Modelo predictivo de deserción de estudiantes universitarios utilizando técnicas de minería de datos

Trabajo de Grado

Kristein Johan Ordoñez López
Jhonathan Astudillo Astudillo

Director: MSc. Jimena Adriana Timaná Peña
Co-Director: PhD. Carlos Alberto Cobos Lozada

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información
Línea de Investigación: Sistemas Inteligentes
Popayán, Julio de 2021

Agradecimientos

El presente trabajo lo dedicamos principalmente a Dios, por ser el inspirador y darnos fuerza para continuar en este proceso de obtener uno de los anhelos más deseados.

Agradecemos a nuestra directora de tesis MSc. Jimena Adriana Timaná Peña y a nuestro co-director PhD. Carlos Alberto Cobos Lozada quienes, con su experiencia, conocimiento y motivación nos orientaron en la investigación.

A nuestros padres quienes nos apoyaron todo el tiempo.

A nuestros hermanos, familia y amigos, infinitas gracias por todo el apoyo brindado durante este proceso.

Tabla de contenido

Contenido

Lista de Tablas

Lista de Figuras

Lista de Fórmulas

Capítulo 1

Introducción

Uno de los principales desafíos que enfrenta el Sistema de Educación Superior Colombiano es disminuir los altos niveles de deserción académica que se presenta en el pregrado. Pese a que en los últimos años se han implementado estrategias que incluyen entre otros: 1) la mejora de la calidad y el volumen de la información que se entrega a los aspirantes sobre los programas ofrecidos, 2) la creación de programas de ayuda financiera para los estudiantes de bajos recursos o que provienen de otras ciudades, y 3) el acompañamiento psicológico de nivelación y orientación al estudiante, todavía el número de alumnos que no logra culminar sus estudios superiores es alto [1]. Según estadísticas del Ministerio de Educación Nacional, de cada cien estudiantes que ingresan a una institución de Educación Superior, cerca de la mitad no logra culminar su ciclo académico y obtener la graduación [2]. Gran parte de éstos, abandona sus estudios, principalmente en los primeros semestres.

Según El Ministerio de Educación Nacional la deserción universitaria en Colombia para año 2018 alcanzó una tasa del 8.79% [3]. Los expertos indican que la deserción en las instituciones universitarias se centra en los primeros años, concretamente en el primer o segundo semestre del programa de educación superior, bien sea porque los estudiantes no encuentran lo que esperaban al ingresar a determinado programa o porque la situación socioeconómica no les permite continuar. Así lo avalan estudios realizados en diferentes universidades a nivel mundial, las cuales han logrado identificar los factores más influyentes para que los estudiantes abandonen sus estudios, dentro de estos se encuentran: 1) el estrato socioeconómico, 2) el nivel de educación de los padres, 3) el puntaje con el que ingresó a la universidad, 4) limitaciones económicas y financieras, 5) bajo rendimiento académico, 6) desorientación profesional y vocacional, 7) dificultades para adaptarse al entorno universitario, 8) la larga duración de algunos de los programas y 10) la falta de flexibilidad para cambiar de carrera, entre otros [4, 5].

Otro resultado que preocupa hace referencia al tiempo que tardan los estudiantes de América Latina en completar una carrera profesional, con un promedio de un 36% más que en el resto del mundo. Por otra parte, los estudiantes muchas veces necesitan salir a trabajar para completar sus estudios, pero esto no siempre termina como se planeó, pues la mayoría de las veces los estudiantes terminan abandonando sus estudios por cumplir con responsabilidades laborales [6].

Las consecuencias de la deserción incluyen entre varios aspectos: 1) pérdidas financieras, tanto para el estudiante como para la institución y 2) menores tasas de graduación e indicadores no favorables para temas relacionados a la acreditación de alta calidad de un programa. Si una institución pierde a un estudiante por cualquier motivo, la institución tiene una tasa de abandono más alta. La identificación temprana de los estudiantes que están en riesgo de abandono, es fundamental para el éxito de cualquier estrategia que busque combatir la deserción [2].

1.1 Planteamiento del problema

Debido a la alta deserción de estudiantes universitarios, se han realizado diferentes investigaciones en todo el mundo que han llevado al desarrollo de modelos predictivos que tienen en cuenta datos sociodemográficos de estudiantes y que utilizan técnicas de minería de datos como los árboles de decisión, las redes neuronales, el algoritmo de los k vecinos más cercanos (K-nn), el algoritmo Random Forest, entre otros. Sin embargo, algunos de estos modelos implementados trabajan con un número limitado de datos, por lo que sus resultados no son totalmente confiables. Gran parte de estos modelos son alimentados por datos recopilados mediante formularios de registros de la propia universidad o a través de encuestas por parte de organizaciones gubernamentales de cada país. En general, estos modelos alcanzan una precisión que oscila entre el 74% y el 95%, pero utilizan generalmente pocos registros, mientras que los modelos que usan grandes volúmenes de datos su precisión no supera el 86%. Estos grupos de registros están desbalanceados, los datos desbalanceados provocan que los modelos tengan mayor dificultad cuando tratan de clasificar un abandono positivo, pues los algoritmos de clasificación tienden a ignorar las clases con menor número de apariciones (clase minoritaria), en este caso los estudiantes que si desertan de la universidad. Los datos de estos modelos tienen la mayoría de los registros como estudiantes que no desertan de la universidad (clase mayoritaria) y pocos registros con estudiantes que sí desertan de la universidad, es por esto que los modelos pueden presentar fallas en la clasificación de nuevas instancias [7].

Considerando que los modelos para la predicción de la deserción estudiantil revisados en el estado del arte con precisión alta son modelos entrenados y probados con pocos registros y datos desbalanceados, se puede decir que estos resultados no son totalmente fiables. Esto ha llevado a la siguiente pregunta de investigación, ¿Es posible mejorar la calidad en la predicción de la deserción estudiantil universitaria, mediante un modelo que hibride las mejores técnicas de minería de datos identificadas en el estado del arte y que opere sobre grandes volúmenes de datos balanceados?

Teniendo en cuenta lo anterior, en esta investigación se presenta una exhaustiva revisión sistemática, una caracterización de los atributos del estudiante que fueron identificados en las muestras obtenidas y un modelo híbrido que trabaja con las tres mejores técnicas de minería de datos reportadas en la revisión sistemática y con una función de optimización.

1.2 Objetivos

1.2.1 Objetivo general

- Proponer un modelo predictivo de deserción de estudiantes universitarios que hibride las mejores técnicas de minería de datos reportadas en el estado del arte y que use grandes volúmenes de datos balanceados buscando mejorar los reportes de calidad de predicción de deserción de estudiantes reportados a la fecha.

1.2.2 Objetivos específicos (OE)

- OE1. Definir las características (sociodemográficas, académicas, financieras) de una vista minable que soporte la predicción de deserción estudiantil universitaria basado en los reportes del estado del arte y las estrategias requeridas para su obtención en el entorno colombiano, con el fin de obtener las más relevantes.

- OE2. Proponer un modelo de predicción híbrido basado en la metodología CRISP-DM, la comparación de los resultados de los algoritmos identificados en el estado del arte y el balanceo previo de los datos suministrados por la Universidad del Cauca[1], para la predicción de la deserción de un estudiante universitario.

- OE3. Evaluar y comparar los resultados de los modelos identificados en el estado del arte y el propuesto, para determinar si hay mejora en la calidad de la predicción a través de la construcción de una matriz de confusión.

[1] En caso de no recibir los datos por parte de la Universidad del Cauca, se buscarán datos de otras fuentes similares o se simularán.

1.3 Aportes de investigación

Desde una perspectiva investigativa, la contribución de este proyecto está dada en el estudio integrativo y retrospectivo realizado, que se logró plasmar en la exhaustiva revisión sistemática de los artículos de la literatura científica relacionados con los modelos predictivos de deserción de estudiantes universitarios utilizando técnicas de minería de datos. También se realizó una caracterización de los atributos más comunes de estos modelos. Se logró además diseñar un modelo híbrido de deserción estudiantil universitaria haciendo uso de varias técnicas de minería de datos y empleando grandes volúmenes de datos previamente balanceados, permitiendo mejorar la calidad de la predicción de deserción de estudiantes universitarios. Se espera que este modelo propuesto pueda ser utilizado en futuras investigaciones a nivel nacional y/o internacional relacionadas con modelos de predicción de deserción estudiantil universitaria y sirva como referente para agilizar y simplificar el trabajo investigativo tanto en tiempo como en esfuerzo.

Desde la perspectiva de innovación se implementó el algoritmo de un modelo híbrido que permitió incorporar las mejores técnicas de minería de datos identificadas en el estado del arte, y a su vez empleando un algoritmo de optimización. Dicho modelo propuesto logró mejorar la calidad de la predicción de deserción de un estudiante universitario, utilizando un gran volumen de datos balanceados. A la fecha, en la literatura revisada no se encontraron modelos que hibriden las mejores técnicas de minería de datos para la predicción de la deserción estudiantil y que cuenten con un gran volumen de datos balanceados. Tanto el modelo como el algoritmo híbrido, estarán disponibles para la comunidad académica y científica, de modo que puedan ser usados en futuras investigaciones referentes a los modelos de predicción de deserción estudiantil universitaria.

1.4 Estructura del documento

A continuación, se presenta la estructura del documento de la monografía:

Capítulo 1 Introducción. En este capítulo se presenta la problemática que motivó a la realización de este proyecto de investigación. Adicionalmente, se presentan el planteamiento del problema, los objetivos y los aportes del proyecto.

Capítulo 2 Revisión sistemática. En este capítulo se presenta una revisión sistemática sobre modelos predictivos guiada por los lineamientos de Kitchenham, además se presenta la caracterización de los atributos o características comunes y más representativos de dichos modelos.

Capítulo 3 Diseño y construcción del componente Bagging. En este capítulo se presenta la definición de un componente Bagging, el diseño en pseudocódigo del componente bagging en sus diferentes versiones y se detalla los procesos utilizados para la implementación del algoritmo del componente que estuvo guiado por la metodología XP [8].

Capítulo 4 Diseño del modelo híbrido basado en CRISP-DM. En este capítulo se presenta la construcción de la vista minable, las etapas y los procesos utilizados para el diseño del modelo propuesto e integración con el componente Bagging, que estuvo guiado por la metodología CRISP-DM [9].

Capítulo 5 Conclusiones, Situaciones Presentadas y Trabajo Futuro. En este capítulo se presentan las conclusiones del trabajo realizado, situaciones que se presentaron en el desarrollo del mismo y trabajo futuro.

Anexos. Los anexos de este documento vienen divididos en 4 partes, A, B, C y D. El Anexo A explica a detalle la fase del ciclo de muerte de la metodología XP. El Anexo B expone el artículo denominado “Modelo híbrido predictivo de deserción de estudiantes universitarios utilizando técnicas de minería de datos” enviado a la revista Investigación e Innovación en Ingenierías, publindex B según Colciencias. El Anexo C expone el artículo denominado “Revisión sistemática de Modelos predictivos de deserción de estudiantes universitarios utilizando técnicas de minería de datos” enviado a la revista Investigación e Innovación en Ingenierías, publindex B según Colciencias y el Anexo D presenta el repositorio de código fuente del modelo propuesto.

Capítulo 2

Revisión sistemática

En este apartado se presenta el proceso realizado para la revisión sistemática de la literatura científica basada en los lineamientos propuestos por Kitchenham [10].

2.1 Revisión sistemática literatura científica

2.1.1 Preguntas de investigación

A continuación, se presenta la pregunta de investigación general que permitirá identificar aquellas propuestas científicas relacionadas con el presente trabajo investigativo:

"¿Qué estudios existen en la literatura científica sobre modelos predictivos de deserción de estudiantes universitarios utilizando técnicas de minería de datos?"

Se han planteado, además, una serie de preguntas de investigación más específicas, que permitirán contestar la pregunta de investigación general:

- PI1. ¿Cuáles de estos modelos están soportados bajo las técnicas de minería de datos?
- PI2. ¿Cuáles son las técnicas de minería de datos usadas en estos modelos predictivos?
- PI3. ¿Cuáles son los atributos más comunes usados en estos modelos predictivos?
- PI4. ¿Cuáles son las técnicas de minería de datos con mejor precisión en estos modelos predictivos?

Con respecto a PI1, se quiere tener un panorama de aquellos modelos predictivos de deserción de estudiantes universitarios existentes que trabajen específicamente bajo técnicas de minería de datos. Con respecto a PI2, se quiere identificar cuáles son específicamente las técnicas de minería de datos que se usan en cada uno de los modelos predictivos de deserción de estudiantes universitarios. Con respecto a PI3, se busca identificar cuáles son los atributos más frecuentes en los modelos predictivos encontrados. Con respecto a PI4, se quiere identificar cuáles son las técnicas de minería de datos que arrojan mejores resultados respecto a la precisión en los modelos predictivos encontrados.

2.1.2 Proceso de búsqueda

En esta sección se define la cadena de búsqueda y los criterios de selección que serán aplicados cuando se ejecute el proceso de búsqueda de la literatura científica.

2.1.2.1 Cadena de búsqueda

La cadena de búsqueda es la siguiente:

("predictive model" OR "predicting model" OR "predict model" OR "prediction model" OR "predictive approach" OR "predicting approach" OR "predict approach" OR "prediction approach" OR "Predicting Academic Performance" OR "Prediction Academic Performance" OR "Predict Academic Performance" OR "Predictive Academic Performance") AND ("student dropout" OR "student retention" OR "student desertion" OR "students dropout" OR "students retention" OR "students desertion" OR "university student dropout" OR "university student retention" OR "university student desertion") AND ("data mining" OR "data mining techniques" OR "educational data mining")

2.1.2.3 Criterios de selección

La Tabla 1, muestra los criterios de selección (inclusión, exclusión y calidad) identificados y que permitirán escoger aquellos estudios más acordes con el tema de investigación y excluir aquellos que no son relevantes para responder la pregunta de investigación.

Criterios de inclusión	
CI-1	artículos en idioma español e inglés
CI-2	artículos que traten de deserción estudiantil en el ámbito universitario
CI-3	full papers, conference papers, book chapter
Criterios de exclusión	
CE-1	artículos inferiores al año 2010
CE-2	artículos que solo presenten revisiones, opiniones, artículos que no detallan las técnicas de minería de datos empleadas.
CE-3	artículos que traten de mooc's o cursos en línea
CE-4	investigaciones similares o duplicadas del mismo autor
Criterios de Calidad	
CQ1	¿El estudio describe adecuadamente el proceso de creación del modelo de minería de datos? Respuestas posibles: (si = 1 / no = 0)
CQ2	¿El estudio utiliza un método de validación (caso de estudio, experimentos) donde se evalúe el modelo de minería de datos? Respuestas posibles: (si=1 / no=0)

CQ3	¿El estudio presenta los resultados detallados obtenidos en una fase de validación del modelo? Respuestas posibles (muy completo =1 medianamente =0.5 y no =0)
CQ4	¿El estudio ha sido publicado en una fuente reconocida? La clasificación del artículo se basa en el cuartil en el que fue publicado y/o en el publindex de la revista en el que fue publicado según Colciencias. Para las conferencias la clasificación se basó en el ranking de Investigación y educación en computación (CORE). (Cuartil 1 o revista/conferencia tipo A =2; cuartil 2 o revista/conferencia tipo B=1.5; cuartil 3 o revista/conferencia tipo C = 1; cuartil 4 =0.5; Sin indexación= 0)

Tabla 1. Criterios de selección

De los criterios de calidad de la Tabla 1, el puntaje total para evaluar una propuesta viene dado por la Fórmula 1.

$$P = CQ1 + CQ2 + CQ3 + CQ4$$

Fórmula 1. Suma de criterios de calidad

Donde *P* es el puntaje total y *CQn* es el puntaje obtenido en cada criterio.

2.1.3 Selección de artículos

La selección de los artículos resultantes de la revisión sistemática se llevó a cabo siguiendo 4 fases a saber: artículos encontrados al probar la cadena de búsqueda, filtrado de los artículos potenciales, aplicación de los criterios de inclusión y exclusión y selección de artículos finales de acuerdo a los criterios de calidad establecidos. En la primera fase, se probó la cadena de búsqueda sobre el indexador de artículos Scopus, dando como resultado una lista de 514 artículos. En la segunda fase, se seleccionó los artículos potenciales, resultado de filtrar aquellas propuestas que contenían al menos dos palabras claves en su título y tres en el resumen (*abstract*), con un resultado de 33 artículos. En la tercera fase, se aplicó los criterios de inclusión y exclusión definidos en la Tabla 1 sobre los artículos potenciales. Finalmente, en la cuarta fase, se aplicó los criterios de calidad a los artículos resultantes de la fase anterior. Dichas propuestas fueron puntuadas a través de la Fórmula 1. Los artículos finales seleccionados son aquellos que superen el umbral definido en el valor 3.0.

2.1.4 Resultados del proceso de búsqueda

En esta sección se presentan los resultados obtenidos en la investigación de la revisión sistemática.

2.1.4.1 Aplicación cadena de búsqueda

Al aplicar la cadena de búsqueda definida en la sección 2.1.2.1 sobre la base de datos bibliográfica Scopus se obtuvo un total de 514 artículos.

2.1.4.2 Selección de artículos potenciales

Para la selección de los artículos potenciales se estableció que estos debían tener al menos dos palabras claves en el título y mínimo tres en el resumen. El número total de propuestas que cumplieron estas condiciones se presentan en la Tabla 2.

Palabras clave	Cantidad de artículos encontrados	Total de artículos potenciales
"predictive model" , "predicting model" , "predict model", "prediction model" , "predictive approach", "predicting approach", "predict approach" , "prediction approach", “Predicting Academic Performance”, “Prediction Academic Performance”, “Predict Academic Performance”, “Predictive Academic Performance”, "student dropout", "student retention", "student desertion", "students dropout", "students retention", "students desertion", "university student dropout", "university student retention", "university student desertion", "university students dropout", "university students retention", "university students desertion", "data mining", "data mining techniques", "educational data mining".	514	33

Tabla 2. Resultados artículos potenciales

En la Tabla 3, se presentan los artículos potenciales seleccionados.

Id. artículo	artículo
1	Trends in information models on retention-university dropout [11]
2	University student retention: Best time and data to identify undergraduate students at risk of dropout [12]
3	Predicting Student Retention Among a Homogeneous Population Using Data Mining [13]
4	Review of techniques, tools, algorithms and attributes for data mining used in student desertion [14]

5	Identifying Students at Risk of Academic Failure Within the Educational Data Mining Framework [15]
6	University dropout prediction through educational data mining techniques: A systematic review [16]
7	Predicting academic performance of tertiary students using classification algorithm [17]
8	A machine learning approach to Predict the Engineering Students at risk of dropout and factors behind: Bangladesh Perspective [18]
9	Research and application of grade prediction model based on decision tree algorithm [19]
10	Students' Success Predictive Models Based on Selected Input Parameters Set [20]
11	From Lab to Production: Lessons Learnt and Real-Life Challenges of an Early Student-Dropout Prevention System [21]
12	An Early Feedback Prediction System for Learners At-Risk within a First-Year Higher Education Course [22]
13	Predictive Analysis for Student Retention by Using Neuro-Fuzzy Algorithm [23]
14	Predictive analytic models of student success in higher education: A review of methodology [24]
15	An analysis of student representation, representative features and classification algorithms to predict degree dropout [25]
16	A Hybrid Prediction Model Integrating a Modified Genetic Algorithm to K-means Segmentation and C4.5 [26]
17	The Analysis of Student Performance Using Data Mining [27]
18	Utilizing feature selection in identifying predicting factors of student retention [28]
19	Dropout situation of business computer students, University of Phayao [29]
20	Students' performance prediction model using meta-classifier approach [30]
21	Detection of desertion patterns in university students using data mining techniques: A case study [31]
22	Application of data mining for the detection of variables that cause university desertion [32]

23	An artificial neural network based early prediction of failure-prone students in blended learning course [33]
24	An approach to educational data mining model accuracy improvement using histogram discretization and combining classifiers into an ensemble [34]
25	A survey of machine learning approaches and techniques for student dropout prediction [35]
26	Self-organising maps and student retention: Understanding multi-faceted drivers [36]
27	Finding the best algorithms and effective factors in classification of Turkish science student success [37]
28	University dropout: A prediction model for an engineering program in Bogotá, Colombia [38]
29	Analytical approach for predicting dropouts in higher education [39]
30	The efficacy of learning analytics interventions in higher education: A systematic review [40]
31	Dropout early warning systems for high school students using machine learning [41]
32	Predictive data modeling: Educational data classification and comparative analysis of classifiers using python [42]
33	A predictive model for student outcomes using sparse coding – Hybrid features selection [43]

Tabla 3. Artículos potenciales seleccionados

2.1.4.3 Aplicación de criterios de inclusión y exclusión

La Tabla 4 presenta la aplicación de los criterios de inclusión y exclusión definidos en la sección 2.1.2.3 Criterios de selección, sobre los artículos potenciales.

ID. Artículo	CI-1	CI-2	CI-3	CE-1	CE-2	CE-3	CE-4	Cumplimiento
1	✓	✓	✓					si
2	✓	✓	✓					si
3	✓	✓						no
4	✓	✓	✓					si
5	✓	✓	✓					si
6	✓	✓	✓					si
7	✓		✓					no

8	✓	✓			X			no
9	✓	✓	✓					si
10	✓	✓	✓				X	no
11	✓	✓	✓					si
12	✓	✓	✓					si
13	✓	✓						no
14	✓							no
15	✓	✓	✓					si
16	✓		✓					no
17	✓		✓					no
18	✓		✓					no
19	✓	✓	✓					si
20	✓		✓					no
21	✓		✓					no
22	✓		✓					no
23	✓	✓	✓					si
24	✓	✓	✓					si
25	✓	✓	✓					si
26	✓							no
27	✓		✓					no
28	✓	✓	✓					si
29	✓	✓	✓					si
30	✓		✓					no
31	✓							no
32	✓	✓	✓					si
33	✓		✓					no

Tabla 4. Criterios de inclusión y exclusión

Las propuestas resultantes después de haber aplicado los criterios de inclusión y exclusión son consideradas como estudios primarios y se encuentran resumidos en la Tabla 5.

Id. Artículo	Autor	Tipo de artículo	Año
1	Guerra, Laura; Rivero, Dulce; Díaz, Eleazar; Arciniegas, Stalin.	Full paper	2020

2	José Maria Ortiz; Antonio Rua; Paloma Bilbao, Martí Casadesús.	Full paper	2020
4	K. Y Diaz Pedroza; B. Y Chindoy Chasoy; A. Rosado Gómez.	Conference paper	2019
5	Annalina Sarra; Lara Fontanella; Simone Di Zio.	Full paper	2019
6	Franceso Agrusti; Gianmarco Bonavolontá; Mauro Mezzini.	Full paper	2019
9	Yaling Zhang; Bei Wu.	Conference paper	2019
11	Alvaro Ortigosa; Rosa M. Carro; Javier Bravo.	Full paper	2019
12	David Baneres; M. Elena Rodriguez; Montse Serra.	Full paper	2019
15	Ruben Manrrique; Bernardo Pereira; Olga Marino.	Conference paper	2019
19	Pratya Nuankaew	Full paper	2019
23	Otgontsetseg Sukhbaatar; Lodoiravsal Choimaa; Tsuyoshi Usagawa	Full paper	2019
24	Dejan Rancić; Olivera Pronić-Rancić; Danijela Milošević.	Conference paper	2019
25	Neema Mduma; Khamisi Kalegele; Dina Machuve.	Full paper	2019
28	Andres Acero; Juan Camilo Achury; Juan Morales Piñero.	Conference paper	2019
29	Garima Jaiswal; Arun Sharma; Sumit Kumar Yadav.	Full paper	2019
32	Pratiyush Guleria; Manu Sood.	Conference paper	2018

Tabla 5. Propuestas primarias

2.1.4.4 Aplicación de criterios de calidad

Los criterios de calidad definidos en la Tabla 1, son aplicados sobre los estudios primarios presentados en la Tabla 5. Sobre cada uno de ellos se aplica la Ecuación 1 para obtener la puntuación final P de cada propuesta, dicha puntuación se presenta en la Tabla 6.

Id. Artículo	Revista de publicación	Puntuación				
		CQ1 si =1 no= 0	**CQ2** si =1 no = 0	**CQ3** muy completo =1 medianamente =0.5 no =0	**CQ4** cuartil 1 o tipo A =2 cuartil 2 o tipo B=1.5 cuartil 3 o tipo C = 1 cuartil 4 = 0.5 Sin indexación= 0	**P**
1	Associacao Iberica de Sistemas e Tecnologias de Informacao	0	1	0,0	1	**2,0**
2	Taylor & Francis Online	1	1	0,5	0	**2,5**
4	IOPScience	0	1	0,5	0	**1,5**
5	Springer Link	1	1	1,0	0	**3,0**
6	Je-LKS	1	1	1,0	0	**3,0**
9	ACM Digital Library	1	1	0,5	0	**2,5**
11	IEEE Xplore	1	1	1,0	0	**3,0**
12	IEEE Xplore	1	1	1,0	0	**3,0**
15	ACM Digital Library	1	1	1	0	**3,0**
19	iJET	1	1	1	0	**3,0**
23	iJET	0	1	0,0	1	**2,0**
24	Springer Link	1	1	0,5	0	**2,5**
25	Data Sicence Journal	0	1	0,5	0	**1,5**
28	ResearchGate	1	1	1,0	0	**3,0**
29	IGI Global	1	1	1,0	0	**3,0**

32	IEEE Explore	1	1	0,5	0	**2,5**

Tabla 6. Evaluación criterios de calidad

Los estudios primarios finalmente seleccionados fueron aquellos que igualaron o superaron un valor mínimo de calidad de 3.0. Las propuestas seleccionadas como estudios primarios de calidad se encuentran resaltadas en la Tabla 6.

2.1.4.5 Solución a las preguntas de investigación

A continuación, se contestan cada una de las preguntas de investigación específicas definidas en la sección 2.1.1 Preguntas de investigación.

PI1. ¿Cuáles de estos modelos están soportados bajo las técnicas de minería de datos?

Id. Artículo	Artículo
1	Trends in information models on retention-university dropout
2	University student retention: Best time and data to identify undergraduate students at risk of dropout
4	Review of techniques, tools, algorithms and attributes for data mining used in student desertion
5	Identifying Students at Risk of Academic Failure Within the Educational Data Mining Framework
6	University dropout prediction through educational data mining techniques: A systematic review
9	Research and application of grade prediction model based on decision tree algorithm
11	From Lab to Production: Lessons Learnt and Real-Life Challenges of an Early Student-Dropout Prevention System
12	An Early Feedback Prediction System for Learners At-Risk within a First-Year Higher Education Course
15	An analysis of student representation, representative features and classification algorithms to predict degree dropout
24	An approach to educational data mining model accuracy improvement using histogram discretization and combining classifiers into an ensemble(Conference Paper)

Tabla 7. Modelos predictivos soportados bajo técnicas de minería de datos

PI2. ¿Cuáles son las técnicas de minería de datos usadas en estos modelos predictivos?

Técnica de minería de datos	Id. artículo

RANDOM FOREST	4,15,24
DECISION TREE	2,4,6,9,11,12,24
NAIVE BAYES	4,5,6,12,15,24
LOGISTIC REGRESION	6
SVM	6,12,15
KNN	6,12
RED NEURONAL	6
NOT SPECIFIC	1
GRADIENT BOOT TREE	15

Tabla 8. Técnicas de minería de datos usadas por modelos predictivos

PI3. ¿Cuáles son los atributos más comunes usados en estos modelos predictivos?

Tipo de atributo	atributo	Id. artículo
No describe	No describe	9
Sociodemográfico	General	1,2,5,12
	Identificación	11
	Nacionalidad	6
	Ciudad procedencia	4
	Género	4,6
	Estado civil	4,6,11
	Edad	4,6,11
	Etnia	4
	Discapacidad física	6
	Ocupación padres	4,6,11
	Tamaño familia	4
	Disciplina	6
Académico	General	1,2,5,12
	Puntaje admisión	11
	Valor matrícula	6
	Edad inscripción	2
	Programa académico	4,
	Promedio académico	4,6,11,15,24
	Semestre	4,6,11
	Número materias	6
	Materias repetidas	4
	Promedio escuela	4
	Tipo admisión	15
	nivel estudios padres	6,11
	nivel inglés	2
	nivel matemáticas	9
	distancia residencia	11

Financiero	general	1
	ayuda familiar	4
	tipo vivienda	6
	trabaja	4
	estrato	4,11
	ingresos familiares	11

Tabla 9. Atributos identificados en los modelos predictivos

PI4. ¿Cuáles son las técnicas de minería de datos con mejor precisión en estos modelos predictivos?

Técnica / Id. Artículo	**1**	**2**	**4**	**5**	**6**	**9**	**11**	**12**	**15**
Random Forest									86%
Árbol Decisión		76%		94%	67%	81%	89%		
Naive Bayes					49%				57%
R. Logística					34%				
SVM									67%
KNN									
MLP									
Red Neuronal					40%				
Gradient Boot Tree									84%
Not Report	X		X					X	

Tabla 10. Precisión modelos predictivos según técnica de minería de datos utilizada

Capítulo 3

Diseño y construcción del componente Bagging

Este capítulo presenta el diseño de un componente Bagging que utilizará diferentes técnicas de minería de datos para predecir la deserción de estudiantes universitarios, basado en la metodología de desarrollo ágil eXtreme Programming (XP), el cual será empleado más adelante para la construcción del modelo híbrido.

Un componente Bagging es un método que permite ensamblar múltiples modelos, construidos a partir del mismo algoritmo (clasificador) base, que utilizan el mismo conjunto de datos para tomar la decisión de la clasificación de un nuevo registro por medio del voto mayoritario [44]. En la Figura 1, se presenta el diseño de un componente Bagging convencional.

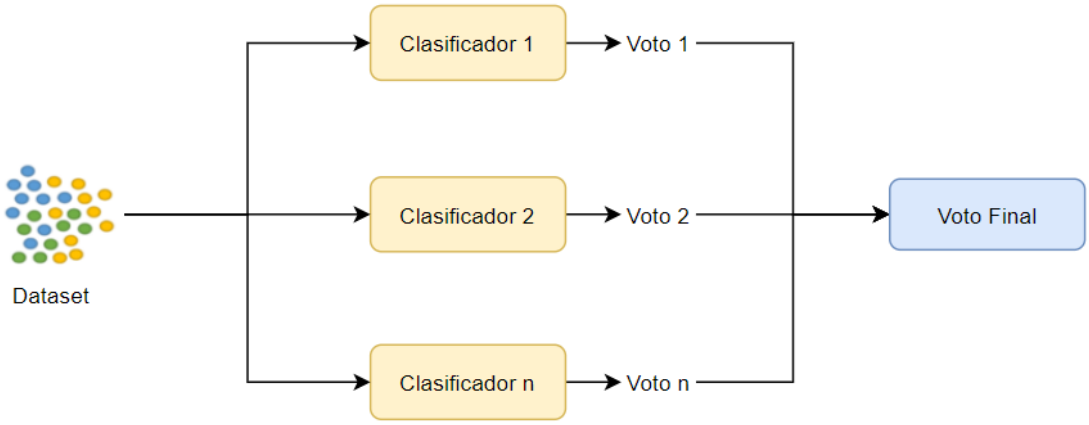

Figura 1. Diseño de un componente Bagging

3.1 Fase de Exploración

En esta fase, se exploraron las herramientas tecnológicas usadas para el diseño e implementación de los modelos Bagging. La herramienta *Rapidminer* [45] fue la herramienta seleccionada para la construcción del modelo, ya que es muy completa y fácil de usar. Además, para la implementación del modelo, se escogió el lenguaje de programación *Python* [46] y con el apoyo de la librería *Scikit-learn* que incluye diferentes algoritmos de clasificación [47].

3.2 Fase de Planificación

En esta fase se definieron las historias de usuario (HU) y se realizó la estimación de esfuerzo para cada una de ellas. Dicha estimación se basó usando la técnica de Planning Poker [48] basada en la secuencia de Fibonacci (1, 2, 3, 5, 8, 13, etc.), siendo 1 un valor de complejidad bajo y 13 un valor de complejidad muy alto.

En la Tabla 11, se presentan las Historias de usuario (HU) identificadas, las cuales fueron priorizadas considerando el orden en que se construiría el modelo Bagging.

Id	Rol	Característica / Funcionalidad	Razón / Resultado	Estimación
HU-01	Yo como desarrollador	Necesito leer el archivo Excel que contiene el conjunto de los datos de los estudiantes	Con el fin de poder usar los datos en el componente Bagging.	8
HU-02	Yo como desarrollador	Necesito separar los datos de los estudiantes en dos grupos	Para obtener los datos de entrenamiento y datos de prueba.	5
HU-03	Yo como desarrollador	Necesito definir el listado de clasificadores	Para poder usarlos en el componente Bagging.	8
HU-04	Yo como desarrollador	Necesito crear el componente bagging	Para entrenar el modelo con los datos de los estudiantes y la técnica de minería datos seleccionada previamente.	13
HU-05	Yo como desarrollador	Necesito obtener los pesos iniciales de los clasificadores	Para poder generar la precisión del modelo.	8
HU-06	Yo como desarrollador	Necesito obtener la precisión del modelo	Para determinar si se debe mejorar la precisión general del modelo.	8

HU-07	Yo como desarrollador	Necesito optimizar los pesos de los clasificadores	Para poder mejorar la precisión del modelo.	13
HU-08	Yo como desarrollador	Necesito generar el modelo Bagging	Para poder determinar la deserción de un estudiante.	13

Tabla 11. Historias de usuario

3.3 Iteración

En esta fase se efectuaron dos iteraciones: la primera iteración, se enfocó en el diseño del pseudocódigo y la implementación de las HU relacionadas al desarrollo del modelo Bagging. En la segunda iteración se refinó el diseño del modelo Bagging y se implementó las HU no abordadas y/o que requerían refinamiento.

3.3.1 Diseño del Pseudocódigo

En esta fase, se realizó el pseudocódigo del modelo Bagging en sus diferentes versiones.

3.3.1.1 Pseudocódigo del modelo Bagging Básico

En este apartado se presenta el pseudocódigo del modelo Bagging básico junto con su respectiva descripción y donde se realizará el proceso de entrenamiento y validación con un solo clasificador. En la Tabla 12, se presenta la descripción del modelo Bagging básico.

Función createBaggingModelBasic

```
function createBaggingModelBasic(ClasifierModel cModel)
1       dataset_train, dataset_test ← loadDatasets()
2       test, target ← splitDataset(dataset_test)
3       model ← buildBagging(dataset_train, cModel.classifier)
4       model.predict(test)
5       return round(model.score(test, target), 2)
end_function
```

Tabla 12. Función createBaggingModelBasic

En la línea 1, se crean las variables *dataset_train* y *dataset_test* donde se asignan los valores que retorna la función *loadDatasets*, la cual divide el conjunto de datos en dos subconjuntos, uno para entrenamiento y otro para pruebas. La función *loadDataset* se muestra en la Tabla 13. En la línea 2, la variable *test* guarda el conjunto de datos que se usarán como prueba en los modelos y la variable *target* guarda el conjunto de datos objetivo que retorna la función *splitDataset* que recibe como parámetro la variable *dataset_test*. La función *splitDataset* separa el conjunto de datos de la variable *dataset_test* en dos arreglos. El primer arreglo guarda los datos que se validarán en el modelo. El segundo arreglo, guarda los datos objetivos. La función *splitDataset* se muestra en la Tabla 15. En la línea 3, la variable *model* guarda el modelo Bagging entrenado que retorna la función *buildBagging* y que recibe como parámetros el dataset de entrenamiento y el objeto del clasificador de tipo *ClassifierModel*. La función *buildBagging* se muestra en la Tabla 16. En la línea 4, el objeto *model* invoca la función *predict* y se envía como parametro la variable *test* que contiene los datos de prueba para realizar la predicción del modelo. La función predict es propia de la librería *scikit-learn [47]*. En la línea 5, se retorna la precisión del modelo redondeado a dos decimales.

Función loadDatasets

En la Tabla 13, se muestra la función *loadDataset* encargada de leer el conjunto de datos del archivo .xlsx, además este conjunto de datos dividido en dos partes, un conjunto de datos para el entrenamiento y otro conjunto de datos para prueba, finalmente la función retorna los dos subconjuntos.

```
function loadDatasets()
1.  dataset ← getFileData()
2.  train, test ← split_train_and_test(dataset, test_size ← 0.30)
3.   return train, test
end_function
```

Tabla 13. Función loadDataSets

En la línea 1, a la variable *dataset* se le asigna los datos que serán usados por el modelo y que retorna la función *getFileData*, la cual se muestra en la Tabla 14. En la línea 2, la variable *train* guarda los datos de entrenamiento y la variable *test* guarda los datos de prueba, que son asignados por defecto en 30% que retorna la función *train_test_split***.** La cual recibe como parámetros la variable *dataset* que contiene el conjunto de datos a usar en el modelo y valor de porcentaje que se usa para repartir los datos entre las variables *train* y *test*. En la línea 3, se retornan los datos para entrenamiento y prueba del modelo en las variables *train* y *test***.**

Función getFileData

En la Tabla 14, se presenta la función *getFileData*, encargada de cargar el dataset, el cual se usará para entrenamiento y prueba del modelo, esta función lee un archivo .xlsx que contiene el conjunto de datos de los estudiantes, asigna sus valores a la variable *dataset* y lo retorna para ser utilizado.

```
function getFileData()
1.  url ← "filename.xlsx"
2.  data_sheet ← pd.read_excel(url)
3.  dataset ← data_sheet.values
4.  return dataset
end_function
```

Tabla 14. Función getFileData

En la línea 1, la variable *url* guarda el nombre del archivo que contiene el conjunto de los datos. En la línea 2, a la variable *data_sheet* se le asigna el objeto de los datos leídos desde el archivo mediante la función *read_excel*, la cual es propia de la librería usada para la construcción del modelo y que recibe como parámetro la variable *url***.** En la línea 3, a la variable *dataset* se le asigna el valor del conjunto de los datos a través de la función *data_sheet.values*. En la línea 4 se retorna la variable *dataset* que contiene el conjunto de datos a usar en el modelo.

Función splitDataset

En la Tabla 15, se presenta la función *splitDataset*, que recibe como parámetro el dataset de prueba y divide dicho dataset en dos conjuntos, uno que contiene únicamente la variable objetivo y otro con los demás atributos, finalmente la función retorna estas dos listas.

```
function splitDataset(dataset)
1.  size ← len(dataset[0]) - 1
2.  data_train ← dataset[0: size]
3.  data_target ← dataset[size]
4.  return data_train, data_target
end_function
```

Tabla 15. Función splitDataset

En la línea 1, la variable *size* guarda el tamaño de las columnas del objeto *dataset* que contiene el conjunto de datos. En la línea 2, a la variable *data_train* se le asigna los datos del objeto *dataset* desde la columna 1 hasta la penúltima columna. En la línea 3, a la variable *data_target* se le asigna los datos de la última columna del objeto *dataset*. En la línea 4 se retornan las variables *data_train* y *data_target*.

Función buildBaggin

En la Tabla 16, se presenta la función *buldBagging* que recibe como parámetro el dataset de pruebas y el clasificador con el que se quiere entrenar y validar el modelo. Esta función se encarga de definir el número de clasificadores con el que se construirá el ensamble Bagging para luego ser entrenado con el dataset.

```
function buildBagging(dataset, classifier)
1.  num_classifiers ← 10
2.  train, target ← splitDataset(dataset)
3.  model_bagging ← BaggingClassifier(base_estimator←classifier,
n_estimators←num_classifiers)
4.  model_bagging.fit(train, target) //training
5.  return model_bagging
end_function
```

Tabla 16. Función buildBaggin

En la línea 1, la variable *num_classifiers* guarda el número de clasificadores que tendrá el componente Bagging. En la línea 2, a la variable *train* se le asigna el conjunto de datos a entrenar y en la variable *target* se le asigna el conjunto de datos objetivo que retorna la función *splitDataset* y que recibe como parámetros la variable *dataset*. La función *splitDataset* se muestra en la tabla 4. En la línea 3, a la variable *model_bagging* se le asigna el objeto del modelo Bagging que retorna la función *BaggingClassifier* y que recibe como parámetro la variable *classifier* que contiene el objeto del clasificador y la variable *num_classifiers* que contiene el número de clasificadores. En la línea 4, la variable *model_bagging* invoca la función *fit* que entrenara el modelo con las variables *train* y *target.* La función *fit* es propia de la librería de entrenamiento. En la línea 5 se retorna la variable *model_bagging* del modelo entrenado.

3.3.1.2 Pseudocódigo del modelo Bagging Híbrido

En este apartado se presenta el modelo Bagging Híbrido, este modelo utiliza tres modelos Bagging internos, donde cada modelo emplea uno de los tres clasificadores que reportaron mejor desempeño en la revisión sistemática. Finalmente, los resultados de cada modelo se evalúan a través del voto mayoritario, esto con el fin de obtener un único resultado para la toma de la decisión. A continuación, en la Figura 2, se presenta el diseño del modelo Bagging Híbrido.

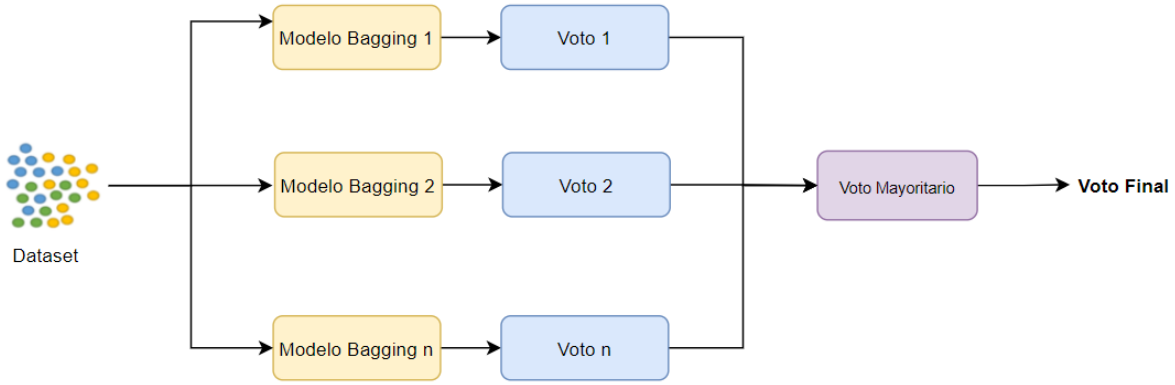

Figura 2. Diseño del modelo Bagging Híbrido

En la Tabla 17, se presenta la descripción del pseudocódigo del modelo Bagging Híbrido.

Función createBaggingHybridModel

```
function createBaggingHybridModel(List[ClasifierModel] models)
1       array_clasiffier ← models
2       dataset_train, dataset_test ← loadDatasets()
3       test, target ← splitDataset(dataset_test)
4       array_predicted ← [ ]
5       for item ← 0 to array_clasiffier-1 do
6               model ← buildBagging(dataset_train, item.classifier)
7               predicted ← model.predict(test)
8             array_predicted.append(predicted)
9       end_for
10      precision ← precisionModelByVote(array_predicted, target)
11      return precision
end_function
```

Tabla 17. Función createBaggingHybridModel

En la línea 1, se crea la variable *array_clasiffier* y se le asigna el listado de clasificadores de la variable *models,* la variable *array_clasiffier* contendrá los clasificadores de tipo *ClassifierModel* a los que se les construirá un ensamble de tipo Bagging. En la línea 2, se asigna en las variables *dataset_train* y *dataset_test* los valores que retorna la función *loadDatasets*, la cual distribuye el dataset en dos subconjuntos, un conjunto para entrenamiento y el otro conjunto para pruebas. La función *loadDataset* se muestra en la Tabla 13. En la línea 3, la variable *test* guarda el conjunto de datos que se usarán como prueba en los modelos y la variable *target* guarda el conjunto de datos objetivo que retorna la función *splitDataset* que recibe como parámetro la variable *dataset_test*. La función *splitDataset* separa el conjunto de datos de la variable *dataset_test* en dos arreglos, en el primer arreglo se guardarán los datos para validar el modelo y el segundo arreglo guardar los datos

objetivos. La función *splitDataset* se muestra en la Tabla 15. En la línea 6, se crea la variable *array_predicted* y se inicializa como una lista vacía donde posteriormente se guardarán los modelos entrenados. De la línea 5 a la línea 9, se crea el proceso iterativo de construcción de ensamble tipo Bagging y se realizará el entrenamiento de cada uno de los modelos guardado en la lista *array_clasiffier*. En la línea 6, la variable *model* guarda el modelo Bagging entrenado que retorna la función *buildBagging* y que recibe como parámetros el dataset de entrenamiento y el objeto del clasificador de tipo *ClassifierModel*. La función *buildBagging* se muestra en la Tabla 16. En la línea 7, el objeto *model* invoca la función *predict* y se envía como parámetro la variable *test* que contiene los datos de prueba para realizar la validación del modelo y finalmente asignar el resultado a la variable *predicted*. En la línea 8, se agrega la variable *predicted* a la lista *array_predicted*. En la línea 13, a la variable *precisión* se le asigna el valor de la precisión del modelo que retorna la función *modelPrecisionByVote* donde se le envían como parámetros la variable *array_predicted* que contiene la lista de los modelos entrenados y la variable *target* que contiene el conjunto de datos objetivo. La función *modelPrecisionByVote* calcula la precisión general del modelo y se muestra en la Tabla 18. En la línea 11, se retorna la precisión del modelo redondeado a dos decimales.

Función modelPrecisionByVote

```
function precisionModelByVote(array_predicted, target)
1       count_hits ← 0
2       num_predicted ← array_predicted-1
3       for i ←0 to target - 1 do
4               value ← 0
5               abandoment ← 0
6               for j ← 0 to num_predicted -1 do
7                   if array_predicted[ j ][ i ] == 1) then
8                      abandonment ← abandonment + 1
9               end_for
10              if abandoment > (num_predicted - abandoment) then
11                  value ← 1
12              end_if
13              if(value == target[ i ]) then
14                  count_hits ← count_hits + 1
15              end_if
16      end_for
17      return round(count_hits / len(target), 2)
end_function
```

Tabla 18. Función precisionModelByVote

En la línea 1, se crea la variable *count_hits* donde se guardará el valor de las coincidencias entre los resultados de los modelos entrenados y los datos objetivo. En la línea 2, a la variable *num_predicted* se asigna el tamaño de la lista

array_predicted. De la línea 3 a la línea 16, se recorren los datos de la lista de *target* para encontrar las coincidencias de los resultados de los modelos. En la línea 4, se crea la variable *value* para guardar el valor de la coincidencia del resultado del modelo. En la línea 5 se crea la variable *abandoment* que contendrá la cantidad de registros de abandono por cada iteración. De la línea 6 a la línea 9 se recorre el listado de los modelos *array_predicted* para obtener los resultados de las predicciones. En la línea 7 pregunta si el valor obtenido en la iteración es igual al valor de abandono. Si es así, en la línea 8, la variable *abandoment* incrementa en 1. En la línea 10 pregunta si el valor de la variable *abandoment* es mayor a la variable *num_predicted* que contiene el tamaño de la lista de modelos entrenados. Si es así, en la línea 11 a la variable *value* le asigna el valor de 1. En la línea 13 pregunta si la variable *value* es igual al valor del *target[i].* Si es así, en la línea 14 a la variable *count_hits* el valor se incrementa en 1. En la línea 17 retorna el resultado de la variable *count_hits* sobre el tamaño de la lista de *target* redondeado a dos decimales.

3.4 Producción

En esta fase se colocó en funcionamiento los tres modelos Bagging básicos y el modelo Bagging Híbrido. Cada modelo Bagging básico, se entrenó y validó con un clasificador específico. Además, el modelo Bagging Híbrido fue entrenado y validado conjuntamente con los tres clasificadores. Los clasificadores usados en los modelos Baging básicos fueron los que reportaron mejores resultados en la revisión sistemática. Estos son: *Árbol de Decisión, Naive Bayes y Support Vector Machine*.

A continuación, se expone el funcionamiento de cada uno de los modelos Bagging.

3.4.1 Funcionamiento de los modelos Bagging básicos

Para cada uno de los modelos Bagging básicos se aplicaron las siguientes métricas: precisión, precisión de los verdaderos positivos, la sensibilidad (Recall) y área bajo la curva (AUC). El conjunto de datos fue balanceado previamente.

En la Figura 3, se presentan los resultados del modelo Bagging Árbol de decisión realizado en Python.

```
C:\Users\ordo112245\AppData\Local\Microsoft\WindowsApps\python3.8.exe
Modelo Bagging Simple
Clasificador:  Árbol de decisión
Precisión: 0.67
Precision TP: 0.7
Recall: 0.64
AUC: 0.75

Process finished with exit code 0
```

Figura 3. Resultados del modelo Bagging Árbol de decisión realizado en Python

En la Figura 4, se presentan los resultados del modelo Bagging Naive Bayes realizado en Python.

.

```
C:\Users\ordo112245\AppData\Local\Microsoft\WindowsApps\python3.8.exe
Modelo Bagging Simple
Clasificador:  Naive Bayes
Precisión: 0.54
Precision TP: 0.54
Recall: 0.83
AUC: 0.52

Process finished with exit code 0
```

Figura 4. Resultados del modelo Bagging Naive Bayes realizado en Python

En la Figura 5, se presentan los resultados del modelo Bagging Support Vector Machine realizado en Python.

```
C:\Users\ordo112245\AppData\Local\Microsoft\WindowsApps\python3.8.exe
Modelo Bagging Simple
Clasificador:  Support Vector Machine
Precisión: 0.54
Precision TP: 0.54
Recall: 0.83
AUC: 0.52

Process finished with exit code 0
```

Figura 5. Resultados del modelo Bagging Support Vector Machine realizado en Python

3.4.2 Funcionamiento del modelo Bagging Híbrido

En la Figura 6, se presentan los resultados del modelo Bagging Híbrido realizado en Python.

```
C:\Users\ordo112245\AppData\Local\Microsoft\WindowsApps\python3.8.exe
Modelo Bagging Híbrido
----------------------------
Clasificador: Árbol de decisión
Precisión: 0.69
----------------------------
Clasificador: Naive Bayes
Precisión: 0.54
----------------------------
Clasificador: Support Vector Machine
Precisión: 0.54
----------------------------
Precisión final: 0.69
Precision TP: 0.7
Recall: 0.67
AUC: 0.76

Process finished with exit code 0
```

Figura 6. Resultados del modelo Bagging Híbrido realizado en Python

Después de haber validado los tres modelos Bagging básicos y el modelo Bagging Híbrido, se observó que los resultados de las métricas aplicadas no fueron los esperados. Debido a esto, se procedió a refinar el modelo Bagging Híbrido.

3.5 Mantenimiento

En este apartado se procedió a refinar el modelo Bagging Híbrido, para ello se agregó un nuevo proceso de optimización que se ejecutará después de entrenar el modelo. Este nuevo proceso se encargará de recorrer el conjunto de resultados del modelo, le asignará un peso correspondiente a cada registro del conjunto y finalmente enviará los datos a una función de optimización, la cual tratará de mejorar la precisión final del modelo.

En el proceso de optimización se usaron tres algoritmos y se crearon tres nuevas versiones del modelo Híbrido. A cada modelo se le agregó una función de optimización diferente. Además, se realizó el proceso de validación cruzada para intentar mejorar los resultados de la predicción. [49]. Las funciones de optimización usadas en las versiones de los modelos son a saber: Hill Climbing [50], Simulated Annealing [51], Step by Grid [52].

A continuación, se presentan los pseudocódigos de las tres versiones de los modelos Bagging Híbridos con su respectiva función de optimización.

3.5.1 Pseudocódigo del modelo Bagging Híbrido Hill Climbing (MH-HC)

En la Figura 7, se presenta el diseño del modelo Bagging Híbrido junto con la función de optimización Hill Climbing.

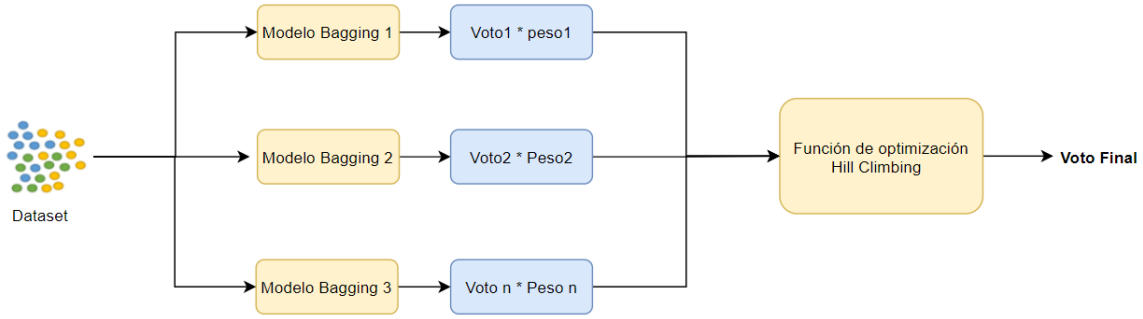

Figura 7. Diseño del modelo Bagging Híbrido con función de optimización Hill Climbing

A continuación, en la Tabla 19, se presenta el algoritmo en pseudocódigo del modelo Bagging Híbrido junto con la función de optimización Hill Climbing.

Función createBaggingHybridModelHillClimbing

```
function createBaggingHybridModelHillClimbing(int IT, boolean weight,
List[ClasifierModel] models)
1       N ← IT
2       weight_model ← weight
3       array_clasiffier ← models
4       dataset ← loadDatasets()
5       train, target ← splitDataset(dataset)
6       array_predicted ← [ ]
7       for item ← 0 to array_clasiffier-1 do
8               model ← buildBagging(dataset, item.classifier)
9               model_predicted ← cross_val_predict(model, train, target, 10)
10              array_predicted.append(model_predicted)
11      end_for
12      array_weight ← initializeWeight(array_clasiffier, weight_model)
13      precision_x ← precisionModelByWeighted (array_predicted, array_weight,target)
14      for i ← to N-1 do
15                      array_s ← optimizeWeight(array_weight)
16                      precision_s ← precisionModelByWeighted(array_predicted,
array_weight, target)
17      end_for
18      if precision_s > precision_x then
19              precision_x ← precision_s
20              array_weight ← array_s
21      end_if
22      return presicion_x
end_function
```

Tabla 19. Función createBaggingHybridModelHillClimbing

En la línea 1, a la variable *N* se asigna el número de iteraciones que se realizarán para que el modelo encuentre los mejores valores de pesos y se evalúen con los resultados, esto con el fin de mejorar la calidad de precisión del modelo. Estos elementos se explican detalladamente en una sección posterior. En la línea 2, a la variable *weigth_model* se asigna un valor de tipo booleano, en caso de ser verdadero el proceso del modelo se desarrollará con pesos iniciales para cada submodelo preestablecido, de lo contrario los pesos para los modelos internos se asignarán por partes iguales. En la línea 3, se guardan en *array_clasiffier* los clasificadores de tipo *ClassifierModel* a los que se les construirá un ensamble de tipo Bagging. En la línea 4, a la variable *dataset* se asigna el conjunto de datos que retorna la función *loadDatasets*. La función *loadDataset* se muestra en la Tabla 13. En la línea 5, la variable *train* guarda el conjunto de datos que se usarán en los modelos y la variable *target* guarda el conjunto de datos objetivo que retorna la función *splitDataset* que recibe como parámetro la variable *dataset*. La función *splitDataset* separa el conjunto de datos de la variable *dataset_test* en dos arreglos, el primer arreglo de datos para probar el modelo y el segundo arreglo los datos objetivos. La función *splitDataset* se muestra en la Tabla 15. En la línea 6, se crea la variable *array_predicted* y se inicializa como una lista vacía donde posteriormente se guardarán los modelos entrenados. De la línea 7 a la línea 11, se realiza el proceso iterativo de construcción de ensamble tipo Bagging y entrenamiento de cada uno de ellos guardado en la lista *array_clasiffier*. En la línea 8, la variable *model* guarda el componente Bagging entrenado que retorna la función *buildBagging* y que recibe como parámetros el dataset de entrenamiento y el objeto del clasificador de tipo *ClassifierModel*. La función *buildBagging* se muestra en la Tabla 16. En la línea 9, a la variable *model_predicted* se asigna el modelo entrenado que retorna la función *cross_val_predict,* a la cual se le envían como parámetros la variable *train* que contiene el conjunto de datos, la variable *target* que contiene los datos objetivos y la constante 10 que indica el número de folders. La función *cross_val_predict* realiza el proceso de validación cruzada y es propia de la librería *scikit-learn*. En la línea 10, el valor de la variable *model_predicted* que contiene el modelo entrenado se agrega la lista de los modelos entrenados *array_predicted*. En la línea 12, a la variable *array_weight* se le asignan los pesos iniciales de los clasificadores que retorna la función *initializeWeight* y que recibe como parámetros la variable *array_clasiffier* que contiene el listado de los clasificadores, y la variable *default_weight_model* que define si toma o no los pesos por defecto del clasificador. La función *initializeWeight* se muestra en la Tabla 20. En la línea 13, a la variable *precision_x* se le asigna el valor global de la precisión del modelo que retorna la función *precisionModelByWeighted* y que se le envían como parámetros la variable *array_predicted* que contiene la lista de los modelos entrenados, la variable *array_weight* que contiene los pesos iniciales de los modelos y la variable *target* que contiene el conjunto de datos objetivo. La función *precisionModelByWeighted*

calcula la precisión general del modelo y se muestra en la Tabla 22. De la línea 14 a la línea 17, se da inicio al proceso iterativo que permite calcular la mejor precisión del modelo. En la línea 15, a la variable *array_s* se le asignan los pesos optimizados de los modelos que retorna la función *optimizeWeight* y que recibe como parámetro la variable *array_weight* que contiene los pesos iniciales de los modelos. La función *optimizeWeight* optimiza los pesos de los modelos y se muestra en la Tabla 21. En la línea 16, a la variable *precision_s* se le asigna el valor la precisión del modelo que retorna la función *modelPrecision*. En la línea 18 se pregunta si la precisión *precision_s* es mayor a la precisión global *precision_x*. si es así, en la línea 19, a la variable *precision_x* se le asigna el valor de la variable *precision_s* y en la línea 20, a la variable *array_weight* se le asigna el valor de la variable *array_s* que contiene los pesos optimizados de los modelos.

Función initializeWeight

La función *initializeWeight* se presenta en la Tabla 20, esta función es la encargada de inicializar los pesos para cada uno de los clasificadores que componen el modelo Bagging Híbrido, en caso de no tener pesos preestablecidos para cada clasificador los asigna por partes iguales.

```
function initializeWeight(array_classifier, weight_model)
1   array_weight ← [ ]
2   if weight_model == False then
3       sum_value ← 0
4       value ← round(1/len(array_classifier), 2)
5       for i←0 to array_classifier-1 do
6                       if i == (len(array_classifier) -1) then
7                       value ← round((1 - sum_value), 2)
8                       end_if
9                       array_weight.append(value)
10              sum_value ←  sum_value + value
11      end_for
12  end_if
13  else then
14      for classifier in array_classifier:
15              if classifier.weight_init is not None do
16                      array_weight.append(classifier.weight_init)
17              end_if
18       end_for
19  end_else
20  return array_weight
end_function
```

Tabla 20. Función initializeWeight

En la línea 1, se crea la variable *array_weight* que contendrá el listado de los pesos. En la línea 2 pregunta si la variable *weight_model* es igual a falso, lo que indica que

debe calcular los pesos. Si es así, en la línea 3 se crea la variable *sum_value* que se utilizará posteriormente para el cálculo de los pesos. En la línea 4, a la variable *value* se le asigna el valor de 1 sobre el número de clasificadores y redondeando a 2 decimales. De la línea 5 a la línea 11 se recorre la lista de los clasificadores *array_classifier* para asignar el peso inicial a cada clasificador. En la línea 6 pregunta si la posición actual del clasificador es igual a la última posición de la lista de clasificadores *array_classifier*. Si es así, entonces en la línea 7, a la variable *value* le asigna 1 menos a la variable sum_value. En la línea 9, a la lista de pesos *array_weight* le agrega el peso de la variable *value*. En la línea 10, a la variable *sum_value* le suma el valor de la variable *value*. Si no, de la línea 14 a la línea 18 se recorre la lista de los clasificadores *array_classifier* para generar los pesos iniciales de los modelos. En la línea 15 pregunta si la variable *weight_init* de cada clasificador es diferente de nulo. Si es así, en la línea 16, se agrega a la lista de pesos *array_weight* el valor del peso que tiene el clasificador *weight_init*. En la línea 20 se retorna la variable *array_weight* que contiene la lista de los pesos.

Función optimizeWeight

En la Tabla 21, se presenta la función *optimizeWeight* que recibe como parámetro la lista de pesos de cada clasificador que contiene el modelo, esta función es la encargada de optimizar los pesos de dichos clasificadores. En primer parte, se selecciona aleatoriamente el clasificador para optimizar su peso, también se selecciona de forma aleatoria el porcentaje a optimizar de ese clasificador y por último asigna los nuevos pesos en la lista y la retorna.

```
function optimizeWeight(array_x)
1.   p ←random.randint(0, (len(array_x)-1))
2.   a ←random.uniform(-0.1, 0.1)
3.   array_x[p] ← array_x[p] + a
4.   if array_x[p] < 0 then
5.      array_x[p] ← 0
6.   end_if
7.   if array_x[p] > 1 then
8.      array_x[p] ← 1
9.    end_if
10.   total ←sum(array_x)
11.   for i in range(len(array_x)) do
12.      array_x[i] ← array_x[i] / total
13.   end_for
14.   return array_x
end_function
```

Tabla 21. Función optimizeWeight

En la línea 1, se asigna a la variable *p* una posición aleatoria del arreglo de pesos, para hacerle una variación a dicho peso. En la línea 2, a la variable *a,* se le asigna la variación que se hará el peso ubicado en la posición previamente escogida

aleatoriamente. En la línea 3 se reemplaza el peso en la posición *p* por el peso en la posición *p* más la variación. En la línea 4, se pregunta si el peso en la posición *p* es menor a 0 y si es así en la línea 5, se le asigna a dicha posición el valor 0. En la línea 6, se pregunta si el peso en la posición p es mayor a 1 y si es así en la línea 7, se le asigna a dicha posición el peso de 1. En la línea 10 a la variable *total* se le asigna la suma de los pesos que están en la lista *array_x*. De la línea 11 a la 13 se recorre el arreglo de pesos para asignar los nuevos pesos normalizados. En la línea 11 en cada posición del *array_x* se asigna su nuevo valor correspondiente. En la línea 14, se retorna los nuevos pesos optimizados que están en la variable *array_x*.

Función precisionModelByWeighted

En la Tabla 22, se presenta la función *precisionModelByWeighted*, que recibe como parámetros la lista de predicciones de los clasificadores, la lista de pesos de los clasificadores y la lista de los variables objetivo. Esta función se encarga de verificar cuantos aciertos tuvo cada modelo y retornar la precisión que obtuvo cada uno de ellos.

```
function precisionModelByWeighted (array_predicted, array_weight, target)
1 umbral ← 0.5
2 count_hits ← 0
3       for i ←0 to array_predicted[0]-1 do
4               value ← 0
5               array_result ← [ ]
6               for j ←0 to array_weight-1 do
7                       array_result.append(array_predicted[j][i] *   array_weight[j])
8               end_for
9               total ← sum(array_result)
10              if total > umbral then
11                      value ← 1
12              end_if
13              if(value == target[i]) then
14                      count_hits +← 1
15              end_if
16      end_for
17 return round(count_hits / len(target), 2)
end_function
```

Tabla 22. Función precisionModelByWeighted

En la línea 1, a la variable *umbral* se le asigna el valor por defecto de 0.5 para posteriormente ser usada en el cálculo de la precisión. En la línea 2, se crea la variable *count_hits* para guardar el valor de las coincidencias entre los datos de los modelos entrenados y los datos objetivo. De la línea 3 a la línea 16, se recorren los datos de la lista de *target* para encontrar las coincidencias de los resultados de los modelos. En la línea 4, se crea la variable *value* para guardar el valor de la

coincidencia del resultado del modelo. En la línea 5 se crea la variable *array_result* que contendrá el listado de los resultados de las predicciones. De la línea 6 a la línea 8 se recorre el listado de los modelos *array_predicted* para obtener los resultados de las predicciones. En la línea 7, a la lista de resultados *array_result* se agrega el resultado de multiplicar el resultado obtenido en cada iteración del modelo por el valor del peso. En la línea 9, a la variable *total* se agrega la suma de los resultados de la lista *array_result*. En la línea 10 pregunta si la variable *total* es mayor a la variable *umbral*. Si es así, en la línea 11 a la variable *value* le asigna el valor de 1, este valor indica que un estudiante deserto. En la línea 13 pregunta si la variable *value* es igual al valor del *target[i]*. Si es así, en la línea 14 a la variable *count_hits* el valor se incrementa en 1. En la línea 17 retorna el resultado de la variable *count_hits* sobre el tamaño de la lista de *target* indicando la precisión del modelo.

3.5.2 Pseudocódigo del modelo Bagging Híbrido Step by Grid (MH-SG)

A continuación, se presenta el algoritmo en pseudocódigo del modelo Bagging Híbrido junto con la función de optimización Step by Grid, dicha función toma los clasificadores del modelo y empieza a iterar de 1 a 100 cada uno. Luego se asigna un peso a cada resultado del clasificador de forma iterativa. Por cada iteración se calcula la precisión y el resultado se guarda en un arreglo junto con los pesos. Finalmente se exporta un archivo txt con los resultados de las precisiones. En la Tabla 24 se presenta su descripción.

En la Figura 8, se presenta el diseño del modelo Bagging Híbrido con la función de optimización Step by Grid.

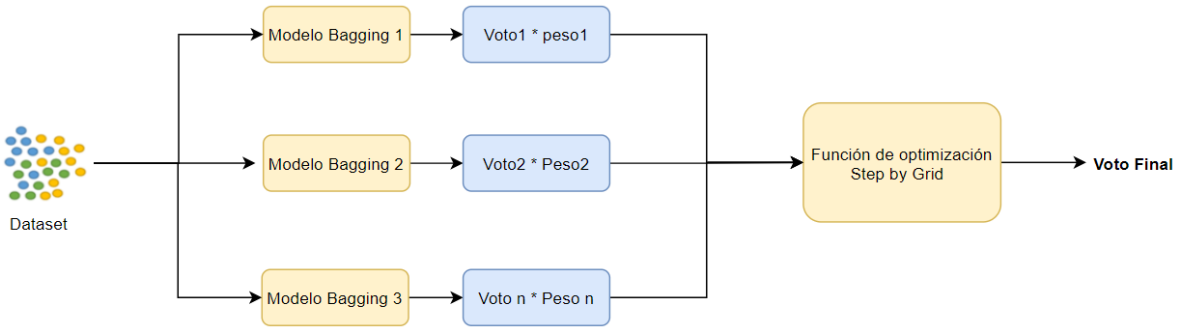

Figura 8. Diseño del modelo Bagging Híbrido con función de optimización Step by Grid

Función createBaggingHybridModelStepByGrid

```
function createBaggingHybridModelStepByGrid(boolean weight, List[ClasifierModel]
clasiffierModels)
1       weight_model ← weight
2       array_clasiffier ← clasifierModels
3       dataset ← loadDatasets()
4       train, target ← splitDataset(dataset)
5       array_predicted ← [ ]
6       for item ← 0 to array_clasiffier-1 do
7               model ← buildBagging(dataset, item.classifier)
8               model_predicted ← cross_val_predict(model, train, target, 10)
9               array_predicted.append(model_predicted)
10      end_for
11    precisionGrid(array_predicted, target)
end_function
```

Tabla 23. Función createBaggingHybridModelStepByGrid

En la línea 1, a la variable *weigth_model* se asigna un valor de tipo booleano, en caso de ser verdadero el proceso del modelo se desarrollará con pesos iniciales para cada submodelo preestablecido, de lo contrario los pesos para los modelos internos se asignarán por partes iguales. En la línea 2, se guardan en *array_clasiffier* los clasificadores de tipo *ClassifierModel* a los que se les construirá un ensamble de tipo Bagging. En la línea 3, a la variable *dataset* se asigna el conjunto de datos que retorna la función *loadDatasets*. La función *loadDataset* se muestra en la Tabla 13. En la línea 4, la variable *train* guarda el conjunto de datos que se usarán en los modelos y la variable *target* guarda el conjunto de datos objetivo que retorna la función *splitDataset* que recibe como parámetro la variable *dataset.* La función *splitDataset* separa el conjunto de datos de la variable *dataset_test* en dos arreglos, el primer arreglo de datos para probar el modelo y el segundo arreglo los datos objetivos. La función *splitDataset* se muestra en la Tabla 15. En la línea 5, se crea la variable *array_predicted* y se inicializa como una lista vacía donde posteriormente se guardarán los modelos entrenados. De la línea 6 a la línea 10, se realiza el proceso iterativo de construcción de ensamble tipo Bagging y entrenamiento de cada uno de ellos guardado en la lista *array_clasiffier*. En la línea 7, la variable *model* guarda el modelo entrenado que retorna la función *buildBagging* y que recibe como parámetros, el dataset de entrenamiento y el objeto del clasificador de tipo *ClassifierModel*. La función *buildBagging* se muestra en la Tabla 16. En la línea 8, a la variable *model_predicted* se asigna el modelo entrenado que retorna la función *cross_val_predict,* a la cual se le envían como parámetros la variable *train* que contiene el conjunto de datos, la variable *target* que contiene los datos objetivos y la constante 10 que indica el número de folders. La función *cross_val_predict* realiza el proceso de validación cruzada y es propia de la librería *scikit-learn*. En la línea 9, el valor de la variable *model_predicted* que contiene el modelo entrenado se agrega la lista de los modelos entrenados *array_predicted*. En la línea 11, se invoca la

función *precisionGrid* que se encargara de optimizar los resultados del modelo donde se envían como parámetros la lista de los modelos entrenados y la lista del target. La función *precisionGrid* se presenta en la Tabla 24.

Función precisionGrid

```
function precisionGrid(array_predicted, target)
1       array_weight ← [0, 0, 0]
2       size_weigth ← len(array_weight)
4       for pos ← 0 to size_weigth -1 do
5               for i ←0 to 99 do
6                       item_value ← (i+1)*0.01
7                       weight ← 1 – ítem_value
8                     value ← weight / (size_weigth-1)
9                       sum_value ← 0
10                    for j ← 0 to size_weigth -1 do
11                      if j == pos
12                          array_weight[j] ←item_value
13                        else
14                          if j == size_weight – 1
15                                value ←weight – value
16                        array_weight[j] ←value
17                        sum_value ← sum_value + value
18                      end_if
19                    end_for
20              end_for
21               precision ← precisionModelByWeighted(array_predicted, array_weight,
target)
22              file_data ←array_weight + precisión
23      end_for
24       writeToFile(file_data)
end_function
```

Tabla 24. Función precisionGrid

En la línea 1, a la variable *array_weight* se crea el arreglo de pesos con el valor por defecto en 0. En la línea 2, se crea la variable *size_weigth* para guardar la cantidad de los clasificadores. De la línea 4 a la línea 23, se recorren los clasificadores. De la línea 5 a la línea 20 se recorre cada peso que va de 1 hasta llegar a 100. En la línea 6, se crea la variable *item_value* para guardar el valor del peso, el cual se irá incrementando. En la línea 7, se crea la variable *weight* que contiene los pesos de los clasificadores restantes. En la línea 8, se crea la variable *value*, donde se obtiene el valor del peso que se guardara en el arreglo de pesos. De la línea 10 a la línea 19 se recorre el listado de los clasificadores para asignar los pesos al arreglo de pesos. En la línea 11, pregunta si la posición *j* del arreglo de pesos es la misma posición del primer ciclo. Si es así, en la línea 12, al arreglo de pesos *array_weight* en la posición *j* se le asigna el valor de la variable *ítem_value*. De lo contrario, en la

línea 14, pregunta si la posición *j* es igual a la última posición del arreglo de pesos *array_weight.* Si es así, en la línea 15, a la variable *value* le asigna el valor de la variable *weight* menos el valor de la variable *value.* En la línea 16, al arreglo de pesos *array_weight* en la posición *j* le asigna el valor de la variable *value*. En la línea 17, a la variable *sum_value* se le suma el valor de la variable *value.* En la línea 21 se obtiene el valor de la precisión, invocando la función *precisionModelByWeighted*, enviando como parámetros el arreglo de modelos entrenados, el arreglo de los pesos y el target. En la línea 22, a la variable *file_data* se le agrega la lista de los pesos y la precisión del modelo por cada iteración. En la línea 24 se crea el archivo txt que contiene el resultado de los pesos y las precisiones.

3.5.3 Pseudocódigo del modelo Bagging Híbrido Simulated Annealing (MH-SA)

En la Figura 9, se presenta el diseño del modelo Bagging Híbrido junto con la función de optimización Simulated Annnealing.

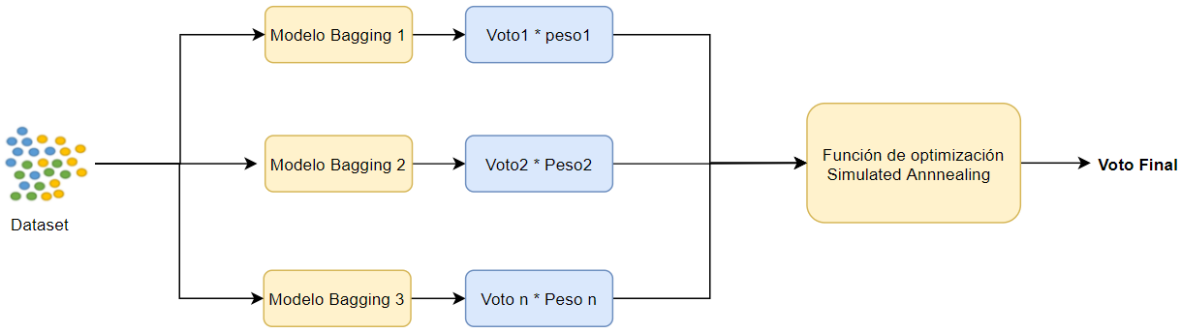

Figura 9. Diseño del modelo Bagging Híbrido con función de optimización Simulated Annealing

A continuación, en la Tabla 25, se presenta el algoritmo en pseudocódigo del modelo Bagging Híbrido junto con la función de optimización Simulated Annnealing.

Función createBaggingHybridModelSimulatedAnnealing

```
CreateBaggingHybridModelSimulatedAnnealing(int IT boolean weight,
List[ClasifierModel] clasiffierModels)
1       N ←IT
2       weight_model ← weight
3       array_clasiffier ← clasifierModels
4       dataset ← loadDatasets()
5       train, target ← splitDataset(dataset)
6       array_predicted ← [ ]
7       for item ← 0 to array_clasiffier-1 do
8               model ← buildBagging(dataset, item.classifier)
9               model_predicted ← cross_val_predict(model, train, target, 10)
```

```
10                array_predicted.append(model_predicted)
11       end_for
12      current_score ←precisionModelByWeighted(array_predicted, array_weight, target)
13       solution_weight ← array_weight
14       for i ← to N-1 do
15          getNeighbor(array_weight)
16          score_diff ←precisionModelByWeighted(array_predicted, array_weight, target)
- current_score
17          if score_diff > 0
18               solution_weight ← array_weight
19               current_score ← precisionModelByWeighted(array_predicted,
array_weight, target)
20          else
21               if random(0,1) < Math.Exp(-score_diff / current_score)
22                    solution_weight ←array_weight
23               end_if
24          end_if
25      end_for
end_function
```

Tabla 25. Función createBaggingHybridModelSimulatedAnnealing

En la línea 1, a la variable *N* se asigna el valor del parámetro de entrada *IT* que contiene el número de iteraciones que se realizaran para que el modelo encuentre los mejores valores de pesos y se evalúen con los resultados, esto con el fin de mejorar la calidad de precisión del modelo. Estos elementos se explican detalladamente en una sección posterior. En la línea 2, a la variable *weigth_model* se asigna un valor de tipo booleano, en caso de ser verdadero el proceso del modelo se desarrollará con pesos iniciales para cada submodelo preestablecido, de lo contrario los pesos para los modelos internos se asignarán por partes iguales. En la línea 3, se guardan en *array_clasiffier* los clasificadores de tipo *ClassifierModel* a los que se les construirá un ensamble de tipo Bagging. En la línea 4, a la variable *dataset* se asigna el conjunto de datos que retorna la función *loadDatasets*. La función *loadDataset* se muestra en la Tabla 13. En la línea 5, la variable *train* guarda el conjunto de datos que se usarán en los modelos y la variable *target* guarda el conjunto de datos objetivo que retorna la función *splitDataset* que recibe como parámetro la variable *dataset*. La función *splitDataset* separa el conjunto de datos de la variable *dataset_test* en dos arreglos, el primer arreglo de datos para probar el modelo y el segundo arreglo los datos objetivos. La función *splitDataset* se muestra en la Tabla 15. En la línea 6, se crea la variable *array_predicted* y se inicializa como una lista vacía donde posteriormente se guardarán los modelos entrenados. De la línea 7 a la línea 11, se realiza el proceso iterativo de construcción del ensamble tipo Bagging y entrenamiento de cada uno de ellos guardado en la lista *array_clasiffier.* En la línea 8, la variable *model* guarda el modelo Bagging entrenado que retorna la función *buildBagging* y que recibe como parámetros el dataset de entrenamiento y el objeto del clasificador de tipo *ClassifierModel*. La función *buildBagging* se muestra en la Tabla 16. En la línea 9, a la variable *model_predicted* se asigna el modelo entrenado que retorna la función

cross_val_predict, a la cual se le envían como parámetros la variable *train* que contiene el conjunto de datos, la variable *target* que contiene los datos objetivos y la constante 10 que indica el número de folders. La función *cross_val_predict* realiza el proceso de validación cruzada y es propia de la librería *scikit-learn*. En la línea 10, el valor de la variable *model_predicted* que contiene el modelo entrenado se agrega la lista de los modelos entrenados *array_predicted*. En la línea 12, a la variable *current_score* se le asigna el valor de la precisión del modelo mediante la función *precisionModelByWeighted*. En la línea 13, se crea la variable *solution_weight* la cual se le asigna el arreglo de pesos *array_weight*. De la línea 14 a la línea 25, se da inicio al proceso iterativo que permite calcular la mejor precisión del modelo basándose en el algoritmo *Simulated Annnealing*. En la línea 15, se invoca la función *getNeighbor* que recibe como parámetro el arreglo de pesos *array_weight*. La función *getNeighbor* se muestra en la Tabla 26. En la línea 16, a la variable *score_diff* se le asigna el valor de la precisión que se obtiene mediante la función *precisionModelByWeighted* menos el valor de la variable *current_score*. En la línea 17 pregunta si el valor de *score_diff* es mayor a cero. Si es así, en la línea 18, a la variable *solition_weight* se le asigna el valor del arreglo de pesos *array_weight*. En la línea 19, a la variable *current_score* se le asigna el nuevo valor de la precisión mediante la función *precisionModelByWeighted* enviando como parámetros el arreglo de modelos *array_predicted* y el nuevo arreglo de pesos *array_weight*. Si no, en la línea 21 pregunta, si el valor decimal aleatorio entre 0 y 1 es menor que el valor de la función exponencial de la variable *score_diff* sobre la variable *current_score*. Si es así, en la línea 22, a la variable *solution_weight* se le asigna el valor del arreglo de pesos *array_weight* que contiene los pesos optimizados de los modelos.

Función getNeighbor

```
function getNeighbor(array_weight)
1    pos ← random(0, len(array_weight)-1)
2    radom_weight ←random(-0.1, 0.1)
3    array_weight[pos] ← array_weight[p] – radom_weight
4     if array_weight[pos] < 0
5         array_weight[pos] ← 0
6     end_if
7     If array_weight[pos] > 1
8         array_weight[pos] ←1
9     end_if
10   total ← suma(array_weight)
11   for i ← to array_weight -1 do
12       array_weight[i] ← array_weight[i] / total
13  end_for
end_function
```

Tabla 26. Función getNeighbor

En la línea 1, a la variable *pos* se le asigna el valor aleatorio entre 0 y el tamaño del arreglo de pesos. Esto para obtener la posición aleatoria del arreglo de pesos a la que se modificara el valor del peso. En la línea 2, a la variable *radom_weight* se le asigna el valor aleatorio entre -0.1 y 0.1. En la línea 3, en la posición *pos* del arreglo de pesos array_weight se asigna el valor de la misma posición arreglo de pesos menos el valor de la variable *radom_weight*. En la línea 4, pregunta si el valor del arreglo de pesos en la posición *pos* es menor a 0. Si es así, en la línea 5, al arreglo de pesos en la posición *pos* se asigna el valor de 0. En la línea 7, pregunta si el valor del arreglo de pesos en la posición *pos* es mayor a 1. Si es así, en la línea 8, al arreglo de pesos en la posición *pos* se asigna el valor de 1. En la línea 10, a la variable *total*, se le asigna el valor de la suma de los pesos. De la línea 11 a la línea 13, se recorre el arreglo de pesos. En la línea 12, al arreglo de pesos en la posición *i* se asigna el valor del peso en la misma posición sobre el valor de la variable *total* que contiene la suma de los pesos. Este proceso se realiza para normalizar cada peso del arreglo.

3.6 Ciclo de muerte

La documentación para el componente Bagging se presenta en el anexo A.

Capítulo 4

Diseño del modelo híbrido basado en CRISP-DM

En este capítulo se presenta el proceso de elaboración de la vista minable y el diseño del modelo de predicción de deserción estudiantil basado en la metodología CRISP-DM. A continuación, se describen cada una de las fases.

4.1 Comprensión del negocio

En esta fase, se definió el plan para la construcción del modelo, cuya finalidad fue determinar los objetivos y requisitos del proyecto. También, en esta fase del proceso, se determinó el objetivo principal que es realizar el modelo de predicción de deserción de estudiantes de la Universidad del Cauca a partir de datos balanceados.

Se estableció como criterio de éxito la posibilidad de realizar la predicción de deserción de un estudiante con un gran porcentaje de precisión. De tal forma que puede ser útil a futuro para la Universidad del Cauca determinar diferentes atributos que pueden definir la deserción de un estudiante. Además, tener una alerta temprana de quienes van a desertar y poder realizar diferentes estrategias para evitar el abandono de los estudiantes.

4.1.1 Valoración de la situación

Se contó con la base de datos de los estudiantes pertenecientes a la Universidad del Cauca, matriculados en diferentes programas de pregrado del año 2018 hasta el año 2019, por lo que se puede afirmar que se dispone de una gran cantidad de datos para resolver el problema. Esta información incluye datos sociodemográficos, datos académicos y datos financieros que son muy útiles al momento de realizar la minería de datos.

4.1.2 Objetivos de la minería de datos

Los objetivos en términos de minería de datos son:

- Predecir si un estudiante abandonará la universidad a partir de sus datos de matrícula.

- Caracterizar los atributos más relevantes relacionados a la deserción del estudiante.

4.1.3 Plan del Proyecto

En el proyecto se contemplan las siguientes etapas generales:

- *Comprensión de los datos,* se llevará a cabo la recopilación inicial de los datos, la descripción de los datos, la exploración y verificación de la calidad los datos.
- *Preparación de los datos*, se llevará a cabo la selección de los datos, limpieza de los datos, construcción de los datos, integración de los datos y balanceo de los datos.
- *Modelado*, se creará un plan de pruebas para determinar la calidad del modelo, se seleccionarán las técnicas de minería de datos y se usará el componente Bagging creado en la primera etapa para realizar la clasificación de los datos con las técnicas identificadas como las adecuadas en el proceso de la revisión sistemática.
- *Evaluación*, se evaluará el modelo, teniendo en cuenta los criterios de éxito definidos en la primera fase. Al final de la fase, si se han generado resultados satisfactorios, se procederá a continuar con la siguiente fase, de lo contrario se iniciará otra iteración desde la fase de modelado.
- *Despliegue,* se ejecutará el modelo con otro conjunto de datos diferente al de entrenamiento y se documentará el proceso de modelado para su posterior implementación.

4.2 Comprensión de los datos

En esta fase se realizó la recolección, descripción, exploración y verificación de los datos. A continuación, se detallan los pasos realizados.

4.2.1 Recolección de los datos

Los datos utilizados en este proyecto, son datos relacionados a estudiantes universitarios de pregrado, los cuales incluyen información sociodemográfica y académica. En este apartado, se extrajeron los atributos relacionados a aquellos atributos más representativos de la revisión sistemática, ya que estos datos son importantes para determinar la deserción de un estudiante. A continuación, en la Tabla 27, se presentan los atributos identificados.

Tipo de atributo	Atributo
Sociodemográfico	Identificación
	Tipo de identificación
	Edad del estudiante
	Género
	Estado civil
	Discapacidad
	Grupo étnico
	Municipio de procedencia
	Departamento de procedencia
	Nombres del estudiante
	Dirección de residencia
Académico	Programa académico
	Facultad
	Estudiante nuevo
	Último semestre cursado
	Puntaje de admisión
	Colegio de procedencia
	Regionalización

Tabla 27. Atributos identificados en el conjunto de datos

4.2.2 Descripción de los datos

En esta fase se presenta la caracterización de los datos que fueron encontrados en archivos Excel separados por periodo desde el año 2018 hasta el año 2019. En un archivo Excel, se encontraron los datos de los estudiantes que se matricularon en los periodos mencionados anteriormente. Mientras que en otro archivo Excel se encontraron los datos de los estudiantes que desertaron en esos mismos periodos.

4.2.2.1 Atributos sociodemográficos

A continuación, en la Tabla 28, se presenta la descripción de los datos sociodemográficos identificados.

Atributo	Descripción
Identificación	tipo de dato alfanumérico, este campo identifica al estudiante
Tipo identificación	Tipo de dato alfanumérico
Edad del estudiante	Tipo de dato numérico.
Género	Tipo de dato alfanumérico
Estado Civil	Tipo de dato alfanumérico

Discapacidad	Tipo de dato alfanumérico, este campo indica si un estudiante matriculado tiene alguna discapacidad.
Grupo Étnico	Tipo de dato alfanumérico, este campo indica si un estudiante pertenece a un grupo étnico.
Municipio de procedencia	Tipo de dato alfanumérico, este campo indica el municipio de procedencia del estudiante.
Departamento de procedencia	Tipo de dato alfanumérico, este campo indica el municipio de procedencia del estudiante.
Nombres del estudiante	Tipo de dato alfanumérico
Dirección de residencia	Tipo de dato alfanumérico, este campo indica el lugar de residencia del estudiante.

Tabla 28. Descripción de los datos sociodemográficos

4.2.2.2 Atributos académicos

A continuación, en la Tabla 29, se presenta la descripción de los datos académicos identificados.

Atributo	Descripción
Programa académico	Tipo de dato numérico, este campo indica el programa al que pertenece el estudiante.
Facultad	Tipo de dato alfanumérico, este campo indica la facultad del programa al que pertenece el estudiante.
Estudiante Nuevo	Tipo de dato alfanumérico, este campo indica si un estudiante es de primer semestre o es un estudiante antiguo.
Último semestre cursado	Tipo de dato numérico, este campo indica el último semestre cursado por el estudiante.
Puntaje de admisión	Tipo de dato numérico, este campo indica el puntaje con el que ingreso el estudiante a la universidad.
Colegio de procedencia	Tipo de dato alfanumérico, este campo indica el colegio donde el estudiante realizo sus estudios de bachiller.
Regionalización	Tipo de dato alfanumérico, este campo indica si el estudiante matriculó en algún programa de regionalización.

Tabla 29. Descripción de los datos académicos

4.3 Exploración de los datos

En este apartado, mediante la herramienta *Rapidminer* se procedió a explorar el conjunto de datos, con la finalidad de observar cómo se distribuye cada uno de los atributos en función a la variable clase. A continuación, se presentan algunos gráficos de los atributos más relevantes.

En la Figura 10, se observa los estudiantes que desertaron por programa académico.

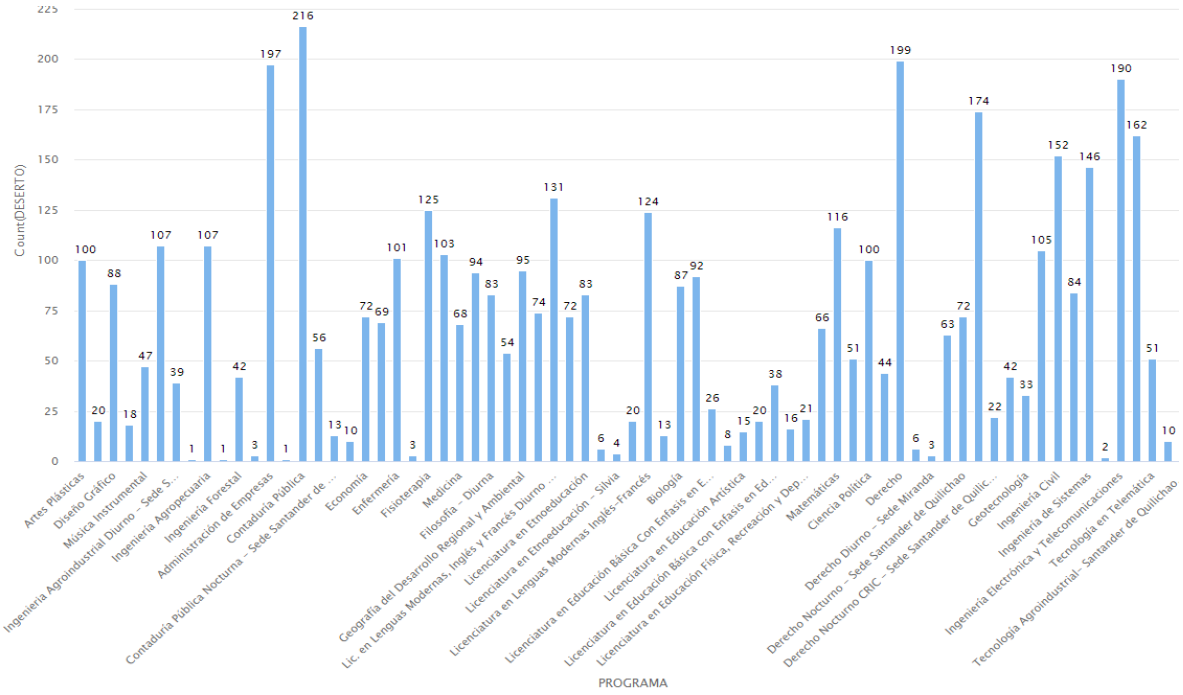

Figura 10. Estudiantes desertores por programa académico

En la Figura 11, se observa la cantidad de estudiantes clasificados por estrato y que desertaron.

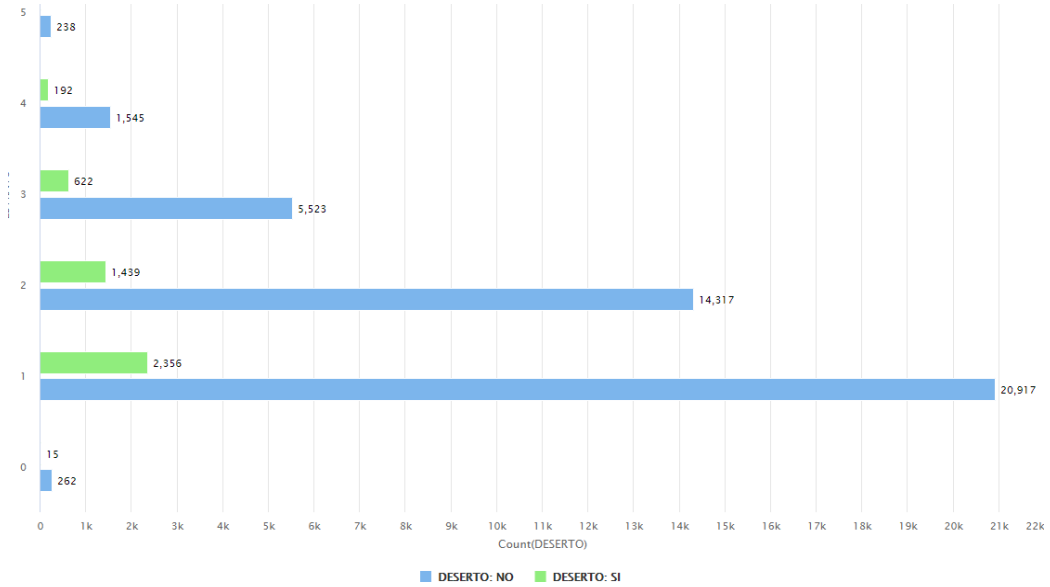

Figura 11. Estudiantes desertores por estrato

En la Figura 12, se observa los estudiantes de pueblos indígenas que desertaron.

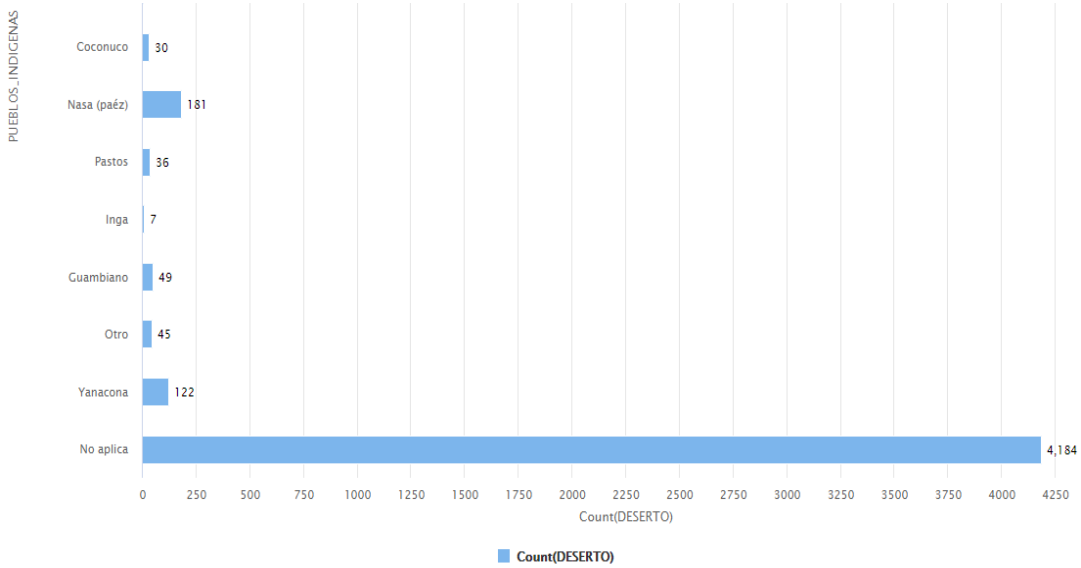

Figura 12. Estudiantes desertores por comunidad indígena

En la Figura 13, se observa los estudiantes con algún tipo de discapacidad que desertaron.

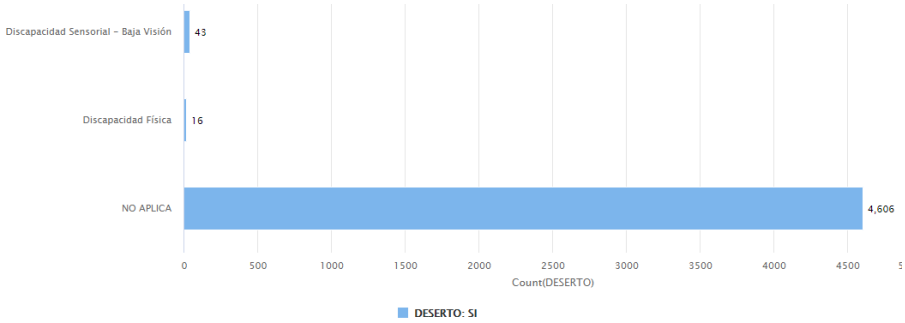

Figura 13. Estudiantes desertores por discapacidad

4.3.1 Calidad de los datos

Después de haber realizado la exploración de los datos, se observó que la gran mayoría de los datos están completos y no contienen errores. Sin embargo, se encontraron algunos atributos con una gran cantidad de valores nulos, donde en una etapa siguiente se procederá a retirarlos del conjunto de datos. También se evidenció que faltan algunos atributos que se consideraron relevantes ya que tenían un gran número de apariciones en los artículos de la revisión sistemática. Estos fueron: Último semestre cursado, promedio académico, edad del estudiante, si el estudiante trabaja. Por lo que más adelante se procederá a simularlos.

4.4 Preparación de los datos

En esta fase se llevó a cabo el proceso de elaboración de la vista minable donde se seleccionaron y prepararon los datos para que pudieran ser usados en las técnicas de minería de datos. Este proceso implicó realizar las siguientes etapas.

4.4.1 Selección de los datos

En esta etapa, se obtiene un conjunto de datos unificado que contiene información de los estudiantes matriculados en el segundo periodo del año 2018 hasta el segundo periodo del año 2019. Sin embargo, en el conjunto de datos se encontraron atributos duplicados o que no fueron relevantes para los objetivos, por lo que se procedió a retirar algunos de ellos.

Los atributos que fueron seleccionados, son aquellos que tienen relación con el objetivo de la minería de datos, que se definió en la fase 1 de la metodología. También se tuvieron en cuenta los atributos que tienen más relevancia en la revisión sistemática. En la Tabla 30, se presentan los atributos seleccionados.

Atributos seleccionados
Identificación
Tipo identificación
Estrato
Género
Estado civil
Discapacidad
Comunidad Negra
Comunidad Indígena
Municipio de procedencia
Departamento de procedencia
Programa académico
Desertó

Tabla 30. Atributos seleccionados

4.4.2 Limpieza de los datos

En esta etapa se logró identificar los atributos que presentan una gran cantidad de datos nulos o vacíos. Por lo tanto, se procede con la eliminación de dichos atributos. Estos se presentan en la Tabla 31.

Atributos eliminados
Nombres del estudiante
Dirección de residencia
Último semestre
Puntaje de admisión
Regionalización
Colegio de procedencia
Edad del estudiante
Facultad
Grupo Étnico

Tabla 31. Atributos eliminados

Con el apoyo de la herramienta *Rapidminer* se identificaron los atributos del conjunto de datos que tienen datos nulos o vacíos, para luego proceder a quitarlos. En la Figura 14, se presenta el proceso realizado. En el paso A, se presenta el conjunto de datos. Seguidamente en el paso B, se identifican los atributos nulos mediante el operador *Missing Values*. En el paso C, se eliminan los tributos con datos nulos usando el operador *Filter Examples*. Finalmente, en el paso D, se obtiene el conjunto de datos filtrado.

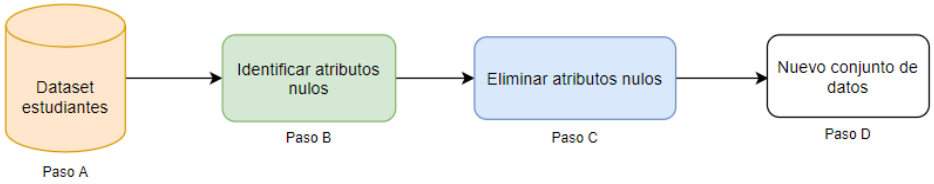

Figura 14. Proceso remoción de datos nulos

Por otro lado, en el atributo de *Programa académico* se identifica que tiene registros de programas pertenecientes a posgrados, por lo que se procedió a eliminar esos registros y dejar solo registros de estudiantes de pregrado.

4.4.3 Construcción de los datos

En esta etapa se crearon algunos de los atributos que fueron eliminados y aquellos atributos que no fue posible obtener de forma real por parte de la Universidad del Cauca. Posteriormente se creó una matriz de correlación para determinar los atributos con más relación a la variable objetivo donde se seleccionaron los siguientes atributos. En la Tabla 32, se presentan los atributos simulados que fueron seleccionados.

Atributos simulados seleccionados
Último semestre cursado
Promedio académico
Estudiante trabaja
Edad del estudiante

Tabla 32. Atributos simulados seleccionados

4.4.4 Integración de los datos

Debido a que en la revisión sistemática se identificaron los atributos relevantes relacionados a la deserción de un estudiante, se observó que algunos de estos atributos no estuvieron presentes en la información suministrada por la Universidad del Cauca. Para ello, fue necesario crear una nueva versión del conjunto de datos que contiene los atributos previamente recopilados junto con los atributos que fueron simulados en la fase anterior. El nuevo conjunto de datos se utilizó en una nueva iteración con el fin de verificar la calidad del modelo. En la Tabla 33, se presenta el nuevo conjunto de atributos.

Nuevo conjunto de atributos
Identificación
Tipo identificación
Estrato

Género
Estado civil
Discapacidad
Comunidad Negra
Comunidad Indígena
Municipio de procedencia
Departamento de procedencia
Programa académico
Último semestre cursado
Promedio académico
Estudiante trabaja
Desertó

Tabla 33. Nuevo conjunto de atributos

4.4.5 Formateo de los datos

Los valores de los atributos que cuentan con información categórica, fueron transformados a valores numéricos mediante el uso de la herramienta *Rapidminer*, ya que algunas de las técnicas de minería de datos no admiten datos categóricos. A continuación, se presenta el proceso realizado.

Se agregó el operador *Nominal to Numerical* para transformar los valores de los atributos categóricos a valores numéricos. En la Tabla 34, se presentan los atributos formateados.

Atributos formateados
Tipo identificación
Estrato
Género
Estado civil
Discapacidad
Comunidad Negra
Comunidad Indígena
Municipio de procedencia
Departamento de procedencia
Programa académico

Tabla 34. Atributos formateados

En la Figura 15, se presenta el proceso realizado en la herramienta *Rapidminer.* En el paso A, se presenta el conjunto de datos. Seguidamente en el paso B, se convierten los datos categóricos a numéricos mediante el operador *Nominal to Numeric*. Finalmente, en el paso C, se obtiene el conjunto de datos numéricos.

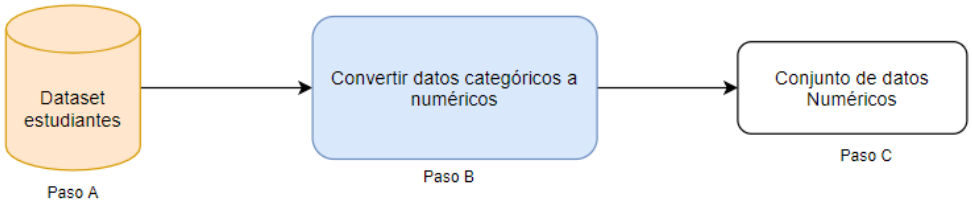

Figura 15. Transformación de atributos categóricos a numéricos

4.4.6 Balanceo de los datos

En esta fase, se observó que el conjunto de datos de los estudiantes tiene un total de 47549 registros. De los cuales se evidencia que la cantidad de estudiantes desertores es de 4677 siendo el 9.8% del conjunto. Los registros restantes corresponden a los estudiantes no desertores, que son en total 42872 abarcando el 95.8% del conjunto de datos como lo muestra la Figura 16 y Figura 17. Lo que significa, que el total de estudiantes desertores es bajo en relación al total de registros del conjunto. De esta manera se llega a la conclusión que el conjunto de datos es desbalanceado.

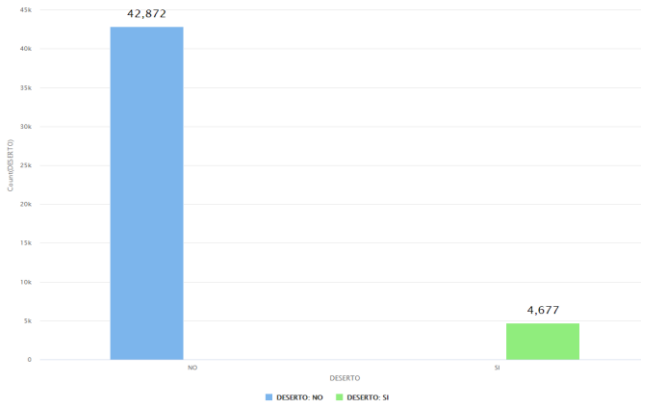

Figura 16. Cantidad de estudiantes no desertores y desertores

Deserción	Cantidad	Porcentaje
NO	**42872**	**90.2%**
SI	**4677**	**9.8%**

Figura 17. Porcentaje equivalente al total de estudiantes no desertores y desertores

Partiendo que el conjunto de datos de los estudiantes no es un conjunto de datos balanceado, se procedió a usar un algoritmo interactivo denominado *Condensed Nearest Neighbor* [53]. Este algoritmo selecciona un subconjunto del conjunto de muestras a fin de reducir los registros de la clase mayoritaria tratando de igualar la cantidad de datos al tamaño de la clase minoritaria.

Este proceso de balanceo de datos se realizó mediante la herramienta *Rapidminer*, haciendo uso el operador *Select By CNN*. Permitiendo dejar un subconjunto de 9779 registros de los cuales 5102 son registros de estudiantes no desertores y 4677 son registros de estudiantes desertores con lo muestra la Figura 18 y la Figura 19.

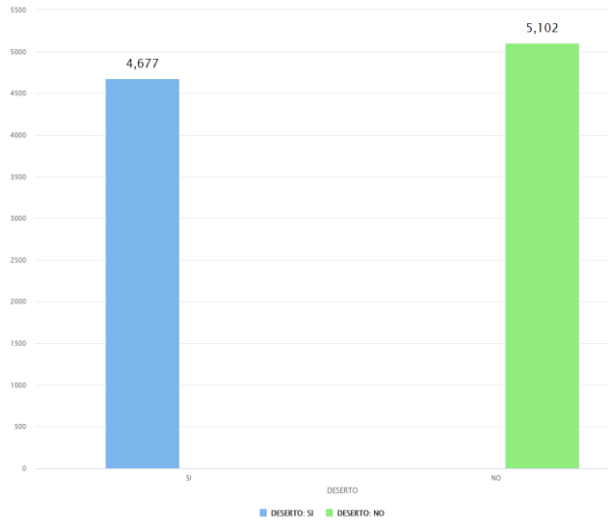

Figura 18. Cantidad de estudiantes no desertores y desertores (Datos balanceados)

Deserción	Cantidad	Porcentaje
NO	**5102**	**52.2%**
SI	**4677**	**47.8%**

Figura 19. Porcentaje equivalente al total de estudiantes no desertores y desertores (Datos balanceados)

En la Figura 20, se muestra el proceso realizado para el balanceo del conjunto de datos haciendo uso de la herramienta *Rapidminer*.

En el paso A, se presenta el conjunto de datos. Seguidamente en el paso B, se identifican las variables de clase con el operador *SetRole*. En el paso C, se usa el operador *Select By CNN* que se encarga de balancear el conjunto de datos. Finalmente, en el paso D, se obtiene el conjunto de datos balanceado.

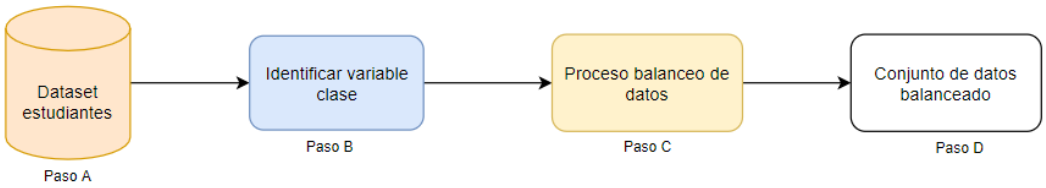

Figura 20. Proceso balanceo del conjunto de datos

4.5 Modelado

En esta fase se escogieron las técnicas de minería de datos más apropiadas, posteriormente se realizó el plan de pruebas, se procedió a aplicar dichas técnicas sobre los datos para generar el modelo y finalmente evaluar si el modelo cumple con los criterios de éxito.

4.5.1 Selección de técnicas de minería de datos

Las técnicas de minería de datos que fueron utilizadas para el modelado son aquellas técnicas que reportaron la mejor precisión en la revisión sistemática. A continuación, en la Tabla 35, se presentan las técnicas de minería de datos seleccionadas.

Técnicas de minería de datos seleccionadas
Decision Tree
Naive Bayes
Support Vector Machine

Tabla 35. Técnicas de minería de datos seleccionadas

4.5.2 Plan de pruebas

En la primera iteración de proceso, se procedió a separar el conjunto de datos en dos subconjuntos de datos. Uno para de entrenamiento que contiene el 70% de los datos y el otro subconjunto para pruebas que contiene el 30% de los datos. Este proceso fue ejecutado en la herramienta *Rapidminer* usando el operador *Split Data.*

En la *Figura 21*, se muestra el proceso realizado.

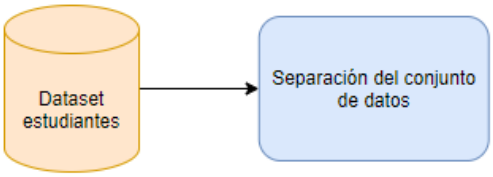

Figura 21. Proceso creación de conjuntos de entrenamiento y prueba

4.5.3 Construcción del modelo

En esta fase se crearon tres modelos Bagging, donde cada modelo usa una técnica de minería de datos seleccionada anteriormente en el apartado 4.4.1 *Técnicas de modelado*. Luego, se aplicó el conjunto de datos de entrenamiento en los tres

modelos. A continuación, se presentan las Figuras de los modelos construidos mediante la herramienta *Rapidminer*.

La Figura 22 muestra el proceso de construcción del modelo Bagging con Árbol de decisión. A continuación, se explica el proceso realizado.

En el paso A, se presenta el conjunto de datos balanceado. Seguidamente en el paso B, se separa el conjunto de datos en dos subconjuntos, uno para pruebas que contiene el 30% de los datos y el otro para entrenamiento que contiene el 70% de los datos como lo presenta el paso C. Posteriormente en el paso D, se crea el modelo Bagging con Árbol de decisión, que se entrena con el subconjunto de datos de entrenamiento. Una vez finalizado el proceso de entrenamiento se procede a validar el modelo con el subconjunto de pruebas como lo indica el paso E. Finalmente, se obtienen los datos de la precisión y matriz de confusión para determinar la calidad del modelo como lo muestra el paso F.

Este mismo proceso se realizó para el modelo Bagging Naive Bayes como se muestra en la Figura 22.

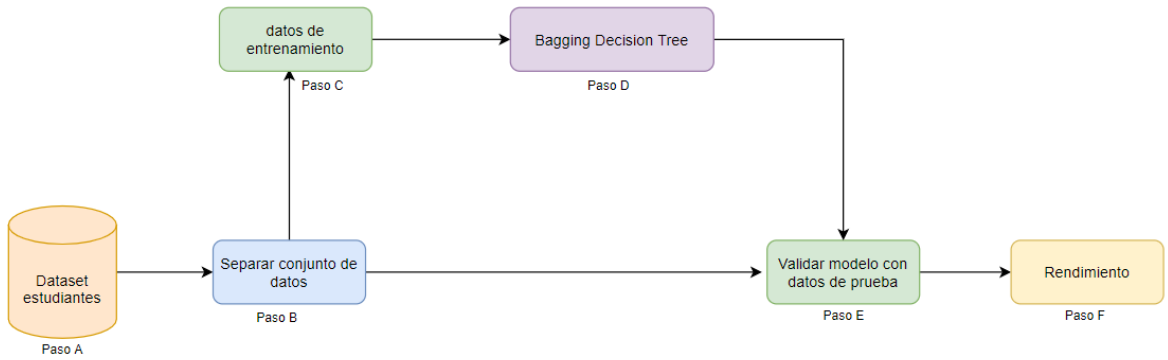

Figura 22. Modelo Bagging Árbol de decisión

En la Figura 23, se presenta el modelo Bagging Naive Bayes.

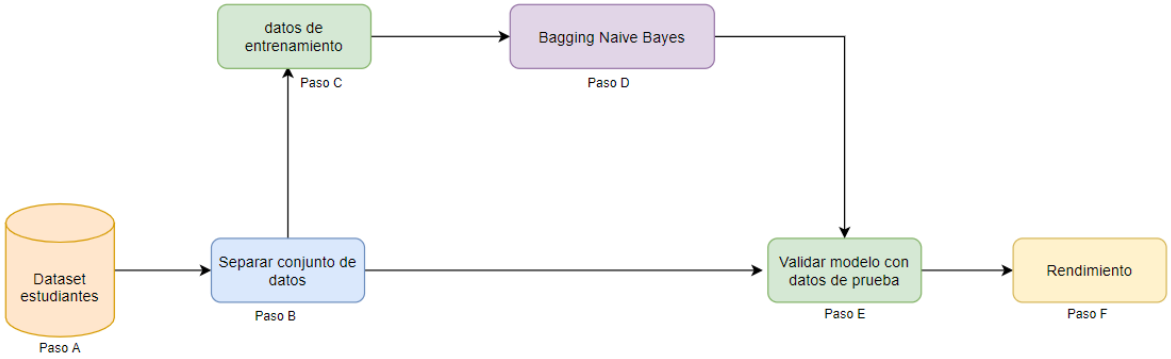

Figura 23. Modelo Bagging Naive Bayes

En la Figura 24, se expone el proceso realizado para la construcción del modelo Bagging con Support Vector Machine. En el paso A, se presenta el conjunto de datos balanceado. Seguidamente en el paso B, se convierten los datos categóricos a datos numéricos ya que el clasificador Support Vector Machine solo admite datos numéricos. En el paso C, se separa el conjunto de datos en dos subconjuntos, uno para pruebas que contiene el 30% de los datos y el otro para entrenamiento que contiene el 70% de los datos como lo presenta el paso D. Posteriormente en el paso E, se crea el modelo Bagging con Árbol de decisión, que se entrena con el subconjunto de datos de entrenamiento. Una vez finalizado el proceso de entrenamiento se procede a validar el modelo con el subconjunto de pruebas como lo indica el paso F. Finalmente se obtienen los datos de la precisión y matriz de confusión para determinar la calidad del modelo como lo muestra el paso G.

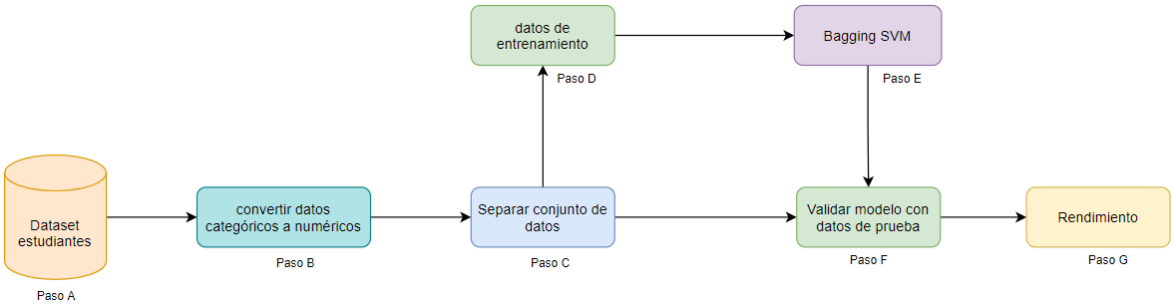

Figura 24. Modelo Bagging Support Vector Machine

En la Figura 25, se presenta el proceso realizado para la construcción del modelo Bagging Híbrido. Este proceso es el mismo de la Figura 22 excepto por el paso D. Donde se crea el modelo Bagging Híbrido que contiene los tres modelos Bagging. Allí se entrenan los tres modelos con el subconjunto de datos de entrenamiento. Una vez finalizado este proceso, se aplica el algoritmo de voto mayoritario [54] que se encarga de evaluar los resultados de los modelos para dejar un único resultado como salida.

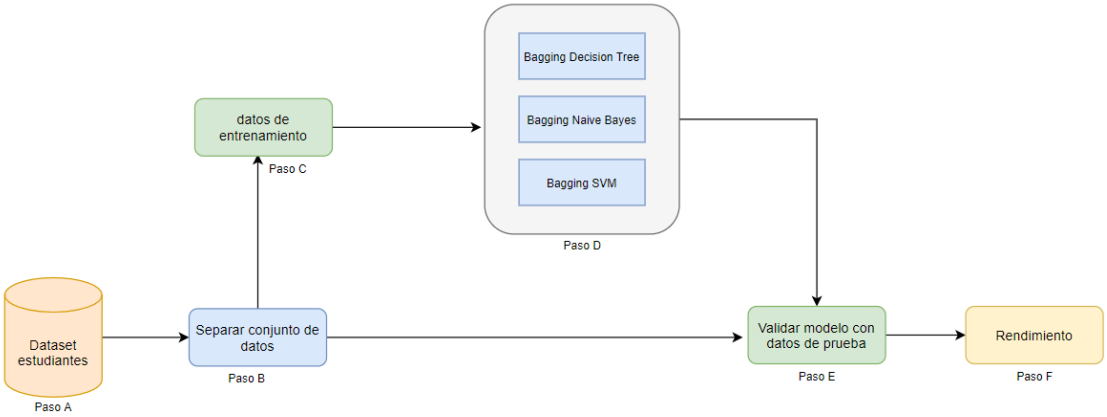

Figura 25. Modelo Bagging Híbrido

4.6 Evaluación del modelo

En esta fase de la metodología se evaluaron los modelos generados anteriormente con el conjunto de datos de pruebas. A continuación, se expone una evaluación de los resultados generales.

4.6.1 Evaluación de los resultados

Para la evaluación de los modelos se apoyó en la herramienta *Rapidminer* haciendo uso del operador *ApplyModel* que valida el modelo entrenado con el conjunto de datos de prueba y el operador *Performance* que se usa para obtener las métricas del modelo. En la Tabla 36, se presenta la precisión, precisión de los verdaderos positivos (Precisión TP), la sensibilidad (Recall), y el área bajo la curva (AUC) de cada modelo generado.

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido	**69%**	**70%**	**67%**	**76%**
Modelo Bagging Árbol de decisión	67%	70%	64%	75%
Modelo Bagging Naive Bayes	54%	54%	73%	52%
Modelo Bagging Support Vector Machine	54%	54%	73%	52%

Tabla 36. Resultados de los modelos Bagging básicos y modelo Bagging Híbrido

A continuación, se presentan los resultados detallados de cada modelo mediante una matriz de confusión [55] y la curva ROC [56].

4.6.1.1 Resultado Modelo Bagging Árbol de decisión

Según la matriz de confusión que se muestra en la Figura 26, los verdaderos positivos corresponden en el modelo a 984 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 412 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 970 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 568 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**984**	**586**
	Negativos	**412**	**970**

Figura 26. Matriz de confusión del modelo Bagging Árbol de decisión

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 70% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 64% de los verdaderos positivos. Las fórmulas para la precisión de los verdaderos positivos se presenta en la Fórmula 2 y la formula de la sensibilidad (Recall) del modelo se presenta en la Fórmula 3.

$$Precisión\ TP = \frac{VerdaderosPositivos}{(VerdaderosPositivos + FalsosPositivos)}$$

Fórmula 2. Precisión verdaderos positivos

$$Sensibilidad = \frac{VerdaderosPositivos}{(VerdaderosPositivos + FalsosNegativos)}$$

Fórmula 3. Sensibilidad (Recall) del Modelo

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 75% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 27.

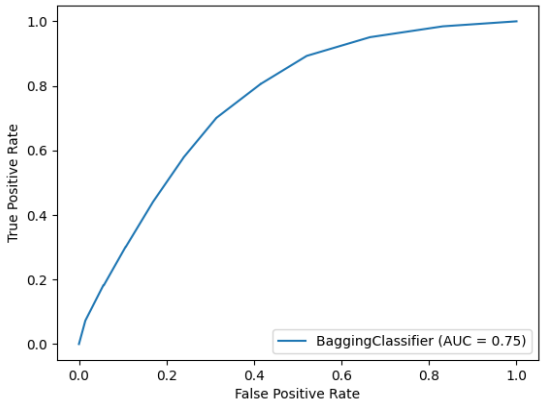

Figura 27. Área bajo la curva (AUC) del modelo Bagging Árbol de decisión

4.6.1.2 Resultado Modelo Bagging Naive Bayes

Según la matriz de confusión que se muestra en la Figura 28, los verdaderos positivos corresponden en el modelo a 1460 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 1251 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 120 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no

desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 103 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1460**	**103**
	Negativos	**1251**	**120**

Figura 28. Matriz de confusión del modelo Bagging Naive Bayes

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 54% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 73% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 52% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 29.

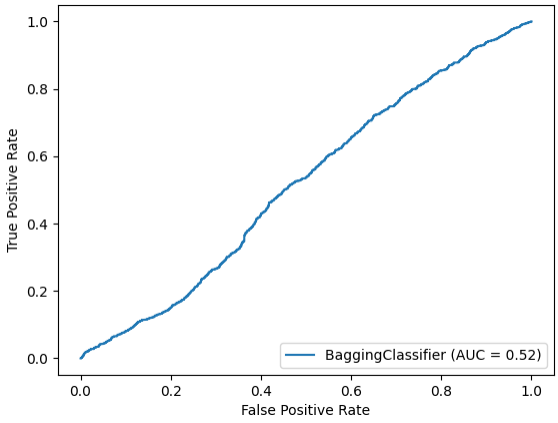

Figura 29. Área bajo la curva (AUC) del modelo Bagging Naive Bayes

4.6.1.3 Resultado Modelo Bagging Support Vector Machine

Según la matriz de confusión que se muestra en la Figura 30, los verdaderos positivos corresponden en el modelo a 1443 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 1271 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 133 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 87 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1443**	**87**
	Negativos	**1271**	**133**

Figura 30. Matriz de confusión del modelo Bagging con Support Vector Machine

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 54% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 73% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 52% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 31.

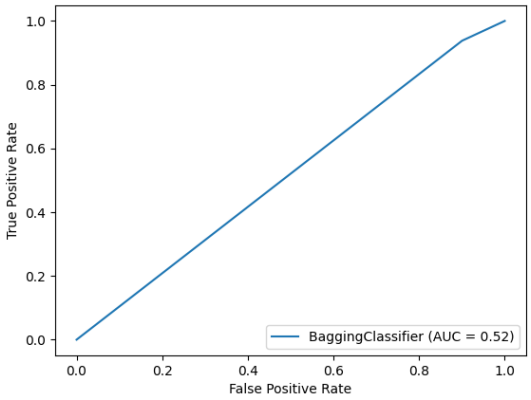

Figura 31. Área bajo la curva (AUC) del modelo Bagging Support Vector Machine

4.6.1.4 Resultado Modelo Bagging Híbrido

Según la matriz de confusión que se muestra en la Figura 32, los verdaderos positivos corresponden en el modelo a 1053 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 463 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 951 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 467 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	1053	467
	Negativos	463	951

Figura 32. Matriz de confusión del modelo Bagging Híbrido

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 70% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 67% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 76% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 33.

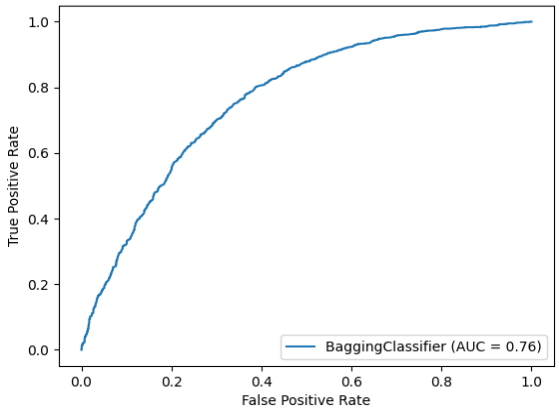

Figura 33. Área bajo la curva (AUC) del modelo Bagging Híbrido

4.6.2 Determinar los pasos siguientes

Después de evaluar los cuatro modelos Bagging creados en la fase anterior, se evidenció que las precisiones de los estos no son las esperadas en comparación a las precisiones reportadas en la revisión sistemática, por lo que fue necesario crear nuevas iteraciones para la construcción del modelo Bagging Híbrido. A continuación, se presenta el proceso realizado.

4.6.2.1 Modelo Bagging Híbrido con funciones de optimización

En este apartado, se procedió a realizar una segunda iteración de la construcción del modelo Bagging Híbrido donde se crearon tres nuevas versiones del modelo. Para cada de ellas, se agregó una función de optimización como se muestra en la

Tabla 37, las cuales tratarán de mejorar la precisión final del modelo. Además, para el entrenamiento y prueba del modelo se agregó el proceso de validación cruzada. Esto con el fin de encontrar nuevos resultados. En la Figura 34, se presenta el modelo Bagging Híbrido con validación cruzada con la función de optimización.

Funciones de optimización
Hill Climbing [50]
Simulated Annealing [51]
Step by Grid [52]

Tabla 37. Funciones de optimización

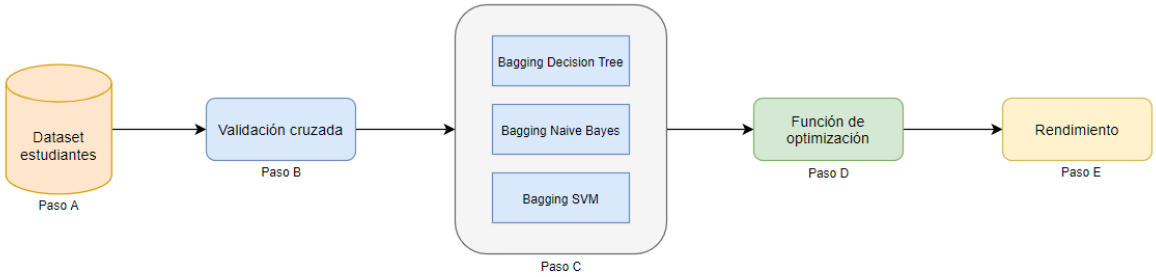

Figura 34. Modelo Bagging Híbrido con validación cruzada y función de optimización

En el paso A se presenta el conjunto completo de estudiantes. En el paso B se realiza el proceso de validación cruzada de 10 Folders para el entrenamiento y prueba del modelo. Seguidamente en el paso C, se crea el modelo Bagging Híbrido que contiene los tres componentes Bagging. Una vez finalizado el proceso de creación del modelo. En el paso D, los resultados generados por el modelo se envían a la función de optimización donde se asigna un peso a cada resultado para luego ser evaluado mediante el voto ponderado como lo muestra la Figura 35. Los resultados finales se envían al paso E, en el que se obtiene la precisión, la matriz de confusión y el área bajo la curva AUC.

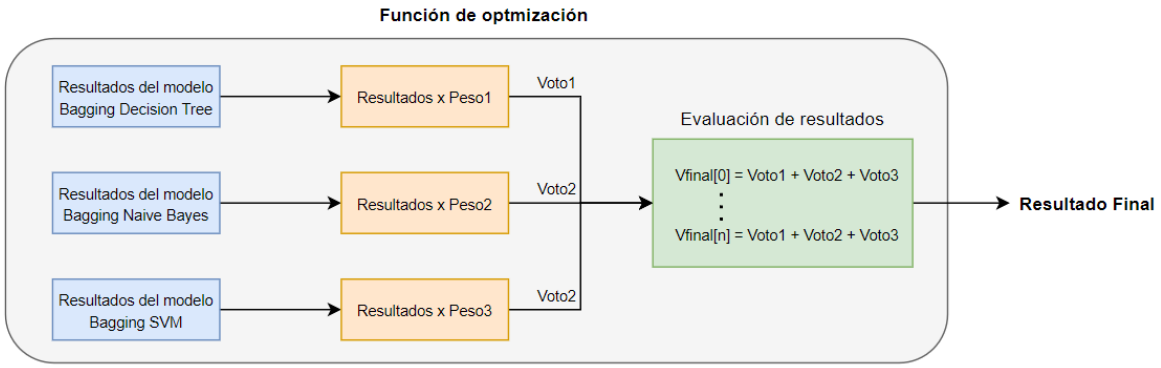

Figura 35. Función de optimización

En la Tabla 38, se presenta la precisión, precisión de los verdaderos positivos, la sensibilidad (Recall), y el área bajo la curva (AUC) de cada modelo generado.

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido Simulated Annealing	**75%**	**75%**	**84%**	**80%**
Modelo Bagging Híbrido Hill Climbing	74%	74%	79%	80%
Modelo Bagging Híbrido Step by Grid	74%	74%	79%	80%

Tabla 38. Resultados de los modelos Bagging Híbridos con función de optimización

A continuación, se exponen los resultados detallados de cada modelo.

4.6.2.1.1 Resultado Modelo Bagging Híbrido Hill Climbing (MH-HC)

Según la matriz de confusión que se muestra en la Figura 36, los verdaderos positivos corresponden en el modelo a 2743 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 1039 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 4063 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 1943 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**2743**	**1943**
	Negativos	**1039**	**4063**

Figura 36. Matriz de confusión del modelo Bagging Híbrido Hill Climbing

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 73% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 79% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 80% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 37.

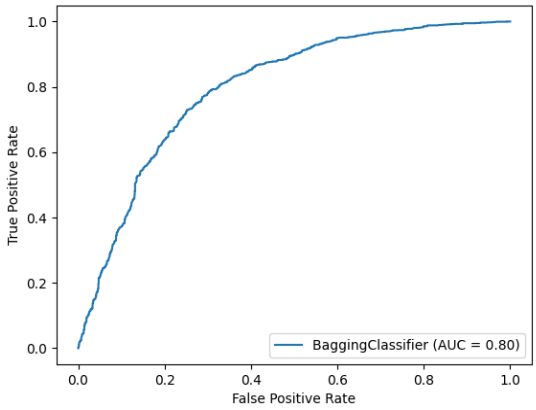

Figura 37. Área bajo la curva (AUC) del modelo Bagging Híbrido Hill Climbing

4.6.2.1.2 Resultado Modelo Bagging Híbrido Simulated Annealing (MH-SA)

Según la matriz de confusión que se muestra en la Figura 38, los verdaderos positivos corresponden en el modelo a 3694 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 1549 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 3553 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 983 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**3694**	**983**
	Negativos	**1549**	**3553**

Figura 38. Matriz de confusión del modelo Bagging Híbrido Simulated Annealing

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 75% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 84% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 80% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 39.

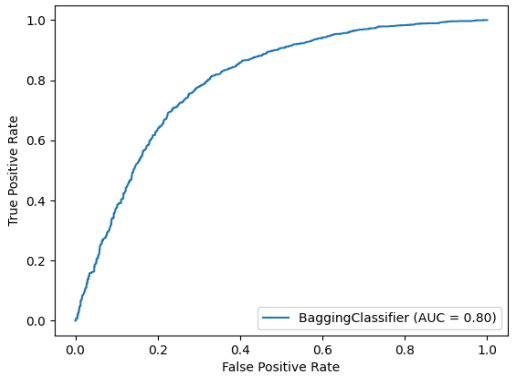

Figura 39. Área bajo la curva (AUC) del modelo Bagging Híbrido Simulated Annealing

4.6.2.1.3 Resultado Modelo Bagging Híbrido Step by Grid (MH-SG)

Según la matriz de confusión que se muestra en la Figura 40, los verdaderos positivos corresponden en el modelo a 2559 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 829 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 4273 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 2118 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**2559**	**2118**
	Negativos	**829**	**4273**

Figura 40. Matriz de confusión del modelo Bagging Híbrido Step by Grid

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 75% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 79% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 80% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 41.

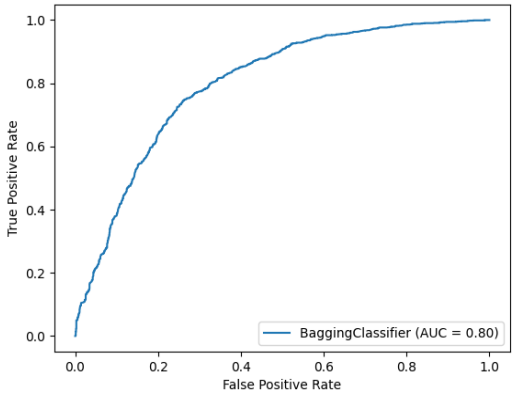

Figura 41. Área bajo la curva (AUC) del modelo Bagging Híbrido Step by Grid

4.6.2.2 Modelo Bagging Híbrido con funciones de optimización y datos simulados

En una tercera iteración, se creó un nuevo conjunto de datos, el cual contiene los atributos del conjunto original y también los atributos simulados que fueron reportados como relevantes en la revisión sistemática y que no fue posible obtener de parte de la universidad. Los atributos simulados se presentan en el apartado *4.3.3 Construcción de los datos*. Este nuevo conjunto de datos se aplicó a las diferentes versiones de los modelos Bagging creados en las anteriores fases. Este proceso fue realizado con el fin de verificar la calidad de los modelos. A continuación, se presentan los resultados obtenidos.

En la Tabla 39, se presentan las precisiones de los modelos Híbridos con su función de optimización y con el nuevo conjunto de datos.

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido Simulated Annealing	**89%**	**87%**	**87%**	**93%**
Modelo Bagging Híbrido Hill Climbing	88%	86%	86%	92%
Modelo Bagging Híbrido Step by Grid	87%	86%	86%	91%
Modelo Bagging Híbrido	86%	86%	83%	89%
Modelo Bagging Árbol de decisión	74%	72%	71%	80%
Modelo Bagging Naive Bayes	73%	70%	71%	79%
Modelo Bagging Support Vector Machine	70%	50%	55%	52%

Tabla 39. Resultados de los modelos Bagging con datos simulados

4.6.2.2.1 Resultado Modelo Bagging Árbol de decisión

Según la matriz de confusión que se muestra en la Figura 42, los verdaderos positivos corresponden en el modelo a 1071 estudiantes que fueron clasificados

correctamente como desertores. Asimismo, se observó que 435 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1085 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 343 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1071**	**343**
	Negativos	**435**	**1085**

Figura 42. Matriz de confusión del modelo Bagging Árbol de decisión con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 72% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 71% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 80% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 43.

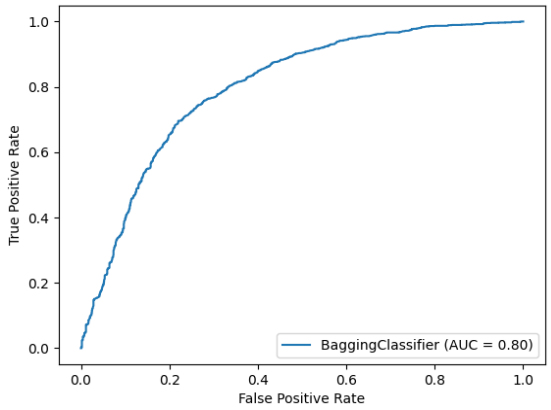

Figura 43. Área bajo la curva (AUC) del modelo Bagging Árbol de decisión con datos simulados

4.6.2.2.2 Resultado Modelo Bagging Naive Bayes

Según la matriz de confusión que se muestra en la Figura 44, los verdaderos positivos corresponden en el modelo a 1049 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 436 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1084

representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 365 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1049**	**365**
	Negativos	**436**	**1084**

Figura 44. Matriz de confusión del modelo Bagging Naive Bayes con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 70% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 71% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 79% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 45.

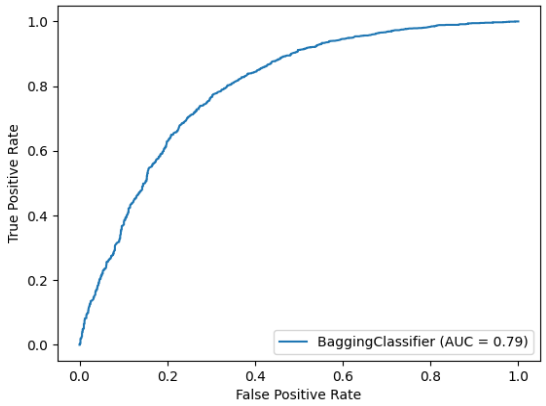

Figura 45. Área bajo la curva (AUC) del modelo Bagging Naive Bayes con datos simulados

4.6.2.2.3 Resultado Modelo Bagging Support Vector

Según la matriz de confusión que se muestra en la Figura 46, los verdaderos positivos corresponden en el modelo a 683 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 673 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 847 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 731 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**683**	**731**
	Negativos	**673**	**847**

Figura 46. Matriz de confusión del modelo Bagging Support Vector Machine con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 50% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 55% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 52% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 47.

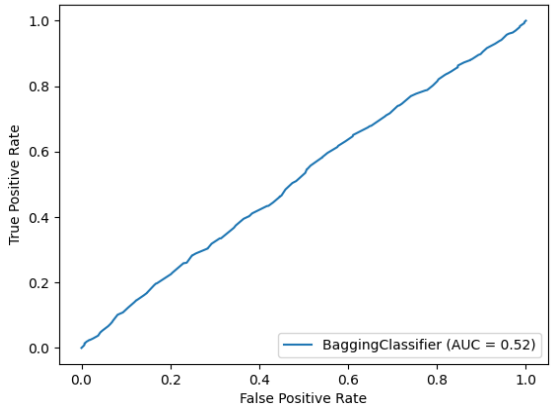

Figura 47. Área bajo la curva (AUC) del modelo Bagging Support Vector Machine con datos simulados

4.6.2.2.4 Resultado Modelo Bagging Híbrido

Según la matriz de confusión que se muestra en la Figura 48, los verdaderos positivos corresponden en el modelo a 1077 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 124 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1470 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 313 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1077**	**313**
	Negativos	**124**	**1470**

Figura 48. Matriz de confusión del modelo Bagging Híbrido con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 86% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 83% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 89% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 49.

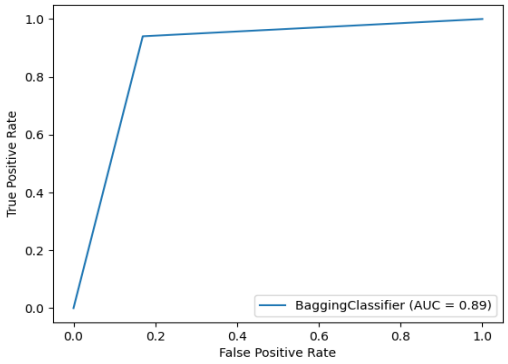

Figura 49. Área bajo la curva (AUC) del modelo Bagging Híbrido con datos simulados

4.6.2.2.5 Resultado Modelo Bagging Híbrido Hill Climbing (MH-SR)

Según la matriz de confusión que se muestra en la Figura 50, los verdaderos positivos corresponden en el modelo a 4005 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 726 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 3776 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 672 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**4005**	**672**
	Negativos	**726**	**3776**

Figura 50. Matriz de confusión del modelo Bagging Híbrido Hill Climbing y datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 86% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 86% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 92% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

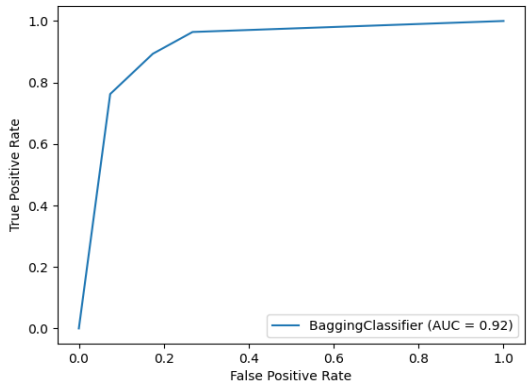

Figura 51. Área bajo la curva (AUC) del modelo Bagging Híbrido Hill Climbing y datos simulados

4.6.2.2.6 Resultado Modelo Bagging Híbrido Simulated Annealing

Según la matriz de confusión que se muestra en la Figura 52, los verdaderos positivos corresponden en el modelo a 4025 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 725 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 4377 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 652 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**4025**	**652**
	Negativos	**725**	**4377**

Figura 52. Matriz de confusión del modelo Bagging Híbrido Simulated Annealing y datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 87% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 87% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 93% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 53.

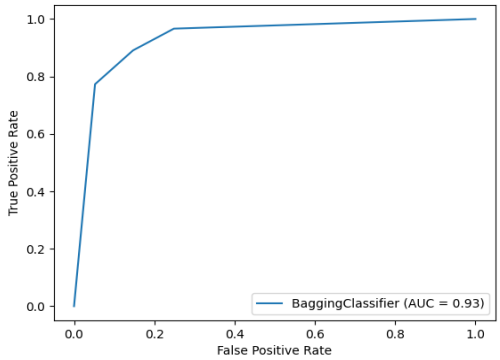

Figura 53. Área bajo la curva (AUC) del modelo Bagging Híbrido Annealing y datos simulados

4.6.2.2.7 Resultado Modelo Bagging Híbrido Step by Grid

Según la matriz de confusión que se muestra en la Figura 54, los verdaderos positivos corresponden en el modelo a 4044 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 744 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 4358 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 633 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**4044**	**633**
	Negativos	**744**	**4358**

Figura 54. Matriz de confusión del modelo Bagging Híbrido Step by Grid y datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 86% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 86% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 91% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 55.

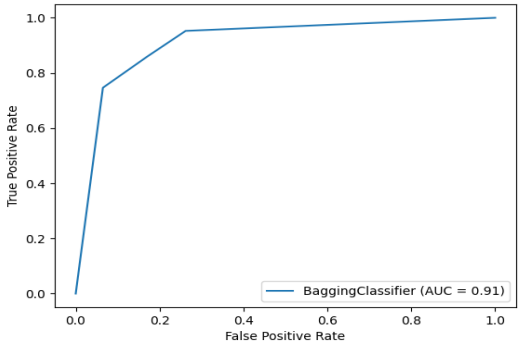

Figura 55. Área bajo la curva (AUC) del modelo Bagging Híbrido Step by Grid y datos simulados

4.6.3 Resultados finales

En esta fase, después de haber realizado el entrenamiento, validación y verificar la calidad de cada modelo mediante las métricas de precisión, precisión de los verdaderos positivos, la sensibilidad (Recall), y el área bajo la curva (AUC). A continuación, se presentan los resultados finales.

En la Tabla 40, se presentan los resultados de los modelos Bagging con el conjunto de datos original.

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido Simulated Annealing	**75%**	**75%**	**84%**	**80%**
Modelo Baggibg Híbrido Step by Grid	74%	74%	79%	80%
Modelo Bagging Híbrido Hill Climbing	74%	72%	79%	80%
Modelo Bagging Híbrido	69%	70%	67%	76%
Modelo Bagging Arbol de decisión	67%	70%	64%	75%
Modelo Bagging Naive Bayes	54%	54%	73%	52%
Modelo Bagging Support Vector Machine	54%	54%	73%	52%

Tabla 40. Resultados finales de los modelos Bagging

En la Tabla 41, se presentan los resultados de los modelos Bagging con el conjunto de datos simulado.

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido Simulated Annealing	**89%**	**87%**	**87%**	**93%**
Modelo Bagging Híbrido Hill Climbing	88%	86%	86%	92%
Modelo Bagging Híbrido Step by Grid	87%	86%	86%	91%
Modelo Bagging Híbrido	86%	86%	83%	89%
Modelo Bagging Árbol de decisión	74%	72%	71%	80%
Modelo Bagging Naive Bayes	73%	70%	71%	79%
Modelo Bagging Support Vector Machine.	70%	50%	55%	52%

Tabla 41. Resultados finales de los modelos Bagging con datos simulados

4.7 Despliegue

4.7.1 Funcionamiento del modelo

En esta fase de despliegue, se creó un nuevo conjunto de datos distinto al conjunto de entrenamiento. El conjunto de datos de prueba seleccionado, es un conjunto de datos de estudiantes del año 2020 con 1472 registros, el cual paso por todas las fases desde la compresión de los datos hasta la fase de modelado. Además, se creó una segunda versión del conjunto de datos con atributos simulados. El proceso para validación del conjunto de datos fue realizado con los tres modelos Bagging básicos (*Bagging Árbol de decisión, Bagging Naive Bayes, Bagging Support Vector Machine*). También se realizó el proceso de validación con modelo Bagging Híbrido con la función de optimización Simulated Annealing, el cual tuvo el mejor desempeño en la fase anterior. A continuación, se presentan los resultados del modelo.

4.7.2 Resultados de los Modelos Bagging

En la Tabla 42, se presentan los resultados de los modelos con las métricas de precisión, precisión de los verdaderos positivos, la sensibilidad (Recall), y el área bajo la curva (AUC).

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido Simulated Annealing	**74%**	**72%**	**70%**	**77%**
Modelo Bagging Árbol de decisión	65%	65%	64%	72%
Modelo Bagging Naive Bayes	56%	56%	70%	55%
Modelo Bagging Support Vector Machine	58%	56%	57%	61%

Tabla 42. Resultados de los modelos Bagging con datos de estudiantes del año 2020

4.7.2.1 Resultado Modelo Bagging Árbol de decisión

Según la matriz de confusión que se muestra en la Figura 56, los verdaderos positivos corresponden en el modelo a 1746 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 1020 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1418 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 606 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1746**	**606**
	Negativos	**1020**	**1418**

Figura 56. Matriz de confusión del modelo Bagging Árbol de decisión

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 65% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 65% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 72% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 57.

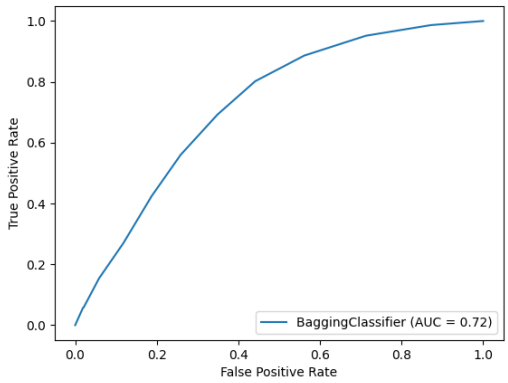

Figura 57. Área bajo la curva (AUC) del modelo Bagging Árbol de decisión

4.7.2.2 Resultado Modelo Bagging Naive Bayes

Según la matriz de confusión que se muestra en la Figura 58, los verdaderos positivos corresponden en el modelo a 419 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 350 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 2368 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 2133 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**419**	**2133**
	Negativos	**350**	**2368**

Figura 58. Matriz de confusión del modelo Bagging Naive Bayes

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 56% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 70% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 55% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 59.

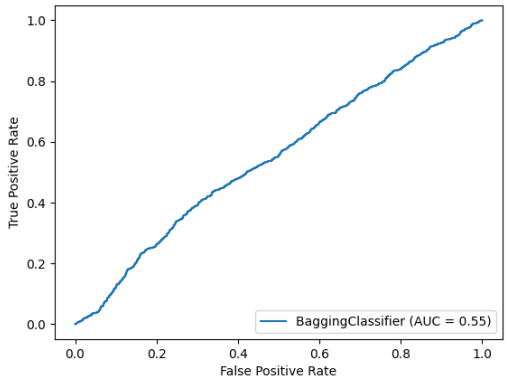

Figura 59. Área bajo la curva (AUC) del modelo Bagging Naive Bayes

4.7.2.3 Resultado Modelo Bagging Support Vector Machine

Según la matriz de confusión que se muestra en la Figura 60, los verdaderos positivos corresponden en el modelo a 1383 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 1078 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1460 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 969 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1383**	**969**
	Negativos	**1078**	**1460**

Figura 60. Matriz de confusión del modelo Bagging Support Vector Machine

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 56% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 57% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 61% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 61.

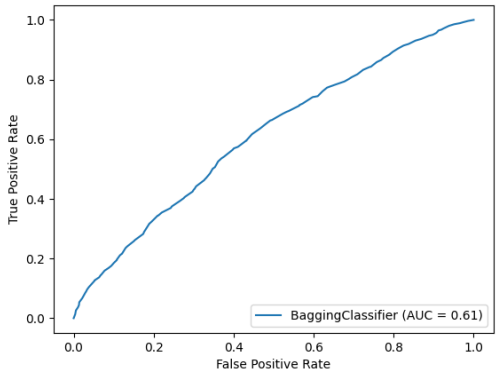

Figura 61. Área bajo la curva (AUC) del modelo Bagging Support Vector Machine

4.7.2.4 Resultado Modelo Bagging Híbrido Simulated Annealing (MH-SA)

Según la matriz de confusión que se muestra en la Figura 62, los verdaderos positivos corresponden en el modelo a 1349 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 512 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1418 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 533 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1349**	**533**
	Negativos	**512**	**1418**

Figura 62. Matriz de confusión del modelo Bagging Híbrido Simulated Annealing

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 72% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 70% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 77% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 63.

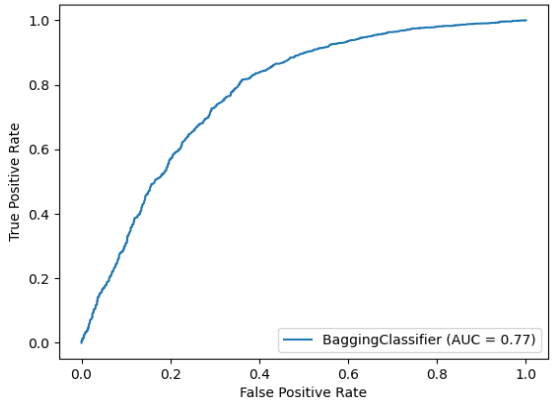

Figura 63. Área bajo la curva (AUC) del modelo Bagging Híbrido Simulated Annealing

4.7.3 Resultado de los Modelos Bagging con datos simulados

En la Tabla 43, se presentan los resultados del modelo con el conjunto de datos simulados. A continuación, se presentan las métricas de precisión, precisión de los verdaderos positivos, la sensibilidad (Recall), y el área bajo la curva (AUC).

Modelo	Precisión	Precisión TP	Recall	AUC
Modelo Bagging Híbrido Simulated Annealing	**88%**	**89%**	**87%**	**96%**
Modelo Bagging Árbol de decisión	83%	83%	84%	88%
Modelo Bagging Naive Bayes	75%	72%	74%	79%
Modelo Bagging Support Vector Machine	62%	60%	70%	60%

Tabla 43. Resultados de los modelos Bagging con datos simulados de estudiantes del año 2020

4.7.3.1 Resultado Modelo Bagging Árbol de decisión

Según la matriz de confusión que se muestra en la Figura 64, los verdaderos positivos corresponden en el modelo a 1945 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 184 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 2354 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 407 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1945**	**407**
	Negativos	**184**	**2354**

Figura 64. Matriz de confusión del modelo Bagging Árbol de decisión con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 83% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 84% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 88% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 65.

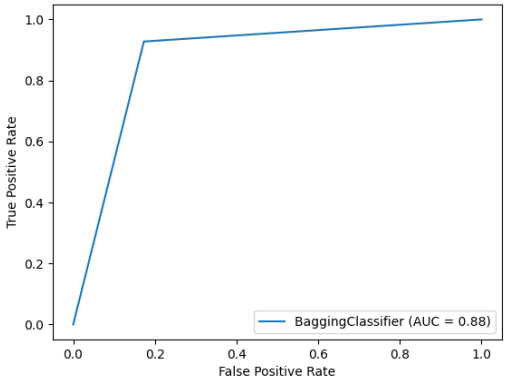

Figura 65. Área bajo la curva (AUC) del modelo Bagging Árbol de decisión con datos simulados

4.7.3.2 Resultado Modelo Bagging Naive Bayes

Según la matriz de confusión que se muestra en la Figura 66, los verdaderos positivos corresponden en el modelo a 1768 estudiantes que fueron clasificados

correctamente como desertores. Asimismo, se observó que 143 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 2415 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 584 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1768**	**584**
	Negativos	**143**	**2415**

Figura 66. Matriz de confusión del modelo Bagging Naive Bayes con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 72% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 74% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 79% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 67.

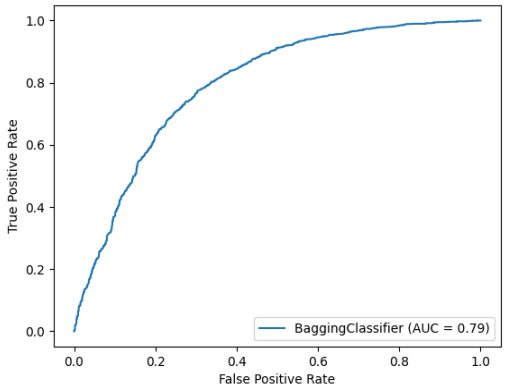

Figura 67. Área bajo la curva (AUC) del modelo Bagging Naive Bayes con datos simulados

4.7.3.3 Resultado Modelo Bagging Support Vector Machine

Según la matriz de confusión que se muestra en la Figura 68, los verdaderos positivos corresponden en el modelo a 1083 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 752 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 1796 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no

desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 1359 registros de estudiantes que el modelo clasifica incorrectamente.

		Predicción	
		Positivos	Negativos
Real	Positivos	**1083**	**1359**
	Negativos	**752**	**1796**

Figura 68. Matriz de confusión del modelo Bagging Support Vector Machine con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 62% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 70% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 60% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 69.

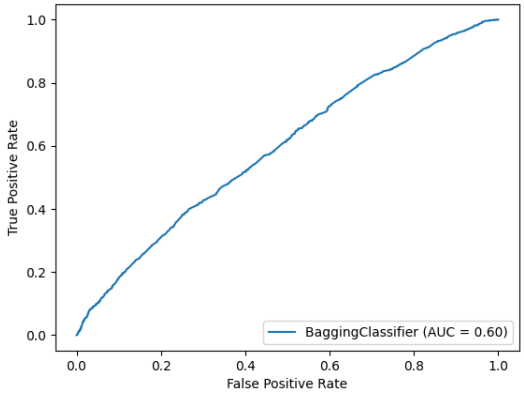

Figura 69. Área bajo la curva (AUC) del modelo Bagging Support Vector Machine con datos simulados

4.7.3.4 Resultado Modelo Híbrido Bagging Simulated Annealing

Según la matriz de confusión que se muestra en la Figura 70, los verdaderos positivos corresponden en el modelo a 1975 estudiantes que fueron clasificados correctamente como desertores. Asimismo, se observó que 221 estudiantes desertores fueron clasificados incorrectamente por el modelo, esta cantidad se muestra como falsos positivos. Por otro lado, se observó que el valor de 2317 representa la cantidad de verdaderos negativos, que son aquellos estudiantes no desertores clasificados de manera correcta por el modelo. Igualmente, los falsos negativos son 377 registros de estudiantes que el modelo clasifica incorrectamente

		Predicción	
		Positivos	Negativos
Real	Positivos	**1975**	**377**
	Negativos	**221**	**2317**

Figura 70. Matriz de confusión del modelo Bagging Híbrido Simulated Annealing con datos simulados

En cuanto a la precisión de los verdaderos positivos, el modelo identificó correctamente el 89% de los estudiantes desertores. En cuanto a la sensibilidad del modelo, este identifica correctamente el 87% de los verdaderos positivos.

Por otro lado, la métrica del área bajo la curva (AUC) presenta un valor de 96% indicando la calidad de precisión que tiene el modelo para identificar los verdaderos positivos y verdaderos negativos, como se muestra en la Figura 71.

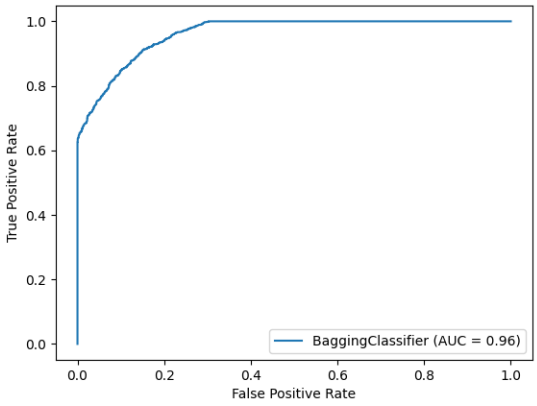

Figura 71. Curva ROC del modelo Bagging Híbrido Simulated Annealing con datos simulados

4.7.4 Documentación del proceso de modelado

Partiendo del proceso realizado anteriormente, para poder implantar este proyecto con otro conjunto de datos, se sugiere seguir los pasos que van desde la fase *4.1 Comprensión de los datos* hasta la fase *4.5 Evaluación del modelo.*

Capítulo 5

Conclusiones, situaciones presentadas y trabajo futuro

5.1 Conclusiones

Se pudo evidenciar una mejora en la calidad de la predicción de la deserción estudiantil universitaria utilizando un modelo que hibrida las técnicas con mejores resultados identificadas en el estado del arte, el cual fue denominado como modelo Bagging Híbrido Simulated Annealing. Este a su vez utilizó un conjunto de datos con 9779 registros balanceados, este conjunto de datos se considera grande en comparación con los utilizados en el estado del arte que tenían en promedio 4900 registros y estaban desbalanceados. Los resultados del modelo híbrido propuesto fueron los siguientes: precisión de 89%, precisión de verdaderos positivos de 87%, sensibilidad (Recall) de 87% y el área bajo la curva (AUC) de 96%, los cuales son mejores que los reportados por los modelos creados con: Árboles de decisión (precisión de 83%, precisión de verdaderos positivos de 83%, sensibilidad (Recall) de 84% y el área bajo la curva (AUC) de 88%), Naive Bayes (precisión de 75%, precisión de verdaderos positivos de 73%, sensibilidad (Recall) de 74% y el área bajo la curva (AUC) de 79%) y SVM (precisión de 62%, precisión de verdaderos positivos de 60%, sensibilidad (Recall) de 70% y el área bajo la curva (AUC) de 60%).

Se evidenció que el modelo Bagging Híbrido Simulated Annealing basado en CRISP-DM, las mejores técnicas identificadas en el estado del arte y el balanceo previo de los datos si fue competitivo frente a los otros modelos construidos y obtuvo buenos resultados.

A través de la construcción de la vista minable para el modelo propuesto de deserción estudiantil universitaria apoyándose en la metodología CRISP-DM, se pudo caracterizar los atributos más comunes relacionados con la deserción de un estudiante universitario basándose en los reportes de la revisión sistemática.

La evaluación y comparación entre los tres modelos básicos, creados con cada una de las mejores técnicas de minería de datos identificadas en la revisión sistemática frente al modelo híbrido propuesto, permitió evidenciar una mejora por parte del modelo MH-SA en la calidad de la predicción de deserción de estudiantes universitarios, reportando una precisión de 75% y una sensibilidad de 84%, en comparación al mejor de los tres modelos de la revisión sistemática, que fue el modelo Bagging Árbol de decisión, el cual obtuvo una precisión del 67% y una

sensibilidad de 64%, utilizando el conjunto original de datos balanceado. También se evidenció, que al validar los mismos modelos con el conjunto de datos balanceado simulado, el modelo híbrido propuesto, alcanzó una precisión de 89% y una sensibilidad de 87%, frente al mejor de los tres modelos de la revisión sistemática, el cual reportó una precisión del 74% y una sensibilidad de 71%.

5.2 Situaciones presentadas

En este proyecto, después de haber identificado las tres mejores técnicas de minería de datos en la revisión sistemática, se procedió a crear tres modelos Bagging, un modelo para cada una de las técnicas de minería datos. Además, se creó un modelo Bagging Híbrido que usó internamente los tres modelos Bagging mencionados anteriormente. Debido a que la calidad del modelo propuesto no fue la mejor, se optó por crear tres nuevas versiones del mismo modelo donde cada versión empleó un algoritmo de optimización diferente, a saber: Hill Climbing, Simulated Annealing y Step by Grid donde finalmente el de mejores resultados fue el modelo Bagging Híbrido con Simulated Annealing.

Al revisar la literatura del estado del arte, se observó que los resultados de la precisión de los modelos es alta, ya que estos trabajan con datos desbalanceados. Los datos desbalanceados provocan que el modelo no sea confiable al momento de tratar de clasificar un verdadero positivo, pues los algoritmos de clasificación tienden a ignorar las clases con menor número de apariciones (clase minoritaria), en este caso los estudiantes que si desertan de la universidad. Sin embargo, después de haber validado los modelos con un conjunto de datos balanceado, se identificó que la precisión bajó, debido a que la cantidad de apariciones de las variables de clase tienden a ser iguales. Sin embargo, al emplear otras métricas como la sensibilidad (Recall) y el área bajo la curva (AUC), los resultados de estas métricas propenden a tener un valor alto, esto le permite al modelo no tener una gran dificultad al tratar de clasificar un abandono positivo.

En el capítulo 4 correspondiente CRISP-DM, la fase de procesamiento de datos fue la más costosa en tiempo, debido a que los datos se encontraron en distintos archivos. También, se encontraron muchos datos nulos. Además, en los archivos provistos cuando eran de diferente periodo académico, se adicionaban nuevos atributos y otros no se encontraban. Asimismo, se encontraron datos de programas de posgrado, lo que redujo la cantidad de los datos. Por otro lado se evidenciaron atributos faltantes y que en la revisión sistemática se consideran importantes para determinar la deserción de un estudiante. Dichos atributos como el promedio académico, semestre actual y la edad del estudiante fueron simulados para luego ser incorporados en un nuevo conjunto de datos.

Cabe mencionar que para la obtención de los datos de los estudiantes se presentó un gran retraso debido a diferentes permisos entre las áreas involucradas de la Universidad del Cauca para brindar la información solicitada, a causa de este retraso las actividades del capítulo 4 CRISP-DM se iniciaron de forma postergada.

Al colocar en funcionamiento los algoritmos de los modelos desarrollados en Python con el conjunto de datos, se observó que se presentaron inconvenientes con los clasificadores de minería de datos, ya que el conjunto de datos cuenta con atributos de tipo fecha y categóricos, mientras que los clasificadores de minería de datos solo soportan datos numéricos, por lo que fue necesario transformar aquellos datos categóricos a numéricos. Sin embargo, es importante tener en cuenta al convertir los datos de naturaleza fecha, ya que al transformarse a datos numéricos, estos datos generan ruido al ser aplicados en los modelos.

5.3 Trabajo futuro

Al observar que existen diferentes atributos que determinan la deserción de un estudiante y que algunas técnicas de minería de datos tienen mejor desempeño con ciertos atributos, en un futuro, el modelo podría escoger las técnicas de minería de datos a usar de acuerdo a la naturaleza de los atributos.

Es necesario tener acceso completo a toda la información relativa a los alumnos de la universidad. La información de cada estudiante debe estar actualizada y debe recolectarse periódicamente. A pesar de haber simulado algunos datos de los estudiantes, no cabe duda que existen múltiples factores que determinan la deserción de un estudiante. Por este motivo es importante tener gran variedad de los datos reales de los alumnos, esto incrementaría aún más la fiabilidad de los modelos.

Referencias

[1] I. A. B. Báez, *et al.*, "Estrategias para la Permanencia en Educación Superior: Experiencias Significativas," *Ministerio de Educación Nacional-MEN-QUALIFICAR. Bogotá: Sanmartín Obregón & Cía Ltda,* 2015.

[2] C. G. Ruiz, *et al.*, "Deserción estudiantil en la educación superior colombiana," *Ministerio de Educación Nacional, Primera edición. Bogotá: Imprenta Nacional de Colombia,* 2009.

[3] "Deserción de la Educación Superior," [Online]. Available: https://www.mineducacion.gov.co/sistemasdeinformacion/1735/w3-article-357549.html?_noredirect=1 [Accessed: Mayo 2020]2018.

[4] E. J. Hernández-Leal, *et al.*, "Educational data mining for the analysis of student deser-tion," *CEUR Workshop Proceedings,* 2018.

[5] D. Heredia, *et al.*, "Student Dropout Predictive Model Using Data Mining Techniques," *IEEE Latin America Transactions,* vol. 13, pp. 3127-3134, 2015.

[6] M. M. FERREYRA, "Momento decisivo: La educación superior en América Latina y el Caribe," *Washington, DC: Banco Mundial,* 2017.

[7] C. M. Vera, "Predicción del fracaso y el abandono escolar mediante técnicas de minería de datos," *Ph.D, Universidad de Córdoba, Córdoba, España,* 2015.

[8] D. W. (1996). "Extreme Programming," http://www.extremeprogramming.org/ [Accessed: Mayo 2020].

[9] "Cross Industry Standard Process for Data Mining (CRISP-DM) " [Online]. Available: http://crisp-dm.eu/home/crisp-dm-methodology/. [Accessed: Mayo 2020].

[10] B. Kitchenham, "Systematic literature reviews in software engineering – A tertiary study," v. Information and Software Technology, pp. 792-805, 2010/08/01/ 2010., Ed., ed, 2010.

[11] L. Guerra, *et al.*, "Trends in information models on retention-university dropout," *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao,* vol. 2020, pp. 55-68, 2020.

[12] J. M. Ortiz-Lozano, *et al.*, "University student retention: Best time and data to identify undergraduate students at risk of dropout," *Innovations in Education and Teaching International,* vol. 57, pp. 74-85, 2020.

[13] G. Bilquise, *et al.*, "Predicting Student Retention Among a Homogeneous Population Using Data Mining," in *Advances in Intelligent Systems and Computing* vol. 1058, ed, 2020, pp. 35-46.

[14] K. Y. Diaz Pedroza, *et al.*, "Review of techniques, tools, algorithms and attributes for data mining used in student desertion," in *Journal of Physics: Conference Series*, 2019.

[15] A. Sarra, *et al.*, "Identifying Students at Risk of Academic Failure Within the Educational Data Mining Framework," *Social Indicators Research,* vol. 146, pp. 41-60, 2019.

[16] F. Agrusti*, et al.*, "University dropout prediction through educational data mining techniques: A systematic review," *Journal of E-Learning and Knowledge Society,* vol. 15, pp. 161-182, 2019.
[17] S. Jayaprakash and V. Jaiganesh, "Predicting academic performance of tertiary students using classification algorithm," *International Journal of Recent Technology and Engineering,* vol. 8, pp. 6558-6561, 2019.
[18] S. A. Ahmed and S. I. Khan, "A machine learning approach to Predict the Engineering Students at risk of dropout and factors behind: Bangladesh Perspective," in *2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019*, 2019.
[19] Y. Zhang and B. Wu, "Research and application of grade prediction model based on decision tree algorithm," in *ACM International Conference Proceeding Series*, 2019.
[20] S. Milinković and V. Vujović, "Students' Success Predictive Models Based on Selected Input Parameters Set," in *2019 18th International Symposium INFOTEH-JAHORINA, INFOTEH 2019 - Proceedings*, 2019.
[21] A. Ortigosa*, et al.*, "From Lab to Production: Lessons Learnt and Real-Life Challenges of an Early Student-Dropout Prevention System," *IEEE Transactions on Learning Technologies,* vol. 12, pp. 264-277, 2019.
[22] D. Baneres*, et al.*, "An Early Feedback Prediction System for Learners At-Risk within a First-Year Higher Education Course," *IEEE Transactions on Learning Technologies,* vol. 12, pp. 249-263, 2019.
[23] M. Adil*, et al.*, "Predictive Analysis for Student Retention by Using Neuro-Fuzzy Algorithm," in *2018 10th Computer Science and Electronic Engineering Conference, CEEC 2018 - Proceedings*, 2019, pp. 41-45.
[24] Y. Cui*, et al.*, "Predictive analytic models of student success in higher education: A review of methodology," *Information and Learning Science,* vol. 120, pp. 208-227, 2019.
[25] R. Manrique*, et al.*, "An analysis of student representation, representative features and classification algorithms to predict degree dropout," in *ACM International Conference Proceeding Series*, 2019, pp. 401-410.
[26] M. Y. Orong*, et al.*, "A Hybrid Prediction Model Integrating a Modified Genetic Algorithm to K-means Segmentation and C4.5," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2019, pp. 1853-1858.
[27] L. W. Santoso and Yulia, "The Analysis of Student Performance Using Data Mining," in *Advances in Intelligent Systems and Computing* vol. 924, ed, 2019, pp. 559-573.
[28] J. D. Febro, "Utilizing feature selection in identifying predicting factors of student retention," *International Journal of Advanced Computer Science and Applications,* vol. 10, pp. 269-274, 2019.
[29] P. Nuankaew, "Dropout situation of business computer students, University of Phayao," *International Journal of Emerging Technologies in Learning,* vol. 14, pp. 117-131, 2019.
[30] H. Hassan*, et al.*, "Students' performance prediction model using meta-classifier approach," in *Communications in Computer and Information Science* vol. 1000, ed, 2019, pp. 221-231.

[31] D. Vila*, et al.*, "Detection of desertion patterns in university students using data mining techniques: A case study," in *Communications in Computer and Information Science* vol. 895, ed, 2019, pp. 420-429.
[32] X. Palacios-Pacheco*, et al.*, "Application of data mining for the detection of variables that cause university desertion," in *Communications in Computer and Information Science* vol. 895, ed, 2019, pp. 510-520.
[33] O. Sukhbaatar*, et al.*, "An artificial neural network based early prediction of failure-prone students in blended learning course," *International Journal of Emerging Technologies in Learning,* vol. 14, pp. 77-92, 2019.
[34] G. Dimić*, et al.*, "An approach to educational data mining model accuracy improvement using histogram discretization and combining classifiers into an ensemble," in *Smart Innovation, Systems and Technologies* vol. 144, ed, 2019, pp. 267-280.
[35] N. Mduma*, et al.*, "A survey of machine learning approaches and techniques for student dropout prediction," *Data Science Journal,* vol. 18, 2019.
[36] D. C. Gibson*, et al.*, "Self-organising maps and student retention: Understanding multi-faceted drivers," in *ASCILITE 2015 - Australasian Society for Computers in Learning and Tertiary Education, Conference Proceedings*, 2019, pp. 112-120.
[37] E. Filiz and E. Öz, "Finding the best algorithms and effective factors in classification of Turkish science student success," *Journal of Baltic Science Education,* vol. 18, pp. 239-253, 2019.
[38] A. Acero*, et al.*, "University dropout: A prediction model for an engineering program in bogotá, Colombia," in *Proceedings of the 8th Research in Engineering Education Symposium, REES 2019 - Making Connections*, 2019, pp. 483-490.
[39] e. a. G. Jaiswal, "Analytical approach for predicting dropouts in higher education," *International Journal of Information and Communication Technology Education, vol. 15, pp. 89-102, 2019*.
[40] A. Larrabee Sønderlund*, et al.*, "The efficacy of learning analytics interventions in higher education: A systematic review," *British Journal of Educational Technology,* vol. 50, pp. 2594-2618, 2019.
[41] J. Y. Chung and S. Lee, "Dropout early warning systems for high school students using machine learning," *Children and Youth Services Review,* vol. 96, pp. 346-353, 2019.
[42] P. Guleria and M. Sood, "Predictive data modeling: Educational data classification and comparative analysis of classifiers using python," in *PDGC 2018 - 2018 5th International Conference on Parallel, Distributed and Grid Computing*, 2018, pp. 740-746.
[43] M. Zaffar*, et al.*, "A predictive model for student outcomes using sparse coding – Hybrid features selection," *Journal of Theoretical and Applied Information Technology,* vol. 96, pp. 7124-7138, 2018.
[44] L. Breiman, "Bagging predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996/08/01 1996.
[45] *Rapidminer*. Available: https://rapidminer.com/
[46] *Python*. Available: https://www.python.org/

[47] *Scikit Learn*. Available: https://scikit-learn.org/stable/
[48] J. Grenning. *Planning Poker*. Available: https://blog.interactius.com/planning-poker-c%C3%B3mo-realizar-la-estimaci%C3%B3n-inicial-de-un-proyecto-de-forma-r%C3%A1pida-y-fiable-3576e9f1a943
[49] *Cross Validation*. Available: https://scikit-learn.org/stable/modules/cross_validation.html
[50] S. Jacobson and E. Yücesan, "Analyzing the Performance of Generalized Hill Climbing Algorithms," *J. Heuristics,* vol. 10, pp. 387-405, 2004.
[51] C. Millán Páramo*, et al.*, "Propuesta y validación de un algoritmo Simulated annealing modificado para la solución de problemas de optimización," *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería,* vol. 30, pp. 264-270, 2014/10/01/ 2014.
[52] *Grid Search Algorithm*. Available: https://www.researchgate.net/publication/286758141_An_Improved_Grid_Search_Algorithm_for_Parameters_Optimization_on_SVM
[53] *Condensed Nearest Neighbor*. Available: Voting over Multiple Condensed Nearest Neighbors - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-Condensed-Nearest-Neighbor-Algorithm-CNN_tbl1_2644628 [accessed 21 Jul, 2021]
[54] R. S. Boyer and J. S. Moore, "MJRTY—A Fast Majority Vote Algorithm," in *Automated Reasoning: Essays in Honor of Woody Bledsoe*, R. S. Boyer, Ed., ed Dordrecht: Springer Netherlands, 1991, pp. 105-117.
[55] S. Visa*, et al.*, *Confusion Matrix-based Feature Selection* vol. 710, 2011.
[56] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition,* vol. 30, pp. 1145-1159, 1997/07/01/ 1997.