EJE_VERTICAL_LESION_1	EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1	0.865451
1	EB_ULCERA_AREA_1	EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1	0.924229
2	EB_ULCERA_AREA_1	EB_LC_ULCERA_EJE_VERTICAL_LESION_1	0.929318
3	EB_LC_LESION_EJE_HORIZONTAL_LESION_1	EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1	0.941248
4	EB_LC_LESION_EJE_HORIZONTAL_LESION_1	EB_LC_ULCERA_EJE_VERTICAL_LESION_1	0.893834
5	EB_LC_LESION_EJE_HORIZONTAL_LESION_1	EB_ULCERA_AREA_1	0.934437
6	EB_LC_LESION_EJE_VERTICAL_LESION_1	EB_LC_ULCERA_EJE_VERTICAL_LESION_1	0.880663
7	EB_LC_LESION_EJE_VERTICAL_LESION_1	EB_LC_LESION_EJE_HORIZONTAL_LESION_1	0.895864
8	EB_LESION_AREA_1	EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1	0.883330
9	EB_LESION_AREA_1	EB_LC_ULCERA_EJE_VERTICAL_LESION_1	0.913126
10	EB_LESION_AREA_1	EB_ULCERA_AREA_1	0.948659
11	EB_LESION_AREA_1	EB_LC_LESION_EJE_HORIZONTAL_LESION_1	0.948841
12	EB_LESION_AREA_1	EB_LC_LESION_EJE_VERTICAL_LESION_1	0.918574
13	EB_LC_TTO_MCTO_GLUCANTIME_DOSIS	EB_LC_PESO	0.973849

Figura 3.4: Atributos altamente correlacionados

Las columnas antes mencionadas son aquellas que se están altamente correlacionadas de forma directa, a medida que aumenta el valor de un atributo el del otro también, y a medida que disminuye también lo hará el otro valor.

10. La cantidad máxima de datos faltantes por registro es 1.
11. Los atributos que presentan datos faltantes son: EF_LC_ESTADO_FINAL_ESTUDIO y EF_LC_ESTADO_FINAL_ESTUDIO.
12. El atributo EF_LC_ESTADO_FINAL_ESTUDIO contiene 5 datos faltantes y el atributo EB_LC_TTO_MCTO_GLUCANTIME_DOSIS contiene 2 datos faltantes.
13. Algunos detalles de los 2 registros que presentan valor nulo en la columna EB_LC_TTO_MCTO_GLUCANTIME_DOSIS son:

	EB_LC_SEXO	EB_LC_ETNIA	EDAD	EB_LC_DPTO_INFECCION	EB_LC_PESO	EB_LC_ESTATURA	EB_LC_TTO_MCTO_PRESCRITO	EB_LC_DX_PREVIO
SU1044	1	3	50	Choco	105.0	177	2	0
SU2164	2	1	34	Narino	75.6	160	2	0

Figura 3.5: Registros con dato faltante

14. Division del conjunto de datos en función de la variable objetivo:

    - La respuesta al tratamiento de leishmaniasis de los pacientes es:
      *13 personas que curaron definitivamente.
      *7 personas que tuvieron falla terapéutica.

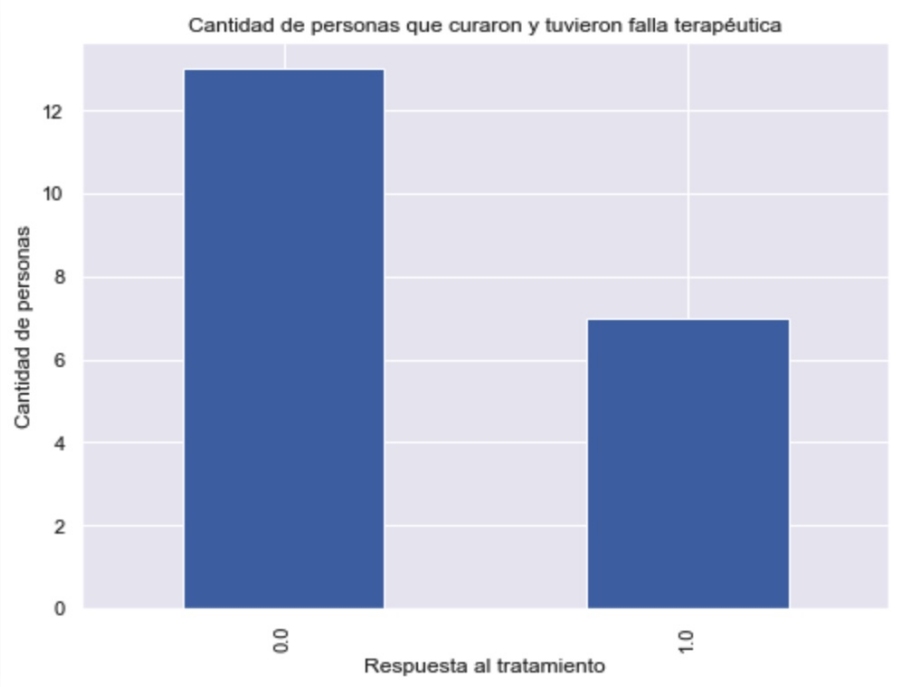

Figura 3.6: Gráfica de respuesta de pacientes al tratamiento de leishmaniasis

3.2. Limpieza de datos

Con base en la descripción inicial y el análisis realizado se procedió a definir el plan de ajuste para la limpieza de los datos:

Plan de ajuste

1. Eliminar los registros que en la columna clasificadora EF_LC_ESTADO_FINAL_ESTUDIO contiene valores nulos.
2. Eliminar los registros que en la columna EB_LC_TTO_MCTO_PRESCRITO contienen el valor 2, el cual indica que la persona se somete a un tratamiento diferente al glucantime, por lo cual no nos interesa.
3. Eliminar las siguientes columnas debido a su alta correlación con otras columnas:

   *EB_LC_ULCERA_EJE_HORIZONTAL_LESION_1.
   *EB_LC_ULCERA_EJE_VERTICAL_LESION_1.
   *EB_LC_LESION_EJE_HORIZONTAL_LESION_1.
   *EB_LC _LESION_EJE_VERTICAL_LESION_1.
   *EB_LESION_AREA_1.
   *EB_LC_PESO.
4. Eliminar la columna EB_LC_TTO_MCTO_PRESCRITO y la columna EB_LC_DX_PREVIO dado que no aportan información porque contienen el mismo valor en todos los registros.
5. Cambiar el tipo de dato de los valores que se encuentran en la columna EB_LC_DPTO_INFECCION. Pasarlos de tipo de dato cadena a entero con el fin de que los algoritmos matemáticos más adelante puedan tenerlo en cuenta.
6. Cambiar el tipo de dato de la columna clasificadora EF_LC_ESTADO_FINAL_ESTUDIO de decimal a entero con el fin de poder realizar más adelante algunas operaciones que solo aceptan este tipo de dato.
7. Calcular de nuevo la matriz de correlación y dado sus resultados eliminar variables si es necesario.
8. Normalizar los datos con el fin de que cada variable use rangos de valores similares, lo cual permitará organizar los datos en grupos lógicos.

3.3. Descripción del conjunto de datos final

Después de que se realizaron los cambios sobre el conjunto de datos incial se obtuvo un conjunto de datos con las siguientes características:

1. Contiene 18 registros y 10 atributos.

2. Los atributos y el tipo de dato de cada uno es:

Nombre del atributo	**Tipo de dato**
EB_LC_SEXO	Entero
EB_LC_ETNIA	Entero
EDAD	Entero
EB_LC_DPTO_INFECCION	Entero
EB_LC_ESTATURA	Entero
EB_LC_NUMERO_LC_ACTIVAS	Entero
EB_LC_TIEMPO_EVOLUCION	Entero
EB_ULCERA_AREA_1	Decimal
EB_LC_TTO_MCTO_GLUCANTIME_DOSIS	Decimal
EF_LC_ESTADO_FINAL_ESTUDIO	Entero

Cuadro 3.4: Tipo de dato de cada atributo del conjunto final

3. Los atributos no contienen datos faltantes.

4. La matriz de correlación es la siguiente:

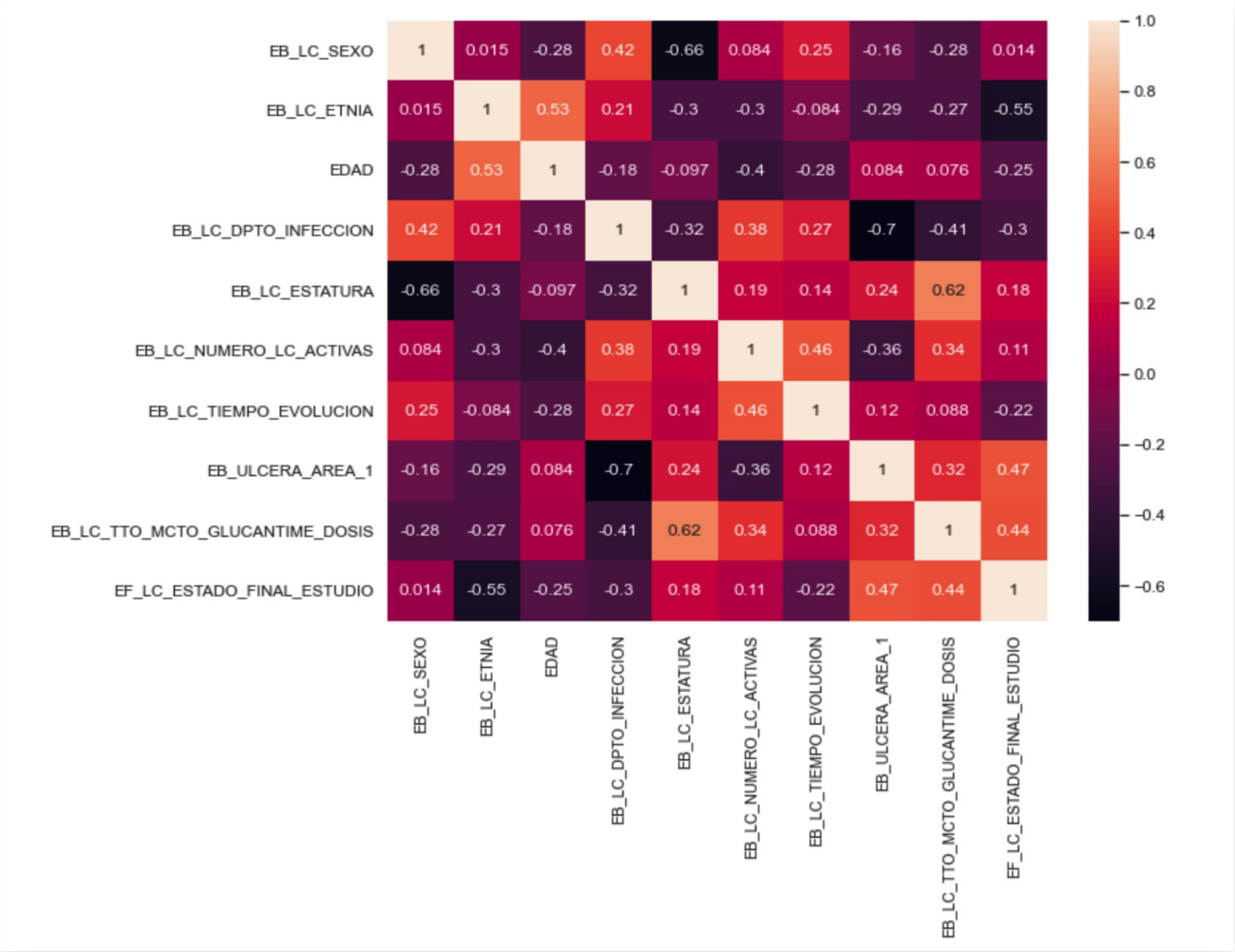

Figura 3.7: Matriz de correlación del conjunto final

No existen columnas que tengan una correlación mayor a 0.85.

5. Division del conjunto de datos final en función de la variable objetivo:

   - La respuesta al tratamiento de leishmaniasis de los pacientes es:
     *11 personas que curaron definitivamente.
     *7 personas que tuvieron falla terapéutica.

     Dado estos resultados se pudo evidenciar que el conjunto de datos está balanceado, por lo tanto, no será necesario aplicar ninguna estrategia de balanceo.

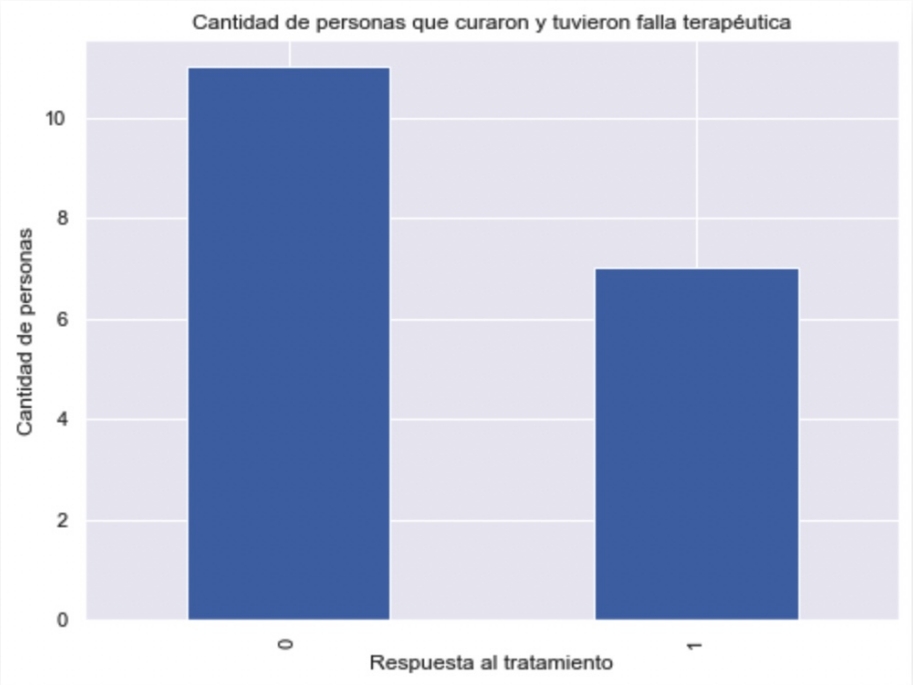

Figura 3.8: Gráfica de respuesta de pacientes al tratamiento de leishmaniasis del conjunto de datos final

Capítulo 4

Características de la construcción y evaluación de los modelos

4.1. Técnica usada para construcción de modelos

Para estimar los parámetros de los modelos base y de los modelos de ensamble se usó una búsqueda exhaustiva sobre unos valores de unos rangos definidos. Esta búsqueda usa internamente validación cruzada de 3 iteraciones sobre todo el conjunto de datos, es decir que de los 18 registros del conjunto, usó en cada iteración 12 para entrenar y 6 para evaluar cada modelo.

La métrica usada en dichas evaluaciones fue F1-score, la cual permite medir el desempeño de los modelos combinando las medidas de precision y recall en un solo valor.

Precision responde a la siguiente pregunta: ¿Qué porcentaje de los que se predijo que curaron definitivamente, realmente curaron definitivamente?

Recall responde a la siguiente pregunta: ¿Qué porcentaje de los que curaron definitivamente el algoritmo logró identificar?

El valor de la métrica está en el rango de 0 y 1, donde 0 indica que se desempeñó muy mal en la clasificación y 1 que se desempeñó muy bien.

4.2. Técnica y métrica usada para evaluación de modelos

Para las evaluaciones de los modelos se usó validación cruzada, dado que se tiene una pequeña cantidad de datos y esta técnica permite garantizar que los resultados obtenidos de la evaluación sean independientes de la partición de los datos de entrenamiento y prueba. Para cada una de las evaluaciones se utilizó también validación cruzada de 3 iteraciones, es decir, se dividió el conjunto de datos en 3 subconjuntos. La cantidad de registros es 18, por lo tanto, para las evaluaciones en cada iteración se consideró 12 registros para entrenar y 6 para evaluar cada modelo como se menciona anteriormente. Se usó la métrica F1 para la evaluación de los modelos, misma métrica que permitió estimar los parámetros.

Capítulo 5

Técnica de bagging

5.1. Bagging Classifier

El principal objetivo de bagging es sacar provecho de forma paralela de los algoritmos simples, reduciendo así el error de las predicciones al combinar los resultados de los modelos construidos.

Los algoritmos simples pertenecen a un mismo tipo, en este caso el algoritmo usado es K vecinos más cercanos. Este fue seleccionado porque se adapta muy bien cuando el conjunto de datos es pequeño.

5.1.1. Estimación de parámetros y construcción del modelo

Inicialmente se realizó la estimación de los valores de los parámetros para el algoritmo K vecinos más cercanos, los parámetros seleccionados fueron:

*** n_neighbors:** Indica el número de vecinos más cercanos que tiene en cuenta cada una de las muestras al momento de clasificar.

*** weights:** Indica la función de peso, la cual define la importancia de los votos de cada muestra dentro del conjunto de los k vecinos más cercanos en la predicción.

*** algorithm:** Indica el algoritmo usado que permite encontrar los k vecinos más cercanos.

*** leaf_size:** Indica la cantidad de hojas del árbol en caso de que el algoritmo que se usa para encontrar los k vecinos más cercanos use un árbol para guardar los valores.

*** p:** Indica el valor que toma el parámetro de la métrica de Minkowski. Esto define la función de distancia que usará el algoritmo.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
n_neighbors	3	5	7	9	11
weights	uniform	distance			
algorithm	ball_tree	kd_tree	brute		
leaf_size	Rango:[2,12]				
p	Rango:[1,15]				

Cuadro 5.1: Posibles valores de los parámetros de K vecinos más cercanos

Dada que la cantidad de posibles valores de los parámetros **leaf_size** y **p** es muy alta, se indicaron a través de rangos como se muestra anteriormente.

Después de probar el algoritmo de K vecinos más cercanos con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron al estimador base, estos fueron:

Nombre del parámetro	Valor seleccionado
n_neighbors	11
weights	distance
algorithm	ball_tree
leaf_size	2
p	4

Cuadro 5.2: Valores de los parámetros de K vecinos más cercanos

Luego de la evaluación, se procedió a estimar los valores de los parámetros de bagging, los parámetros seleccionados fueron:

*** base_estimator:** Indica el modelo simple que se usa de forma paralela. El valor de este parámetro corresponde al algoritmo de K vecinos más cercanos que antes ya fue construido, por lo tanto, este valor ya no se estima.

*** n_estimators:** Indica el número de estimadores base o de modelos simples que usa el algoritmo.

*** max_samples:** Indica el número de muestras o registros que utiliza cada uno de los estimadores base para su entrenamiento. Utiliza reemplazo.

*** max_features:** Indica el número de características o atributos que usa cada estimador base para su entrenamiento. No utiliza reemplazo.

Los posibles valores que tomaron los demás parámetros fueron los siguientes:

Nombre del parámetro	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5	Valor 6	Valor 7
base_estimator	Algoritmo de K vecinos más cercanos						
n_estimators	Valores impares del rango:[3,101]						
max_samples	11	12					
max_features	3	4	5	6	7	8	9

Cuadro 5.3: Posibles valores de los parámetros de Bagging Classifier

Dada que la cantidad de posibles valores del parámetro **n_estimators** es muy alta, se indicó a través de un rango como se muestra anteriormente.

Después de probar el algoritmo de Bagging Classifier con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron al modelo, estos fueron:

Nombre del parámetro	Valor seleccionado
base_estimator	Algoritmo de K vecinos más cercanos
n_estimators	3
max_samples	12
max_features	3

Cuadro 5.4: Valores de los parámetros de Bagging Classifier

Es así como se construye Bagging Classifier con el modelo de k vecinos más cercanos y con los valores de los parámetros de la anterior estimación. Dicho modelo finalmente queda así:

Nombre del parámetro	Valor seleccionado
n_neighbors	11
weights	distance
algorithm	ball_tree
leaf_size	2
p	4
n_estimators	3
max_samples	12
max_features	3

Cuadro 5.5: Valores seleccionados para los parámetros del modelo de Bagging Classifier

Los primeros cinco parámetros hacen parte del algoritmo de K vecinos más cercanos.

5.1.2. Evaluación del modelo

Para evaluar el modelo de bagging, se realizaron primero solo evaluaciones del modelo de K vecinos más cercanos y luego se realizaron evaluaciones del modelo de ensamble que utiliza como estimador base K vecinos más cercanos con el fin de poder realizar una comparación.

Se evaluó tres veces el modelo simple y el modelo de ensamble con los parámetros antes seleccionados, en cada una de las evaluaciones se usó validación cruzada de tres iteraciones y la métrica F1-score.

Los resultados de las evaluaciones fueron los siguientes:

***Modelo K vecinos más cercanos:**

Nro de evaluación	**Media de la validación cruzada con k=3**
Primera evaluación	0.33
Segunda evaluación	0.33
Tercera evaluación	0.33

Cuadro 5.6: Resultados de las tres evaluaciones de K vecinos más cercanos

***Modelo Bagging Classifier:**

Nro de evaluación	**Media de la validación cruzada con k=3**
Primera evaluación	0.13
Segunda evaluación	0.00
Tercera evaluación	0.11

Cuadro 5.7: Resultados de las tres evaluaciones de Bagging Classifier

A continuación se mostrará en detalle los mejores resultados obtenidos de los modelos, es decir, los que presentaron la media más alta.

1) Evaluación del modelo simple: Los experimentos realizados con el modelo de K vecinos más cercanos presentaron los mismos resultados en las tres evaluaciones, como se muestra en la tabla 5.6. A continuación se describe en detalle los resultados de una de ellas:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.00
Segunda iteración	1.00
Tercera iteración	0.00

Cuadro 5.8: Resultados de la evaluación de K vecinos más cercanos durante las tres iteraciones de validación cruzada

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.33
Desviación estándar	0.47

Cuadro 5.9: Media y desviación estándar de los resultados de evaluación de K vecinos más cercanos durante las tres iteraciones de validación cruzada

2) Evaluación del modelo de ensamble: Los experimentos realizados con el modelo de Bagging Classifier presentaron resultados diferentes en las tres evaluaciones, como se muestra en la tabla 5.7. A continuación se describe en detalle los resultados de la primera evaluación, ya que fue la que presentó la media más alta:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.00
Segunda iteración	0.40
Tercera iteración	0.00

Cuadro 5.10: Resultados de la evaluación de Bagging Classifier durante las tres iteraciones de validación cruzada en la primera evaluación

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.13
Desviación estándar	0.19

Cuadro 5.11: Media y desviación estándar de los resultados de evaluación de Bagging Classifier durante las tres iteraciones de validación cruzada de la primera evaluación

La siguiente imagen resume los resultados obtenidos de las tres iteraciones de validación cruzada que anteriormente se presentaron en la tabla 5.8 del modelo simple(1 estimador) y 5.10 del modelo de ensamble(3 estimadores):

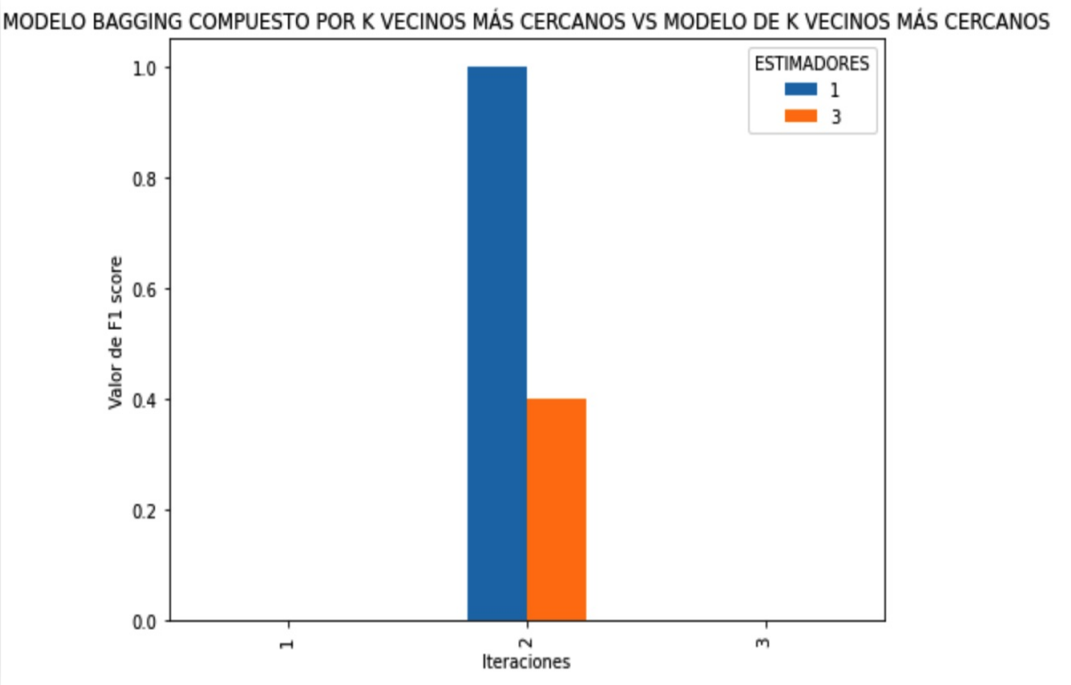

Figura 5.1: Modelo bagging compuesto por k vecinos más cercanos vs modelo de k vecinos más cercanos

Análisis: Al evaluar tres veces el modelo de ensamble con validación cruzada de tres iteraciones la media fue diferente, dado que bagging internamente intenta crear tres estimadores base del algoritmo de k vecinos más cercanos lo más diferentes posibles. Así que cada uno de los estimadores selecciona aleatoriamente diferentes conjuntos de muestras y características para entrenarlos, esa aleatoriedad hace que cada resultado de evaluación sea diferente. Sin embargo, los tres resultados de las evaluaciones tienen algo en común, su media es baja, ni siquiera el modelo que presenta la media más alta presenta un buen desempeño. Muy probablemente el mal desempeño del modelo no se deba por el tema de aleatoriedad sino al tema de que no cuentan con la cantidad suficiente de datos. Resulta que bagging solo está entrenando con doce registros, según los parámetros que mejor se ajustaron a cada estimador base, cada uno de estos debe entrenarse con doce registros que usan reemplazo, es decir, utilizan una muestra igual al tamaño del conjunto de entrenamiento, pero no contiene todas las muestras ya que algunas se repiten. Es así como cada estimador base recibe tan pocos datos que la posibilidad de que cada modelo se sobreajuste es muy alta, de tal forma, que cada modelo no pueda aprender lo suficiente para poder reconocer las clases de forma correcta.

Al realizar las tres evaluaciones del modelo simple se confirma la teoría, este modelo no hace selecciones aleatorias internas sobre las muestras y características, sino que usa el conjunto completo de entrenamiento para la construcción del modelo, por ello, es que en dichas evaluaciones la media es la misma. Esta media indica que el modelo simple no se desempeña bien.

Finalmente se permite establecer la siguiente hipótesis: En este caso utilizar algoritmos de forma paralela para obtener una predicción final no mejora el desempeño del modelo, en comparación a que sí solo se usara un solo algoritmo para entrenarlo. Sin embargo, a pesar de que el modelo simple se desempeña un poco mejor que el modelo de ensamble, ambos se siguen desempeñando muy mal, y todo esto se debe a que la cantidad de datos disponibles para entrenar cada algoritmo es muy pequeña.

Capítulo 6

Técnica de boosting

6.1. Boosting Classifier

El principal objetivo de boosting es sacar provecho al utilizar algoritmos simples de forma secuencial reduciendo el sesgo. La idea es que el modelo simple posterior le dé más importancia a los errores que cometió un modelo simple anterior.

6.1.1. Gradient Boosting Classifier

Un modelo Gradient Boosting tiene un estimador base por defecto, usa los árboles de decisión como modelos simples. Para determinar los parámetros que usará cada árbol de decisión se usa el algoritmo de descenso de gradiente que a través de derivadas, minimizará la función de pérdida. Es así que de forma secuencial se agregan árboles con distintos parámetros con la intención de que al final la combinación de dichos árboles pueda minimizar la pérdida del modelo.

6.1.1.1. Estimación de parámetros y construcción del modelo

Inicialmente se realizó la estimación de los parámetros de boosting. Dentro de estos parámetros están también los del modelo de árbol de decisión debido al hecho de que por defecto se establece como modelo simple. Los parámetros seleccionados fueron:

*** loss:** Es una función que permite medir el error del modelo, de forma indirecta mide la precisión del modelo.

*** learning_rate:** Es una medida que determina el impacto de cada árbol en la estimación final. Controla la magnitud del cambio en cada una de las estimaciones.

*** n_estimators:** Indica el número de árboles secuenciales que se modelan.

*** criterion:** Permite definir la estrategia para medir la calidad de una división al momento de construir un nodo de un árbol.

*** min_samples_split:** Define el número mínimo de registros que necesita un nodo para considerarse para la división y construcción de nuevos nodos.

*** min_samples_leaf:** Define el número mínimo de registros que necesita un nodo para que se considere un nodo hoja.

*** max_depth:** Define la profundidad máxima de un árbol.

*** max_features:** Define la cantidad de atributos que se deben considerar al momento de buscar la mejor división para la construcción de nodos en un árbol.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	**Valor 1**	**Valor 2**	**Valor 3**	**Valor 4**	**Valor 5**	**Valor 6**	**Valor 7**
loss	deviance	exponential					
learning_rate	0.001	0.01	0.1	0.5	1		
n_estimators	3	5	15	33	51	71	97
criterion	friedman_mse	mse					
min_samples_split	2	5	8	10	12		
min_samples_leaf	1	4	7	9	11		
max_depth	1	3	5	7	9		
max_features	1	3	5	7	9		

Cuadro 6.1: Posibles valores de los parámetros de Gradient Boosting Classifier

Después de probar el algoritmo de Gradient Boosting Classifier con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron al algoritmo, estos fueron:

Nombre del parámetro	**Valor seleccionado**
loss	exponential
learning_rate	0.5
n_estimators	3
criterion	mse
min_samples_split	10
min_samples_leaf	4
max_depth	5
max_features	3

Cuadro 6.2: Valores de los parámetros de Gradient Boosting Classifier

Es así como se construye el modelo de Gradient Boosting Classifier a partir de los parámetros antes seleccionados.

Para poder realizar una comparación del modelo simple con respecto al modelo de ensamble, es necesario realizar la estimación de los valores de los parámetros para el algoritmo árbol de decisión por separado, es así como se sigue con dicha estimación, los parámetros seleccionados fueron:

*** criterion:** Es un valor que permite medir la pureza de un nodo, es decir, la calidad de la división de datos al momento de construir los nodos del árbol de decisión.

*** splitter:** Es la estrategia usada para la división de los nodos. Define si el algoritmo debe considerar todas las características para encontrar la mejor división de los nodos o solamente la cantidad especificada en el parámetro max_features.

*** max_depth:** Define la profundidad máxima del árbol.

*** min_samples_split:** El número de datos mínimo requerido por un nodo para realizar una división.

*** min_samples_leaf:** El número mínimo de datos para que un nodo sea considerado hoja.

*** max_features:** El número máximo de atributos o variables a considerar cuando el splitter está buscando la mejor división.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	**Valor 1**	**Valor 2**	**Valor 3**	**Valor 4**	**Valor 5**
criterion	gini	entropy			
splitter	best	random			
max_depth	1	3	5	7	9
min_samples_split	2	5	8	10	12
min_samples_leaf	1	4	7	9	11
max_features	1	3	5	7	9

Cuadro 6.3: Posibles valores de los parámetros de árbol de decisión

Después de probar el algoritmo de árbol de decisión con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron a este algoritmo, estos fueron:

Nombre del parámetro	Valor seleccionado
criterion	entropy
splitter	best
max_depth	7
min_samples_split	8
min_samples_leaf	1
max_features	5

Cuadro 6.4: Valores de los parámetros de árbol de decisión

Es así como se construye el modelo de árbol de decisión a partir de los parámetros antes seleccionados.

6.1.1.2. Evaluación del modelo

Para evaluar el modelo de Gradient Boosting, primero se realizaron solo evaluaciones del modelo simple, en este caso del modelo de árbol de decisión y luego se realizaron evaluaciones del modelo de ensamble que utiliza como estimador base el modelo de árbol de decisión, con el fin de poder realizar una comparación.

Se evaluó tres veces el modelo simple y el modelo de ensamble con los parámetros antes seleccionados, en cada una de las evaluaciones se usó validación cruzada de tres iteraciones y la métrica F1-score.

Los resultados de las evaluaciones fueron los siguientes:

***Modelo de árbol de decisión:**

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.29
Segunda evaluación	0.38
Tercera evaluación	0.25

Cuadro 6.5: Resultados de las tres evaluaciones de árbol de decisión

***Modelo Gradient Boosting Classifier:**

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.67
Segunda evaluación	0.45
Tercera evaluación	0.39

Cuadro 6.6: Resultados de las tres evaluaciones de Gradient Boosting Classifier

A continuación se mostrará en detalle los mejores resultados obtenidos de los modelos, es decir, los que presentaron la media más alta.

1) Evaluación del modelo simple: Los experimentos realizados con el modelo de árbol de decisión presentaron resultados diferentes en las tres evaluaciones, como se muestra en la tabla 6.5. A continuación se describe en detalle los resultados que corresponden a la segunda evaluación, ya que fue la que presentó la media más alta:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.40
Segunda iteración	0.00
Tercera iteración	0.75

Cuadro 6.7: Resultados de la evaluación de árbol de decisión durante las tres iteraciones de validación cruzada en la segunda evaluación

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.38
Desviación estándar	0.31

Cuadro 6.8: Media y desviación estándar de los resultados de evaluación de árbol de decisión durante las tres iteraciones de validación cruzada de la segunda evaluación

2) Evaluación del modelo de ensamble: Los experimentos realizados con el modelo de Gradient Boosting Classifier presentaron resultados diferentes en las tres evaluaciones, como se muestra en la tabla 6.6. A continuación se describe en detalle los resultados que corresponden a la primera evaluación, ya que fue la que presentó la media más alta:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.50
Segunda iteración	0.67
Tercera iteración	0.86

Cuadro 6.9: Resultados de la evaluación de Gradient Boosting Classifier durante las tres iteraciones de validación cruzada en la primera evaluación

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.67
Desviación estándar	0.15

Cuadro 6.10: Media y desviación estándar de los resultados de evaluación de Gradient Boosting Classifier durante las tres iteraciones de validación cruzada de la primera evaluación

La siguiente imagen resume los resultados obtenidos de las tres iteraciones de validación cruzada que anteriormente se presentaron en la tabla 6.7 del modelo simple (1 estimador) y 6.9 del modelo de ensamble (3 estimadores):

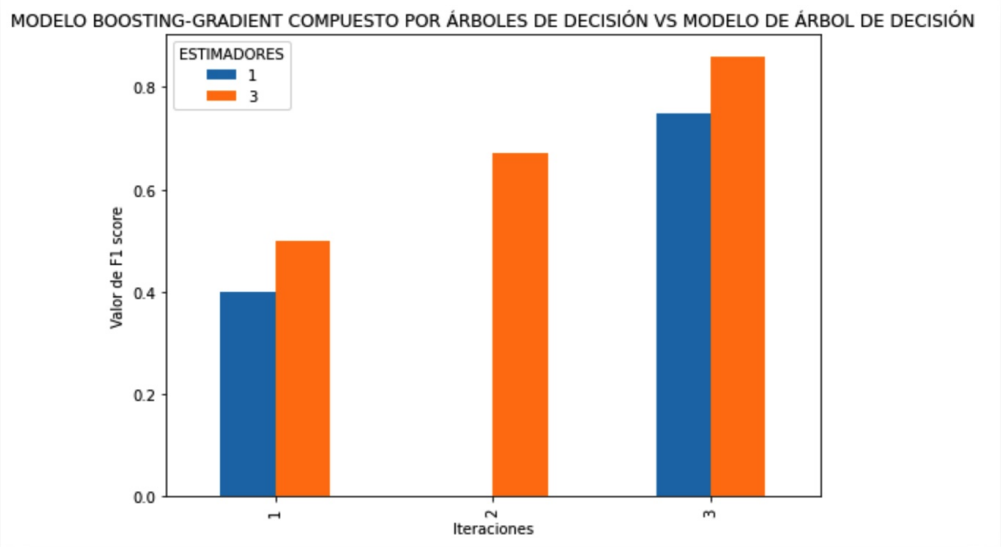

Figura 6.1: Modelo Boosting-Gradient compuesto por árboles de decisión vs modelo de árbol de decisión

Análisis: Al evaluar tres veces el modelo de ensamble con validación cruzada de tres iteraciones la media fue diferente, debido a que gradient boosting al crear los tres estimadores base del algoritmo de árboles de decisión selecciona aleatoriamente características para entrenarlos, dicha aleatoriedad hace que el resultado en cada evaluación sea diferente. La media de las evaluaciones no se considera alta para determinar que el modelo construido presenta un buen desempeño, así que probablemente el mal desempeño del modelo no se deba por elegir aleatoriamente las características para entrenar cada estimador base, sino que se deba a la pequeña cantidad de datos que hay para entrenar el algoritmo de ensamble. A pesar de que este algoritmo usa los doce registros del conjunto de entrenamiento para entrenar los estimadores base, esta cantidad no permite que dichos estimadores base puedan ajustar poco a poco los parámetros que harían que la función de pérdida disminuya considerablemente, ya que no permiten construir hipótesis fuertes para reconocer de forma correcta las clases a la que pertenecen los registros.

Al evaluar tres veces el modelo simple, es decir, el modelo de árbol de decisión, la media de las tres evaluaciones fue diferente, lo anterior se debe a que el algoritmo al construir los árboles de decisión debe usar un valor que le permita medir la calidad de una división al momento de construir los nodos, dicha división es dada por alguna característica, resulta que este valor es igual para algunas de las características del conjunto de datos, por lo cual, el algoritmo debe seleccionar aleatoriamente la característica que usará para la división de datos inicial. A continuación se muestran dos árboles de decisión en el que su nodo inicial se divide con diferentes características dado que su índice el cual se llama **entropy** es el mismo para ambas:

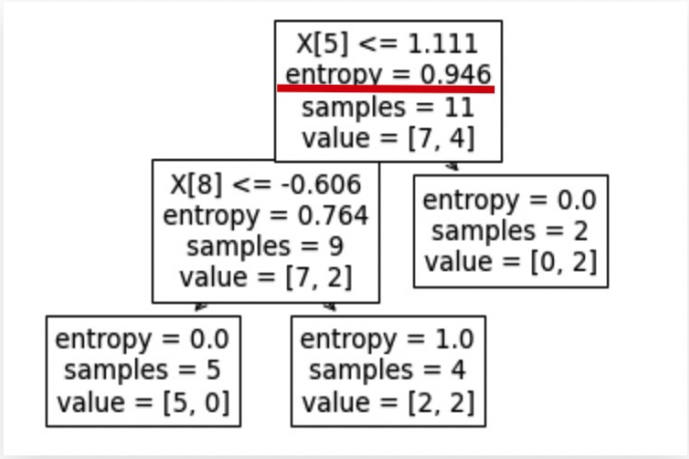

Figura 6.2: Primer modelo de árbol de decisión

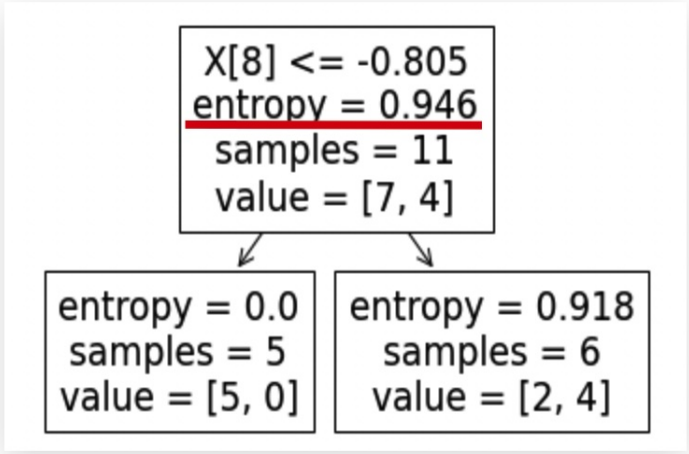

Figura 6.3: Segundo modelo de árbol de decisión

Las medias obtenidas al evaluar el modelo simple, indica que este no se desempeña bien.

Finalmente se permite establecer la siguiente hipótesis: En este caso utilizar algoritmos simples de forma secuencial utilizando la estrategia del algoritmo de Gradient Boosting Classifier para obtener una predicción final si mejora el desempeño del modelo, en comparación a que sí solo se usara un

solo algoritmo para entrenarlo. Sin embargo, a pesar de que el modelo de ensamble se desempeña un poco mejor que el modelo simple, la media no indica un buen desempeño, solo indica que le hace falta una cantidad de datos mayor para mejorar dicho desempeño.

6.1.2. AdaBoost Classifier

Un modelo de AdaBoost Classifier no tiene predefinido el estimador base que usará. Por lo tanto, en este caso se define los árboles de decisión como el modelo simple. Este algoritmo funciona aplicando el estimador base n veces al conjunto de entrenamiento, y cada vez aumentando los pesos de los ejemplos incorrectamente clasificados, de tal forma, que en cada iteración el estimador base se entrene con un conjunto de datos con pesos actualizados con el fin de mejorar las predicciones erróneas.

6.1.2.1. Estimación de parámetros y construcción del modelo

Inicialmente se debió realizar la estimación de los parámetros para el algoritmo de árbol de decisión, pero dado que en el anterior algoritmo de boosting, es decir, en el algoritmo de Gradient Boosting Classifier se realizó esta tarea utilizando la misma cantidad de registros para el entrenamiento de estimadores base, entonces simplemente se usaron los valores de los parámetros ya encontrados, estos fueron:

Nombre del parámetro	**Valor seleccionado**
criterion	entropy
splitter	best
max_depth	7
min_samples_split	8
min_samples_leaf	1
max_features	5

Cuadro 6.11: Valores de los parámetros de árbol de decisión

Luego se procedió a estimar los valores de los parámetros de AdaBoost Classifier, los parámetros seleccionados fueron:

*** base_estimator:** Indica el modelo simple que se usa de forma secuencial. El valor de este parámetro corresponde al algoritmo de árbol de decisión que antes ya fue construido, por lo tanto, este valor ya no se estima.

*** n_estimators:** Indica el número de estimadores base que usa el algoritmo.

*** learning_rate:** Es una medida que señala el grado en que los pesos aumentarán o disminuirán a medida que se construye un estimador base.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
base_estimator	Algoritmo de árbol de decisión				
n_estimators	Valores impares del rango:[3,101]				
learning_rate	0.001	0.01	0.1	0.5	1

Cuadro 6.12: Posibles valores de los parámetros de AdaBoost Classifier

Dada que la cantidad de posibles valores del parámetro n_estimators es muy alta, se indicó a través de un rango como se muestra anteriormente.

Después de probar el algoritmo de AdaBoost Classifier con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron al modelo, estos fueron:

Nombre del parámetro	Valor seleccionado
base_estimator	Algoritmo de árbol de decisión
n_estimators	5
learning_rate	0.01

Cuadro 6.13: Valores de los parámetros de AdaBoost Classifier

Es así como se construye AdaBoost Classifier con el modelo de árbol de decisión y con los valores de los parámetros de la anterior estimación. El modelo finalmente queda así:

Nombre del parámetro	Valor seleccionado
criterion	entropy
splitter	best
max_depth	7
min_samples_split	8
min_samples_leaf	1
max_features	5
n_estimators	5
learning_rate	0.01

Cuadro 6.14: Valores seleccionados para los parámetros del modelo AdaBoost Classifier

Los primeros seis parámetros corresponden al algoritmo de árbol de decisión.

6.1.2.2. Evaluación del modelo

Para evaluar el modelo de AdaBoost, se realizaron comparaciones de las evaluaciones del modelo simple, es decir, del modelo de árbol de decisión que anteriormente ya se habían realizado, con respecto a las evaluaciones del modelo de ensamble que utiliza como estimador base el modelo de árbol de decisión.

Se evaluó tres veces el modelo simple y el modelo de ensamble con los parámetros antes seleccionados, en cada una de las evaluaciones se usó validación cruzada de tres iteraciones y la métrica F1-score.

Los resultados de las evaluaciones fueron los siguientes:

***Modelo de árbol de decisión:**

Nro de evaluación	**Media de la validación cruzada con k=3**
Primera evaluación	0.29
Segunda evaluación	0.38
Tercera evaluación	0.25

Cuadro 6.15: Resultados de las tres evaluaciones de árbol de decisión

***Modelo AdaBoost Classifier:**

Nro de evaluación	**Media de la validación cruzada con k=3**
Primera evaluación	0.29
Segunda evaluación	0.47
Tercera evaluación	0.25

Cuadro 6.16: Resultados de las tres evaluaciones de AdaBoost Classifier

A continuación se mostrará en detalle los mejores resultados obtenidos de los modelos, es decir, los que presentaron la media más alta.

1)Evaluación del modelo simple: Los experimentos realizados con el modelo de árbol de decisión presentaron resultados diferentes en las tres evaluaciones, como se muestra en la tabla 6.15. A continuación se describe en detalle los resultados que corresponden a la segunda evaluación, ya que

fue la que presentó la media más alta:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.40
Segunda iteración	0.00
Tercera iteración	0.75

Cuadro 6.17: Resultados de la evaluación de árbol de decisión durante las tres iteraciones de validación cruzada en la segunda evaluación

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.38
Desviación estándar	0.31

Cuadro 6.18: Media y desviación estándar de los resultados de evaluación de árbol de decisión durante las tres iteraciones de validación cruzada de la segunda evaluación

2) Evaluación del modelo de ensamble: Los experimentos realizados con el modelo de AdaBoost Classifier presentaron resultados diferentes en las tres evaluaciones, como se muestra en la tabla 6.16. A continuación se describe en detalle los resultados que corresponden a la segunda evaluación, ya que fue la que presentó la media más alta:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.00
Segunda iteración	0.67
Tercera iteración	0.75

Cuadro 6.19: Resultados de la evaluación de AdaBoost Classifier durante las tres iteraciones de validación cruzada en la segunda evaluación

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.47
Desviación estándar	0.34

Cuadro 6.20: Media y desviación estándar de los resultados de evaluación de AdaBoost Classifier durante las tres iteraciones de validación cruzada en la segunda evaluación

La siguiente imagen resume los resultados obtenidos de las tres iteraciones de validación cruzada que anteriormente se presentaron en la tabla 6.17 del modelo simple (1 estimador) y 6.19 del modelo de ensamble (5 estimadores):

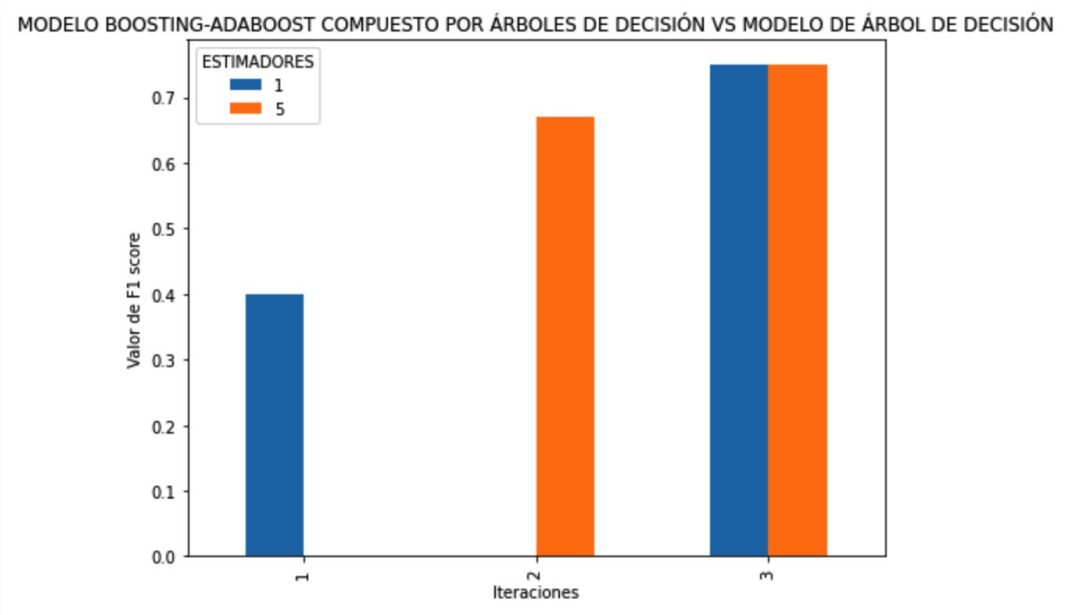

Figura 6.4: Modelo Boosting-Adaboost compuesto por árboles de decisión vs modelo de árbol de decisión

Análisis: Durante las tres evaluaciones del modelo de ensamble la media de las tres iteraciones de validación cruzada fue diferente, esto debido a que el algoritmo al construir los estimadores base selecciona aleatoriamente características para entrenarlos, por ello es que son diferentes los resultados en cada evaluación. La media de las evaluaciones es baja, lo que indica que el modelo de ensamble no se desempeña bien, una razón del mal desempeño probablemente sea por la cantidad tan pequeña de datos que tiene disponible los estimadores base para su entrenamiento, que no permite que dichos estimadores base puedan poco a poco corregir las predicciones mal clasificadas, dado que el algoritmo desde un inicio no puede establecer hipótesis fuertes que permitan clasificar cada registro

de forma correcta.

Al evaluar tres veces el modelo simple, la media de las tres evaluaciones fue diferente, como se mencionó anteriormente en el apartado del modelo de Gradient Boosting Classifier esto se debe a que el algoritmo al construir los árboles de decisión debe usar un valor que permita medir la calidad de una división al momento de construir un nodo, dicha división es dada por alguna característica, resulta que puede haber más de una característica con el mismo índice, por lo cual el algoritmo debe seleccionar aleatoriamente alguna de las características para la división inicial, es por esto que la media resulta diferente en cada una de las evaluaciones. Las medias de este modelo simple no indica que el modelo se desempeña bien.

Finalmente se permite establecer la siguiente hipótesis: En este caso utilizar algoritmos simples de forma secuencial utilizando la estrategia de AdaBoost Classifier para obtener una predicción final resulta igual al desempeño que pueda presentar un modelo que utiliza un solo algoritmo simple. Sin embargo, las medias obtenidas al evaluar tanto el modelo simple como el modelo de ensamble, indican que no se desempeñan bien dado que le hace falta una mayor cantidad de datos para poder establecer hipótesis fuertes que permitan mejorar las predicciones realizadas.

Capítulo 7

Técnica de stacking

7.1. Stacking Classifier

Stacking usa la predicción de varios modelos diferentes de forma secuencial y paralela, de tal forma, que dichas predicciones en conjunto con los datos originales permitan crear modelos mucho más completos.

En este caso, se crea un modelo de stacking que usa dos niveles, es decir, usa un primer nivel que corresponde a varios algoritmos entrenados sobre el conjunto de datos de entrenamiento, y termina con un segundo nivel, el cual usa un modelo final que entrena con los datos originales de entrenamiento y además como características adicionales usa las predicciones obtenidas por los modelos de primer nivel. Esta técnica usa validación cruzada para entrenar y evaluar cada modelo del primer nivel.

Los modelos construidos para el primer nivel son: Máquinas de vectores de soporte, K vecinos más cercanos y árbol de decisión.

El modelo usado para el segundo nivel es: Regresión logística.

7.1.1. Estimación de parámetros y construcción del modelo

Inicialmente se realizó la estimación de los valores de los parámetros para los algoritmos de primer nivel.

Se empezó con el algoritmo de máquinas de vectores de soporte. Los parámetros seleccionados para este algoritmo fueron:

*** C:** Define el parámetro para la función de costo de margen blando, que tiene como fin controlar la influencia de cada vector de soporte individual, que señala una penalización a cualquier punto de datos mal clasificado.

*** kernel:** Específica el método que usará el algoritmo para transformar el espacio de los datos de entrada en la forma requerida.

*** gamma:** Determina el valor del parámetro de los siguientes kernel: Función de base radial('rbf'), polinómica('poli') y sigmoide('sigmoide'). Es un factor que permite definir cómo un vector da forma

al límite de decisión en su vecindario cercano.

*** degree:** Determina el grado de la función del kernel polinómico('poli').

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5	Valor 6	Valor 7	Valor 8	Valor 9
C	0.0001	0.001	0.01	0.02	0.1	0.3	1.0	10	100
kernel	linear	poly	rbf	sigmoid					
gamma	0.0001	0.001	0.01	0.02	0.1	0.3	1.0	10	100
degree	2	3	4	5					

Cuadro 7.1: Posibles valores de los parámetros de máquinas de vectores de soporte

Después de probar el algoritmo de máquinas de vectores de soporte con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron al algoritmo, estos fueron:

Nombre del parámetro	Valor seleccionado
C	1.0
kernel	linear
gamma	0.0001
degree	2

Cuadro 7.2: Valores de los parámetros de máquinas de vectores de soporte

Luego, se seleccionaron los parámetros para el algoritmo de k vecinos más cercanos, se hace nuevamente la construcción de este modelo, en lugar de utilizar el modelo construido en la técnica de Bagging Classifier, dado que la técnica de Stacking Classifier tiene como estrategia usar validación cruzada para entrenar y evaluar los estimadores base, por lo tanto, dicha cantidad de registros que hacen parte del conjunto de entrenamiento es menor, por ende es necesario ajustar ciertos posibles valores de parámetros. Los parámetros seleccionados para este algoritmo fueron:

*** n_neighbors:** Indica el número de vecinos más cercanos que tiene en cuenta cada una de las muestras al momento de clasificar.

*** weights:** Indica la función de peso, la cual define la importancia de los votos de cada muestra dentro del conjunto de los k vecinos más cercanos en la predicción.

*** algorithm:** Indica el algoritmo usado que permite encontrar los k vecinos más cercanos.

*** leaf_size:** Indica la cantidad de hojas del árbol en caso de que el algoritmo que se usa para encontrar los k vecinos más cercanos use un árbol para guardar los valores.

*** p:** Indica el valor que toma el parámetro de la métrica de Minkowski. Esto define la función de distancia que usará el algoritmo.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	**Valor 1**	**Valor 2**	**Valor 3**	**Valor 4**	**Valor 5**
n_neighbors	3	5			
weights	uniform	distance			
algorithm	ball_tree	kd_tree	brute		
leaf_size	2	3	4	5	6
p	Rango:[1,15]				

Cuadro 7.3: Posibles valores de los parámetros del algoritmo de K vecinos más cercanos

Dada que la cantidad de posibles valores del parámetro **p** es muy alta, se indicó a través de un rango como se muestra anteriormente.

Después de probar el algoritmo de K vecinos más cercanos con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron al estimador base, estos fueron:

Nombre del parámetro	**Valor seleccionado**
n_neighbors	3
weights	distance
algorithm	ball_tree
leaf_size	2
p	9

Cuadro 7.4: Valores de los parámetros del algoritmo de K vecinos más cercanos

Después, se seleccionaron los parámetros para el algoritmo de árbol de decisión, se hace nuevamente la construcción de este modelo, en lugar de utilizar el modelo construido en la técnica boosting, dado que la técnica de Stacking Classifier tiene como estrategia usar validación cruzada para entrenar y evaluar los estimadores base, por lo tanto, dicha cantidad de registros que hacen parte del conjunto de entrenamiento es menor, por ende es necesario ajustar ciertos posibles valores de parámetros.

Los parámetros seleccionados para este algoritmo fueron:

*** criterion:** Es un valor que permite medir la pureza de un nodo, es decir, la calidad de la división de datos al momento de construir los nodos del árbol de decisión.

*** splitter:** Es la estrategia usada para la división de los nodos. Define si el algoritmo debe considerar todas las características para encontrar la mejor división de los nodos o solamente la cantidad especificada en el parámetro max_features.

*** max_depth:** Define la profundidad máxima del árbol.

*** min_samples_split:** El número de datos mínimo requerido por un nodo para realizar una división.

*** min_samples_leaf:** El número mínimo de datos para que un nodo sea considerado hoja.

*** max_features:** El número máximo de atributos o variables a considerar cuando el splitter está buscando la mejor división.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Nombre del parámetro	**Valor 1**	**Valor 2**	**Valor 3**	**Valor 4**	**Valor 5**
criterion	gini	entropy			
splitter	best	random			
max_depth	1	3	5	7	9
min_samples_split	2	4	6		
min_samples_leaf	1	3	5		
max_features	1	3	5	7	9

Cuadro 7.5: Posibles valores de los parámetros del algoritmo de árbol de decisión

Después de probar el algoritmo de árbol de decisión con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron a este algoritmo, estos fueron:

Nombre del parámetro	Valor seleccionado
criterion	gini
splitter	best
max_depth	9
min_samples_split	6
min_samples_leaf	1
max_features	5

Cuadro 7.6: Valores de los parámetros del algoritmo de árbol de decisión

Una vez se realizó la estimación de los valores de los parámetros para los algoritmos de primer nivel, se siguió con la estimación de los valores de los parámetros para el algoritmo de segundo nivel, es decir, para el algoritmo de regresión logística. El parámetro seleccionado para este algoritmo fue:

*** solver:** Determina el algoritmo de optimización para la función de pérdida.

Los posibles valores que tomó el anterior parámetro fueron los siguientes:

Nombre del parámetro	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
solver	newton-cg	lbfgs	liblinear	sag	saga

Cuadro 7.7: Posibles valores del parámetro del algoritmo de regresión logística

Después de probar el algoritmo de regresión logística con todas las combinaciones de los valores antes señalados, se seleccionó el valor del parámetro que mejor se ajustó a este algoritmo, este fue:

Nombre del parámetro	Valor seleccionado
solver	newton-cg

Cuadro 7.8: Valor del parámetro del algoritmo de regresión logística

Finalmente se procedió a estimar los valores de los parámetros de Stacking Classifier, los parámetros seleccionados fueron:

*** estimators:** Define la lista de los algoritmos de primer nivel. El valor de este parámetro corresponde a una lista que contiene los algoritmos de máquinas de vectores de soporte, k vecinos más cercanos y árbol de decisión que antes ya fueron construidos, por lo tanto, esta lista ya quedó construida.

*** final_estimator:** Define el algoritmo final, es decir, el algoritmo de segundo nivel. El valor de este parámetro corresponde al algoritmo de regresión logística que ya fue construido.

*** cv:** Determina la estrategia de validación cruzada utilizada para entrenar y evaluar los algoritmos de primer nivel.

*** passthrough:** Determina si el algoritmo final, es decir, el algoritmo de segundo nivel se entrena solo con las predicciones hechas por los modelos de primer nivel o se entrena también con los datos del conjunto de entrenamiento original.

Los posibles valores que tomaron los anteriores parámetros fueron los siguientes:

Máquinas de vectores de soporte = MVS.
K vecinos más cercanos = Knn.
Árbol de decisión = arb.

Nombre del parámetro	**Valor 1**	**Valor 2**	**Valor 3**
estimators	[Algoritmo MVS, Knn , arb]		
final_estimator	Algoritmo de regresión logística		
cv	2	3	4
passthrough	True	False	

Cuadro 7.9: Posibles valores de los parámetros del algoritmo de Stacking Classifier

Después de probar el algoritmo de Stacking Classifier con todas las combinaciones de los valores antes señalados, se seleccionaron los valores de los parámetros que mejor se ajustaron a este algoritmo, estos fueron:

Nombre del parámetro	**Valor seleccionado**
estimators	[Algoritmo MVS, Knn , arb]
final_estimator	Algoritmo de regresión logística
cv	3
passthrough	True

Cuadro 7.10: Valores de los parámetros del algoritmo de Stacking Classifier

Es así como se construye Stacking Classifier con los algoritmos de primer y segundo nivel, y con los valores de los parámetros de la anterior estimación. El modelo finalmente queda así:

Nombre del parámetro	Valor seleccionado
C	1.0
kernel	linear
gamma	0.0001
degree	2
n_neighbors	3
weights	distance
algorithm	ball_tree
leaf_size	2
p	9
criterion	gini
splitter	best
max_depth	9
min_samples_split	6
min_samples_leaf	1
max_features	5
solver	newton-cg
cv	3
passthrough	True

Cuadro 7.11: Valores de los parámetros del algoritmo de Stacking Classifier

Los primeros dieciséis parámetros hacen parte de los algoritmos de primer y segundo nivel.

7.1.2. Evaluación del modelo

Para evaluar el modelo de Stacking Classifier se realizaron comparaciones de las evaluaciones de los modelos simples, con respecto a las evaluaciones del modelo de ensamble. Para ello, se evaluó el modelo de máquinas de vectores de soporte, k vecinos más cercanos, árbol de decisión y regresión logística, y cada uno de estos se comparó con el desempeño del modelo de ensamble, es decir, con el modelo de Stacking Classifier que combina todos aquellos modelos simples para obtener predicciones.

Se evaluó tres veces el modelo simple y el modelo de ensamble con los parámetros antes seleccionados, en cada una de las evaluaciones se usó validación cruzada de tres iteraciones y la métrica F1-score.

Los resultados de las evaluaciones fueron los siguientes:

*Modelo de máquinas de vectores de soporte:

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.77
Segunda evaluación	0.77
Tercera evaluación	0.77

Cuadro 7.12: Resultados de las tres evaluaciones del modelo de máquinas de vectores de soporte

*Modelo de k vecinos más cercanos:

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.17
Segunda evaluación	0.17
Tercera evaluación	0.17

Cuadro 7.13: Resultados de las tres evaluaciones del modelo de k vecinos más cercanos

*Modelo de árbol de decisión:

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.22
Segunda evaluación	0.25
Tercera evaluación	0.39

Cuadro 7.14: Resultados de las tres evaluaciones del modelo de árbol de decisión

*Modelo de regresión logística:

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.49
Segunda evaluación	0.49
Tercera evaluación	0.49

Cuadro 7.15: Resultados de las tres evaluaciones del modelo regresión logística

***Modelo de Stacking Classifier:**

Nro de evaluación	Media de la validación cruzada con k=3
Primera evaluación	0.27
Segunda evaluación	0.27
Tercera evaluación	0.27

Cuadro 7.16: Resultados de las tres evaluaciones del modelo Stacking Classifier

A continuación se mostrará en detalle los mejores resultados obtenidos de los modelos, es decir, los que presentaron la media más alta.

1)Evaluación del modelo de máquinas de vectores de soporte: Los experimentos realizados con el modelo de máquinas de vectores de soporte presentaron los mismos resultados en las tres evaluaciones, como se muestra en la tabla 7.12. A continuación se describe en detalle los resultados de una de ellas:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.50
Segunda iteración	1.00
Tercera iteración	0.80

Cuadro 7.17: Resultados de la evaluación del modelo de máquinas de vectores de soporte durante las tres iteraciones de validación cruzada

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.77
Desviación estándar	0.21

Cuadro 7.18: Media y desviación estándar de los resultados de la evaluación del modelo de máquinas de vectores de soporte durante las tres iteraciones de validación cruzada

2)Evaluación del modelo de k vecinos más cercanos: Los experimentos realizados con el modelo de k vecinos más cercanos presentaron los mismos resultados en las tres evaluaciones, como se muestra en la tabla 7.13. A continuación se describe en detalle los resultados de una de ellas:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.00
Segunda iteración	0.00
Tercera iteración	0.50

Cuadro 7.19: Resultados de la evaluación del modelo de k vecinos más cercanos durante las tres iteraciones de validación cruzada

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.17
Desviación estándar	0.24

Cuadro 7.20: Media y desviación estándar de los resultados de la evaluación del modelo de k vecinos más cercanos durante las tres iteraciones de validación cruzada

3)Evaluación del modelo de árbol de decisión: Los experimentos realizados con el modelo de árbol de decisión presentaron resultados diferentes en las tres evaluaciones, como se muestra en la tabla 7.14. A continuación se describe en detalle los resultados que corresponden a la tercera evaluación, ya que fue la que presentó la media más alta:

Nro iteración de validación cruzada	F1-score
Primera iteración	0.00
Segunda iteración	0.67
Tercera iteración	0.50

Cuadro 7.21: Resultados de la evaluación del modelo de árbol de decisión durante las tres iteraciones de validación cruzada de la tercera evaluación

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.39
Desviación estándar	0.28

Cuadro 7.22: Media y desviación estándar de los resultados de la evaluación del modelo de árbol de decisión durante las tres iteraciones de validación cruzada de la tercera evaluación

4)Evaluación del modelo de regresión logística: Los experimentos realizados con el modelo de regresión logística presentaron los mismos resultados en las tres evaluaciones, como se muestra en la tabla 7.15. A continuación se describe en detalle los resultados de una de ellas:

Nro iteración de validación cruzada	**F1-score**
Primera iteración	0.00
Segunda iteración	0.67
Tercera iteración	0.80

Cuadro 7.23: Resultados de la evaluación del modelo de regresión logística durante las tres iteraciones de validación cruzada

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.49
Desviación estándar	0.35

Cuadro 7.24: Media y desviación estándar de los resultados de la evaluación del modelo de regresión logística durante las tres iteraciones de validación cruzada

5)Evaluación del modelo de Stacking Classifier: Los experimentos realizados con el modelo de Stacking Classifier presentaron los mismos resultados en las tres evaluaciones, como se muestra en la tabla 7.16. A continuación se describe en detalle los resultados de una de ellas:

Nro iteración de validación cruzada	**F1-score**
Primera iteración	0.00
Segunda iteración	0.00
Tercera iteración	0.80

Cuadro 7.25: Resultados de la evaluación del modelo de Stacking Classifier durante las tres iteraciones de validación cruzada

La media y la desviación estándar que se obtuvieron con respecto a las tres iteraciones de validación cruzada antes mencionadas fueron:

Media	0.27
Desviación estándar	0.38

Cuadro 7.26: Media y desviación estándar de los resultados de la evaluación del modelo de Stacking Classifier durante las tres iteraciones de validación cruzada

A continuación se presentan imágenes que muestran los resultados obtenidos de las tres iteraciones de validación cruzada de cada uno de los modelos simples con respecto a los obtenidos por el modelo de ensamble.

Imagen que presenta los resultados de la tabla 7.17 que pertenece a las tres iteraciones de validación cruzada al evaluar el modelo de máquinas de vectores de soporte(1 estimador) con respecto a los resultados de la tabla 7.25 que fueron los obtenidos por el modelo Stacking Classifier (4 estimadores) :

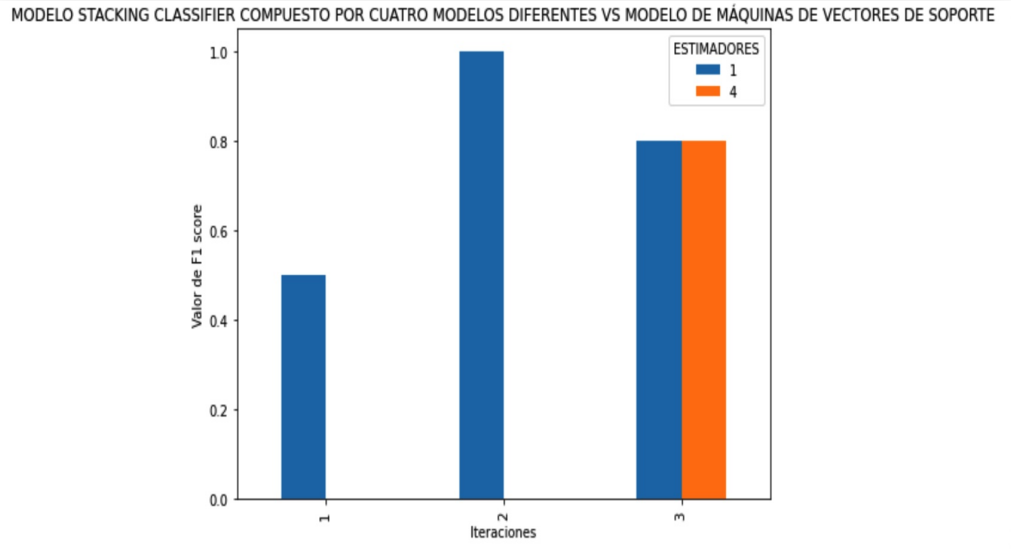

Figura 7.1: Modelo Stacking Classifier compuesto por cuatro modelos diferentes vs modelo de máquinas de vectores de soporte

Imagen que presenta los resultados de la tabla 7.19 que pertenece a las tres iteraciones de validación cruzada al evaluar el modelo de k vecinos más cercanos(1 estimador) con respecto a los resultados de la tabla 7.25 que fueron los obtenidos por el modelo Stacking Classifier (4 estimadores) :

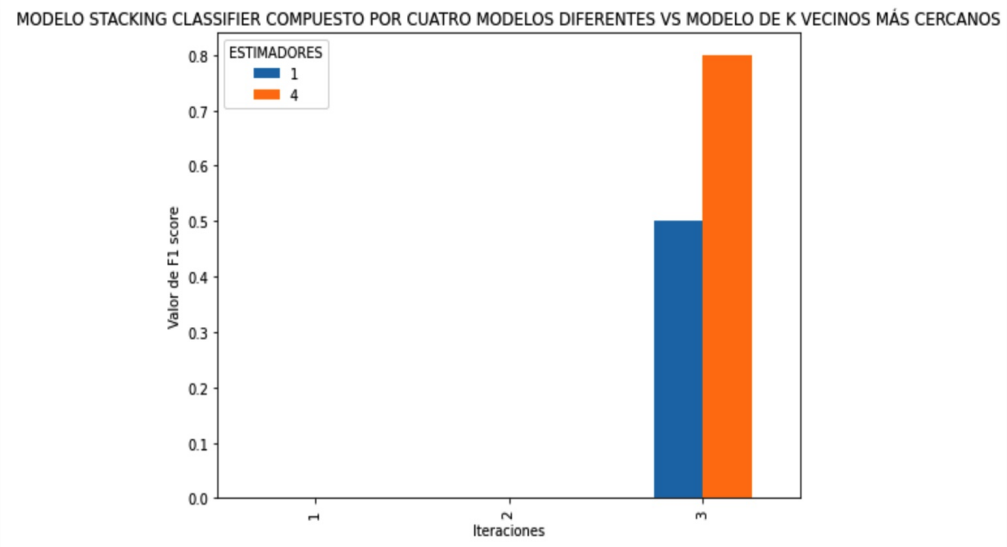

Figura 7.2: Modelo Stacking Classifier compuesto por cuatro modelos diferentes vs modelo de k vecinos más cercanos

Imagen que presenta los resultados de la tabla 7.21 que pertenece a las tres iteraciones de validación cruzada al evaluar el modelo de árbol de decisión(1 estimador) con respecto a los resultados de la tabla 7.25 que fueron los obtenidos por el modelo Stacking Classifier (4 estimadores) :

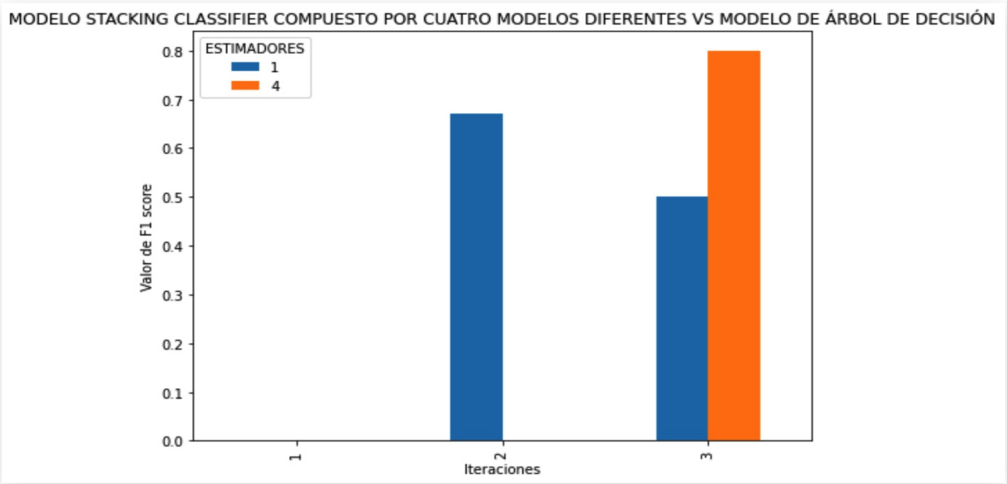

Figura 7.3: Modelo Stacking Classifier compuesto por cuatro modelos diferentes vs modelo de árbol de decisión

Imagen que presenta los resultados de la tabla 7.23 que pertenece a las tres iteraciones de validación cruzada al evaluar el modelo de regresión logística(1 estimador) con respecto a los resultados de la tabla 7.25 que fueron los obtenidos por el modelo Stacking Classifier (4 estimadores) :

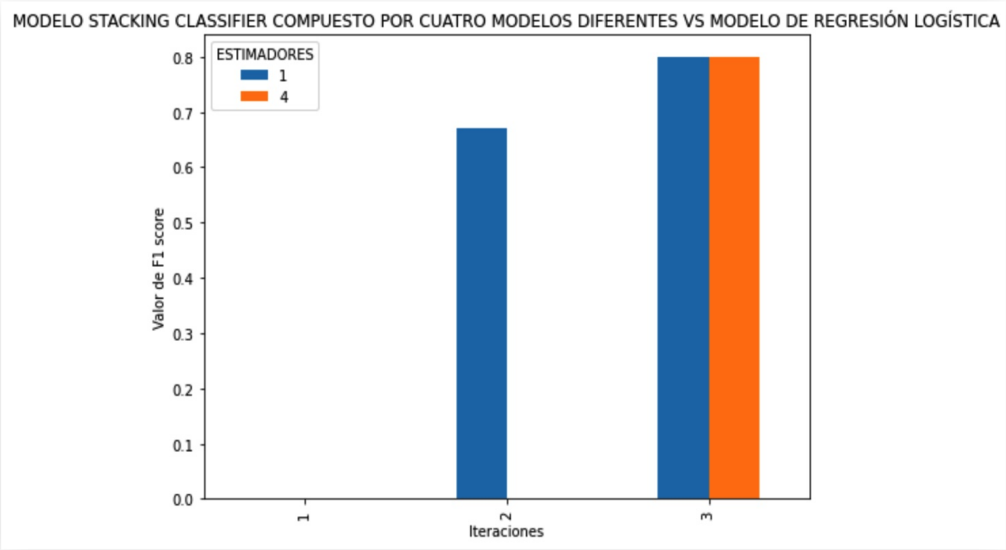

Figura 7.4: Modelo Stacking Classifier compuesto por cuatro modelos diferentes vs modelo de regresión logística

Análisis: Durante las tres evaluaciones del modelo de ensamble la media de las tres iteraciones de validación cruzada es la misma, esto dado a que el algoritmo internamente no selecciona un conjunto de características y registros sobre los cuales entrena los estimadores base, sino por el contrario, usa validación cruzada y entrena los algoritmos de primer nivel con los datos disponibles del conjunto de entrenamiento. La media de las evaluaciones es baja, lo que indica que el modelo de Stacking Classifier no se desempeña bien, una razón probablemente del mal desempeño sea a que este algoritmo cuenta con muy pocos datos para entrenar cada estimador base, cuenta con doce registros en el conjunto de entrenamiento y sobre estos realiza validación cruzada para dicho entrenamiento, por ende, la cantidad que usa cada algoritmo de primer nivel para su entrenamiento es aún más pequeña, así que el hecho de que los modelos se sobreajusten es muy alta. El sobreajuste de los estimadores base lleva a que estos no puedan establecer hipótesis fuertes, por lo tanto, las predicciones que estos lleguen a dar resultan poco valiosas para el estimador final.

Por otro lado, al evaluar tres veces el modelo de máquinas de vectores de soporte, k vecinos más cercanos, y regresión logística se puede notar que la media de las tres evaluaciones no cambia, esto dado que para la construcción de dichos modelos simples siempre se usa la misma cantidad de atributos y registros. El único modelo simple que presentó media diferente durante las tres iteraciones

de validación cruzada, fue el modelo de árbol de decisión, y no porque use un subconjunto de las características y registros del conjunto de entrenamiento para su construcción, sino porque como se mencionó anteriormente, al construir los nodos del árbol, se debe seleccionar una característica según un índice que mide la calidad de la división de los datos, hay que tener en cuenta que a veces este índice puede ser el mismo para más de una característica, así que el algoritmo selecciona aleatoriamente la división. Las evaluaciones de los modelos simples indican que hay algoritmos que construyen modelos que se desempeñan mejor que otros, hay estrategias de entrenamiento que funcionan mucho mejor que otras. Es el caso del modelo de máquinas de vectores de soporte que indica una media de 0.77 mientras que el modelo de k vecinos más cercanos indica una media de 0.17. A pesar de que hay unos modelos que se desempeñan mucho mejor que otros, las medias obtenidas al evaluar los modelos simples siguen siendo bajas, lo que indican un mal desempeño de todos estos.

Finalmente se permite establecer la siguiente hipótesis: En este caso, utilizar algoritmos simples de forma paralela y secuencial utilizando la estrategia de Stacking Classifier para obtener una predicción final en algunos casos puede resultar mejor que utilizar un solo algoritmo, en otros casos el desempeño de un algoritmo es mucho mayor que el de este modelo de ensamble. Sin embargo, las medias obtenidas al evaluar tanto los modelos simples como el modelo de ensamble indican que no tienen un buen desempeño dado que necesita una mayor cantidad de datos para la construcción de dichos modelos.

Capítulo 8

Análisis comparativo

8.1. Comparación de modelos construidos

Al evaluar cada uno de los modelos de ensamble el desempeño que mejor presentaron fue:

Modelo de ensamble	**F1 score**
Bagging Classifier	0.13
Gradient Boosting Classifier	0.67
AdaBoost Classifier	0.47
Stacking Classifier	0.27

Cuadro 8.1: Resultados de la evaluación de los modelos de ensamble

Con base en los resultados, la estrategia usada por boosting classifier, la cual es crear modelos de forma secuencial con el fin de que el modelo posterior intente corregir los errores del modelo anterior construye estimadores base con hipótesis muchos más fuertes que los estimadores que se construyen utilizando la estrategia de bagging classifier o stacking classifier.

Entendiendo que el concepto fundamental de las técnicas de ensamble es construir modelos relativamente no correlacionados que permitan juntos producir predicciones más precisas, gracias a que la no correlación permitirá que los modelos se protejan mutuamente de sus errores individuales, entonces se puede decir que la estrategia de boosting classifier es la que ha permitido construir los estimadores base lo más diversos posibles, lo menos correlacionados.

La técnica de boosting classifier ha permitido construir modelos basados en estimadores simples mucho más precisos que los construidos con las otras técnicas y esto probablemente se deba a que aprovecha en mejor medida los registros que tiene disponible. Mientras que boosting construye cada estimador base con doce registros del conjunto de entrenamiento, bagging y stacking utilizan solo una porción de estos, así que la cantidad de datos para el entrenamiento es aún más pequeña lo que aumenta la probabilidad de que estos modelos se sobreajusten, y por ende entreguen predicciones que son menos precisas.

8.2. Selección del mejor modelo

De las técnicas de boosting classifier, la que construye el modelo con el mejor desempeño es Gradient Boosting Classifier, esta técnica usa el algoritmo de descenso de gradiente para minimizar la función de pérdida y mejorar las predicciones, lo que demuestra que es un algoritmo bastante poderoso, en este caso demuestra que es mucho más poderoso al que usa la técnica AdaBoost Classifier.

El desempeño que presenta Gradient Boosting Classifier sigue siendo bajo, es decir, este modelo no se debería usar aún. La métrica indica que es el modelo que mejor desempeño presenta, pero no es un buen modelo, solo indica que probablemente es la técnica que con un poco más de datos puede llegar a desempeñarse muy bien.

Capítulo 9

Conclusiones

9.1. Conclusiones

Se construyeron varios modelos de ensamble para predecir el desenlace terapéutico del tratamiento de glucantime para la leishmaniasis cutánea.

Se logró seleccionar la representación de la información para facilitar el desempeño de los algoritmos que usan las técnicas de ensamble. Es decir, se logró un proceso de limpieza de los datos, que implicó identificar, sustituir o eliminar aquellos datos incompletos, incorrectos, o inexactos con el fin de alcanzar datos de calidad. Es así como se ataca uno de los problemas del conjunto de datos que fue disminuir la dimensionalidad de estos mismos.

Después del proceso de preparación de los datos, se realizó la estimación de los parámetros de los algoritmos de Bagging Classifier, Gradient Boosting Classifier, AdaBoost Classifier y Stacking Classifier. Esta estimación fue un proceso cuidadoso en el que se seleccionaron primeramente los parámetros y posibles valores que estos podrían tener según la documentación, y luego usando una búsqueda exhaustiva, combinada con la técnica de validación cruzada se obtuvieron los valores que mejor se ajustaron a los parámetros de dichos algoritmos. Luego de estimar los valores de los parámetros de las técnicas, se construyeron los cuatro modelos usando los algoritmos ya mencionadas.

Finalmente se realizaron evaluaciones de los distintos modelos construidos, para dichas evaluaciones se usó validación cruzada para garantizar que los resultados obtenidos sean independientes de las partición de los datos. Es así como se obtuvieron cuatro medidas que indican el desempeño de cada modelo.

Con las medidas obtenidas se permitió inferir que ningún modelo presentó un desempeño que permita confiar en sus predicciones. El modelo que presentó el mejor desempeño fue Gradient Boosting Classifier, sin embargo, la métrica indica que aún le falta construir hipótesis más fuertes para mejorar las predicciones. También indica que es la técnica que probablemente con un una cantidad un poco mayor de datos pueda entregar mejores predicciones.

Capítulo 10

Trabajos futuros

10.1. Trabajos futuros

Una vez se tenga una mayor cantidad de datos podría ser interesante repetir el mismo experimento descrito, ya que las técnicas de ensamble podrían construir estimadores base lo más diversos posibles, lo que aumentaría la probabilidad de que dichos estimadores base no se sobreajusten y puedan entregar predicciones un poco más precisas.

Además sería bueno experimentar la construcción de modelos usando técnicas de ensamble con un conjunto de datos que contengan características genéticas de los pacientes en lugar de demográficas, o una mezcla de ambas.

Otro experimento que también puede resultar interesante es aplicar técnicas de aumento de datos en conjunto con técnicas de ensamble, ya que como se mencionó anteriormente, con mayor cantidad de datos dichas técnicas de ensamble probablemente no se sobreajusten, y logren una construcción de estimadores base mucho más diversos que puedan establecer hipótesis un poco más fuertes.

Capítulo 11

Bibliografía

[1] "Leishmaniosis cutánea en Colombia y género."[Online]. Available:https://www.scielosp.org/article/csp/2001.v17n1/171-180/.[Accessed:02-Nov-2020].

[2] "Leishmaniasis: Qué es, Síntomas, Tratamientos e Información." [Online]. Available: https://cuidateplus. marca.com/enfermedades/ infecciosas/leishmaniasis.html. [Accessed: 05-Oct-2020].

[3] "Leishmaniasis." [Online]. Available: https://www.who.int/es/news-room/fact-sheets/detail/leishmaniasis. [Accessed: 05-Oct-2020].

[4] "Centro Internacional de Entrenamiento e Investigaciones Médicas." [Online]. Available: http://www.cideim.org.co/cideim/. [Accessed: 05-Oct-2020].

[5] "Leishmaniasis." [Online]. Available: http://www.cideim.org.co/cideim/es /investigacion/enfoquesdeinvestigacion/leishmaniasis.html. [Accessed: 05-Oct-2020].

[6] "SciELO - Salud Pública - Aspectos socioepidemiológicos y culturales de la leishmaniasis cutánea: concepciones, actitudes y prácticas en las poblaciones de Tierralta y Valencia, (Córdoba, Colombia) Aspectos socioepidemiológicos y culturales de la leishmaniasis cutánea: concepciones, actitudes y prácticas en las poblaciones de Tierralta y Valencia, (Córdoba, Colombia)." [Online]. Available: https://www.scielosp.org/article/scol/2017.v13n1/123-138/. [Accessed: 05-Oct-2020].

[7] INS, "Leishmaniasis-2018 Boletín epidemiologico Semana 31," pp. 1–58, 2018.

[8] "Definición de un modelo de aprendizaje automático".[Online]. Available:https://docs.microsoft.com/es-es/windows/ai/windows-ml/what-is-a-machine-learning-model. [Accessed: 02-Nov-2020].

[9] ."Ensembles: voting, bagging, boosting, stacking".[Online]. Available:https://www.iartificial.net/ensembles-voting-bagging-boosting-stacking/#Bagging. [Accessed:02-Nov-2020].

[10] Soumya(2021).Bagging vs Boosting in Machine Learning.[Figura].Recuperado de https://www.geeksforgeeks. org/bagging-vs-boosting-in-machine-learning/.

[11] Grootendorst, M.(2019).Stacking made easy with Sklearn.[Figura]. Recuperado de https://towardsdatascience.com/stacking-made-easy-with-sklearn-e27a0793c92b.

[12] Hidalgo Solís MJ, Víquez Redondo KF, Barrantes Valverde SM. "Leishmaniasis cutánea". Rev.méd.sinerg. [Online]. 1 de mayo de 2021 [citado 24 de noviembre de 2021];6(5):e674.

[13] Jacobs, A (2018).Leishmaniasis.[Figura].Recuperado de https://www.cronicascientificas.com/index.php/ediciones/edicion-ix-mayo-agosto-2018/221-leishmaniasis.

[14] N. Flores Balmaseda,Identificación de nuevos compuestos con potencial actividad antileishmaniásica medianteestudios in silico", tesis de maestría, Universidad Central Marta Abreu de las Villas, Facultad de Química y Farmacia, Santa Clara, Cuba, 2015.

[15] O. Salgado Vernier, Empleo de las técnicas de inteligencia artificial en la identificación de nuevos compuestos con potencial actividad contra Leishmania major", tesis de prgrado, Universidad Central Marta Abreu de las Villas, Facultad de Química y Farmacia, Santa Clara, Cuba, 2018.

[16] C. Muskus, "PREDICCIÓN DE MOLÉCULAS CON POTENCIAL ACTIVIDAD ANTI-LEISHMANIAEMPLEANDO ESTRATEGIAS COMPUTACIONALES", *Vitae*, vol. 19, pp 58-59, sep 2012.