Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Tesis presentada para obtener el grado de Maestría en Ciencias de la Computación.

Título: Metodología para la clasificación de alumnos en riesgo de abandonar la universidad a través del análisis de datos con técnicas de Inteligencia Artificial.

Presenta: Liosbel Cabrera Hernández.

Director de Tesis: Dr. Mario Rossainz López

Asesor de Tesis: Dr. Ivo Humberto Pineda Torres

Agosto de 2023

Resumen.

Uno de los grandes problemas que afecta a la educación es la deserción escolar. Es alto el número de estudiantes que abandonan las instituciones educativas sin haber culminado sus estudios. En particular, en la educación superior, cerca del 50% de los alumnos de Latinoamérica que inician una carrera universitaria no llegan a graduarse, tal situación impacta de manera directa en la vida de los jóvenes y sus familias, en las universidades y en la sociedad en general. Por esto, en los últimos años son varios los autores que realizan investigaciones encaminadas a tratar de detectar de manera temprana a aquellos estudiantes que tienen riesgo de abandonar la educación superior. Gracias al desarrollo creciente de los equipos de cómputo y de las herramientas de software, hoy es posible almacenar grandes volúmenes de información relacionada con los universitarios y se cuenta con técnicas avanzadas para procesar esta información. Esto ha posibilitado el surgimiento de varios sistemas computacionales que utilizando datos de los estudiantes intentan clasificar a alumnos en riesgo de abandonar la universidad, aplicando técnicas de Inteligencia Artificial. Pero aún queda mucho por trabajar en esta área, la diversidad de criterios en cuanto a la recopilación de información y su posterior uso para efectos de predicción, las diversas vertientes en el preprocesamiento de los datos, la escasa utilización de técnicas Aprendizaje Profundo (Deep Learning) para la clasificación así como la variedad de posibles métricas a utilizar para medir la calidad de los resultados; son algunos de los aspectos identificados en la revisión bibliográfica y que dejan aún un amplio margen de mejora.

Por esta razón, la presente investigación propone una metodología de clasificación de los estudiantes en riesgo de abandonar la universidad, basándose en el análisis de datos académicos y no académicos, para el caso particular del Tecnológico de Monterrey. El conjunto de datos seleccionado para probar la metodología propuesta es el que puso a disposición de la comunidad científica dicha institución el 25 de agosto del 2022 en un artículo llamado "*Student Dataset from Tecnologico de Monterrey in Mexico to Predict Dropout in Higher Education*". Se realizó un amplio preprocesamiento para mejorar la calidad de los datos y quedaron conformados cuatro conjuntos para realizar una clasificación de abandono a largo plazo y también una en el primer año de la carrera, de igual modo la clasificación se ejecuta en dos momentos, luego de la matrícula y posterior al primer semestre en la universidad. Se compararon los resultados de la clasificación obtenida a partir del uso de cinco técnicas Random Forest, XGBoost, Support Vector Machine, Deep Neural Network y ResNet. Los mejores resultados se lograron con el clasificador Support Vector Machine, el cual logró un accuracy del 97.657 % para el Conjunto 1, 95.770 % para el Conjunto 2, 99.493 % para el Conjunto 3 y un 98.718 % para el Conjunto 4. Adicionalmente, se aplica un modelo de votación compuesto por los tres clasificadores de mejores resultados para cada conjunto**.** Este modelo demostró poder llegar a tener muy buenos resultados en situaciones específicas.

Índice

1 Introducción. 1
1.1 Antecedentes del Proyecto. 1
1.2 Infraestructura. 2
1.3 Impacto Socioeconómico. 2
2 Estado del Arte 4
2.1 Datos y predictores 4
2.2 Técnicas empleadas y tipos de análisis 6
3 Metodología Propuesta 11
3.1 Análisis preliminar de datos. 11
3.2 Preprocesamiento de los datos. 11
3.2.1 Procesamiento inicial. 12
3.2.2 Ingeniería de características. 12
3.2.3 Ingeniería de instancias. 12
3.2.4 Procesamiento final. 13
3.3 Selección de las técnicas. 13
3.3.1 *Redes Neuronales Profundas*. 13
3.3.2 DNN que imita el comportamiento de ResNet. 14
3.3.3 *Random Forest*. 14
3.3.4 *Support Vector Machines*. 14
3.3.5 *XGBoost*. 14
3.3.6 Modelo integrado. 14
3.4 Evaluación de resultados. 14
4 Implementación y Resultados 16
4.1 Conjunto de Datos. 16
4.2 Preparación del Conjunto 1: 16
4.2.1 Procesamiento inicial: 16
4.2.2 Ingeniería de características. 17
4.2.3 Ingeniería de instancias. 23
4.2.4 Procesamiento final. 28
4.2.5 Resumen de preprocesamiento en Conjunto 1. 29
4.3 Preparación del Conjunto 2. 29
4.3.1 Resumen de preprocesamiento en Conjunto 2. 29
4.4 Preparación del Conjunto 3. 29
4.4.1 Procesamiento inicial. 30
4.4.2 Ingeniería de características. 30
4.4.3 Ingeniería de instancias 32

4.4.4 Procesamiento final. 34
4.4.5 Resumen de preprocesamiento en Conjunto 3. 34
4.5 Preparación del Conjunto 4. 34
4.5.1 Procesamiento inicial. 35
4.5.2 Ingeniería de características. 35
4.5.3 Ingeniería de instancias 37
4.5.4 Procesamiento final. 38
4.5.5 Resumen de preprocesamiento en Conjunto 4. 38
4.6 Implementación de Clasificadores. 38
4.6.1 División estratificada de los conjuntos. 39
4.6.2 Validación cruzada. 41
4.6.3 Clasificación inicial. 41
4.6.4 Optimización de Hiperparámetros. 42
4.7 Implementación del Modelo Integrado. 70
4.8 Discusión de resultados. 74
5 Conclusiones. 76
6 Referencias. 77

1 Introducción.

1.1 Antecedentes del Proyecto.

La deserción estudiantil es uno de los grandes problemas existentes en el ámbito educativo. La educación superior no escapa de ello, según reportes del Banco Mundial, se calcula que en América Latina solo el 50% de los estudiantes que inician sus estudios superiores llegan a terminar y se gradúan [1]. México el escenario es similar, en 2015 la Organización para la Cooperación y el Desarrollo Económicos (OCDE) informaba que México presenta un índice de deserción escolar de casi 50% [2], incluyendo todos los niveles educativos, las cifras de la educación superior evidencian la magnitud del problema, por ejemplo, en el ciclo escolar 2018-2019 previo a las afectaciones provocadas por la Covid-19, abandonaron la educación superior aproximadamente 327314 alumnos, según datos presentados por la SEP para ese periodo [3].

Esta situación se ha visto agudizada aún más como resultado de la pandemia, provocando el aumento de los índices de abandono escolar y de las consecuencias que este problema trae consigo. Un ejemplo del impacto que puede tener la deserción lo tenemos en el ámbito económico, la OCDE estima que en los próximos 80 años este fenómeno podría costarle a México un monto acumulado de hasta el 136 % del producto interno bruto de 2019, lo que equivale a una pérdida de 3,426 miles de millones de USD para el país en ese periodo [4].

El Tecnológico de Monterrey presenta un escenario favorable en comparación con otras instituciones de educación superior, sobre todo con respecto a otras universidades privadas donde los indicadores de deserción son más elevados. Pero, aun así, la deserción no deja de ser una preocupación para la dirección de la universidad, producto de esto se pone a disposición de la comunidad científica internacional un conjunto de datos recopilado para hacer frente al fenómeno de la deserción.

A partir del documento publicado por el Tecnológico de Monterrey, se evidencia que un número no despreciable de alumnos dejan cada año las aulas. Porque, aunque la tasa de abandono en el nivel de licenciatura es relativamente baja con un 8%, las altas matrículas que se manejan en la institución generan un alto número de estudiantes no completen su periodo universitario.

Es evidente la necesidad de disponer de mecanismos automáticos que permitan anticipar la deserción escolar; desarrollar investigaciones y proyectos que aporten soluciones para contribuir a la paulatina disminución de los índices de deserción, y de todos los problemas que esto trae consigo.

Problema: Existencia de altas tasas de deserción estudiantil en la educación superior, lo cual afecta a los estudiantes y sus familias, a universidades y a la sociedad.

Pregunta de Investigación: ¿Será posible crear una metodología para la identificación de los estudiantes que presenten un mayor riesgo de abandonar la universidad utilizando sus datos?

Objetivos Generales y Específicos del Proyecto.

Objetivo General:

- Diseñar una metodología de análisis de datos para la clasificación de alumnos en riesgo de abandonar la universidad utilizando técnicas de inteligencia artificial.

Objetivos específicos:

- Preparar los datos necesarios para el entrenamiento de los modelos aplicando métodos de procesamiento de datos.
- Seleccionar técnicas de clasificación en el área del Machine Learning y Deep Learning a partir del análisis de investigaciones anteriores para obtener los mejores resultados posibles.
- Seleccionar las métricas que evalúen los resultados de la clasificación en diferentes aspectos para verificar el correcto funcionamiento de los modelos.

1.2 Infraestructura.

El proyecto se llevará a cabo utilizando el equipamiento del *Laboratorio Nacional de Supercómputo del Sureste de México*, el cual se encuentra ubicado en la BUAP y dispone de avanzados recursos de cómputo, con altas prestaciones. Los recursos específicos de los que se harán uso para el desarrollo de la investigación dependerán de la asignación particular que se destine para este trabajo.

1.3 Impacto Socioeconómico.

La deserción escolar tiene un impacto directo en el alumno, al cual se le frustran sus posibilidades de superación y sus opciones futuras de tener acceso a un mejor empleo. De acuerdo con el Banco Mundial, en estudio realizado para simular el impacto de la Covid-19 en la educación, se estima que el estudiante promedio que sale del sistema educativo pierde el equivalente al 8% del ingreso que podría haber percibido a lo largo de su vida [5].

El Informe sobre la Educación Superior en América Latina y el Caribe 2000-2005, “La Metamorfosis de la Educación Superior”, patrocinado por la UNESCO, comenta que las implicancias de la deserción en la educación superior se pueden distinguir tres categorías: sociales, institucionales y personales [6].

- Entre las sociales está la retroalimentación del círculo de la pobreza y la gestación de una “capa social” de frustrados profesionales, con posible disminución del aporte intelectual y el potencial aumento del subempleo. Adicionalmente, se incrementa el costo para el país de la educación

asociada a una suboptimización de los recursos debido al costo de la deserción.

- Entre las institucionales está la limitación para cumplir la misión institucional y un descenso en los índices de eficiencia y calidad. De igual manera, tiene implicaciones económicas debido a los menores ingresos por matrícula.
- Entre las personales está el disgusto, la frustración y la sensación de fracaso de los repitentes y desertores con los consiguientes efectos en su salud física y mental. Asimismo, se produce una pérdida de oportunidades laborales dadas las menores posibilidades de conseguir empleos satisfactorios y la postergación económica por salarios más bajos.

Como se puede apreciar, el problema de la deserción escolar tiene consecuencias de gran magnitud, que abarcan a la sociedad en general. Por tanto, desarrollar proyectos que puedan contribuir a mejorar esta realidad, puede tener gran repercusión en presente y futuro de la sociedad. Con cada alumno en riesgo de abandonar la universidad, que se logre identificar de manera temprana, con la metodología que se está proponiendo, existirá la posibilidad de aplicar medidas institucionales que intenten evitar la salida del estudiante del sistema educativo y con ello influir de manera positiva en la vida del alumno y la de su familia, mejorar los parámetros de la institución educativa, así como disminuir sus gastos y los gastos del presupuesto nacional, en resumen, se estará contribuyendo a tener un mejor país.

2 Estado del Arte

En los últimos años, han crecido el número de investigaciones que tratan de realizar una predicción temprana de la deserción escolar a través del análisis de datos de los estudiantes utilizando técnicas de aprendizaje automático. La detección temprana de estudiantes en riesgo permite a las instituciones de educación superior ofrecer asistencia individualizada en diversas formas, incluidos cursos de recuperación y sesiones de tutoría para mitigar el fracaso académico [7]. Se decide analizar el estado actual de la investigación siguiendo dos líneas a) Tipos de datos y predictores y b) Técnicas empleadas y tipos de análisis.

2.1 Datos y predictores

Un primer punto que se tiene en cuenta para este tipo de problemas es la selección de los atributos para tener en cuenta o predictores, al respecto existe una gran diversidad en la bibliografía consultada. Los autores [8-10] solo utilizan información académica de los estudiantes en sus modelos de predicción, en aras de trabajar con información que normalmente resulta más asequible. En el análisis de los datos escolares se reportan resultados sorprendentes en [11] demostrando que el rendimiento escolar en humanidades tiene impacto incluso en el desempeño universitario en carreras de ingeniería. En [10] se concluye que el rendimiento académico previo, el rendimiento académico de primer año, la asistencia a clases y la fecha de matriculación son variables que se relacionan directamente con la deserción.

Pero queda demostrado en numerosas investigaciones la importancia de información adicional para mejorar los resultados de la detección del abandono escolar. En [7] aportan un análisis estadístico acerca de las variaciones de las tasas de deserción en dependencia de las actividades extracurriculares realizadas por los alumnos. En [12] detectan que el puntaje educativo preuniversitario y el índice de pobreza comunitaria son variables predictivas importantes. Adicionalmente, [13, 14] demostraron la importancia de los parámetros no académicos para predecir o estimar el rendimiento de los estudiantes. Específicamente, en [13] se diseñaron dos conjuntos de datos para demostrar la importancia de los atributos no académicos, utilizando únicamente datos académicos lograron un 79,6 % en la clasificación y usando todos los parámetros se alcanzó un 93,8 %. Otro grupo de publicaciones como [15-17] incluyen varios tipos de variables sociodemográficas, socioeconómicas y de comportamiento estudiantil.

En lo referente al número de atributos a utilizar existe una gran diversidad en los artículos sobre el tema, la mayoría de los consultados reportan haber tenido en cuenta entre 10 y 20 parámetros [9, 13, 15, 16]. Existen otros que consideran más aspectos para la detección de abandono, ubicándose en un rango entre 21 y 40 [8, 11, 12].

Otro aspecto muy variable en los datos es la cantidad de estudiantes que se tienen en los diferentes experimentos. Varias referencias cuentan con cifras cercanas a las 6000 instancias para sus pruebas [12-14]. También se encuentran trabajos con menor número de individuos, por ejemplo [10, 16] tienen entre 2500 y 4500 alumnos. Pero en este tipo de problema está demostrado que cuanto mayor sea el número de instancias con que se cuenta para el entrenamiento, más precisos suelen ser los resultados, en esta revisión los valores más altos encontrados rondan los 15000 y son los casos de [11, 15].

Los modelos revisados siguen diferentes criterios con respectos a cuantos semestres del alumno en la universidad se deben incluir en los conjuntos de datos para efectuar la predicción. En varios casos se crean varios modelos donde cada uno se entrena con un conjunto de datos distinto. En [8] se crea 5 modelos predictivos distintos, uno para el momento de la inscripción, otro terminado el primer semestre donde ya van incluyendo los resultados del alumno en esa etapa, terminado el primer año, terminado el tercer semestre y terminado el segundo año. Algo similar hace [12] evaluando cuatro modelos predictivos, un modelo global para la predicción de la retención estudiantil, independientemente del nivel (año) en el que se produzca la deserción. El segundo modelo para los que abandonaron sus estudios durante el primer año. Los modelos tercero y cuarto para predecir la retención durante el segundo y tercer año de estudio. Otros autores como [11] utilizan otro enfoque en el que solo utilizan datos captados en el momento de la inscripción. El resumen de los datos utilizados en la bibliografía consultada se muestra en la Tabla 1.

En esta investigación se pretenden utilizar datos académicos y no académicos debido a que se ha demostrado que esto influye positivamente en la predicción. Por tanto, se empleará información general del estudiante, sociodemográfica, datos de la educación anterior, resultados de las pruebas de ingreso, resultados del primer semestre y participación en actividades extracurriculares. Se pretende analizar un gran número de atributos de cada alumno, buscando obtener mayor exactitud en la clasificación. Se crearán cuatro conjuntos de datos, que efectuarán una predicción en diferentes momentos y con distintos objetivos. Con estos cuatro modelos se pretende incidir de manera oportuna en la deserción general y en la del primer año de estudio, que según estadísticas internacionales es la que más impacta en la educación superior.

El siguiente aspecto que se tiene en cuenta en la literatura una vez que se determinan los predictores que se utilizaran como entrada del modelo es la preparación de los datos. En [8] se hace hincapié en el procesamiento de los datos debido a que cuentan con pocos atributos, aplicaron todas las técnicas de ingeniería de características e ingeniería de instancias para obtener el mayor provecho posible del conjunto de características. Además, aplican otras técnicas como son la codificación y la normalización. De igual manera, [10, 11] acometen acciones para manejar los datos faltantes, transformar y generar atributos, reducir dimensiones y redundancias y la limpieza de los datos. Los autores de

[15] demuestran que es conveniente excluir a los estudiantes activos, porque no se sabe de antemano si abandonarán o se graduarán en el futuro. En [8, 9, 11-13] se reporta el problema del desequilibrio entre clases, dado porque el porciento de estudiantes que terminan sus estudios universitarios es normalmente mayor que el de los que abandonan, estas investigaciones prueban que esta diferencia produce problemas en la clasificación de la clase minoritaria y aplican diversas técnicas para el equilibrio de clases.

Se demuestra en la revisión realizada la importancia del preprocesamiento de los datos y la necesidad de resolver problemas siempre presentes en este tipo de problemas como el desequilibrio de clases y los ruidos que producen los estudiantes activos.

Tabla 1. Características de los datos en el estado del arte.

Artículo	Datos utilizados		# Atributos	# Alumnos	Periodo de análisis				
	Académico	No académico			Previo	1.er	2.o	3.er	4.o
[8]	X		22	1418	X	X	X	X	X
[9]	X		16	N/A	X	X	X	X	
[10]	X		N/A	4434		X	X		
[7]	X	X	50	143326	N/A				
[12]	X	X	41	6656		X		X	
					X				
[15]	X	X	19	15720	Todos los semestres de la carrera				
[11]	X	X	35	15825	X				
					Todos los semestres de la carrera				
[13]	X	X	20	6807	X				
[14]	X	X	30	6000	X				
					Todos los semestres de la carrera				
[16]	X	X	11	2670	N/A				
[17]	X	X	23	6078	X	X		X	
[18]	X	X	27	8319	X				

2.2 Técnicas empleadas y tipos de análisis

En cuanto a las técnicas de aprendizaje automático, para resolver este tipo de problemas, existe una gran diversidad de opiniones en la comunidad científica. Muchas fuentes eligen algoritmos llamados clásicos, por ejemplo [8] compara 3 algoritmos disponibles en la librería *scikit-learn* de *Python* que fueron: *Gradient Boosting* (GB), *Random Forest* (RF) y *Support Vector Machine* (SVM), se

aplicaron diferentes conjuntos de datos y las tres técnicas se superaron indistintamente dependiendo de los datos de entrada. También [12] compara varias técnicas utilizando *Weka* y agrega a dos de las mencionadas anteriormente (RF y SVM) las siguientes: *Decision Trees* (DT), *K-nearest Neighbors*, *Logistic Regression* (LR) y *Naive Bayes* (NB). RF fue la que presentó los mejores resultados en la clasificación. En [10] se repite la utilización de cuatro de estos algoritmos LR, NB, SVM y RF, agregando además la técnica de *Adaptive Boosting* (AB), la biblioteca *scikit-learn* vuelve a ser la elegida por los autores para implementar estas técnicas. Trabajaron con dos conjuntos de datos, para el más pequeño los mejores resultados se obtuvieron con LR y para el más grande con AB. Otros autores siguen la tendencia de emplear algoritmos clásicos, es el caso de [9] que compara LR y DT, este último demostró tener ligeramente mejores resultados.

Existe otro grupo de investigadores que incorpora el uso de redes neuronales para resolver este problema. Este es el caso de [13, 15] que además de continuar el trabajo con RF, LR y SVM, implementan una red neuronal de tipo *Perceptrón Multicapa* (PM), aunque los mejores resultados los obtuvieron con RF. La investigación de [16] basa su trabajo solo en las redes neuronales, comparando el rendimiento de un PM y una *Red de Base Radial* (RBR), la precisión de esta última es ligeramente superior a la del PM. En el caso de [11] ya emplean un algoritmo de *Deep Learning* (DL), una *Deep Neural Network* (DNN) que compara contra RF y GB, trabajaron con dos conjuntos de datos, uno con más datos que el otro, sus resultados evidencian que la DNN obtiene mejor resultado que el resto cuando se emplea en el conjunto de datos más grandes. Otra investigación que compara algoritmos clásicos con técnicas de DL es [18] que utiliza RF, GB y además implementa tres arquitecturas de DL, que son *Fully Connected Neural Network* (FCNN), *TabNet* y *BaggingFCNN*, la que logró mejores métricas en la clasificación fue la FCNN. En [14] continúa aplicando técnicas de DL comparando la efectividad de una CNN con una *Red Bayesiana* (RB). Demostrando que usar CNN da mejores resultados. En las CNN profundiza [17] comparando tres arquitecturas distintas, *ResNet*, *VGG* y una combinación de ambas propuesta por los autores; los resultados de las 3 redes fueron muy similares, siendo ligeramente mejor *ResNet*. El resumen de las diferentes técnicas empleadas por los investigadores en la revisión que se realizó se muestra en la Tabla 2.

De esta parte de la investigación se ha podido concluir que la gran mayoría de los investigadores eligen los algoritmos clásicos como método para determinar la deserción escolar. Esta elección suele estar dada por varios aspectos, el bajo consumo de recursos computacionales, la simplicidad de la implementación de este tipo de técnicas y los buenos resultados que presentan estos algoritmos en mucho de los casos. En algunos artículos se introduce el uso de las redes neuronales tradicionales. Son muy pocas las publicaciones que utilizan técnicas de DL, pero en ellas se evidencia que pueden ser muy efectivas cuando se tienen

conjuntos de datos de gran magnitud. Solo una de las bibliografías consultadas [8] emplea un método que integra más de una técnica de Aprendizaje Automático para dar un resultado conjunto, se toman las salidas de tres técnicas aplicadas al mismo grupo de datos y se da como salida del sistema el resultado donde coincidan dos o más de ellas.

Tabla 2. Técnicas utilizadas en el estado del arte.

	Referencia									
Técnica	**[8]**	**[12]**	**[10]**	**[9]**	**[20, 21]**	**[16]**	**[11]**	**[14]**	**[17]**	**[18]**
Gradient Boosting	X						X			X
Random Forest	X	X	X		X		X			X
Support Vector Machine	X	X	X		X					
K-nearest Neighbors		X								
Decision Trees		X		X						
Logistic Regression		X	X	X	X					
Naive Bayes		X	X							
Adaptive Boosting			X							
Perceptrón Multicapa					X	X				
Red de Base Radial						X				
Red Neuronal Profunda							X			X
CNN								X	X	
Red Bayeciana								X		

Para este proyecto se prevé contar con una gran cantidad de datos, por ello se cree que utilizar técnicas de DL, como las Redes Neuronales Convolucionales y Redes Neuronales Profundas, pueden producir muy buenos resultados. Por otra parte, para esta investigación se cuenta con grandes recursos computacionales, esto elimina la principal limitación que tienen los autores para aplicar métodos que implican el uso simultáneo de varios algoritmos, lo cual puede llevar a obtener una mayor exactitud en la predicción.

Un último debate que se repite en la bibliografía es el referente al tipo de métricas a utilizar para medir la calidad de los resultados. Aunque la mayoría de los autores expresan sus resultados finales en términos de *Accuracy*, se utilizan otras fórmulas que cobran gran importancia a la hora de determinar el rendimiento de un algoritmo. En [10, 14] se emplean *Accuracy*, *Precision*, *Recall* y *F1* para evaluar sus resultados. Los primeros dan mayor importancia a los valores de *F1*, que se usa para mezclar la *Precision* y el *Recall* en un único valor. Mientras que los segundos considera la deserción como la clase positiva y la no deserción como la clase negativa. Y quieren minimizar los falsos negativos (los estudiantes que abandonan la escuela y se predicen como estudiantes que no abandonan) por tanto, seleccionan los modelos con mayor *Recall*. *F1* también es el valor priorizado para comparar los diferentes resultados obtenidos por [13].

En [8, 11] utilizan las 3 primeras métricas de la 4 citadas inicialmente. Los segundos además incorporan el análisis de las curvas *ROC* para evaluar el comportamiento de sus modelos. Tendencia que siguen [9, 16] incluyendo de igual manera la curvas *ROC* que son una representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador binario, además ambas utilizan la *Especificidad* y el Recall. Otro caso es el de [15] que emplea estas dos últimas y además incorpora el coeficiente *Kappa* que mide el grado de concordancia de las evaluaciones nominales u ordinales realizadas por múltiples evaluadores cuando se evalúan las mismas muestras. Los investigadores de [12] van un poco más allá y utilizan otros instrumentos como *Error Cuadrático Medio*, *k-statistic* y *Valor de Friedman*, además de seguir empleando otras de las mediciones aquí expuestas. En prácticamente todas las publicaciones se hace uso de la Matriz de Confusión para tener mayor claridad del comportamiento de los modelos planteados. La Tabla 3 resume las métricas empleadas en la bibliografía consultada.

Tabla 3. Métricas utilizadas en el estado del arte.

Métricas	**[10, 14]**	**[13]**	**[5]**	**[18]**	**[9, 16]**	**[15]**	**[12]**
Accuracy	X	X	X	X			X
Precision	X	X	X	X			X
Recall	X	X	X	X	X	X	
F1	X	X					X
Curvas ROC				X	X		
Especificidad					X	X	
Kappa						X	
Error cuadrático medio							X
k-statistic							X
Valor de Friedman							X

Para nuestra propuesta se ha decidido adoptar las métricas más empleadas en la literatura que fueron *Accuracy*, *Precision*, *Recall* y *F1*.

Fueron consultados artículos que estudian el problema específicamente en México, en los que proponen sistemas para la detección temprana de las bajas en el nivel superior. En [19] se propone un modelo que utiliza información obtenida del examen de admisión EXANI II, su base de catos cuenta con 671 registros. Utilizando el software Weka se comparan los comportamientos de 5 algoritmos NB, PM, *J48*, RF y *Random Tree*. Los mejores resultados se obtuvieron con RF mostrando una precisión de un 86%. Por su parte, [20] realiza un estudio comparativo de cuatro algoritmos: LR, *Clúster*, DT y *Red Neuronal de Microsoft*. Se utiliza información personal y académica de los alumnos. Los resultados indican que el algoritmo de LR es el más adecuado. El análisis se

lleva a cabo con 134 alumnos, obteniéndose valores de Accuracy cercanos al 95 %.

Ambas investigaciones utilizan algoritmos clásicos en sus modelos, con la problemática de que en los dos casos se cuenta con los datos de pocos estudiantes para el entrenamiento de los modelos, lo cual le resta fiabilidad a los valores que obtuvieron en la clasificación. En el primer artículo solo emplean datos de los exámenes de ingreso, los cuales pueden ser de mucha utilidad en la predicción, pero ha quedado demostrada la importancia de incorporar datos no académicos de los estudiantes. En el segundo no se describen con claridad los predictores que utilizan, se menciona que se emplea también información personal de los alumnos, pero no se detallan los atributos.

Es evidente que en este campo aún existe mucho trabajo por hacer para lograr sistemas más robustos y que arrojen resultados realmente certeros que ayuden a las universidades a enfrentar el problema de la deserción. Muchas instituciones en México muestran una preocupación creciente en este aspecto, muestra de ello es la reciente publicación realizada por el Tecnológico de Monterrey en el pasado mes de agosto [7], donde ponen a disposición de la comunidad científica un Dataset de 143,326 registros que contiene datos de estudiantes del TEC, para que puedan ser estudiados y realizar propuestas que intenten predecir de forma efectiva el problema aquí tratado.

3 Metodología Propuesta

Se propone una metodología para la detección de alumnos en riesgo de abandonar la universidad. La misma está compuesta por cuatro etapas, las cuales se describen a continuación.

3.1 Análisis preliminar de datos.

En este primer momento se definen los parámetros necesarios que permitirán la detección de estudiantes en riesgo de abandonar la universidad. En la revisión realizada, se han identificado categorías de atributos que impactan de forma positiva en la solución de este problema, por ejemplo:

- Generales (edad, sexo, carrera, etc.).
- Sociodemográficas (municipio, colonia, ingreso mensual familiar, etc.).
- Educación anterior y exámenes de ingreso (escuela de procedencia, tipo de educación, promedio final, etc.).
- Resultados del primer semestre (promedio, ausencias, inscripción a actividades deportivas, etc.).

Luego de contar con un conjunto de datos que cuente con estas categorías. Se separan para la obtención de cuatro conjuntos.

- Conjunto 1: Contendrá todos los atributos válidos del alumno para realizar una predicción del abandono a lo largo de la carrera.
- Conjunto 2: Contendrá todos los atributos válidos del alumno hasta el momento de su ingreso a la universidad para realizar una predicción temprana del abandono a lo largo de la carrera.
- Conjunto 3: Contendrá el total de atributos válidos del alumno para realizar una predicción del abandono en el primer año.
- Conjunto 4: Contendrá el total de atributos válidos del alumno hasta el momento de su ingreso a la universidad para realizar una predicción del abandono en el primer año.

Los conjuntos 1 y 2 permitirán realizar una predicción del abandono a largo plazo, en dos momentos distintos, en el caso del primero se ejecutará una vez terminado el primer semestre del primer año y el segundo se ejecutará antes de que el alumno comience la carrera.

Los conjuntos 3 y 4 permitirán realizar una predicción del abandono en el primer año. De igual forma en dos momentos, el Conjunto 3 luego del primer semestre y el 4 antes de iniciar la carrera. Ambos conjuntos servirán para realizar un pronóstico en una de las etapas críticas en cuanto a deserción universitaria se refiere, el primer año, donde como se afirma en [24], se producen el 58% de los abandonos.

3.2 Preprocesamiento de los datos.

Para esta etapa se emplearán varias técnicas identificadas en la revisión de la literatura, que trabajan a profundidad la limpieza y preparación de los datos.

3.2.1 Procesamiento inicial.

- Valores atípicos: Valores que se encuentren fuera de rango o resulten ilógicos dentro de la variable que lo contenga.
- Codificación: Los modelos de aprendizaje automático requieren que todas las variables de entrada y salida sean numéricas. Esto significa que si los datos contienen datos categóricos, deben codificarse en números antes de poder probar y evaluar un modelo [21].

3.2.2 Ingeniería de características.

Es la idea que subyace tras la búsqueda de características particulares que puedan resultar clave para resolver problemas concretos [22].

- Características inútiles: Serán aquellas variables que por una razón u otra no aporte absolutamente nada al conjunto de datos.
- Correlación: La correlación es una medida de cómo dos variables cambian juntas. Quizás la medida de correlación más común es la correlación de *Pearson,* que supone una distribución gaussiana para cada variable e informa sobre su relación lineal. Las puntuaciones de correlación lineal suelen tener un valor entre -1 y 1, donde 0 no representa ninguna relación [21].
- Características de alta cardinalidad: Algunas variables pueden tener una alta cardinalidad, donde las variables categóricas tienen numerosos niveles. La alta cardinalidad puede ser muy problemática en modelos analíticos como regresión o redes neuronales porque para cada nivel crea una estimación de parámetro o un parámetro ponderado, respectivamente. Estos niveles esencialmente crean modelos que son mucho más complejos al final del proceso [23].
- Valores faltantes: Los datos del mundo real a menudo tienen valores faltantes. Esto puede suceder por varias razones, como observaciones que no se registraron y corrupción de datos. El manejo de los datos faltantes es importante, ya que muchos algoritmos de aprendizaje automático no admiten datos con valores faltantes [21].

3.2.3 Ingeniería de instancias.

Tiene la capacidad de mejorar el rendimiento de los modelos a través de la selección y manipulación adecuada de las instancias [8].

- Agrupamiento: También conocido como categorización o discretización, es el proceso de traducir una variable cuantitativa en un conjunto de dos o más cubos cualitativos (es decir, categorías). Los valores de la variable se agrupan en contenedores discretos y a cada contenedor se le asigna un número entero único, de modo que se conserva la relación ordinal entre los contenedores. El mapeo proporciona una clasificación de valores de alto orden que puede suavizar las relaciones entre las observaciones [21].

- Desequilibrio de clases: En los problemas de clasificación, tradicionalmente, se supone que disponemos de ejemplos de distintas clases. En determinadas aplicaciones, unas clases son mucho más frecuentes que otras. Entonces es cuando hablamos de clasificación no balanceada [22]. El desequilibrio de clases puede tener un impacto negativo en la precisión y el rendimiento de los modelos de aprendizaje automático. Cuando las clases están desequilibradas, los modelos tienden a sesgarse hacia la clase mayoritaria y tienen dificultades para predecir con precisión las muestras de la clase minoritaria.

3.2.4 Procesamiento final.

Última técnica que se aplicaran al conjunto de datos.

- Escalado: Los modelos de aprendizaje automático aprenden una asignación de variables de entrada a una variable de salida. Como tal, la escala y la distribución de los datos extraídos del dominio pueden ser diferentes para cada variable. Las variables de entrada pueden tener diferentes unidades (por ejemplo, pies, kilómetros y horas) que, a su vez, pueden significar que las variables tienen diferentes escalas. Las diferencias en las escalas entre las variables de entrada pueden aumentar la dificultad del problema que se está modelando [21].

3.3 Selección de las técnicas.

La carrera universitaria de un estudiante puede resultar en la graduación o la deserción. Por lo tanto, el problema se puede plantear como un modelo de clasificación binaria supervisada [9].

Se utilizarán dos técnicas de *Deep Learning*, debido a los resultados favorables que producen este tipo de algoritmos con grandes cantidades de datos. También se utilizarán tres de los algoritmos clásicos de *Machine Learning* con mayor presencia en la literatura revisada.

Además de utilizar estas cinco técnicas por separado, se propone analizar los resultados de un modelo que integre las tres técnicas de mejores resultados para cada conjunto, modelo que se aplica en la investigación llevada a cabo por [8], pero en su caso integra solo algoritmos clásicos de *Machine Learning*.

Las técnicas seleccionadas serán implementadas en el lenguaje Python, el cual es el más utilizado en este tipo de investigaciones según la bibliografía consultada e incluye librerías que facilitan la implementación de cada una de estas tecnologías.

3.3.1 *Redes Neuronales Profundas*.

Las DNN son modelos computacionales que consisten en muchas unidades de procesamiento simples que funcionan en paralelo y están dispuestas en capas interconectadas. Las redes neuronales simples constan de una capa de entrada

y una capa de salida; cuando se apilan más capas, las redes se denominan profundas [26].

Para la selección de este modelo se tuvo en cuenta un análisis comparativo entre varias arquitecturas de DNN realizado por [18], donde se emplean tres tipos de DNN en la predicción de la deserción estudiantil. Los mejores resultados se obtuvieron con una red neuronal profunda simple totalmente conectada capaz de lograr un rendimiento alto en conjuntos de datos tabulares.

3.3.2 DNN que imita el comportamiento de ResNet.

En este proyecto se utilizará la arquitectura que imita el comportamiento de ResNet propuesta por [37], variantes de Resnet han sido probadas para este tipo de problemas por [14, 17], obteniendo buenos resultados.

3.3.3 *Random Forest*.

RF es un clasificador que consta de una colección de clasificadores con estructura de árbol donde cada árbol es un vector aleatorio independiente distribuido de forma idéntica y cada uno emite un voto unitario para la clase más popular en la entrada x [27].

3.3.4 *Support Vector Machines*.

Las máquinas de vectores de soporte son una familia de métodos de aprendizaje automático, introducidas por *Vapnik* y colaboradores en la década de 1990. Se basan en los principios de la teoría del aprendizaje estadístico y la optimización convexa para encontrar un límite adecuado en el espacio de datos para separar las dos clases de puntos [28].

3.3.5 *XGBoost*.

XGBoost es la abreviatura de *eXtreme Gradient Boosting*. Es una implementación eficiente y escalable del marco de refuerzo de gradiente propuesto por Friedman et al. en el año 2000. El paquete incluye un eficiente solucionador de modelos lineales y un algoritmo de aprendizaje de árboles. Es compatible con varias funciones objetivas, incluidas la regresión y la clasificación [29].

3.3.6 Modelo integrado.

En este modelo de las cinco técnicas anteriormente mencionadas se elegirán las tres que obtengan mejores resultados y cada una realizará una clasificación sobre cada una de las instancias del conjunto de datos. La salida del modelo para cada registro será aquella en la que coincidan al menos dos de las tres tecnologías.

3.4 Evaluación de resultados.

Solo existen cuatro posibilidades en la clasificación binaria, que podemos representar en una matriz de confusión:

- Verdaderos positivos [TP: True Positive]: Los ejemplos de la clase positiva que nuestro clasificador es capaz de clasificar correctamente.
- Falsos positivos [FP: False Positive]: Los ejemplos que, aun no siendo de la clase positiva, nuestro clasificador predice que sí lo son.
- Falsos negativos [FN: False Negative]: Los errores que comete nuestro clasificador en sentido contrario, diciéndonos que no son de la clase positiva cuando en realidad sí que lo son.
- Verdaderos negativos [TN: True Negative]: Los ejemplos de la clase negativa que nuestro clasificador clasifica correctamente.[22]

Tabla 4. Matriz de confusión.

		Predicción	
		P	N
Clase Real	P	TP	FN
	N	FP	TN

Partiendo de esto se analizarán los resultados con las métricas de:

- $acurracy = \frac{\mathrm{TP+TN}}{T\,P + T\,N + F\,P + F\,N}$
- $precision = \frac{TP}{TP+FP}$
- $recall = \frac{TP}{TP+FN}$
- $F1 = \frac{2*precision*recall}{precision+recall}$ [22]

Las cuales son las elegidas por la mayoría de los autores en las investigaciones de este tipo. Porque permiten tener una idea clara del comportamiento de los modelos.

4 Implementación y Resultados

4.1 Conjunto de Datos.

El conjunto de datos seleccionado para probar la metodología propuesta es el que puso a disposición de la comunidad científica el Tecnológico de Monterrey el 25 de agosto del 2022 en un artículo llamado "*Student Dataset from Tecnologico de Monterrey in Mexico to Predict Dropout in Higher Education*".

La conformación del conjunto de datos es la siguiente:

- 121,584 estudiantes de preparatoria y/o licenciatura.
- 143,326 registros.
- 50 variables.
- 77,517 registros de pregrado.
- 65,809 registros de preparatoria.
- Siete cohortes de ingreso a la institución del 2014 al 2020.

Para el caso específico de los datos del Tecnológico de Monterrey descartaremos los 65809 registros pertenecientes a preparatoria y se comenzará el procesamiento de datos con 50 variables y 77,517 instancias.

A lo largo de esta etapa de preprocesamiento se necesita medir el impacto de cada una de las pruebas que se realizarán en la clasificación, para esto se utilizará *XGBClassifier* perteneciente a la librería *xgboost* con sus parámetros por defecto, que es el único de los algoritmos clásicos seleccionados que admite datos faltantes, que aún poseen nuestro conjunto. La métrica de evaluación a utilizar para comparar resultados será el *Accuracy*, empleando *accuracy_score* perteneciente a la librería *sklearn.metrics*.

4.2 Preparación del Conjunto 1:

Como se explica en la metodología, este es un conjunto para una predicción a lo largo de la duración de la carrera. Por tanto, solo debemos tener estudiantes que hayan matriculado en una fecha anterior a 4 años, para eliminar la incertidumbre que producen los estudiantes que no cumplen con el tiempo mínimo para graduarse, los cuales no conocemos si terminaran la carrera o abandonaran. Se eliminan las instancias correspondientes a las generaciones 2019 y 2020. Con la eliminación quedan 56233 instancias.

4.2.1 Procesamiento inicial:

- **Valores atípicos.**

Se analizaron los valores dentro del conjunto y se identificó que *admission.test* presenta dos escalas distintas. Según la descripción de los datos, una escala es de 1-100 (en periodo COVID-19) y otra de 400-1600 (examen presencial en periodo normal). Resulta conveniente convertir el primer rango al segundo para lograr homogeneidad en los datos.

Para este fin se diseñó una función en Python que escala los valores pertenecientes al examen en línea, llevándolos al rango entre 400 y 1600.

- **Codificación.**

La conversión de datos categóricos a datos numéricos es un proceso común en este tipo de investigaciones. En muchos casos ente proceso se realiza en una etapa posterior, en este trabajo se decide efectuar al inicio, porque muchas de las herramientas de Python que se utilizaran en los próximos puntos del preprocesamiento solo admiten datos numéricos. A continuación, se describen los métodos más comunes de codificación.

- Codificación ordinal.

En la codificación ordinal, a cada valor de categoría único se le asigna un valor entero. Por ejemplo, el rojo es 1, el verde es 2 y el azul es 3. Esto se denomina codificación ordinal o codificación entera y es fácilmente reversible [21].

- Codificación en caliente.

Cada bit representa una categoría posible. Si la variable no puede pertenecer a varias categorías a la vez, entonces solo un bit del grupo puede estar *on*. En el ejemplo de la variable de color, hay tres categorías y, por lo tanto, se necesitan tres variables binarias. Se coloca un valor 1 en la variable binaria para el color y valores 0 para los otros colores [21].

Se ejecutan estas dos variantes de codificación utilizando *OrdinalEncoder* incluido en *sklearn.preprocessing* y *OneHotEncoder* perteneciente a la misma librería. En este primer momento las dos variantes lograron un *accuracy* del 100 %. Como el impacto de la codificación no demostró ser relevante para este caso en particular, se decide adoptar *OrdinalEncoder* como técnica de codificación. Debido a ser un conjunto de grandes dimensiones, el método *OneHotEncoder* aumenta en gran medida el número de columnas llevándolo hasta 4387, lo cual dificulta el trabajo con los datos.

4.2.2 Ingeniería de características.

- **Características inútiles.**

Se eliminarán los siguientes atributos:

- *level* - Al eliminar los estudiantes de preuniversitario del conjunto de datos, *level* asume un mismo valor para el resto de las instancias.
- *educational.model* - Al eliminar las generaciones 2019 y 2020, *educational.model* asume un mismo valor para el resto de las instancias.
- *online.test* - Al eliminar las generaciones 2019 y 2020, *online.test* asume un mismo valor para el resto de las instancias.

- **Valores faltantes, eliminación de variables.**

Aunque este punto del análisis está planteado en la metodología como último en la ingeniería de características, en particular este conjunto de datos contiene

variables que tienen un porcentaje de datos faltantes extremadamente altos, estos casos vale la pena analizarlos en este momento por si es pertinente eliminarlas del conjunto y del resto del análisis.

Realizando un análisis lógico y analizando las recomendaciones de autores que han profundizado en el tema [30], se determinó que las variables que posean más de un 50 % de valores faltantes serán eliminadas. Para conocer el total de datos faltantes que presenta cada una de las variables se utilizará la función *isnull* de la librería Pandas en Python.

Se eliminarán los siguientes atributos:

Las siguientes seis variables tienen un 100 % de datos faltantes, esto se debe a que es información que solo se capta en las generaciones 2019 y 2020, las cuales fueron eliminadas para este primer conjunto.

- *social.lag* – 100 %.
- *average.first.period* – 100 %
- *failed.subject.first.period* - 100 %
- *dropped.subject.first.period* - 100 %
- *zone.type* - 100 %
- *socioeconomic.level* - 100 %

Las siguientes seis variables tienen un 79 % de datos faltantes, esto se debe a que es información que solo se capta en las generaciones 2018, 2019 y 2020, de las cuales solo se mantuvo la 2018 para este primer conjunto.

- *art.culture* - 79 %
- *total.life.activities* - 79 %
- *wellness.activities* - 79 %
- *athletic.sports* - 79 %
- *life.work.mentoring* - 79 %
- *student.society.leadership* - 79 %

En el caso de *first.generation* tienen un 70 % de datos faltantes, esto se debe a que es información que solo se capta en las generaciones 2017, 2018, 2019 y 2020, de las cuales solo se mantuvieron las de 2017 y 2018 para este primer conjunto.

- *first.generation* - 70 %

De las cinco características siguientes se encuentran datos en todas las generaciones, pero en las últimas es donde existe mayor presencia, por esto luego de la eliminación de estas el porcentaje de datos faltantes es cercano al 70 %.

- *mother.education.summary* - 68 %
- *mother.education.complete* - 68 %
- *father.education.summary* - 68 %
- *father.education.complete* - 68 %

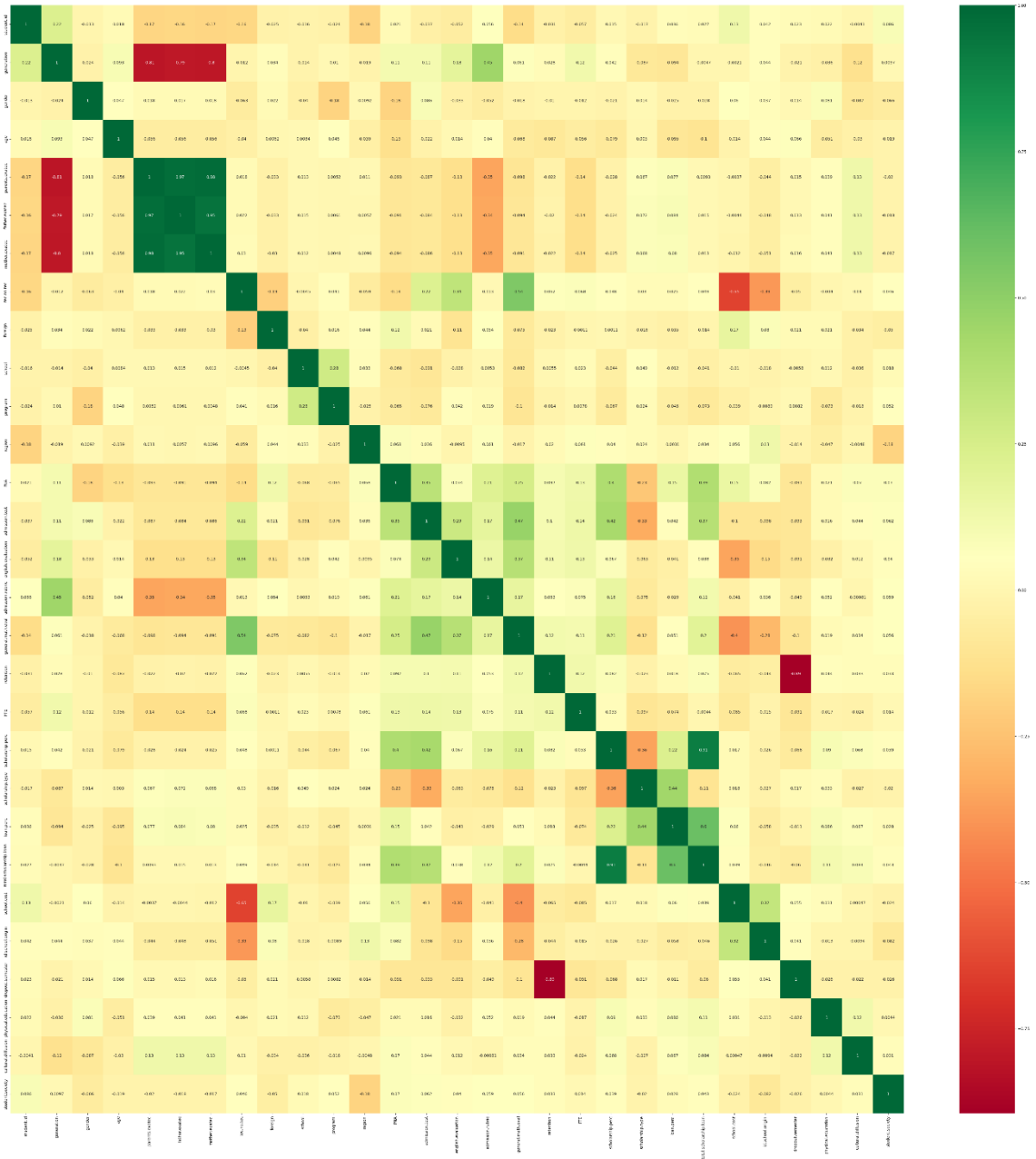
- *max.degree.parents* - 66 %

Para el resto de las variables con ausencia de datos se probarán diferentes técnicas de completamiento al final de la etapa de ingeniería de características, cuando ya se hayan ejecutado el resto de los pasos, para solo aplicarlo en las columnas seleccionadas para la versión final de este conjunto. Además de evitar que los valores generados para completar la información influyan en el resto de proceso.

- **Correlación.**

Con los análisis realizados en las etapas anteriores se eliminaron 21 atributos del total de 50. Se medirá la correlación entra las restantes 29 variables utilizando la función *corr* de la librería Pandas, los resultados se muestran en la Figura 1.

Figura 1.Matriz de correlación del Conjunto 1.



Como se puede apreciar, hay variables que presentan una correlación próxima al 90 % la cual es muy elevada. Se realizará un análisis de cada caso y la eliminación de alguna si procede.

Caso 1: *parents.exatec* tiene una correlación del 97 % con *father.exatec* y un 98 % con *mother.exatec*. Esta variable muestra si uno o ambos padres son egresados del Tecnológico de Monterrey, que es lo mismo que seleccionar un valor positivo de una de las dos variables o negativo en caso de que ambas columnas *mother.exatec* y *father.exatec* tengan valor negativo.

Se decide eliminar:

- *parents.exatec*: porque además de estar altamente correlacionada con otras dos columnas, repite el valor de una de estas, sin aportar nueva información.

Caso 2: *total.scholarship.loan* y *scholarship.perc* presentan una correlación de un 91 %, analizando la información que se proporciona en la descripción del conjunto de datos, los autores explican que: *total.scholarship.loan* es el resultado de sumar *scholarship.perc* y *loan.perc*. Esto justifica la alta correlación entre estos dos atributos.

Se decide eliminar:

- *total.scholarship.loan*: porque además de estar altamente correlacionada con *scholarship.perc*, es una característica calculable que no aporta información nueva.

Caso 3: *dropout.semester* y *retention* presentan una correlación de un 89 %, se revisaron los diferentes valores que puedes asumir ambas variables y el problema está dado porque en todos los casos en los que el alumno permanece en la carrera (*retention* = 1), la variable *dropout.semester* = 0 debido a que es el valor determinado para los casos de no abandono. Esta correlación elevada con la variable objetivo *retention*, es muy perjudicial para la clasificación porque solo con la variable *dropout.semester* ya se obtendrían altos resultados en la predicción de abandono o retención, pero estos resultados carecerían totalmente de validez.

Se decide eliminar:

- dropout.semester: porque además de estar altamente correlacionada con retention, presenta un impacto negativo para la clasificación.

- **Características de alta cardinalidad.**

Primero se contarán la cantidad de valores únicos que presentan las diez variables categóricas con mayor cardinalidad. Como se aprecia en la Tabla 5, hay varios atributos que presentan una alta cardinalidad, se analizarán los casos con más de 1000 valores únicos y se excluirán las variables pertinentes.

Tabla 5. Cantidad de valores únicos en variables categóricas del Conjunto 1.

Variable	**Valores Únicos**	**Variable**	**Valores Únicos**
student.id	52962	*school.cost*	6
id.school.origin	4208	*region*	5
program	66	*school*	5
scholarship.type	9	*Generation*	5
english.evaluation	8	*foreign*	3

student.id: presenta la mayor cardinalidad del conjunto con 52962, como era de esperar por ser el identificador del alumno y es único, pero es el atributo que permite detectar al estudiante dentro del grupo, por esto se prefiere mantenerlo.

id.school.origin: tiene una gran variedad, 4208 valores distintos que son muchos siendo una variable categórica, esto puede dificultar la capacidad de inferir conocimiento de los algoritmos. Pero antes de decidir extraer esta característica del conjunto se analizará el impacto de la variable en el conjunto de datos.

Correlación con la variable objetivo: en una etapa anterior se midió la correlación entre variables, analizando los valores de cada una con respecto a *retention*. Si se toman todos los valores como absolutos, *id.school.origin* tiene una correlación de 0.044 con *retention*, ocupando el puesto número 12 si se ordenan las 24 características en orden descendente. Esto la ubica a mediado de tabla, dejando a otras en una posición muy por detrás.

Evaluación en modelo: utilizando un modelo de clasificación se realizará la predicción con el conjunto de datos, incluyendo *id.school.origin* y excluyéndolo, para determinar si beneficia o perjudica a la clasificación.

Los resultados son:

Accuracy: 91.473 %, con *id.school.origin.*

Accuracy: 91.632, sin *id.school.origin*.

Se decide eliminar:

1. *id.school.origin:* Porque se demostró que la alta cardinalidad de esta variable afecta los resultados de la clasificación.

- **Valores faltantes, completamiento de variables.**

En un análisis anterior se eliminaron las características del conjunto que contenían más de un 50% de datos faltantes. En este momento se trabajará con aquellas que tienen una ausencia de datos inferior al 50%, para las cuales se probaran diferentes técnicas de completamiento y se probaran los conjuntos resultantes en un clasificador. Primeramente, se muestra en la Tabla 6 un listado de las variables que quedan con datos faltantes y su porcentaje.

Tabla 6. Valores faltantes del Conjunto 1.

Variable	Total de faltantes	Porcentaje (%)
admission.rubric	24376	46
father.exatec	20866	39
mother.exatec	20704	39
admission.test	18124	34
student.society	11854	22
cultural.diffusion	11854	22
physical.education	11854	22
general.math.eval	7847	15

Existen una gran variedad de técnicas para solucionar el problema de los datos faltantes, a continuación, se mencionan algunas de las más utilizadas.

- Imputación estadística.

Un enfoque simple y popular para la imputación de datos implica el uso de métodos estadísticos para estimar un valor para una columna a partir de los valores que están presentes, luego reemplazar todos los valores que faltan en la columna con la estadística calculada. Las estadísticas comunes calculadas incluyen el valor medio de la columna, el valor de la mediana de la columna y el valor de la moda de la columna [21].

Este es un método bastante simple y rápido, pero no se utilizará porque no tiene en cuenta las relaciones entre los datos, a diferencia de los dos que se enuncian a continuación.

- Imputación *K- Nearest Neighbors* (KNN).

Una técnica popular para la imputación es un modelo de KNN. Se imputa una nueva muestra encontrando las muestras en el conjunto de entrenamiento "más cercano" y se promedian estos puntos cercanos para completar el valor. La imputación KNN parece proporcionar un método más sólido y sensible para la estimación de valores faltantes [21].

Para la implementación de esta técnica se utilizará la clase *KNNImputer* que ofrece la librería *sklearn*. El parámetro fundamental de *KNNImputer* es *n_neighbors* que indica el número de vecinos que tendrá en cuenta el algoritmo para el completamiento de los datos, por defecto este valor es 5, pero resulta conveniente realizar pruebas con varios valores. Para poder evaluar el impacto de cada una de las pruebas de completamiento se implementará XGBClassifier.

Se diseñó un experimento donde se asignó a *n_neighbors* todos los valores impares entre 5 y el total de variables, sin incluir este último. Los resultados se muestran en la Tabla 7.

Tabla 7. Resultados de aplicar distintos valores de n_neighbors en KNNImputer.

n_neighbors	Accuracy	n_neighbors	Accuracy
5	91.375 %	15	91.496 %
7	91.443 %	17	91.488 %
9	91.549 %	19	91.496 %
11	91.450 %	21	91.503 %
13	91.345 %	23	91.526 %

El mayor *accuracy* obtenido fue de 91.549 % y se consiguió con tres valores distintos de *n_neighbors = 9*.

- Imputación iterativa.

La imputación iterativa se refiere a un proceso en el que cada característica se modela en función de las otras características, un problema de regresión en el que se predicen los valores faltantes. Cada característica se imputa secuencialmente, una tras otra, lo que permite que los valores imputados anteriores se utilicen como parte de un modelo para predecir características posteriores. Es iterativo porque este proceso se repite varias veces, lo que permite calcular estimaciones cada vez mejores de los valores perdidos a medida que se estiman los valores perdidos en todas las características [21].

En la implementación de este método se continúa haciendo uso de la librería *sklear,* pero esta vez con la clase *IterativeImputer*. La clase contiene varios parámetros, pero las pruebas a realizar se concentrarán en variar el número de iteraciones del algoritmo, esto se puede lograr modificando el valor de *max_iter* que por defecto es 10. De igual manera que en el caso anterior, la evaluación de los resultados se realizará con *XGBClassifier*. El experimento diseñado evalúa los resultados de los conjuntos obtenidos al realizar corridas del algoritmo variando *max_iter* entre 5 y 25.

El mejor resultado se repitió para varios valores de *max_iter,* incluyendo en valor por defecto, lográndose un accuracy de 92.612 %. Se puede apreciar que los resultados obtenidos al aplicar imputación iterativa fueron mejores que los logrados con KNN, por tanto, se define *IterativeImputer* con un *max_iter* por defecto como la técnica de imputación a aplicar en el conjunto de datos.

4.2.3 Ingeniería de instancias.

- **Agrupamiento.**

Se pueden usar diferentes métodos para agrupar los valores en *k* contenedores discretos. Las técnicas comúnmente incluyen los siguientes métodos:

- Uniforme: cada contenedor tiene el mismo ancho en el intervalo de valores posibles para la variable.
- Cuantil: cada contenedor tiene el mismo número de valores, divididos en percentiles.

- Agrupados: se identifican los clústeres y se asignan ejemplos a cada grupo [21].

Analizando las variables que aún permanecen en el conjunto, se encontró el caso de la columna *age* que por la diversidad de valores que presenta y experiencias de otros autores en trabajos similares, puede verse favorecida al aplicarle un agrupamiento y pudiera mejorar los resultados de clasificación del conjunto.

Se probarán las tres técnicas de discretización mencionadas anteriormente utilizando *KBinsDiscretizer* que se encuentra en la librería *sklearn.* Inicialmente, se mantendrán todos sus parámetros por defecto, exceptuando *strategy*, el cual se variará para poder aplicar el agrupamiento uniforme, cuantil y agrupados. Luego se realizará la clasificación con los tres conjuntos resultantes de aplicar *KBinsDiscretizer* con las tres opciones de *strategy*.

Los resultados fueron los siguientes:

Con *strategy = 'uniform'*, accuracy = 92.590 %.

Con *strategy = ' quantile'*, accuracy = 92.718 %.

Con *strategy = ' kmeans'*, accuracy = 92.771 %.

El mejor resultado se obtuvo con *strategy = 'kmeans'* logrando un 92.771 % que mejora el 92.612 % que se había obtenido en el conjunto sin aplicar la técnica de agrupamiento en la variable *age.*

A continuación, se realizan pruebas modificando otro parámetro importante como *n_bins* que especifica el número de contenedores en los que se dividirá una característica durante la discretización. Se probarán valores de *n_bins* desde 2 hasta 10, siempre con *strategy = 'kmeans'* que fue la técnica seleccionada. Los resultados se muestran en la Tabla 8.

Tabla 8. Resultados de aplicar distintos valores de n_bins en KBinsDiscretizer.

n_bins	Accuracy	n_bins	Accuracy
2	92.703 %	7	92.794 %
3	92.703 %	8	92.673 %
4	92.741 %	9	92.612 %
5	92.771 %	10	92.612 %
6	92.771 %		

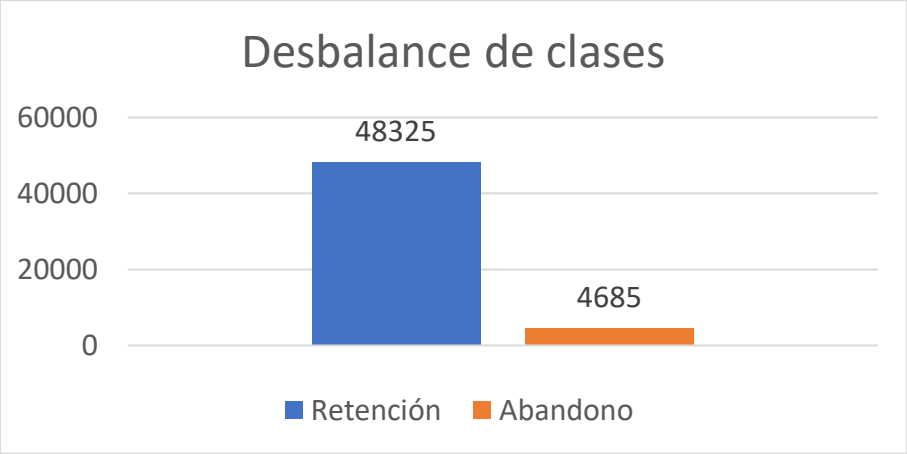
El mayor valor de accuracy fue de 92.794 % y se obtuvo con *n_bins = 7*. Por tanto, se decide aplicar *KBinsDiscretizer* para la variable *age* con *strategy = 'kmeans'* y *n_bins = 7.*

- **Desequilibrio de clases.**

En el estudio del abandono estudiantil, los conjuntos de datos en la mayoría de las ocasiones presentan desequilibrio de clases, como se pudo constatar en la

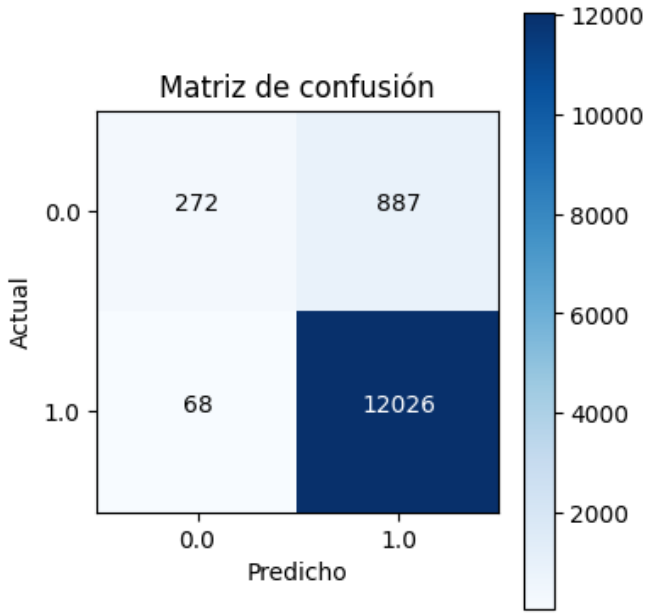
revisión bibliográfica. Este conjunto no es la excepción, la Figura 2 muestra el comportamiento del conjunto en este aspecto.

Figura 2. Representación de clases en el Conjunto 1.



Esto constituye un gran problema porque, aunque se están logrando porcientos de clasificación cercanos al 93 %, para la clase minoritaria que es la de mayor interés las tasas son muy bajas.

Figura 3. Matriz de confusión del Conjunto 1.



Como se puede apreciar en la matriz de confusión, el clasificador falla en la predicción de la gran mayoría de casos de la clase minoritaria, logrando solo un 23.46%. Esto también se puede evidenciar utilizando la métrica *Recall* que para este conjunto con clases desbalanceadas es de 61%. Existen varias vías para abordar el problema del desequilibrio de clases en conjuntos de datos. Dos de las principales técnicas son las siguientes:

Sobremuestreo: funciona volviendo a muestrear las instancias de la clase minoritaria para que el conjunto de entrenamiento resultante contenga el mismo número de instancias positivas y negativas [31].

Submuestreo: funciona al disminuir el número de instancias de la clase mayoritaria. Elimina elementos de la clase mayoritaria hasta que haya el mismo número de instancias positivas y negativas [31].

Existe una gran variedad de técnicas en Python para enfrentar este problema, a continuación, se mencionan las que serán empleadas en este trabajo, todas están incluidas en la librería *imblearn*.

RandomOverSampler: clase para realizar un sobremuestreo aleatorio. Para realizar sobremuestreo de la(s) clase(s) minoritaria(s) seleccionando muestras al azar con reemplazo [32]. Funciona duplicando aleatoriamente ejemplos de la(s) clase(s) minoritaria(s) hasta que el número de ejemplos en cada clase esté más equilibrado. Esto se hace seleccionando aleatoriamente muestras de la(s) clase(s) minoritaria(s) con reemplazo, lo que significa que algunos ejemplos pueden duplicarse, mientras que otros pueden no ser seleccionados. El proceso de sobremuestreo continúa hasta que el número de ejemplos en cada clase alcance una proporción deseada o hasta que la(s) clase(s) minoritaria(s) tengan el mismo tamaño que la clase mayoritaria.

RandomUnderSampler: clase para realizar submuestreo aleatorio. Submuestreo de la(s) clase(s) mayoritaria(s) seleccionando muestras al azar con o sin reemplazo [33]. Aborda el desequilibrio eliminando aleatoriamente ejemplos de la clase mayoritaria hasta alcanzar un nivel de equilibrio deseado. Esto se logra seleccionando de forma aleatoria los ejemplos de la clase mayoritaria hasta que el número de ejemplos de la clase mayoritaria se reduce para igualar el número de ejemplos de la clase minoritaria o alcanzar el ratio deseado.

SMOTE (Synthetic Minority Over-sampling Technique): Propuesto por Chawla, N.V. et al. La clase minoritaria se sobremuestrea creando ejemplos "sintéticos" en lugar de sobremuestrear con reemplazo [34]. El proceso de SMOTE comienza seleccionando un ejemplo de la clase minoritaria y buscando sus vecinos más cercanos en el espacio de características. Luego, se elige aleatoriamente uno de esos vecinos y se genera un nuevo ejemplo sintético en el espacio entre el ejemplo seleccionado y su vecino. Este proceso se repite para un número deseado de ejemplos sintéticos.

SMOTEENN: Sobremuestreo con *SMOTE* y limpieza con *Edited Nearest Neighbours (ENN)*. Combina el sobremuestreo y el submuestreo con *SMOTE* y *ENN* [35]. La primera etapa de SMOTEENN es aplicar SMOTE, donde se generan ejemplos sintéticos de la clase minoritaria. La segunda etapa es la aplicación de ENN. Se utiliza el algoritmo de vecinos más cercanos para evaluar cada ejemplo sintético y los ejemplos originales en el conjunto de datos. Los

ejemplos que son clasificados incorrectamente o que tienen vecinos más cercanos de la clase mayoritaria se eliminan del conjunto de datos.

TomekLinks: Submuestreo eliminando los enlaces llamados Tomek [36], apellido del investigador que los describe en 1976. La idea principal detrás de este método es que si hay un par de ejemplos de clases diferentes que son vecinos cercanos, se considera que esos ejemplos están en una región de incertidumbre o superposición entre las clases. Estos pares de ejemplos se llaman enlaces de Tomek. Finalmente, los ejemplos que forman parte de los enlaces de Tomek se eliminan del conjunto de datos.

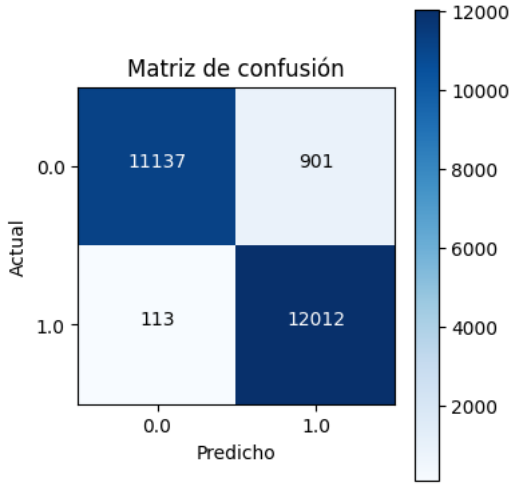
Se aplicaron las cinco técnicas anteriormente mencionadas en el conjunto de datos y posteriormente se realizó la clasificación de los conjuntos resultantes, los resultados se muestran en la Tabla 9.

Tabla 9. Comparación de técnicas para tratar el desbalance de clases en el Conjunto 1.

Técnica	Instancias	Accuracy	Recall
Sobremuestreo			
RandomOverSampler	96650	90.733 %	91 %
SMOTE	96650	95.824 %	96 %
Submuestreo			
RandomUnderSampler	9370	72.684 %	76 %
TomekLinks	50853	92.276 %	62 %
Combinado			
SMOTEENN	66776	95.357 %	96 %

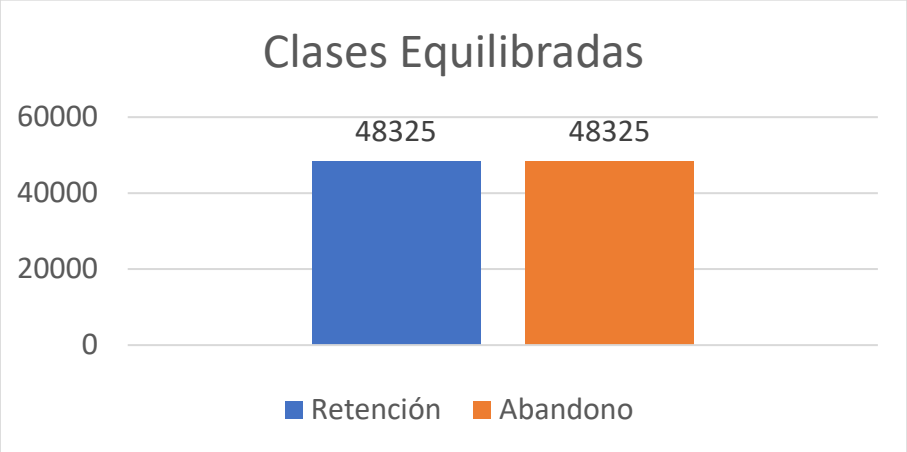
Como demuestra la comparativa anterior, los mejores resultados fueron obtenidos al aplicar SMOTE con un accuracy resultante de 95.824 %. El problema de la clasificación de la clase minoritaria se ha resuelto como evidencia el 96 % de *recall* logrado por este mismo método.

Figura 4. Matriz de confusión posterior a resolver el desbalance del Conjunto 1.



Este efecto positivo se puede apreciar también en la matriz de confusión representada en la Figura 4. Finalmente, la Figura 5 muestra como quedaron representadas las clases luego de aplicar la técnica de balanceo de clases *SMOTE* en este Conjunto 1.

Figura 5. Representación de clases después del balanceo en el Conjunto 1.



4.2.4 Procesamiento final.

- **Escalado**

Las dos técnicas más populares para escalar datos numéricos antes del modelado son la normalización y la estandarización.

Normalización: escala cada variable de entrada por separado al rango 0-1, que es el rango de valores de punto flotante donde tenemos la mayor precisión.

Estandarización: escala cada variable de entrada por separado restando la media (llamado centrado) y dividiendo por la desviación estándar para cambiar la distribución y tener una media de cero y una desviación estándar de uno [21].

Tanto la normalización como la estandarización se pueden lograr mediante la biblioteca *sklearn*. En el caso de la primera se puede ejecutar utilizando la clase *MinMaxScaler*, mientras que para la segunda se puede aplicar *StandardScaler*. A continuación, se realiza el escalamiento del conjunto de datos empleando estas dos variantes y posteriormente se evalúa el impacto de cada transformación para la clasificación. Para realizar la clasificación se empleará Support Vector Machine con sus parámetros por defecto, por ser más sensible al escalado de los datos, en su implementación se utiliza la librería sklearn.

Los resultados fueron los siguientes:

- Clasificación de conjunto sin escalar: 52.783 % de accuracy.
- Clasificación de conjunto escalado con *MinMaxScaler*: 87.915 % de accuracy.
- Clasificación de conjunto escalado con *StandardScaler*: 89.558 % de accuracy.

Como se evidencia para el caso del clasificador SVM, el escalado tiene un gran impacto positivo en los resultados. La técnica de estandarización *StandardScaler* logró el mayor accuracy que fue de 89.558 %, por tanto, es la seleccionada para aplicar al conjunto.

4.2.5 Resumen de preprocesamiento en Conjunto 1.

- En la etapa de ingeniería de características se eliminaron por diferentes razones 25 variables de las 50 iniciales.
- Durante la ingeniería de instancias el número de registros aumentó hasta 96650.
- Para favorecer el proceso de clasificación, el conjunto fue codificado, escalado y reequilibrado.

4.3 Preparación del Conjunto 2.

Como se explica en la metodología, este conjunto es para lograr una predicción temprana del abandono a lo largo de la carrera. Por tanto, al igual que en el Conjunto 1, solo debemos tener estudiantes que hayan matriculado en una fecha anterior a 4 años. Por otra parte, el objetivo de este conjunto es poder realizar la predicción en el momento de la matrícula, entonces se deben eliminar las variables que correspondan al estudiante dentro de la universidad.

Por la similitud del Conjunto 2 con el Conjunto 1, se tomará como punto de partida el conjunto anterior ya preprocesado y la única operación que realizará es la eliminación de las variables que pertenecen a un momento posterior a la matrícula.

Se eliminarán las siguientes variables:

- *cultural.diffusion.*
- *student.society.*
- *scholarship.type.*
- *loan.perc.*
- *physical.education.*

4.3.1 Resumen de preprocesamiento en Conjunto 2.

- De las 25 variables del Conjunto 1 se eliminaron 5 quedando en 20.
- Se mantienen las 96650 instancias.

4.4 Preparación del Conjunto 3.

Como se explica en la metodología, este es un conjunto para realizar una predicción del abandono en el primer año. En este caso el requisito para poder integrar el conjunto es llevar más de un año en la universidad, como la última generación del conjunto de datos original es la correspondiente al 2020, y los datos fueron actualizados hasta 2022, inicialmente todos los estudiantes pueden integrar el conjunto. Se quieren emplear todas las variables válidas del conjunto, incluyendo varios resultados del primer semestre que solamente están disponibles para las generaciones 2019 y 2020. Por esta razón, se eliminan el

resto de las generaciones, el conjunto resultante consta de 24507 instancias. La predicción se realizará una vez concluido el primer semestre para poder emplear los resultados académicos del semestre y otras variables relacionadas con la vinculación del alumno a la vida universitaria. Entonces debemos excluir también del análisis los estudiantes que abandonaron durante el primer semestre, que en el caso del total de 24507 solo fueron 257, al eliminarlos quedan 24250. Hay que realizar otro análisis con los registros de los estudiantes que abandonaron en el segundo año, como la intención es clasificar el abandono del primer año, estas instancias deben modificarse a no abandono, debido a que en el primer año no dejaron la carrera.

Como el Conjunto 3 tiene características diferentes a los dos anteriores, se repetirán las etapas del preprocesamiento, pero aplicando las técnicas que dieron los mejores resultados anteriormente.

4.4.1 Procesamiento inicial.

Se repiten los pasos ejecutados en el Conjunto 1 para esta etapa.

4.4.2 Ingeniería de características.

- **Características inútiles.**

Se eliminarán los siguientes atributos:

  - *level* - Al eliminar los estudiantes de preuniversitario del conjunto de datos, *level* asume un mismo valor para el resto de las instancias.
  - *educational.model* - Al solo tener las generaciones 2019 y 2020, *educational.model* asume un mismo valor para todas las instancias.
  - *online.test* - De las 24250 instancias que tiene este conjunto se repite el mismo valor para 23530, por lo que aporta muy poca información.

- **Valores faltantes, eliminación de variables.**

Se eliminarán los siguientes atributos:

Las siguientes tres variables tienen un 100 % de datos faltantes, esto se debe a que esta información se captura de la generación 2014 a la 2017, las cuales fueron eliminadas para este tercer conjunto.

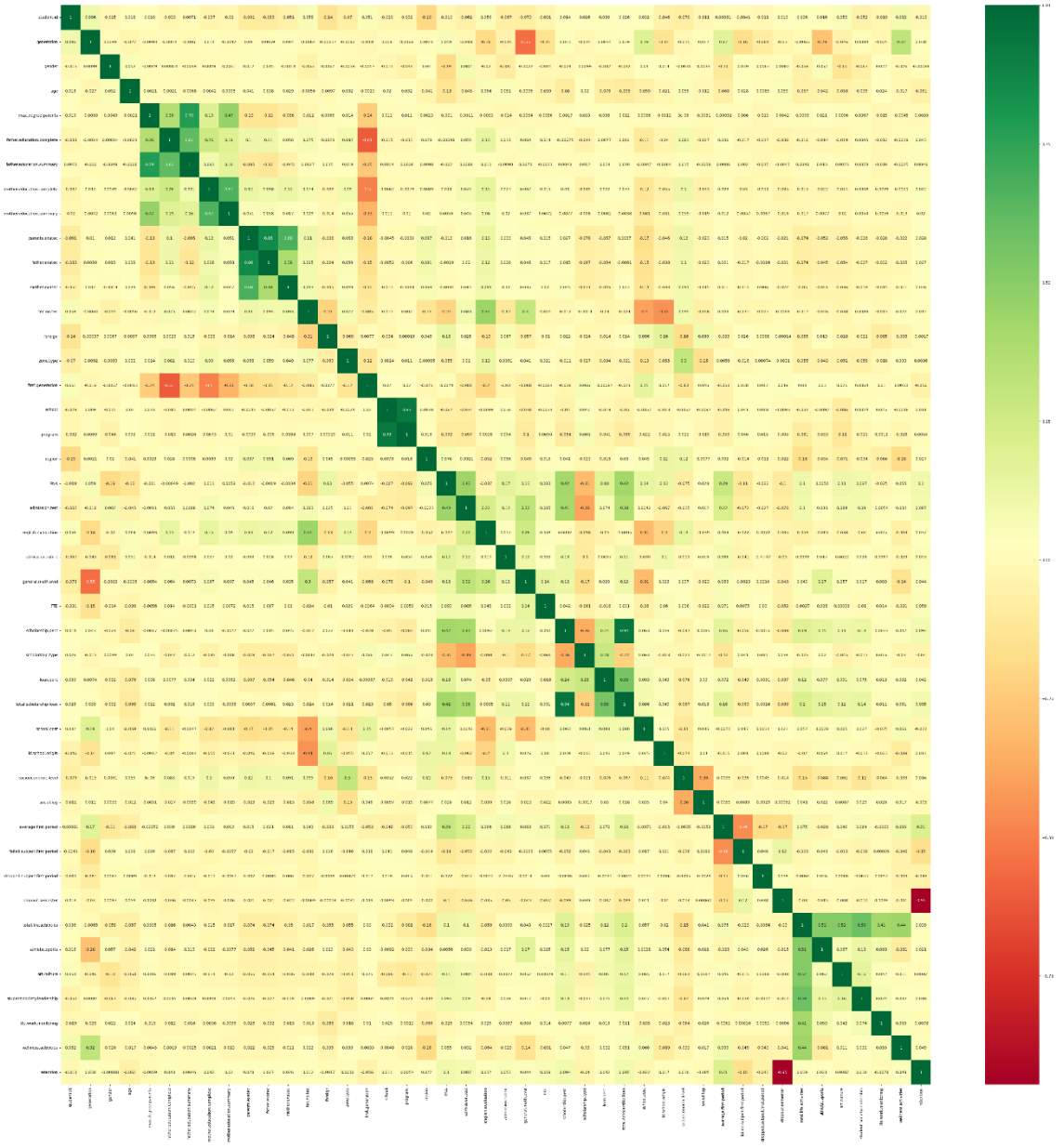
  - *cultural.diffusion*– 100 %.
  - *physical.education*– 100 %.
  - *student.society*- 100 %.

La variable que le sigue en porcentaje de datos faltantes es *socioeconomic.level* que presenta un 55 %, valor que se encuentra un poco por encima del umbral que se definió para eliminar una columna por datos faltantes en el análisis inicial. Pero como no se excede por mucho, se decide mantener en el conjunto. El resto de las variables presentan porcentajes menores y serán completadas posteriormente.

- **Correlación.**

Se mide la correlación entra las restantes 44 variables en la Figura 6.4

Figura 6. Matriz de correlación del Conjunto 2.



Hay variables que presentan una correlación muy elevada. Se realizará un análisis de cada caso y la eliminación de alguna si procede.

Caso 1: *parents.exatec* tiene una correlación del 86 % con *father.exatec* y un 68 % con *mother.exatec*.

Se decide eliminar:

- *parents.exatec*: por el análisis realizado para el Conjunto 1.

Caso 2: *program* y *school* presentan una correlación de un 93 %, se revisaron los diferentes valores que puedes asumir ambas variables y el problema está dado porque la variable *program* nombra el programa académico al que pertenece el estudiante y *school* nombra la escuela a la que pertenece el

programa académico del estudiante. Y particularmente para las generaciones que contiene este conjunto, estas dos columnas tienen una estrecha relación.

Se decide eliminar:

- *program*: porque además de estar altamente correlacionada con *school*, presenta una cardinalidad alta.

Caso 3: *total.scholarship.loan* y *scholarship.perc* presentan una correlación de un 94 %.

Se decide eliminar:

- *total.scholarship.loan*: por el análisis realizado para el Conjunto 1.

Caso 4: *dropout.semester* y *retention* presentan una correlación de un 95 %.

Se decide eliminar:

- *dropout.semester*: por el análisis realizado para el Conjunto 1.

- **Características de alta cardinalidad.**

Se realiza un conteo de los valores únicos que presenta las diez variables con mayor cardinalidad de este conjunto, los resultados se muestran en la Tabla 10.

Tabla 10. Cantidad de valores únicos del Conjunto 3.

Variable	# Únicos	Variable	# Únicos
student.id	24244	*father.education.complete*	8
id.school.origin	4248	*socioeconomic.level*	7
scholarship.type	10	*english.evaluation*	7
dropped.subject.first.period	10	*failed.subject.first.period*	6
mother.education.complete	8	*school*	6

Se decide eliminar:

- *id.school.origin:* Por el análisis realizado para el Conjunto 1.

- **Valores faltantes, completamiento de variables.**

En la Tabla 11 se presenta un listado de las variables que quedan con datos faltantes y su porcentaje. Para resolverlo se aplica IterativeImputer con un max_iter por defecto como la técnica de imputación, la cual fue elegida anteriormente.

4.4.3 Ingeniería de instancias

- **Agrupamiento.**

Se aplica *KBinsDiscretizer* para la variable *age* con *strategy = 'kmeans'* y *n_bins = 7.*

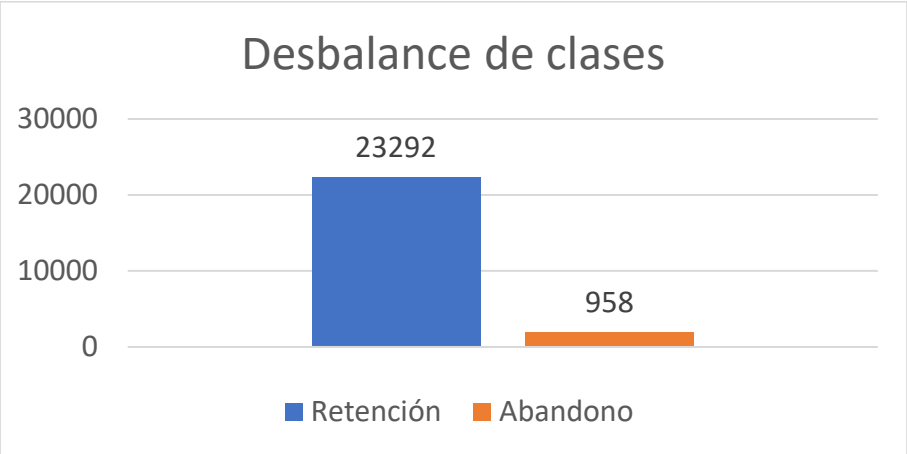
- **Desequilibrio de clases.**

Al igual que en el conjunto anterior, en este existe un gran desequilibrio de clases como muestra la Figura 7.

Tabla 11. Valores faltantes en el Conjunto 3.

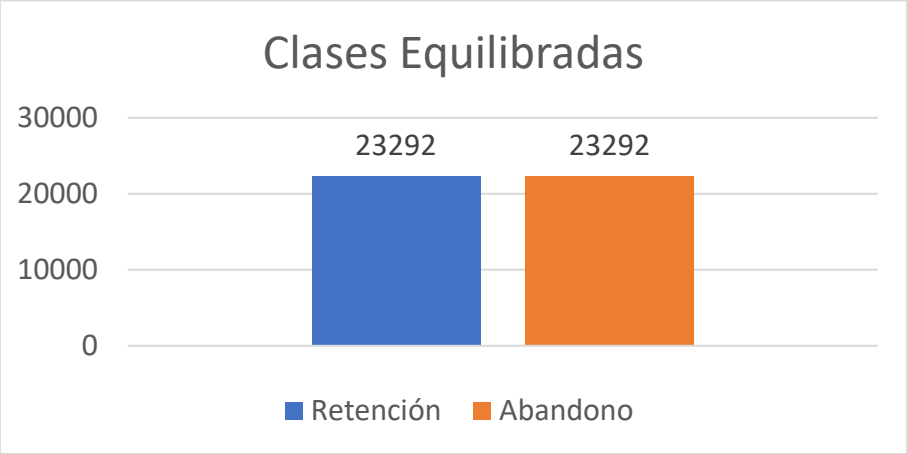
Variable	Total de faltantes	Porcentaje (%)
socioeconomic.level	13297	55
social.lag	10855	45
admission.test	7055	29
admission.rubric	5441	22
zone.type	1695	7
mother.education.complete	974	4
mother.education.summary	974	4
father.education.complete	745	3
father.education.summary	745	3
father.exatec	254	1
mother.exatec	83	0.3
max.degree.parents	54	0.2
first.generation	54	0.2
athletic.sports	30	0.1
total.life.activities	30	0.1
art.culture	30	0.1
student.society.leadership	30	0.1
life.work.mentoring	30	0.1
wellness.activities	30	0.1

Figura 7. Representación de clases en el Conjunto 3.



La Figura 8 muestra como quedó el conjunto luego de aplicar la técnica de balanceo de clases *SMOTE*.

Figura 8. Representación de clases luego del balanceo en el Conjunto 3.



Después de esta etapa el conjunto queda con 46584 instancias.

4.4.4 Procesamiento final.

- **Escalado**

Como último paso se aplica StandardScaler al Conjunto 3.

4.4.5 Resumen de preprocesamiento en Conjunto 3.

- En la etapa de ingeniería de características se eliminaron por diferentes razones 11 variables, quedando con 39 finalmente.
- Durante la ingeniería de instancias el número de registros aumentó hasta 46584.
- Como en los casos anteriores, el conjunto fue codificado, escalado y reequilibrado.

4.5 Preparación del Conjunto 4.

Como se explica en la metodología, este es un conjunto para realizar una predicción del abandono en el primer año, pero solo con los datos registrados hazte el momento de la matrícula. Al igual que en el anterior, el requisito para poder integrar este conjunto es llevar más de un año en la universidad e inicialmente todos los estudiantes están incluidos. Como no se empleará ninguna información del estudiante dentro de la universidad, datos que solo se registran en los últimos años, de momento no se elimina ninguna generación como si se hizo en el Conjunto 3, se parte de un total de 77517 instancias. La predicción se realizará en un momento posterior a la matrícula, antes del ingreso a la universidad. Los registros de los estudiantes que abandonaron en el segundo año deben modificarse a no abandono, debido a que en el primer año no dejaron la carrera.

Se repetirán las etapas del preprocesamiento, aplicando las técnicas que dieron los mejores resultados anteriormente. Pero primero se eliminarán las variables que aportan información del alumno dentro de la universidad, que son un total de 16.

Se eliminarán los siguientes atributos:

- *scholarship.perc.*
- *scholarship.type*.
- *loan.perc*.
- *total.scholarship.loan*.
- *average.first.period*.
- *failed.subject.first.period*.
- *dropped.subject.first.period*.
- *physical.education*.
- *cultural.diffusion*.
- *student.society*.
- *total.life.activities*.
- *athletic.sports*.
- *art.culture*.
- *student.society.leadership.*
- *life.work.mentoring*.
- *wellness.activities*.

4.5.1 Procesamiento inicial.

Se repite los pasos ejecutados en el conjunto 1 para esta etapa.

4.5.2 Ingeniería de características.

- **Características inútiles.**

Se eliminarán los siguientes atributos:

- *level* - Al eliminar los estudiantes de preuniversitario del conjunto de datos, *level* asume un mismo valor para el resto de las instancias.
- *online.test* - De las 77517 instancias que tiene este conjunto se repite el mismo valor para 76776, por lo que aporta muy poca información.

- **Valores faltantes, eliminación de variables.**

Se eliminarán los siguientes atributos:

Las siguientes tres variables tienen más de un 70 % de datos faltantes.

- *socioeconomic.level* – 86 %.
- *social.lag* – 83 %.
- *zone.type* – 71 %.

- **Correlación.**

Se mide la correlación entra las restantes 29 variables, la matriz se muestra en la Figura 9. Hay variables que presentan una correlación muy elevada. Se realizará un análisis de cada caso y la eliminación de alguna si procede.

Caso 1: *parents.exatec* tiene una correlación del 86 % con *father.exatec* y un 67 % con *mother.exatec*.

Se decide eliminar:

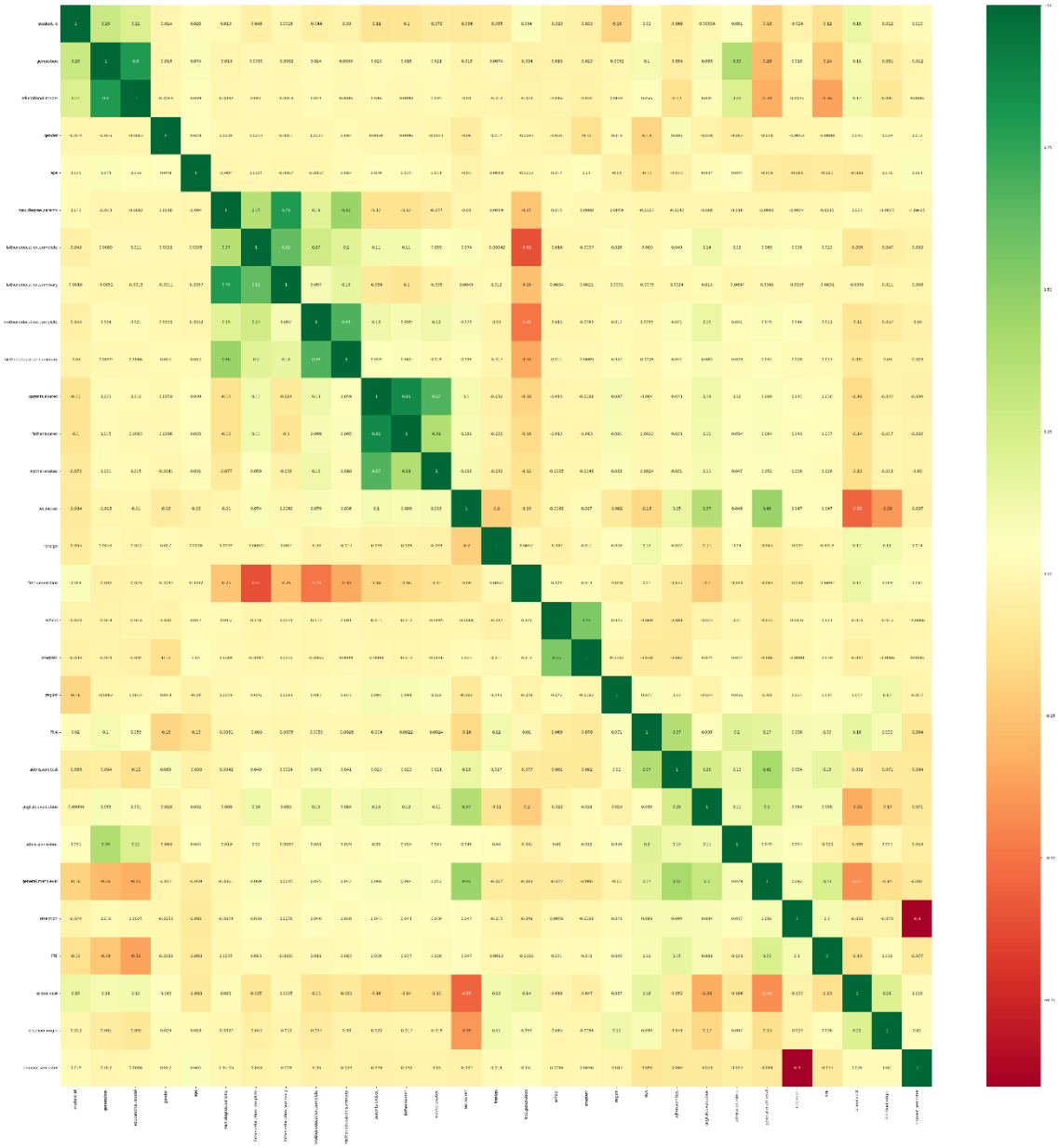
- *parents.exatec*: por el análisis realizado para el Conjunto 1.

Caso 2: *dropout.semester* y *retention* presentan una correlación de un 90 %.

Se decide eliminar:

- *dropout.semester*: por el análisis realizado para el Conjunto 1.

Figura 9. Matriz de correlación del Conjunto 4.



- **Características de alta cardinalidad.**

Se realiza un conteo de los valores únicos que presenta las diez variables con mayor cardinalidad de este conjunto, los resultados se muestran en la Tabla 12.

Tabla 12. Cantidad de valores únicos del Conjunto 4.

Variable	# Únicos	Variable	# Únicos
student.id	77437	*father.education.complete*	8
id.school.origin	6534	*Generation*	7
Program	72	*School*	6
english.evaluation	8	*school.cost*	5
mother.education.complete	8	*region*	5

Se decide eliminar:

- *id.school.origin:* Por el análisis realizado para el conjunto 1.

- **Valores faltantes, completamiento de variables.**

La Tabla 13 muestra un listado de las variables que quedan con datos faltantes y su porcentaje.

Tabla 13. Valores faltantes en el Conjunto 4.

Variable	Total de faltantes	Porcentaje (%)
first.generation	37372	48
mother.education.complete	37282	48
mother.education.summary	37282	48
father.education.summary	36742	47
father.education.complete	36742	47
max.degree.parents	35255	45
admission.rubric	29851	39
admission.test	25255	33
father.exatec	21123	27
mother.exatec	20789	27
general.math.eval	7847	10
school.cost	5687	7

4.5.3 Ingeniería de instancias

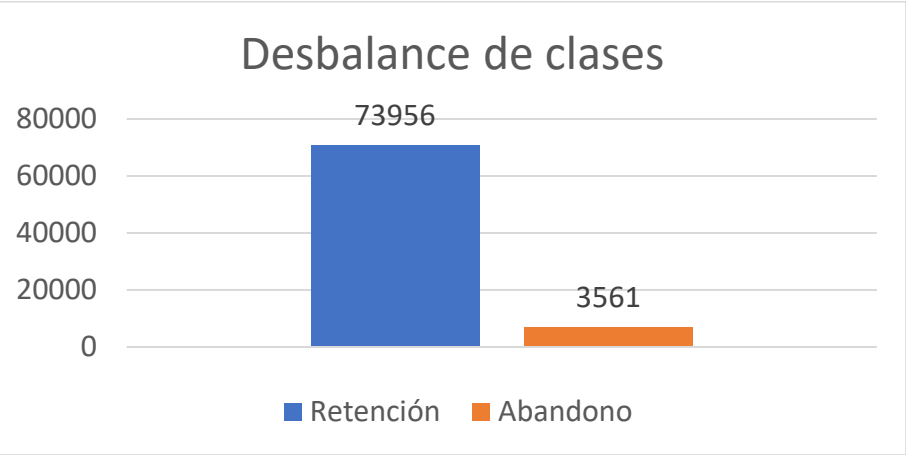
- **Agrupamiento.**

Se aplica *KBinsDiscretizer* para la variable *age.*

- **Desequilibrio de clases.**

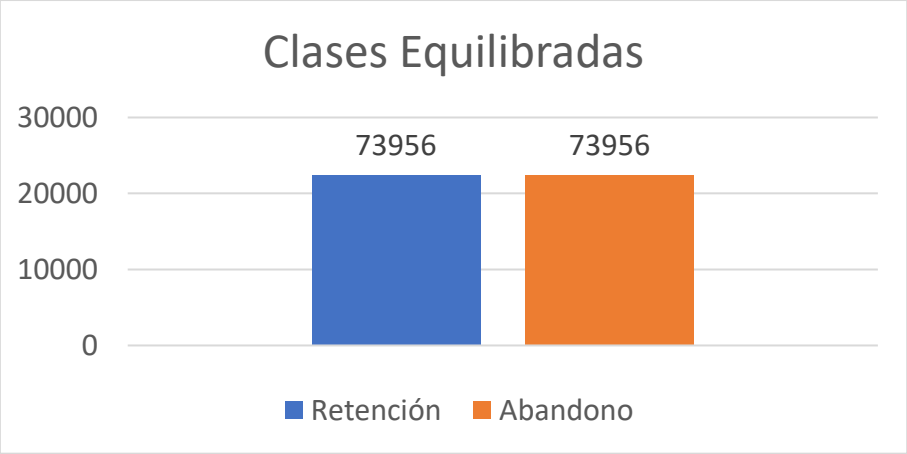
En este conjunto también existe un gran desequilibrio de clases como muestra la Figura 10.

Figura 10. Representación de clases en el Conjunto 4.



Luego de aplicar la técnica de balanceo de clases *SMOTE* en conjunto, quedó representado como muestra la Figura 11.

Figura 11. Representación de clases luego del balanceo en el Conjunto 4.



Después de esta etapa el conjunto queda con 147912 instancias.

4.5.4 Procesamiento final.

- **Escalado**

Como último paso se aplica *StandardScaler* al Conjunto 4.

4.5.5 Resumen de preprocesamiento en Conjunto 4.

- En la etapa de ingeniería de características se eliminaron por diferentes razones 23 variables, quedando con 27 finalmente.
- Durante la ingeniería de instancias el número de registros aumentó hasta 147912.
- Como en los casos anteriores, el conjunto fue codificado, escalado y reequilibrado.

4.6 Implementación de Clasificadores.

Como se describe en la metodología, se implementarán cinco clasificadores y un modelo integrado, a continuación, se describe las herramientas que se emplearán para la implementación de cada uno de los algoritmos.

Ramdon Forest se implementará utilizando la clase *RandomForestClassifier* disponible en la librería *sklearn.* En esta misma librería se encuentra la clase *SVC* que es la elegida para el trabajo con *Support Vector Machine*. Distinto es el caso de *XGBoost* donde se utilizará la clase *XGBClassifier* de la librería *xgboost*. Para la *Deep Neural Network* se empleará la librería *Keras.* En el caso del modelo *ResNet* se trabajará de forma diferente porque las librerías que la implementan están enfocadas al trabajo con imágenes, se aplicará una propuesta de red que imita el comportamiento de *ResNet* y fue concebida para el trabajo con datos tabulares por [37] en el sitio web *Kaggle*.

4.6.1 División estratificada de los conjuntos.

Para la división de cada conjunto en entrenamiento y prueba se utilizará una proporción 75% - 25%, que está dentro del rango que recomiendan la mayoría de los autores. Para realizar esta división se diseñó una partición estratificada que consta de los siguientes pasos:

Paso 1: se realiza un análisis de la importancia de cada una de las variables del conjunto que se vaya a particiona. Para esto se utilizará la importancia basada en árboles, esta se calcula al medir cuánto contribuye cada variable a la reducción de la impureza (como la ganancia de Gini o la ganancia de información) en los nodos de los árboles. Las variables que más contribuyen a la reducción de la impureza se consideran más importantes.

Paso 2: una vez que las variables se encuentren organizadas por su importancia de la más a la menos, se tomarán las 5 variables más importantes y se identifican los distintos valores que puede asumir cada una.

Paso 3: se realiza una partición inicial utilizando *train_test_split* perteneciente a la librería *sklearn*. Una vez realizado este procedimiento se calcula la proporción en porciento que representa la cantidad de cada valor único de las variables más importante con respecto al total de instancias, esto se ejecutará para el conjunto de entrenamiento y el de prueba resultantes de la partición.

Paso 4:

a) si los valores únicos de las variables más importantes se encuentran proporcionalmente repartidos. Se mantienen la división realizada en el paso 3.

b) si los valores únicos de las variables más importantes no se encuentran proporcionalmente repartidos. Empezando por la variable más importante de las cinco, se reparte proporcionalmente entre entrenamiento y prueba las instancias de cada valor distinto que pueda asumir la variable. Este proceso se repite para las restantes 4 variables en orden de importancia, manteniendo la proporción de las variables ya ubicadas.

4.6.1.1 Implementación de la división estratificada.

Conjunto 1: se ejecutaron los pasos 1, 2 y 3 de la estrategia planteada con los cuales se obtuvieron las 5 variables identificadas como más importantes para este conjunto, los valores más representativos que puede tomar cada una de las variables y además el porcentaje que representan estos valores en el conjunto de prueba y el conjunto de entrenamiento. Los resultados obtenidos se muestran en la Tabla 14.

Como se demuestra para el Conjunto 1, en los conjuntos de entrenamiento y prueba los valores más representativos de las variables más importantes se encuentran proporcionalmente repartidos después de aplicar el paso 3, por tanto, se mantiene esta partición como indica el paso 4 a).

Tabla 14. Resultados de la aplicación de la división estratificada en el Conjunto 1.

Variable	Valor	X_train	X_test
physical.education	-1.486455724217135	21.40 %	21.27 %
	1.0232588135211829	36.53 %	36.71 %
gender	0.9533037967935296	45.59 %	45.48 %
	-1.2260169992458114	32.98 %	33.04 %
english.evaluation	0.5990151250033899	47.04 %	46.77 %
	1.16533612895379	6.03 %	6.03 %
cultural.diffusion	-1.047677840839518	31.60 %	31.76 %
	1.381935842633231	26.77 %	26.60 %
scholarship.type	0.0227687768343917	43.99 %	44.16 %
	0.9884659403108952	21.58 %	21.50 %

Conjunto 2: se ejecutaron los pasos 1, 2 y 3 de la estrategia y se analizan los resultados que se muestran en la Tabla 15.

Tabla 15. Resultados de la aplicación de la división estratificada en el Conjunto 2.

Variable	Valor	X_train	X_test
gender	0.9533037967935296	45.59 %	45.48 %
	-1.2260169992458114	32.98 %	33.04 %
school	1.4596128936326829	20.70 %	19.91 %
	-0.2850369090918992	33.64 %	34.17 %
english.evaluation	0.5990151250033899	47.04 %	46.77 %
	1.16533612895379	6.03 %	6.03 %
FTE	0.1893104259068211	60.69 %	60.72 %
	1.3101812142649123	9.94 %	9.89 %
school.cost	-0.787989874982546	50.90 %	50.82 %
	0.424663211759172	8.11 %	8.24 %

Como se demuestra, el Conjunto 2 muestra el mismo comportamiento que el anterior, por tanto, se mantiene esta partición como indica el paso 4 a).

Conjunto 3: se ejecutaron los pasos 1, 2 y 3 de la estrategia y se analizan los resultados que se muestran en la Tabla 16. Como se demuestra, el Conjunto 3 muestra el mismo comportamiento que los anteriores, por tanto, se mantiene esta partición como indica el paso 4a).

Conjunto 4: se ejecutaron los pasos 1, 2 y 3 de la estrategia y se analizan los resultados que se muestran en la Tabla 17. Como se demuestra, el Conjunto 4 muestra el mismo comportamiento que los anteriores, por tanto, se mantiene esta partición como indica el paso 4a). Así concluye la partición de los conjuntos en entrenamiento y prueba, quedando listos para su posterior uso.

Tabla 16. Resultados de la aplicación de la división estratificada en el Conjunto 3.

Variable	Valor	X_train	X_test
max.degree.parents	0.9533786899068616	41.09 %	41.76 %
	-1.546213684960993	17.45 %	17.19 %
mother.education.summary	0.994532773836934	38.86 %	39.17 %
	-0.9144009231947134	21.21 %	20.65 %
gender	1.025836198818541	41.25 %	41.38 %
	-1.1589695007856335	35.21 %	35.33 %
average.first.period	0.657234347119095	2.02 %	1.78 %
	0.4858188561799288	1.91 %	1.65 %
father.education.summary	1.0116487837157186	37.44 %	38.06 %
	-0.7405340697255566	14.80 %	14.71 %

Tabla 17. Resultados de la aplicación de la división estratificada en el Conjunto 4.

Variable	Valor	X_train	X_test
gender	0.9934889252774424	42.93 %	42.70 %
	-1.1939660965287109	33.75 %	33.84 %
mother.education.summary	1.3662207768468704	19.69 %	19.47 %
	-1.2079643977946293	11.30 %	11.50 %
school.cost	-0.8694540459214085	45.08 %	44.92 %
	0.6629849214142396	8.38 %	8.25 %
school	-0.2686266163868226	31.90 %	32.21 %
	1.4322834380177374	19.87 %	20.00 %
father.education.complete	1.177520543269303	19.17 %	19.50 %
	-0.0875002838191877	7.08 %	6.91 %

4.6.2 Validación cruzada.

En todas las pruebas se utilizará la validación cruzada para evaluar los modelos de manera robusta y evitar el sobreajuste. Se decidió trabajar con un *K = 5.* El conjunto de datos se divide en 5 pliegues y se realizan 5 iteraciones, utilizando un pliegue diferente como conjunto de prueba en cada iteración.

4.6.3 Clasificación inicial.

Inicialmente, se clasificarán los cuatro conjuntos conformados en la sección anterior con cada uno de los clasificadores. En todos los casos los algoritmos se emplearán con sus datos por defecto. Para la DNN se iniciará con una capa oculta y una de salida, la capa de entrada tendrá 32 neuronas y la capa oculta 16, la función de activación de la capa oculta será *relu* y en la de salida *sigmoid.* En el caso de *ResNet* se tomará como configuración inicial la obtenida de

Kaggle. Los resultados en cuanto a accuracy de los clasificadores para cada conjunto se muestra en la Tabla 18.

Tabla 18. Clasificación de cada conjunto con los hiperparámetros por defecto de cada clasificador.

	RF	**SVM**	**XGBoost**	**DNN**	**ResNet**
Conjunto 1	95.849 %	89.591 %	95.998 %	85.804 %	77.166 %
Conjunto 2	94.959 %	4.145 %	94.673 %	79.261 %	76.431 %
Conjunto 3	97.887 %	95.998 %	97.879 %	84.475 %	85.766 %
Conjunto 4	98.204 %	92.022 %	98.247 %	83.043 %	80.648 %

4.6.4 Optimización de hiperparámetros.

Existen varios métodos para la optimización de los hiperparámetros de los algoritmos de clasificación, en este caso se decidió utilizar la Búsqueda en cuadrícula (Grid Search): Esta técnica implica definir una cuadrícula de posibles combinaciones de valores de hiperparámetros y evaluar el rendimiento del modelo para cada combinación utilizando validación cruzada. Se selecciona la combinación de hiperparámetros que proporciona el mejor rendimiento. Para su implementación se utilizará la clase *GridSearchCV* perteneciente a la librería *sklearn.*

4.6.4.1 Optimización de hiperparámetros de Random Forest.

RandomForestClassifier que fue la herramienta seleccionada para la implementación de RF presenta un alto número de hiperparámetros, tratar de ajustarlos todos sería un proceso extremadamente largo y muy costoso computacionalmente. Por esto se han elegido cuatro hiperparámetros de los más significativos para la clasificación, con los cuales se realizarán los ajustes pertinentes.

Según un análisis comparativo realizado en [38] donde se evalúa el impacto de ajustar diferentes parámetros de RF en una gran muestra de 38 conjuntos datos, se demuestra que *Random Forest* es un algoritmo en el que la optimización no impacta mucho, logrando en general una mejora de un 0.006. Los hiperparámetros más importantes a tener en cuenta según el estudio comparativo fueron:

- *mtry*: controla el número máximo de características o variables predictoras que se consideran en cada árbol de decisión dentro del bosque aleatorio.
- *sample.fraction*: controla la proporción de muestras que se seleccionan al azar del conjunto de datos de entrenamiento para construir cada árbol dentro del bosque aleatorio.
- *replace*: indica si se permite el muestreo con reemplazo durante la construcción de cada árbol en el bosque aleatorio.
- *num.trees*: especifica el número de árboles que se construirán en el bosque aleatorio.

- *min.node.size*: especifica el tamaño mínimo que debe tener un nodo terminal (hoja) en un árbol de decisión dentro del bosque aleatorio.

Se identificaron los hiperparámetros equivalentes a los mencionados anteriormente para la herramienta *RandomForestClassifier.* Para *mtry es max_features,* el de *num.trees* es *n_estimators,* en el caso de *min.node.size* su par es *min_samples_leaf.* No se encontró un equivalente específico para *sample.fraction* y *replace*, pero se identificó que con *bootstrap* se puede obtener un efecto similar. Por tanto, los cuatro seleccionados son:

- *n_estimators:* valor por defecto 100.
- *max_features:* valor por defecto *sqrt.*
- *bootstrap:* valor por defecto *True.*
- *min_samples_leaf:* valor por defecto 1.

En el experimento diseñado se incluye siempre el valor por defecto de cada hiperparámetro debido a lo bien que funcionan normalmente, y se agregan otros que también se evaluarán. Los valores seleccionados se muestran a continuación:

- *n_estimators* [100, 150, 200, 250, 300, 350, 400]. En general, un valor más alto mejora la precisión del modelo.
- *max_features* [*"sqrt", "log2", None,* 3, 7, 9]. Aunque se colocan enteros, como el número exacto de características que se deben considerar en cada división. Decidimos también mantener *"sqrt"* que es el valor por defecto y devuelve raíz cuadrada del número total de características. Se probará también *"log2"* logaritmo en base 2 del total de características y *None* donde se consideran todas las características en cada división.
- *bootstrap:* [*True, False*]. Al ser un parámetro booleano solo admite estos dos valores.
- *min_samples_leaf* [1, 2, 3]. Valores más grandes generan árboles más pequeños y evitan un sobreajuste excesivo, por este motivo además de 1 que es el valor por defecto se probará con 2 y 3.

En las tablas 19, 20, 21 y 22 se muestran las cinco mejores combinaciones de hiperparámetros obtenidas para cada uno de los conjuntos.

Tabla 19. Las 5 mejores combinaciones de hiperparámetros para RF del Conjunto 1.

Orden	***bootstrap***	***max_features***	***min_samples_leaf***	***n_estimators***
1	False	7	1	100
2	False	7	1	150
3	False	7	1	400
4	False	7	1	200
5	False	7	1	350

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 95.998 %.

Tabla 20. Las 5 mejores combinaciones de hiperparámetros para RF del Conjunto 2.

Orden	*bootstrap*	*max_features*	*min_samples_leaf*	*n_estimators*
1	False	log2	1	300
2	False	Sqrt	1	300
3	False	Sqrt	1	400
4	False	log2	1	250
5	False	log2	1	400

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 95.195 %.

Para el Conjunto 3 se incluye en *max_features* el valor 11.

Tabla 21. Las 5 mejores combinaciones de hiperparámetros para RF del Conjunto 3.

Orden	*bootstrap*	*max_features*	*min_samples_leaf*	*n_estimators*
1	False	log2	1	200
2	False	5	1	300
3	False	9	1	300
4	False	log2	1	400
5	False	Sqrt	1	400

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 97.947.

Tabla 22. Las 5 mejores combinaciones de hiperparámetros para RF del Conjunto 4.

Orden	*bootstrap*	*max_features*	*min_samples_leaf*	*n_estimators*
1	False	9	1	400
2	False	9	1	250
3	False	9	1	200
4	False	9	1	350
5	False	9	1	300

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 98.263.

4.6.4.2 Optimización de hiperparámetros de Support Vector Machine.

SVC de la librería *sklearn* fue la herramienta seleccionada para la implementación, presenta también un alto número de hiperparámetros. Por esto se han elegido algunos de los hiperparámetros de los más significativos para la clasificación, con los cuales se realizarán los ajustes pertinentes.

Se continúa empleando el análisis comparativo realizado en [38], en este caso, su evaluación del impacto de ajustar diferentes parámetros de SVM, para este

algoritmo la optimización logra una mejora de un 0.042. Los hiperparámetros de mayor importantes para tener en cuenta según el estudio comparativo fueron:

- *cost*: parámetro de regularización que controla el equilibrio entre la capacidad del modelo para clasificar correctamente los datos de entrenamiento y la simplicidad del modelo.
- *kernel*: tipo de función de *kernel* que se utiliza para transformar los datos de entrada en un espacio de características de mayor dimensión.
- *gamma*: utilizado en algunas variantes de *kernel* para controlar la forma de la función de base radial.
- *degree*: es utilizado con el *kernel* polinómico. Especifica el grado del polinomio que se utiliza para calcular la función de decisión en SVM. No se utilizará en las pruebas por ser un parámetro empleado solo para un tipo de *kernel.*

Se identificaron los hiperparámetros equivalentes a los mencionados anteriormente para la herramienta *SVM.*

- *C:* valor por defecto 1.
- *kernel:* valor por defecto *rbf.*
- *gamma:* valor por defecto *scale.*

Los valores seleccionados se muestran a continuación:

- *C* [1, 5, 10]. Un valor más alto de C dará un margen más estrecho y un modelo más ajustado a los datos de entrenamiento, mientras que un valor más bajo permitirá un margen más amplio y un modelo más suave.
- *kernel* ['linear', 'rbf', 'sigmoid']. Son varios de los *kernel* que posee la clase que más recomiendan en la web.
- *gamma* ['scale', 'auto', 0.5]: Un valor bajo significa que los puntos lejanos tienen una influencia alta y el hiperplano de decisión será más suave, mientras que un valor alto significa que los puntos cercanos tienen una influencia alta y el hiperplano de decisión será más ajustado a los datos de entrenamiento.

En las tablas 23, 24, 25 y 26 se muestran las cinco mejores combinaciones de hiperparámetros obtenidas para cada uno de los conjuntos.

Tabla 23. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 1.

Orden	***C***	***gamma***	***kernel***
1	10	0.5	*rbf*
2	5	0.5	*rbf*
3	1	0.5	*rbf*
4	10	*scale*	*rbf*
5	10	*auto*	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 96.498 %.

Tabla 24. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 2.

Orden	***C***	***gamma***	***kernel***
1	5	0.5	*rbf*
2	10	0.5	*rbf*
3	1	0.5	*rbf*
4	10	*scale*	*rbf*
5	10	*auto*	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 95.671 %.

Tabla 25. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 3.

Orden	***C***	***gamma***	***kernel***
1	10	*auto*	*rbf*
2	10	*scale*	*rbf*
3	5	*scale*	*rbf*
4	5	*auto*	*rbf*
5	5	0.5	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 98.213 %.

Tabla 26. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 4.

Orden	***C***	***gamma***	***Kernel***
1	5	0.5	*rbf*
2	10	0.5	*rbf*
3	1	0.5	*rbf*
4	10	*scale*	*rbf*
5	10	*auto*	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 98.213 %.

Como se puede apreciar, en todos los casos los mejores resultados se obtuvieron empleando *kernel = rbf.* En los casos de los mejores valores resultantes para *C* y *gamma,* variaron en cada conjunto. Se diseñó un segundo experimento donde se decidió mantener fijo el *kernel = rbf,* que fue la mejor opción en todos los casos. Para los valores de *C* y *gamma* se mantendrán los valores con los que se lograron los tres mejores resultados en cada conjunto y se agregarán otros, para tratar de mejorar aún más la clasificación.

Los nuevos valores para evaluar serán:

- *C* [15].
- *gamma* [0.01, 0.1, 0.3, 0.7].

En las tablas 27, 28, 29 y 30 se muestran las cinco mejores combinaciones de hiperparámetros obtenidas para cada uno de los conjuntos en el segundo experimento.

Tabla 27. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 1, experimento 2.

Orden	***C***	***gamma***	***kernel***
1	5	0.3	*rbf*
2	15	0.3	*rbf*
3	10	0.3	*rbf*
4	1	0.3	*rbf*
5	10	0.5	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 97.657 %.

Tabla 28. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 2, experimento 2.

Orden	***C***	***gamma***	***kernel***
1	15	0.5	*rbf*
2	5	0.5	*rbf*
3	10	0.5	*rbf*
4	10	0.3	*rbf*
5	10	0.7	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 95.770 %.

Tabla 29. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 3, experimento 2.

Orden	***C***	***gamma***	***kernel***
1	5	0.1	*rbf*
2	15	0.1	*rbf*
3	10	0.1	*rbf*
4	15	*scale*	*rbf*
5	15	*auto*	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 99.493 %.

Tabla 30. Las 5 mejores combinaciones de hiperparámetros para SVM del Conjunto 4, experimento 2.

Orden	*C*	*gamma*	*kernel*
1	5	0.3	*rbf*
2	10	0.3	*rbf*
3	15	0.3	*rbf*
4	1	0.3	*rbf*
5	15	0.1	*rbf*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 98.718 %.

4.6.4.3 Optimización de hiperparámetros de XGBoost.

XGBClassifier de la librería *xgboost* fue la herramienta seleccionada para la implementación, presenta también un alto número de hiperparámetros. Por esto se han elegido los hiperparámetros más significativos para la clasificación, con los cuales se realizarán los ajustes pertinentes.

Se continúa empleando el análisis comparativo realizado en [38], en este caso, su evaluación del impacto de ajustar diferentes parámetros de *XGBoost*, para este algoritmo la optimización logra una mejora de un 0.014. Los hiperparámetros más importantes para tener en cuenta según el estudio comparativo fueron:

- *booster:* determina el tipo de modelo de refuerzo que se utilizará durante el entrenamiento.
- *eta:* controla la contribución de cada árbol al modelo final. Determina la magnitud con la que los pesos de los árboles se actualizan en cada iteración durante el entrenamiento.
- nrounds: especifica la cantidad de árboles de decisión que se deben construir durante el entrenamiento.
- *subsample:* controla la proporción de muestras utilizadas para entrenar cada árbol de decisión, determina la fracción de los datos de entrenamiento que se utiliza en cada ronda de construcción del árbol.
- min_child_weight: es un parámetro de regularización que especifica el peso mínimo necesario para que un nodo hijo sea considerado en la construcción de un árbol.

Se identificaron los hiperparámetros equivalentes a los mencionados anteriormente para la herramienta *XGBClassifier.*

- *booster:* valor por defecto *gbtree*.
- *learning_rate:* valor por defecto 0.1.
- *n_estimators*: valor por defecto 100.
- *subsample:* valor por defecto 1.
- *min_child_weight:* valor por defecto 1.

Los valores seleccionados se muestran a continuación:

- *booster* [gbtree, gblinear, dart]. Son todas las opciones que tiene esta clase.
- *learning_rate* [0.05, 0.1, 0.5]. Un valor más alto permite que el modelo aprenda más rápidamente, por otro lado, un valor más bajo hace que el modelo aprenda más lentamente.
- *n_estimators* [100, 200, 300]. Un valor más alto puede resultar en un modelo más complejo y potencialmente de mejor rendimiento, pero también puede aumentar el tiempo de entrenamiento.
- *subsample* [0.6, 0.8, 1]. El uso de un valor menor a 1 se conoce como "*subsampling*" o "*bagging*", y puede ayudar a reducir el sobreajuste al introducir variabilidad en cada árbol y promediar sus predicciones.
- *min_child_weight* [1, 2, 3]. Un valor más alto implica que se requiere una mayor cantidad de instancias en un nodo hoja para que se realice una división adicional. Esto puede ayudar a prevenir la creación de particiones con muy pocos ejemplos.

En las tablas 31, 32, 33 y 34 se muestran las cinco mejores combinaciones de hiperparámetros obtenidas para cada uno de los conjuntos.

Tabla 31. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 1.

Orden	***booster***	***learning_rate***	***min_child_weight***	***n_estimators***	***subsample***
1	*gbtree*	0.5	3	300	1
2	*dart*	0.5	3	300	1
3	*gbtree*	0.5	2	300	1
4	*dart*	0.5	2	300	1
5	*gbtree*	0.5	3	200	1

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 96.370 %.

Tabla 32. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 2.

Orden	***booster***	***learning_rate***	***min_child_weight***	***n_estimators***	***subsample***
1	*gbtree*	0.1	3	300	1
2	*dart*	0.1	3	300	1
3	*gbtree*	0.1	1	300	1
4	*dart*	0.1	1	300	1
5	*gbtree*	0.1	2	300	1

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 94.702 %.

Tabla 33. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 3.

Orden	*booster*	*learning_rate*	*min_child_weight*	*n_estimators*	*subsample*
1	*gbtree*	0.05	3	300	1
2	*dart*	0.05	3	300	1
3	*gbtree*	0.1	1	200	1
4	*dart*	0.1	1	200	1
5	*gbtree*	0.1	2	300	1

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 97.853 %.

Tabla 34. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 4.

Orden	*booster*	*learning_rate*	*min_child_weight*	*n_estimators*	*subsample*
1	*gbtree*	0.5	2	300	1
2	*dart*	0.5	2	300	1
3	*gbtree*	0.5	3	300	1
4	*dart*	0.5	3	300	1
5	*gbtree*	0.5	1	300	1

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 98.298 %.

En busca de mejorar aún más los resultados obtenidos y observando que en algunos casos los mejores resultados se lograron con valores que se encuentran en los límites planteados en el experimento inicial. Se decidió realizar una segunda etapa de experimentos para este clasificador en la cual se mantendrán los valores con los que se lograron los tres mejores resultados y además se incluirán otros que amplíen los límites establecidos inicialmente teniendo en cuenta el caso particular de cada conjunto. En esta segunda etapa de experimentos se excluye del análisis el hiperparámetro *subsample,* ya que para todos los conjuntos su valor óptimo fue 1, que es el máximo que puede tomar.

En las tablas 35, 36, 37 y 38 se muestran las cinco mejores combinaciones de hiperparámetros obtenidas para cada uno de los conjuntos en el segundo experimento.

Conjunto 1. Los valores seleccionados se muestran a continuación:

- *booster* [gbtree, dart].
- *learning_rate* [0.4, 0.5, 0.6].
- *n_estimators* [300, 400, 500].
- *min_child_weight* [2, 3, 4].

Tabla 35. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 1, experimento 2.

Orden	*booster*	*learning_rate*	*min_child_weight*	*n_estimators*
1	*gbtree*	0.3	4	400
2	*dart*	0.3	4	400
3	*gbtree*	0.3	3	500
4	*dart*	0.3	3	500
5	*gbtree*	0.3	4	500

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 96.473 %.

Conjunto 2. Los valores seleccionados se muestran a continuación:

- *booster* [gbtree, dart].
- *learning_rate* [0.09, 0.1, 0.2].
- *n_estimators* [300, 400, 500].
- *min_child_weight* [1, 3, 4].

Tabla 36. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 2, experimento 2.

Orden	*booster*	*learning_rate*	*min_child_weight*	*n_estimators*
1	*gbtree*	0.2	1	300
2	*dart*	0.2	1	300
3	*gbtree*	0.1	4	500
4	*dart*	0.1	4	500
5	*gbtree*	0.2	1	400

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 94.839 %.

Conjunto 3. Los valores seleccionados se muestran a continuación:

- *booster* [gbtree, dart].
- *learning_rate* [0.05, 0.07, 0.1].
- *n_estimators* [200, 300, 400].
- *min_child_weight* [1, 4, 5].

Tabla 37. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 3, experimento 2.

Orden	*booster*	*learning_rate*	*min_child_weight*	*n_estimators*
1	*gbtree*	0.07	1	300
2	*dart*	0.07	1	300
3	*gbtree*	0.05	1	400
4	*dart*	0.05	1	400

5	*gbtree*	0.07	1	400

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 97.887 %.

Conjunto 4. Los valores seleccionados se muestran a continuación:

- *booster* [gbtree, dart].
- *learning_rate* [0.3, 0.5, 0.7].
- *n_estimators* [300, 400, 500].
- *min_child_weight* [2, 3, 4].

Tabla 38. Las 5 mejores combinaciones de hiperparámetros para XGBoost del Conjunto 4, experimento 2.

Orden	***booster***	***learning_rate***	***min_child_weight***	***n_estimators***
1	*gbtree*	0.3	2	400
2	*dart*	0.3	2	400
3	*gbtree*	0.3	3	500
4	*dart*	0.3	3	500
5	*gbtree*	0.3	4	400

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 98.390 %.

En todos los casos los resultados de esta segunda etapa superaron a los primeros experimentos realizados, por tanto, se decide mantener la mejor combinación de hiperparámetros obtenidos para cada conjunto.

4.6.4.4 Optimización de hiperparámetros de DNN.

En la implementación de la red neuronal se empleará la librería *Keras*, que cuenta con varias herramientas que facilitan la creación del modelo. Se han elegido los hiperparámetros más significativos para la clasificación, con los cuales se realizarán los ajustes pertinentes.

Los hiperparámetros con los que se estará trabajando son:

- *capas ocultas:* son capas intermedias entre la capa de entrada y la capa de salida de una red.
- *neuronas:* son los bloques de construcción básicos de una red neuronal.
- *activation:* es una función matemática que se aplica a la salida de una neurona.
- *optimizer:* algoritmo utilizado para ajustar los pesos y los sesgos de una red neuronal.
- *batch_size:* se refiere al número de muestras de datos que se utilizan en cada paso de entrenamiento.
- *epochs:* se refiere a una pasada completa por todo el conjunto de datos de entrenamiento durante el proceso de entrenamiento.

Valores por defecto de los hiperparámetros que tienen esta característica en la librería *Keras.*

- *activation:* valor por defecto *relu*.
- *optimizar:* valor por defecto *adam.*
- *batch_size:* valor por defecto 32.
- *epochs:* valor por defecto 10.

Las pruebas con todos estos hiperparámetros son computacionalmente costosas y tardadas en tiempo, por esta razón inicialmente se mantendrán algunos fijos y se probarán las distintas combinaciones de otros. Los valores seleccionados para esta primera etapa son:

- capas ocultas [1, 2, 3]. Una mayor cantidad de capas ocultas proporciona una mayor capacidad para aprender características complejas, pero también aumenta complejidad y el tiempo de entrenamiento de la red.
- *neuronas* [16, 32, 64, 128]. Problemas más complejos pueden requerir más neuronas en las capas ocultas. Por otro lado, problemas más simples pueden requerir menos neuronas.
- *activation* [*relu, tanh, sigmoid*]. Se probarán la mayor parte de las funciones de activación disponibles en *Keras*.

En las tablas 39, 40, 41, 42, 43, 44, 45 y 46 se muestran las mejores combinaciones de hiperparámetros obtenidas para el Conjunto 1 en los 8 experimentos realizados.

Tabla 39. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 1.

Orden	***activation***	***hidden_layers***	***hidden_units***
1	*relu*	3	128
2	*relu*	2	128
3	*relu*	3	64
4	*relu*	2	64
5	*tanh*	3	128

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 84.913 %.

Se mantendrán los valores obtenidos en las 3 mejores combinaciones y se probarán otros.

- capas ocultas [2, 3, 4].
- *neuronas* [64, 128, 256, 512].
- *activation* [*relu, selu, elu*].

Tabla 40. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 1, experimento 2.

Orden	***activation***	***hidden_layers***	***hidden_units***

1	*relu*	3	256
2	*relu*	4	512
3	*relu*	2	512
4	*relu*	3	512
5	*relu*	4	256

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 84.943 %.

Seguidamente, se vuelven a mantener los valores obtenidos en las 3 mejores combinaciones en los casos de las capas ocultas y neuronas. Y se probarán las opciones restantes de funciones de activación.

- capas ocultas [2, 3, 4].
- *neuronas* [256, 512].
- *activation* [*relu, softplus, softsign, exponential*].

Tabla 41. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 1, experimento 3.

Orden	***activation***	***hidden_layers***	***hidden_units***
1	*relu*	4	512
2	*relu*	3	256
3	*relu*	3	512
4	*relu*	2	512
5	*relu*	4	256

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 85.647 %.

Seguidamente, se vuelven a mantener los valores obtenidos en las 3 mejores combinaciones. Y se probarán funciones de activación en la capa de salida.

- capas ocultas [2, 3, 4].
- *neuronas* [256, 512].
- *activation* [*relu*].
- *output_activation* [*sigmoid, softmax*]. Se probarán dos de las funciones más empleadas en la web para problemas de clasificación binaria.

Tabla 42. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 1, experimento 4.

Orden	***activation***	***hidden_layers***	***hidden_units***	***output_activation***
1	*relu*	3	256	*sigmoid*
2	*relu*	3	512	*sigmoid*
3	*relu*	4	256	*sigmoid*
4	*relu*	4	512	*sigmoid*

5	*relu*	2	512	*sigmoid*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros: 85.788 %.

Se continúa con el mismo procedimiento y ahora se agregará el parámetro *optimizer* a las pruebas.

- capas ocultas [3, 4].
- *neuronas* [256, 512].
- *activation* [relu].
- *output_activation* [*sigmoid, softmax*].
- *optimizer* [*adam, adagrad, adamax, adafactor, nadam, ftrl, sgd, rmsprop, adamW, adadelta*]. Se probarán todas las opciones disponibles para la librería *Keras*.

Tabla 43. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 1, experimento 5.

Orden	***activation***	***hidden_layers***	***hidden_units***	***output_activation***	***optimizer***
1	*relu*	3	512	*sigmoid*	*nadam*
2	*relu*	4	512	*sigmoid*	*adam*
3	*relu*	4	512	*sigmoid*	*nadam*
4	*relu*	3	512	*sigmoid*	*adam*
5	*relu*	3	256	*sigmoid*	*nadam*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 86.483 %.

Se continúa con el mismo procedimiento y ahora se agregará el parámetro *batch_size* a las pruebas.

- capas ocultas [3, 4].
- *neuronas* [256, 512].
- *activation* [relu].
- output_activation [sigmoid].
- optimizer [adam, nadam].
- batch_size [16, 32, 64]. Un tamaño de lote más grande puede ayudar a obtener estimaciones de gradiente más estables y precisas. También puede haber casos en los que un tamaño de lote más pequeño resulte beneficioso.

Tabla 44. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 1, experimento 6.

Orden	**activation**	**hidden layers**	**hidden units**	**output activation**	**optimizer**	**batch_size**
1	*relu*	3	512	*sigmoid*	*nadam*	16

2	*relu*	3	512	*sigmoid*	*nadam*	32
3	*relu*	3	512	*sigmoid*	*nadam*	64
4	*relu*	4	512	*sigmoid*	*adam*	16
5	*relu*	3	256	*sigmoid*	*nadam*	16

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 88.325 %.

Después de los experimentos realizados se establecen los siguientes hiperparámetros para el Conjunto 1:

- *capas ocultas* [3].
- *activation* [relu].
- *optimizer* [nadam].
- *batch_size* [16].
- *output_activation* [sigmoid].

En el caso del número de neuronas los mejores resultados llegaron al utilizar los valores 256 y 512. Por tanto, se probarán todas las combinaciones existentes en la red neuronal resultante compuesta por la capa de entrada, tres capas ocultas y la capa de salida.

Tabla 45. Las 5 mejores combinaciones de capas para la DNN en el Conjunto 1, experimento 7.

Orden	**Capa 1**	**Capa 2**	**Capa 3**	**Capa 4**
1	512	512	512	256
2	512	256	256	512
3	512	256	256	512
4	512	512	256	256
5	512	512	256	512

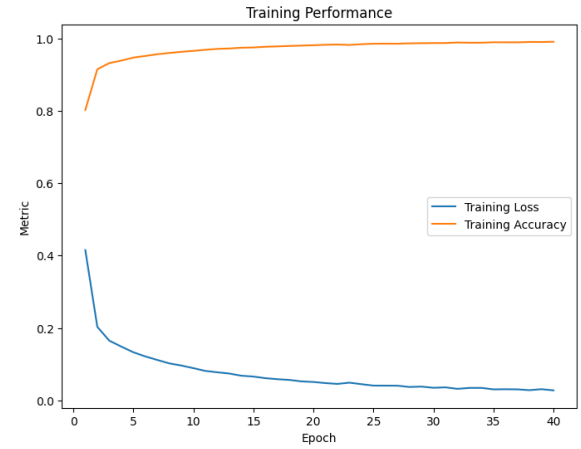
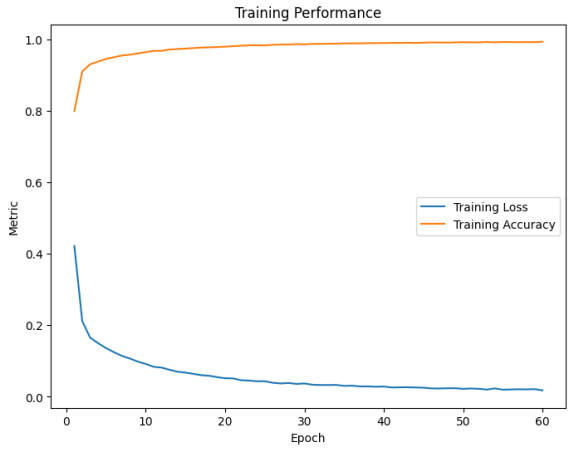
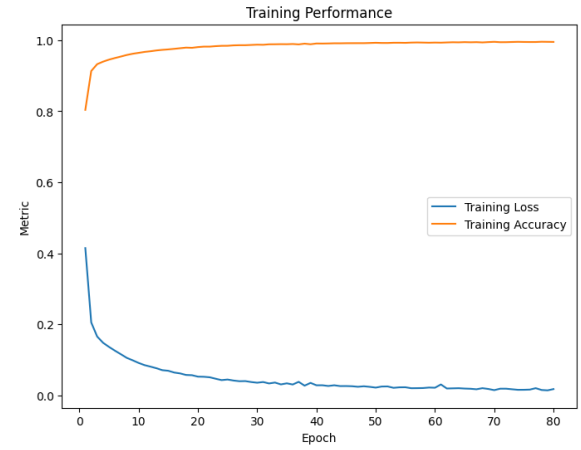
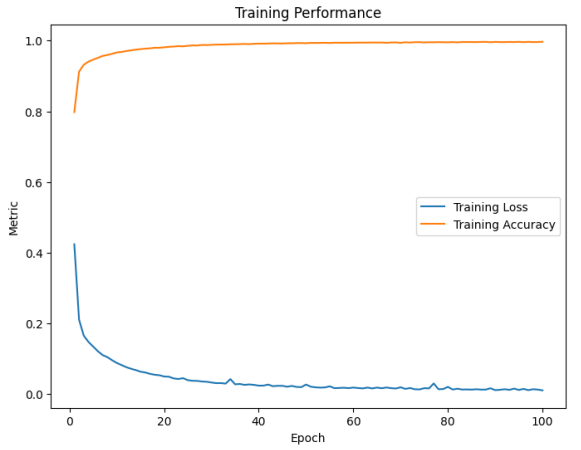
Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de neuronas: 89.541 %.

Para terminar el ajuste de la red para el Conjunto 1 se probará el entrenamiento con los siguientes valores del hiperparámetro *epochs*.

- *epochs* [40, 60, 80, 100]. Al aumentar el número de épocas, la red neuronal tiene la oportunidad de ajustar los pesos y mejorar gradualmente su rendimiento. Sin embargo, es importante tener en cuenta que un número excesivo de épocas puede llevar al sobreajuste.

Tabla 46. Resultados de diferentes valores de epochs para la DNN en el Conjunto 1, experimento 8.

***epochs* = 40**	***epochs* = 60**

	
accuracy = 96.464 %	*accuracy* = 96.876 %
***epochs* = 80**	***epochs* = 100**
	
accuracy = 97.075 %	*accuracy* = 96.895 %

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 80 épocas. Así termina el ajuste de hiperparámetros de la red neuronal para el Conjunto 1.

- *capas ocultas* [3].
- *activation* [*relu*].
- *optimizer* [*nadam*].
- *batch_size* [16].
- *output_activation* [*sigmoid*].
- *neuronas* [512 - 512 – 512 - 256].
- *epochs* [80].

En el ajuste de la red para el resto de los conjuntos se definirán algunos hiperparámetros desde el inicio, con base en la experiencia obtenida en el trabajo con el Conjunto 1. Los valores que se mantendrán fijos serán:

- *activation* [*relu*].
- *output_activation* [sigmoid].

Con el resto de hiperparámetros se probarán los siguientes valores:

- *capas ocultas* [2, 3, 4].
- *optimizer* [*nadam, adam*].
- *neuronas* [128, 256, 512].

En las tablas 47, 48, y 49 se muestran las mejores combinaciones de hiperparámetros obtenidas para el Conjunto 2 en los 3 experimentos realizados.

Tabla 47. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 2.

Orden	***hidden_layers***	***hidden_units***	***optimizer***
1	3	256	*adam*
2	2	512	*adam*
3	4	512	*adam*
4	4	512	*nadam*
5	2	512	*nadam*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 77.680 %.

Se fijan otros dos hiperparámetros:

- *capas ocultas* [3].
- *optimizer* [adam].

Y con esa estructura de tres capas ocultas, al igual que en el Conjunto 1, se probarán todas las combinaciones posibles de neuronas con los dos valores que mejores resultaron 256 y 512.

Tabla 48. Las 5 mejores combinaciones de capas para la DNN en el Conjunto 2, experimento 2.

Orden	**Capa 1**	**Capa 2**	**Capa 3**	**Capa 4**
1	256	256	512	512
2	512	512	512	256
3	256	512	512	512
4	256	512	256	256
5	512	256	256	256

Se elige la mejor configuración obtenida que para este conjunto fue:

- *neuronas* [256 - 256 – 512 - 512].

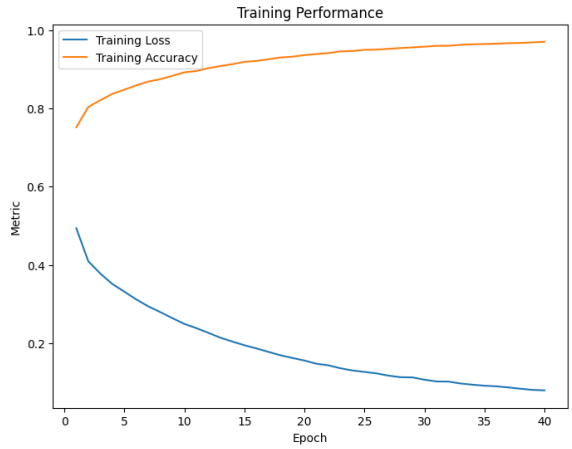
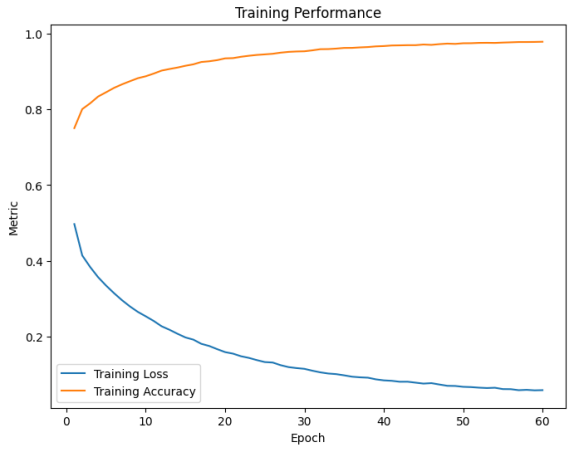
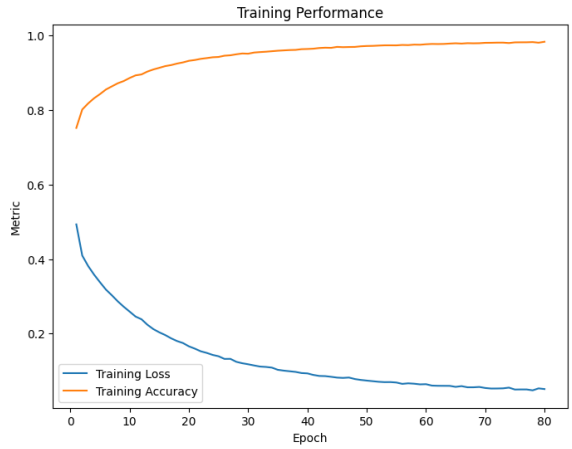
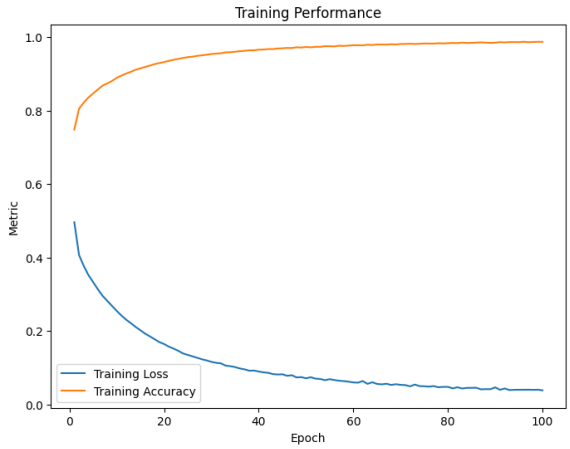
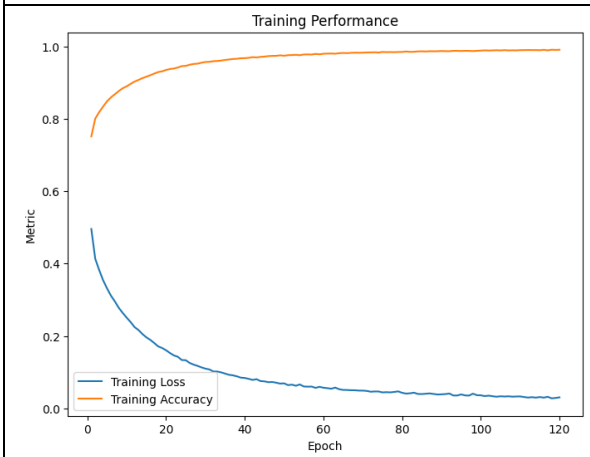
A continuación, se prueban varias opciones de configuración del batch_size:

- *batch_size* [16, 32, 64].

Los mejores resultados se lograron con un *batch_size = 16.*

Tabla 49. Resultados de diferentes valores de epochs para la DNN en el Conjunto 2, experimento 3.

***epochs* = 40**	***epochs* = 60**

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>accuracy = 90.169 %</td><td>accuracy = 90.687 %</td></tr>
<tr><td>epochs = 80</td><td>epochs = 100</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>accuracy = 91.248 %</td><td>accuracy = 91.479 %</td></tr>
<tr><td colspan="2">epochs = 120</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">accuracy = 91.242 %</td></tr>
</table>

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 100 épocas. Así termina el ajuste de hiperparámetros de la red neuronal para el Conjunto 2.

- *capas ocultas* [3].

- *activation* [*relu*].
- *optimizer* [*adam*].
- *batch_size* [16].
- *output_activation* [*sigmoid*].
- *neuronas* [256 - 256 – 512 - 512].
- *epochs* [100].

Conjunto 3, se probarán los siguientes valores:

- *capas ocultas* [2, 3, 4].
- *optimizer* [*nadam, adam*].
- *neuronas* [128, 256, 512].

En las tablas 50, 51 y 52 se muestran las mejores combinaciones de hiperparámetros obtenidas para el Conjunto 3 en los 3 experimentos realizados.

Tabla 50. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 3.

Orden	***hidden_layers***	***hidden_units***	***optimizer***
1	4	256	*nadam*
2	4	256	*adam*
3	4	512	*nadam*
4	3	256	*adam*
5	3	256	*nadam*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 91.911 %.

Se fijan otros dos hiperparámetros:

- *capas ocultas* [4].
- *optimizer* [nadam].

Y con esa estructura de cuatro capas ocultas, se probarán todas las combinaciones posibles de neuronas con los dos valores que mejores resultaron 256 y 512.

Tabla 51. Las 5 mejores combinaciones de capas para la DNN en el Conjunto 3, experimento 2.

Orden	**Capa 1**	**Capa 2**	**Capa 3**	**Capa 4**	**Capa 5**
1	256	512	512	512	256
2	256	256	512	512	512
3	256	256	512	256	512
4	256	256	256	256	256
5	256	512	256	512	256

Se elige la mejor configuración obtenida que para este conjunto fue:

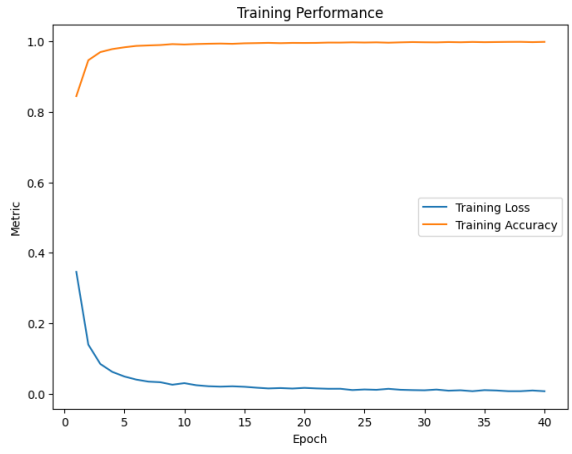
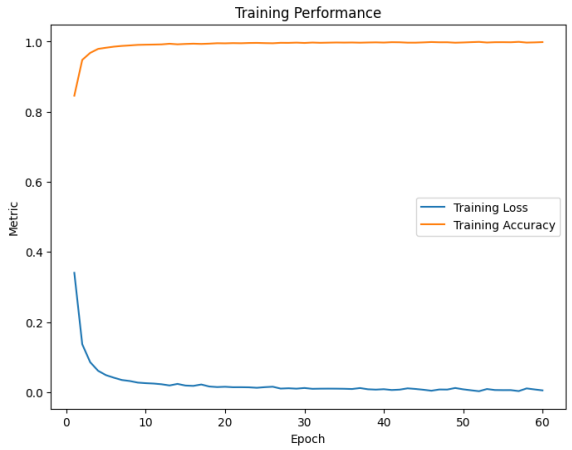
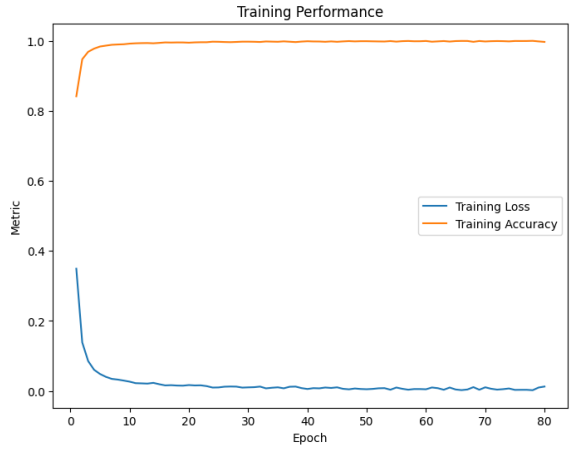
- *neuronas* [256 - 512 - 512 - 512 - 256].

A continuación, se prueban varias opciones de configuración del *batch_size*:

- *batch_size* [16, 32, 64].

Los mejores resultados se lograron con un *batch_size = 64.*

Tabla 52. Resultados de diferentes valores de epochs para la DNN en el Conjunto 3, experimento 3.

***epochs* = 40**	***epochs* = 60**
	
accuracy = 97.930 %	*accuracy* = 98.128 %
***epochs* = 80**	
	
accuracy = 97.781 %	

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 60 épocas. Así termina el ajuste de hiperparámetros de la red neuronal para el Conjunto 3.

- *capas ocultas* [4].
- *activation* [*relu*].
- *optimizer* [*nadam*].
- *batch_size* [64].
- *output_activation* [*sigmoid*].
- *neuronas* [256 - 512 – 512 – 512 - 256].
- *epochs* [60].

Conjunto 3, se probarán los siguientes valores:

- *capas ocultas* [2, 3, 4].
- *optimizer* [*nadam, adam*].
- *neuronas* [128, 256, 512].

En las tablas 53, 54 y 55 se muestran las mejores combinaciones de hiperparámetros obtenidas para el Conjunto 4 en los 3 experimentos realizados.

Tabla 53. Las 5 mejores combinaciones de hiperparámetros para DNN del Conjunto 4.

Orden	*hidden_layers*	*hidden_units*	*optimizer*
1	4	512	*nadam*
2	4	256	*nadam*
3	3	512	*nadam*
4	4	256	*adam*
5	4	512	*adam*

Resultados del *accuracy* alcanzado al realizar la clasificación con la mejor combinación de hiperparámetros obtenida: 90.451 %.

Se fijan otros dos hiperparámetros:

- *capas ocultas* [4].
- *optimizer* [*nadam*].

Y con esa estructura de cuatro capas ocultas, se probarán todas las combinaciones posibles de neuronas con los dos valores que mejores resultaron 256 y 512.

Tabla 54. Las 5 mejores combinaciones de capas para la DNN en el Conjunto 4, experimento 2.

Orden	Capa 1	Capa 2	Capa 3	Capa 4	Capa 5
1	512	512	512	256	512
2	512	512	512	512	256
3	512	512	256	256	512
4	512	256	256	256	512
5	256	512	256	512	256

Se elige la mejor configuración obtenida que para este conjunto fue:

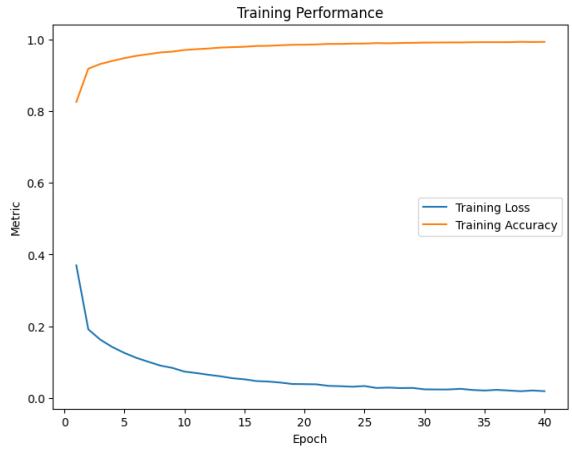
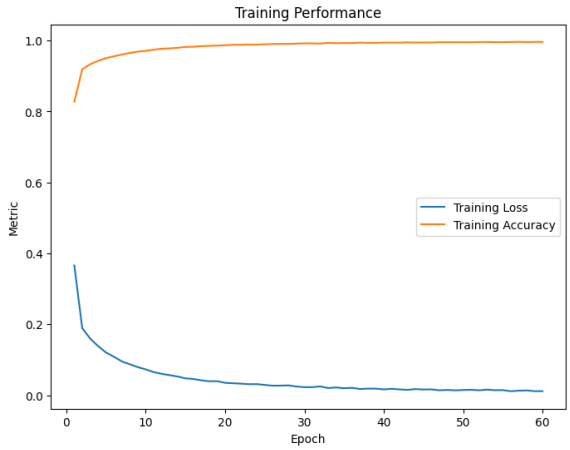
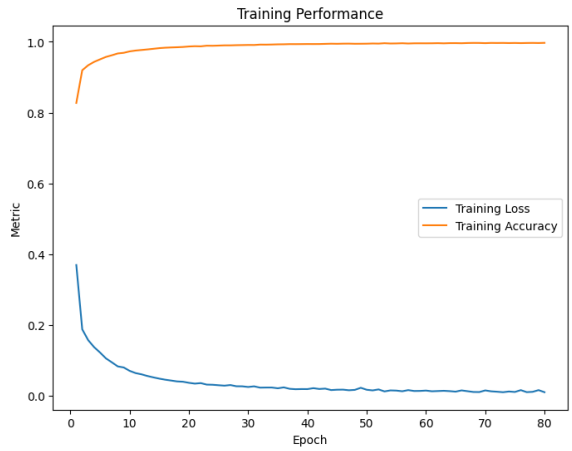
- *neuronas* [512 – 512 – 512 – 256 – 512].

A continuación, se prueban varias opciones de configuración del batch_size:

- *batch_size* [16, 32, 64].

Los mejores resultados se lograron con un *batch_size = 32.*

Tabla 55. Resultados de diferentes valores de epochs para la DNN en el Conjunto 4, experimento 3.

***epochs* = 40**	***epochs* = 60**
	
accuracy = 97.710 %	*accuracy* = 97.748 %
***epochs* = 80**	
	
accuracy = 97.848 %	

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 80 épocas. Así termina el ajuste de hiperparámetros de la red neuronal para el Conjunto 4.

- *capas ocultas* [4].
- *activation* [*relu*].
- *optimizer* [*nadam*].
- *batch_size* [32].
- *output_activation* [*sigmoid*].
- *neuronas* [512 – 512 – 512 – 256 – 512].
- *epochs* [80].

4.6.4.5 Optimización de hiperparámetros de ResNet.

Este es un caso particular porque como se menciona anteriormente, se aplicará una propuesta de red que imita el comportamiento de *ResNet* y fue concebida

para el trabajo con datos tabulares por [37] en el sitio web *Kaggle*. La red original está compuesta por nueve capas *Dense* divididas en tres bloques, acompañadas por capas *BatchNormalization* y *Dropout*, todo esto implementado en la librería *Keras.*

Se eliminaron dos capas *Dense* del segundo bloque que contaba con cuatro capas de este tipo. Con esto se logró mejorar del 77.166 % obtenido inicialmente a un 78.224 %.

La red original usaba una combinación de funciones de activación empleando funciones de tipo *elu*, *relu* y *selu*. Por experiencia obtenida en el trabajo con la red profunda de la sección anterior se probará la utilización de la función *relu* para todos los casos.

- *activation* [*relu*].

Con esto se mejora el accuracy logrando obtener un 81.025 %.

La red original utilizaba un optimizador *SWA* que ya se encuentra en desuso. Se prueba con:

- *optimizer* [*adam, nadam*].

El mejor resultado se obtuvo con *adam* logrando un 81.291 %.

A continuación, se prueban varias opciones de configuración del *batch_size*:

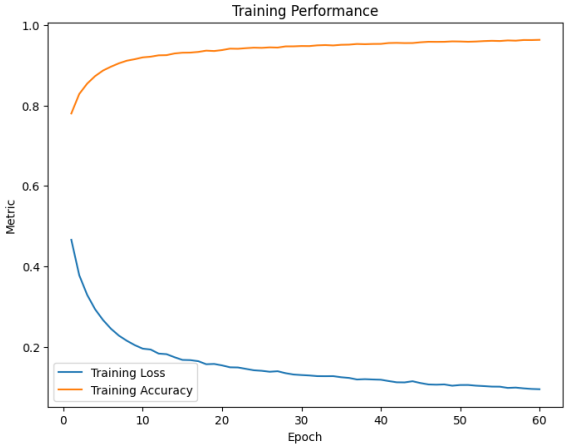
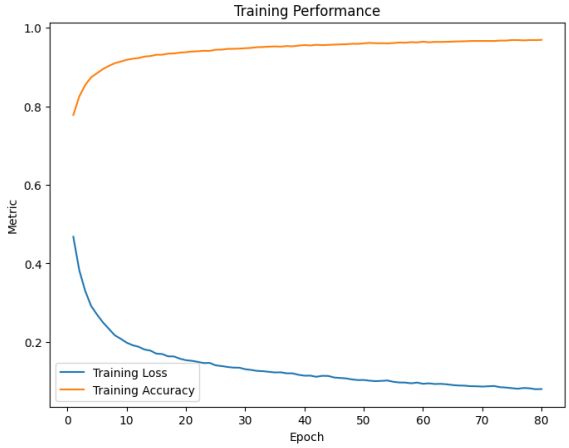
- *batch_size* [16, 32, 64].

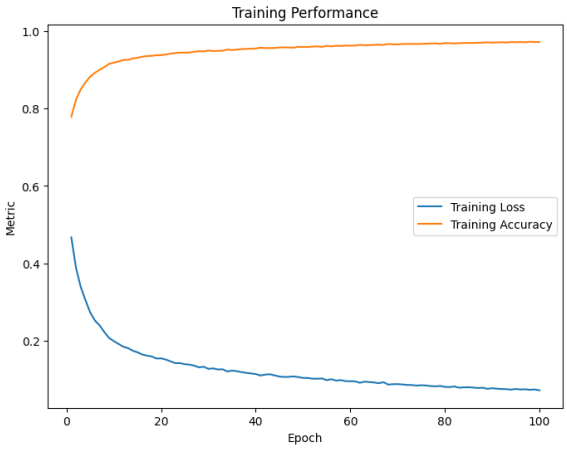
El mejor resultado se obtuvo con un *batch_size* de 64, logrando un 82.034 %.

Para terminar el ajuste de la red para el Conjunto 1 se probará el entrenamiento con los siguientes valores de *epochs*.

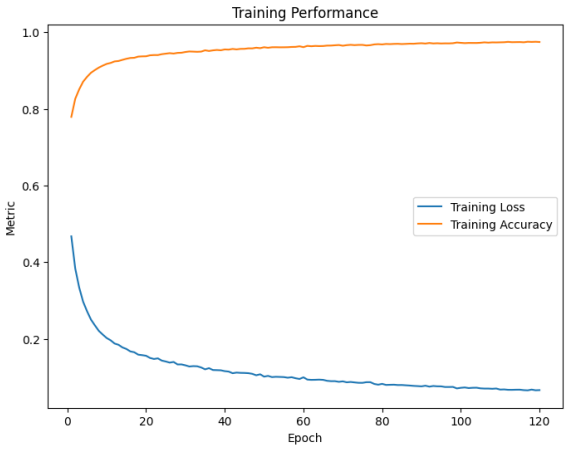
- *epochs* [40, 60, 80, 100, 120, 140].

Tabla 56. Resultados de diferentes valores de epochs para ResNet en el Conjunto 1.

***epochs* = 60**	***epochs* = 80**
	
accuracy = 96.069 %	accuracy = 96.113 %
***epochs* = 100**	***epochs* = 120**

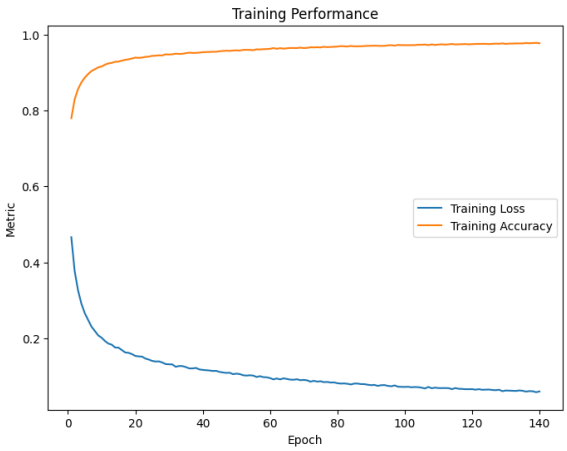


accuracy = 96.785 %



accuracy = 96.511 %

***epochs* = 140**



accuracy = 96.785 %

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 100 épocas. Así termina el ajuste de hiperparámetros de *ResNet* para el Conjunto 1.

Conjunto 2.

En el resto de los conjuntos se eliminan las mismas capas del bloque dos de la red con la que se mejoró el funcionamiento para el Conjunto 1 y también se mantendrá *relu* como la función de activación de todo el modelo.

Se prueban dos variantes de *optimizer*:

- *optimizer* [*adam, nadam*].

El mejor resultado se obtuvo con *adam* logrando un 77.045 %.

A continuación, se prueban varias opciones de configuración del *batch_size*:

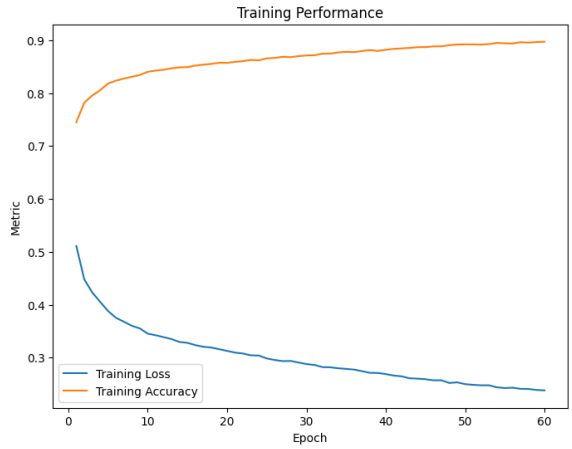
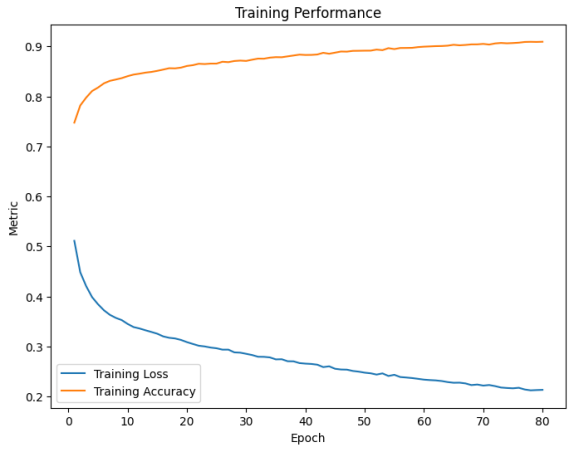
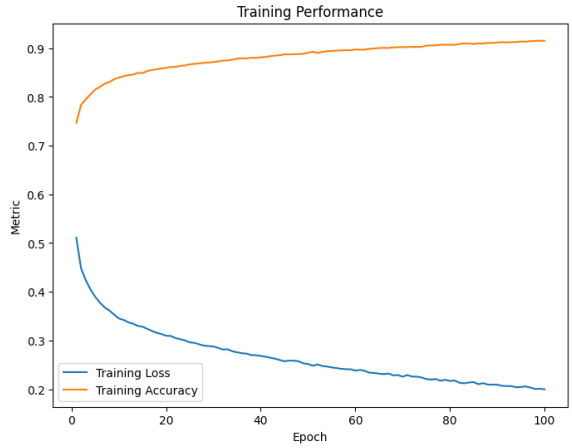
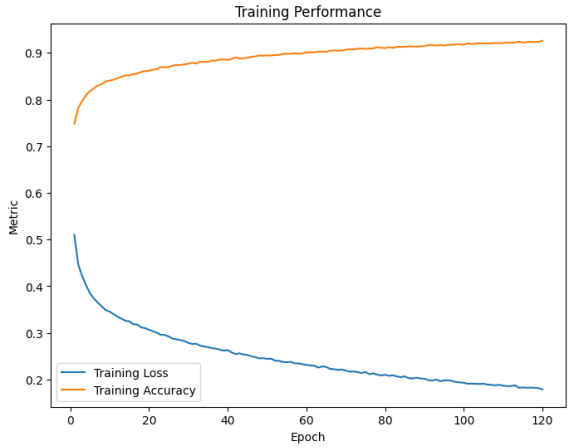
- *batch_size* [16, 32, 64].

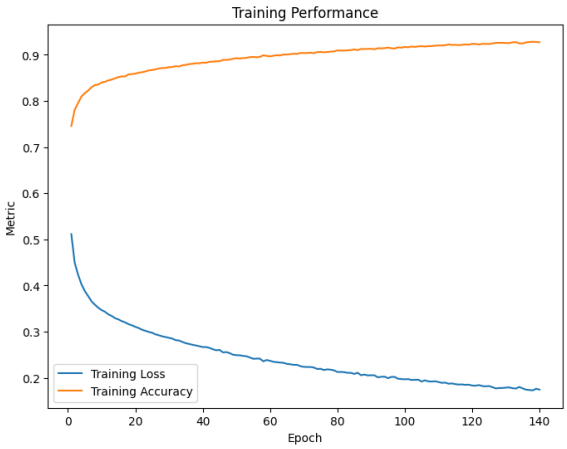
El mejor resultado se obtuvo con un *batch_size* de 64, logrando un 77.749 %.

Para terminar el ajuste de la red para el Conjunto 2 se probará el entrenamiento con los siguientes valores de *epochs*.

- *epochs* [60, 80, 100, 120, 140].

Tabla 57. Resultados de diferentes valores de epochs para ResNet en el Conjunto 2.

***epochs* = 60**	***epochs* = 80**
	
accuracy = 88.469 %	accuracy = 89.668 %
***epochs* = 100**	***epochs* = 120**
	
accuracy = 90.271 %	accuracy = 90.733 %
***epochs* = 140**	



accuracy = 91.126

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 140 épocas donde ya el gráfico muestra un comportamiento bastante estable. Así termina el ajuste de hiperparámetros de *ResNet* para el Conjunto 2.

Conjunto 3.

Se prueban dos variantes de *optimizer*:

- *optimizer* [*adam, nadam*].

El mejor resultado se obtuvo con *nadam* logrando un 87.240 %.

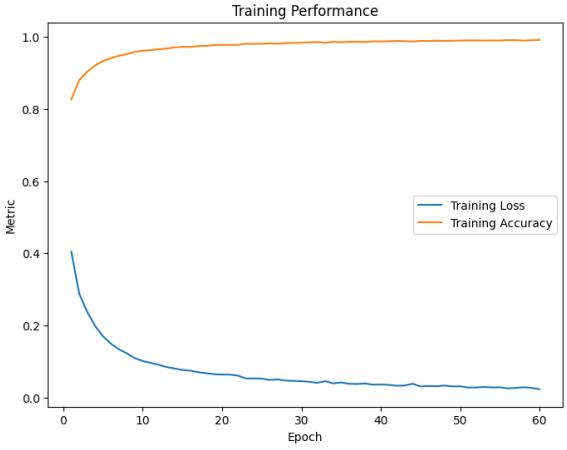
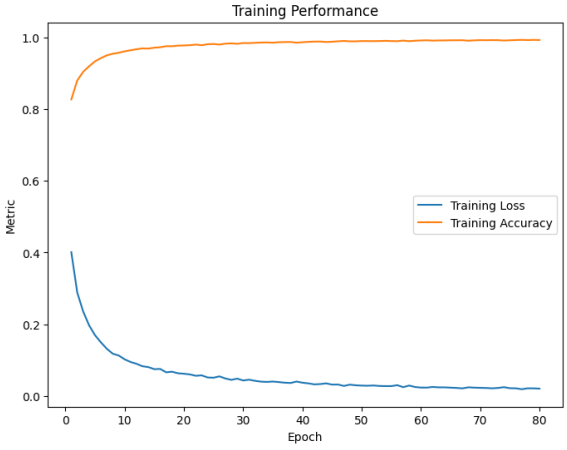
A continuación, se prueban varias opciones de configuración del *batch_size*:

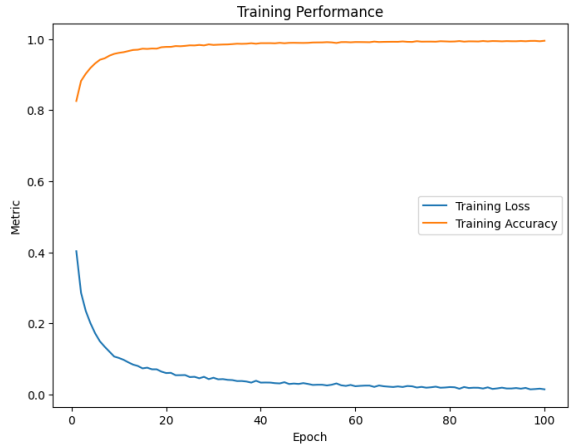
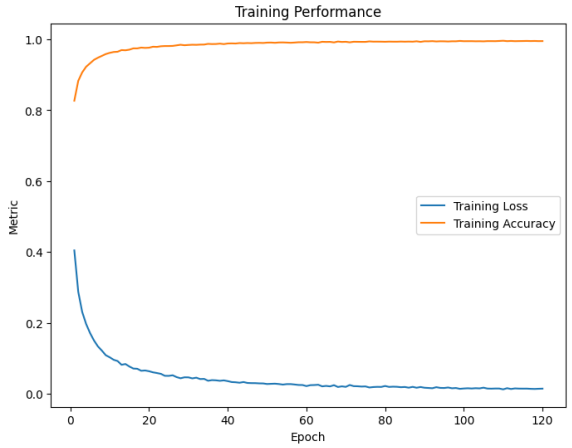
- *batch_size* [16, 32, 64].

El mejor resultado se obtuvo con un *batch_size* de 64, logrando un 87.652 %.

Para terminar el ajuste de la red para el Conjunto 2 se probará el entrenamiento con los siguientes valores de *epochs*.

- *epochs* [60, 80, 100, 120].

***epochs* = 60**	***epochs* = 80**
	
accuracy = 98.096 %	accuracy = 98.202 %

***epochs* = 100**	***epochs* = 120**
	
accuracy = 98.377 %	accuracy = 98.279 %

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 100 épocas. Así termina el ajuste de hiperparámetros de la *ResNet* para el Conjunto 3.

Conjunto 4.

Se prueban dos variantes de *optimizer*:

- *optimizer* [*adam, nadam*].

El mejor resultado se obtuvo con *nadam* logrando un 82.078 %.

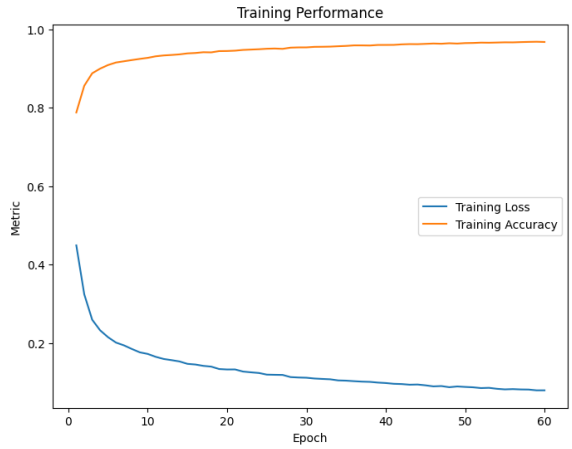
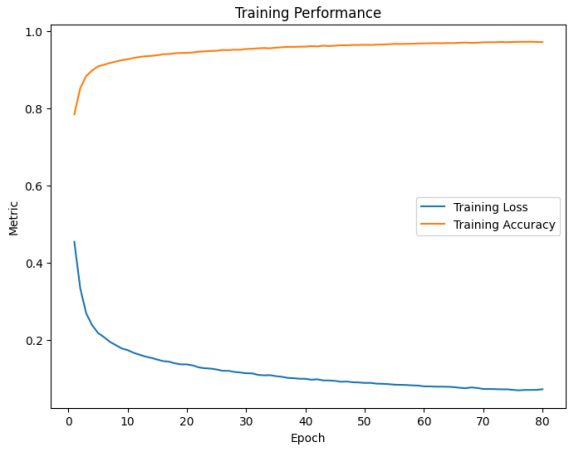
A continuación, se prueban varias opciones de configuración del *batch_size*:

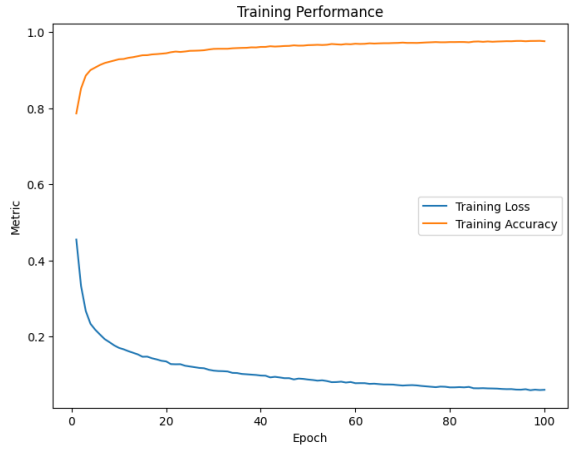
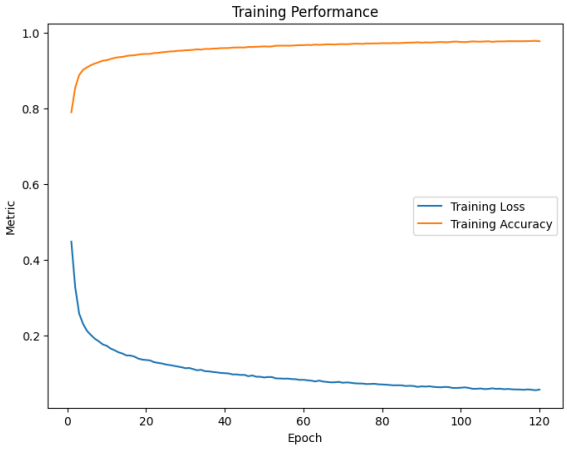
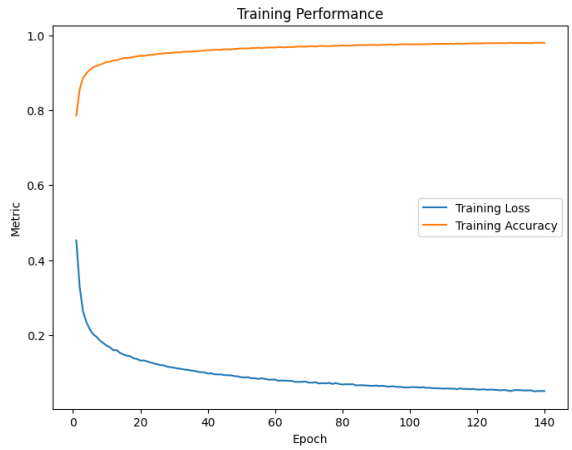
- *batch_size* [16, 32, 64].

El mejor resultado se obtuvo con un *batch_size* de 64, logrando un 83.741 %.

Para terminar el ajuste de la red para el Conjunto 4 se probará el entrenamiento con los siguientes valores de *epochs*.

- *epochs* [60, 80, 100, 120, 140].

***epochs* = 60**	***epochs* = 80**
	

accuracy = 97.055 %	accuracy = 97.260 %
***epochs* = 100**	***epochs* = 120**
	
accuracy = 97.474 %	accuracy = 97.734 %
***epochs* = 140**	
	
accuracy = 97.670 %	

Como se puede apreciar en los gráficos y los valores obtenidos, los mejores resultados se logran con un entrenamiento de 120 épocas. Así termina el ajuste de hiperparámetros de ResNet para el Conjunto 4.

Resultados obtenidos en la optimización de hiperparámetros.

Tabla 58. Resultados obtenidos luego de la optimización de hiperparámetros por las 5 técnicas en los 4 conjuntos de datos.

	RF	SVM	XGBoost	DNN	ResNet
Conjunto 1	95.998 %	97.657 %	96.473 %	97.075 %	96.785 %
Conjunto 2	95.195 %	95.770 %	94.839 %	91.479 %	91.126 %
Conjunto 3	97.947 %	99.493 %	97.887 %	98.128 %	98.377 %
Conjunto 4	98.263 %	98.718 %	98.390 %	97.848 %	97.734 %

4.7 Implementación del Modelo Integrado.

Como se describe en la metodología propuesta, se implementará un modelo integrado que para cada conjunto estará compuesto por las tres técnicas que lograron los mejores resultados, por tanto, este modelo quedará conformado de la siguiente forma:

- Conjunto 1 (SVM, DNN, ResNet).
- Conjunto 2 (SVM, RF, XGBoost).
- Conjunto 3 (SVM, ResNet, DNN).
- Conjunto 4 (SVM, XGBoost, RF).

Se programó una clase en Python donde se creó una función para cada una de las cinco técnicas utilizadas anteriormente. De las cuales se llaman las tres que corresponden específicamente para el trabajo con cada uno de los conjuntos. Las tres técnicas con las que se trabajará en cada caso se ejecutarán en paralelo utilizando la librería *threading.* Una vez que las tres técnicas hayan realizado sus predicciones, estas se guardan en tres listas y se crea otra en la que se almacenará la predicción final del modelo integrado. Como nos encontramos ante un problema de clasificación binaria, las opciones que existe en cada una de las listas solo son 0 y 1. Por tanto, se procede a ejecutar un sistema de votación donde se recorren las tres listas obtenidas y si hay dos o tres listas que en la posición *i* contienen el valor 0, se agrega el valor 0 en la posición *i* de la lista del modelo integrado, si en el caso contrario el valor de dos o tres listas es 1, se realiza la misma operación pero agregando en esta ocasión el valor 1.

Se realizó este procedimiento para los cuatro conjuntos en cada caso usando las tres técnicas definidas para cada caso. Para estas pruebas no se realizó validación cruzada, solo se usaron cada uno de los conjuntos particionados en entrenamiento y prueba. Los resultados fueron los siguientes:

Conjunto 1. De la Tabla 59 a la 66 se muestran la matriz de confusión y el resumen de los resultados obtenidos con cada técnica para este conjunto.

SVM - accuracy: 97.657 %.

Tabla 59. Matriz de confusión del Conjunto 1 con SVM.

	P	N
P	11833	249
N	317	11764

Tabla 60. Resumen del Conjunto 1 con SVM.

Clase	precision	recall	f1-score
0	97 %	98 %	98 %
1	98 %	97 %	98 %
ambas	98 %	98 %	98 %

DNN - accuracy: 96.784 %.

Tabla 61. Matriz de confusión del Conjunto 1 con DNN.

	P	N
P	11906	176
N	601	11480

Tabla 62. Resumen del Conjunto 1 con DNN.

Clase	precision	recall	f1-score
0	95 %	99 %	97 %
1	98 %	95 %	97 %
ambas	97 %	97 %	97 %

ResNet - accuracy: 97.218 %.

Tabla 63. Matriz de confusión del Conjunto 1 con ResNet.

	P	N
P	11634	448
N	224	11857

Tabla 64. Resumen del Conjunto 1 con ResNet.

Clase	precision	recall	f1-score
0	98 %	96 %	97 %
1	96 %	98 %	97 %
ambas	97 %	97 %	97 %

Modelo Integrado - accuracy: 98.547 %.

Tabla 65. Matriz de confusión del Conjunto 1 con el Modelo Integrado.

	P	N
P	11917	165
N	186	11895

Tabla 66. Resumen del Conjunto 1 con el Modelo Integrado.

Clase	precision	recall	f1-score
0	98 %	99 %	99 %
1	99 %	98 %	99 %
ambas	99 %	99 %	99 %

Conjunto 2. De la Tabla 67 a la 82 se muestran la matriz de confusión y el resumen de los resultados obtenidos con cada técnica para este conjunto.

SVM - accuracy: 95.770 %.

Tabla 67. Matriz de confusión del Conjunto 2 con SVM.

	P	N
P	11667	415
N	607	11474

Tabla 68. Resumen del Conjunto 2 con SVM.

Clase	precision	recall	f1-score
0	95 %	97 %	96 %
1	97 %	95 %	96 %
ambas	96 %	96 %	96 %

RF - accuracy: 95.219 %.

Tabla 69. Matriz de confusión del Conjunto 2 con RF.

	P	N
P	11006	1076
N	79	12002

Tabla 70. Resumen del Conjunto 2 con RF.

Clase	precision	recall	f1-score
0	99 %	91 %	95 %
1	92 %	99 %	95 %
ambas	96 %	95 %	95 %

XGBoost - accuracy: 94.839 %

Tabla 71. Matriz de confusión del Conjunto 2 con XGBoost.

	P	N
P	10926	1156
N	91	11990

Tabla 72. Resumen del Conjunto 2 con XGBoost.

Clase	precision	recall	f1-score
0	99 %	90 %	95 %
1	91 %	99 %	95 %
ambas	95 %	95 %	95 %

Modelo Integrado - accuracy: 95.580 %.

Tabla 73. Matriz de confusión del Conjunto 2 con el Modelo Integrado.

	P	N
P	11066	1016
N	52	12029

Tabla 74. Resumen del Conjunto 2 con el Modelo Integrado.

Clase	precision	recall	f1-score
0	100 %	92 %	95 %
1	92 %	100 %	96 %
ambas	96 %	96 %	96 %

Conjunto 3. De la Tabla 75 a la 66 se muestran la matriz de confusión y el resumen de los resultados obtenidos con cada técnica para este conjunto.

SVM - accuracy: 95.580 %.

Tabla 75. Matriz de confusión del Conjunto 3 con SVM.

	P	N
P	5817	6
N	53	5770

Tabla 76. Resumen del Conjunto 3 con SVM.

Clase	precision	recall	f1-score
0	99 %	100 %	99 %
1	100 %	99 %	99 %
ambas	99 %	99 %	99 %

ResNet - accuracy: 98.660 %.

Tabla 77. Matriz de confusión del Conjunto 3 con ResNet.

	P	N
P	5823	0
N	156	5667

Tabla 78. Resumen del Conjunto 3 con Resnet.

Clase	precision	recall	f1-score
0	97 %	100 %	99 %
1	100 %	97 %	99 %
ambas	99 %	99 %	99 %

DNN - accuracy: 97.861 %.

Tabla 79. Matriz de confusión del Conjunto 3 con DNN.

	P	N
P	5816	7
N	242	5581

Tabla 80. Resumen del Conjunto 3 con DNN.

Clase	precision	recall	f1-score
0	96 %	100 %	98 %
1	100 %	96 %	98 %
ambas	98 %	98 %	98 %

Modelo Integrado - accuracy: 99.192 %.

Tabla 81. Matriz de confusión del Conjunto 3 con el Modelo Integrado.

	P	N
P	5822	1
N	93	5730

Tabla 82. Resumen del Conjunto 3 con el Modelo Integrado.

Clase	precision	recall	f1-score
0	98 %	100 %	99 %
1	100 %	98 %	99 %
ambas	99 %	99 %	99 %

Conjunto 4. De la Tabla 83 a la 90 se muestra la matriz de confusión y el resumen de los resultados obtenidos con cada técnica para este conjunto.

SVM - accuracy: 98.718 %.

Tabla 83. Matriz de confusión del Conjunto 4 con SVM.

	P	N
P	18359	130
N	344	18145

Tabla 84. Resumen del Conjunto 4 con SVM.

Clase	precision	recall	f1-score
0	98 %	99 %	99 %
1	99 %	98 %	99 %
ambas	99 %	99 %	99 %

XGBoost - accuracy: 98.390 %.

Tabla 85. Matriz de confusión del Conjunto 4 con XGBoost.

	P	N
P	17929	560
N	35	18454

Tabla 86. Resumen del Conjunto 4 con XGBoost.

Clase	precision	recall	f1-score
0	100 %	97 %	98 %
1	97 %	100 %	98 %
ambas	98 %	98 %	98 %

RF - accuracy: 98.247 %.

Tabla 87. Matriz de confusión del Conjunto 4 con RF.

	P	N
P	17859	630
N	18	18471

Tabla 88. Resumen del Conjunto 4 con RF.

Clase	precision	recall	f1-score
0	100 %	97 %	98 %
1	97 %	100 %	98 %
ambas	98 %	98 %	98 %

Modelo Integrado - accuracy: 98.553 %.

Tabla 89. Matriz de confusión del Conjunto 4 con el Modelo Integrado.

	P	N
P	17960	529
N	6	18483

Tabla 90. Resumen del Conjunto 4 con Modelo Integrado.

Clase	precision	recall	f1-score
0	100 %	97 %	99 %
1	97 %	100 %	99 %
ambas	99 %	99 %	99 %

Resumen final.

Tabla 91. Resultados obtenidos luego de aplicar las 5 técnicas y en Modelo Integrado en los 4 conjuntos de datos.

	RF	SVM	XGBoost	DNN	ResNet	Integrado
Conjunto 1		97.657 %		96.784 %	97.218 %	98.547 %
Conjunto 2	95.219 %	95.770 %	94.839 %			95.580 %
Conjunto 3		99.493 %		97.861 %	98.660 %	99.192 %
Conjunto 4	98.247 %	98.718 %	98.390 %			98.553 %

4.8 Discusión de resultados.

Se aplicaron las cinco técnicas propuestas en la metodología y el Modelo Integrado a los cuatro conjuntos conformados en la etapa de preparación de datos. Los resultados obtenidos luego de ajustar los hiperparámetros fueron muy buenos en todos los casos, todos los clasificadores obtienen porcentajes altos de clasificación en todos los conjuntos. De las cinco técnicas iniciales, la que

mostró mayor efectividad fue *Suppor Vector Machine* que logró el mayor accuracy en los cuatro conjuntos como muestra la Tabla 58.

En el caso del modelo integrado, para la partición realizada en las pruebas que se registraron en este documento, solo logra superar a SVM en el Conjunto 1 como se evidencia en la Tabla 91. Resultado que se valora como lógico debido a que, en un sistema de votación con tres participantes, en los casos donde dos clasificadores obtienen resultados similares superiores a un tercero, el sistema de votación tiende a mejorar los resultados de los dos mejores predictores. Por el contrario, los casos donde dos modelos obtienen resultados similares inferiores a los de un tercero, el sistema de votación tiende a obtener resultados que mejoran al de los dos clasificares de resultados similares, pero que están por debajo del clasificador de mejor evaluación. Por lo que la aplicación de este método es recomendable solo en aquellos casos donde se presente la primera situación descrita.

En cuanto a los resultados por conjunto, las mejores clasificaciones se logran para el Conjunto 3, esto se debe a que es el conjunto que posee un mayor número de variables con un total de 33. Esa misma lógica se cumple en el resto de los casos, el orden de los conjuntos mejor clasificados continúa con el Conjunto 4 que tiene 27 variables, el Conjunto 1 con 25 variables y por último el Conjunto 2 que solo cuenta con 20 variables.

5 Conclusiones.

En el presente trabajo se aborda el problema de la deserción escolar en la educación superior, para intentar impactar de forma positiva en esta situación desde la perspectiva de las ciencias computacionales. Para esto se realizó una revisión bibliográfica dentro de la literatura científica relacionada con la predicción del abandono escolar a partir del trabajo con datos de alumnos. En la búsqueda efectuada se identificaron las principales categorías de predictores que deben estar presente en un conjunto de datos para obtener los mejores resultados, quedó demostrado que incluir tanto dados académicos como datos no académicos permite ejecutar clasificaciones más precisas. En esta etapa inicial también se identificó que los algoritmos clásicos de *Machine Learning* son los más empleados en la literatura para este tipo de problemas. De igual forma, se notó que las métricas más empleadas por los distintos autores para medir la calidad de los resultados obtenidos son *accuracy, recall, precision* y *F1*.

Con base en el análisis realizado durante el estudio del estado del arte, se propuso una metodología para la clasificación de alumnos en riesgo de abandonar la universidad a través del análisis de datos con técnicas de *Inteligencia Artificial*. La metodología propuesta está integrada por cuatro etapas: análisis preliminar de los datos, preprocesamiento de los datos, selección de las técnicas y evaluación de los resultados.

Dicha metodología se aplicó en un conjunto de datos publicado por el Tecnológico de Monterrey en agosto del 2022. El conjunto contiene todas las categorías que se determinó que eran necesarias para obtener los mejores resultados posibles. Se particionó el conjunto original en cuatro subconjuntos para lograr realizar predicciones del abandono a lo largo de la carrera y en el primer año de esta, además de poder ejecutarlas en diferentes momentos. Durante la etapa de preprocesamiento se aplicaron un gran número de técnicas encaminadas a mejorar la calidad de los datos, entre otros aspectos, se logró resolver el problema de los datos faltantes que presentaban el conjunto, se solucionó la problemática del desbalance entre las dos clases objetivo que contiene el conjunto y se evidenció el impacto positivo que tiene para la clasificación trabajar con clases equilibradas. En la etapa de selección de las técnicas, debido a las características particulares de este conjunto, se decidió trabajar con tres algoritmos clásicos de *Machine Learning,* dos algoritmos de *Deep Learning* y un Modelo Integrado de votación compuesto por las tres técnicas que presentaron mejores resultados. Con estas cinco técnicas se llevó a cabo una amplia optimización de hiperparámetros y las mejoras obtenidas evidenciaron la importancia de ejecutar este proceso. SVM demostró ser la herramienta que alcanza mejores resultados en las pruebas individuales con los cuatro conjuntos con los que se trabajó. Mientras que el método integrado probó potencial para mejorar los resultados de clasificación en casos específicos.

6 Referencias.

1. Mundial, B.: Graduarse: solo la mitad lo logra en América Latina. Recuperado de https://www. bancomundial. org/es/news/feature/2017/05/17/graduating-only-half-of-latin-american-students-manage-to-do-so Universidad Femenina del Sagrado Corazón. Recuperado de http://repositorio. unife. edu. pe/repositorio/bitstream/handle/20.500 11955, 94 (2017)

2. OCDE: México: políticas prioritarias para fomentar las habilidades y el conocimiento de los mexicanos para la productividad y la innovación. Serie Mejores Políticas (2015)

3. Publica, S.d.E.: Principales cifras del sistema educativo nacional 2018-2019. Autor (2019)

4. Hanushek, E.A., Woessmann, L.: The economic impacts of learning losses. (2020)

5. Azevedo, J.P., Hasan, A., Goldemberg, D., Geven, K., Iqbal, S.A.: Simulating the potential impacts of COVID-19 school closures on schooling and learning outcomes: A set of global estimates. The World Bank Research Observer 36, 1-40 (2021)

6. Caribe, I.I.d.l.U.p.l.E.S.e.A.L.y.e.: Informe sobre la Educación Superior en América Latina y el Caribe. 2000-2005: La Metamorfosis de la Educación Superior. Iesalc Caracas (2006)

7. Alvarado-Uribe, J., Mejía-Almada, P., Masetto Herrera, A.L., Molontay, R., Hilliger, I., Hegde, V., Montemayor Gallegos, J.E., Ramírez Díaz, R.A., Ceballos, H.G.: Student Dataset from Tecnologico de Monterrey in Mexico to Predict Dropout in Higher Education. Data 7, 119 (2022)

8. Fernández-García, A.J., Preciado, J.C., Melchor, F., Rodriguez-Echeverria, R., Conejero, J.M., Sánchez-Figueroa, F.: A real-life machine learning experience for predicting university dropout at different stages using academic data. IEEE Access 9, 133076-133090 (2021)

9. Kemper, L., Vorhoff, G., Wigger, B.U.: Predicting student dropout: A machine learning approach. European Journal of Higher Education 10, 28-47 (2020)

10. Rovira, S., Puertas, E., Igual, L.: Data-driven system to predict academic grades and dropout. PLoS one 12, e0171207 (2017)

11. Nagy, M., Molontay, R.: Predicting dropout in higher education based on secondary school performance. In: 2018 IEEE 22nd international conference on intelligent engineering systems (INES), pp. 000389-000394. IEEE, (Year)

12. Palacios, C.A., Reyes-Suárez, J.A., Bearzotti, L.A., Leiva, V., Marchant, C.: Knowledge discovery for higher education student retention based on data mining: Machine learning algorithms and case study in Chile. Entropy 23, 485 (2021)

13. Aggarwal, D., Mittal, S., Bali, V.: Significance of non-academic parameters for predicting student performance using ensemble learning techniques. International Journal of System Dynamics Applications (IJSDA) 10, 38-49 (2021)

14. Agrusti, F., Mezzini, M., Bonavolontà, G.: Deep learning approach for predicting university dropout: A case study at Roma Tre University. Journal of e-Learning and Knowledge Society 16, 44-54 (2020)

15. Solis, M., Moreira, T., Gonzalez, R., Fernandez, T., Hernandez, M.: Perspectives to predict dropout in university students with machine learning. In: 2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI), pp. 1-6. IEEE, (Year)

16. Alban, M., Mauricio, D.: Neural networks to predict dropout at the universities. International Journal of Machine Learning and Computing 9, 149-153 (2019)

17. Mezzini, M., Bonavolontà, G., Agrusti, F.: Predicting university dropout by using convolutional neural networks. In: INTED2019 Proceedings, pp. 9155-9163. IATED, (Year)

18. Baranyi, M., Nagy, M., Molontay, R.: Interpretable deep learning for university dropout prediction. In: Proceedings of the 21st annual conference on information technology education, pp. 13-19. (Year)

19. Rodrıguez-Maya, N.E., Lara-Álvarez, C., May-Tzuc, O., Suárez-Carranza, B.A.: Modeling students' dropout in mexican universities. Research in Computing Science 139, 163-175 (2017)

20. Gonzalez, A.G.H., Armenta, R.A.M., Rosales, L.A.M., Barrientos, A.G., Xihuitl, J.L.T., Algredo, I.: Comparative Study of Algorithms to Predict the Desertion in the Students at the ITSM-Mexico. IEEE Latin America Transactions 14, 4573-4578 (2016)

21. Brownlee, J.: Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery (2020)

22. Berzal, F.: Redes neuronales & deep learning: Volumen I. Independently published (2018)

23. Reis, C.A., Patetta, M.: Introduction to Statistical and Machine Learning Methods for Data Science (2021)

24. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. (Year)

26. Cichy, R.M., Kaiser, D.: Deep neural networks as scientific models. Trends in cognitive sciences 23, 305-317 (2019)

27. Breiman, L.: Random forests. Machine learning 45, 5-32 (2001)

28. Mammone, A., Turchi, M., Cristianini, N.: Support vector machines. Wiley Interdisciplinary Reviews: Computational Statistics 1, 283-289 (2009)

29. Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T.: Xgboost: extreme gradient boosting. R package version 0.4-2 1, 1-4 (2015)

30. Kumar, A.: Practical Full Stack Machine Learning: A Guide to Build Reliable, Reusable, and Production-Ready Full Stack ML Solutions (English Edition). BPB Publications (2021)

31. Han, J., Kamber, M., Pei, J.: Data mining concepts and techniques third edition. University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University (2012)

32. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html

33. developers, T.i.-l.: RandomUnderSampler. (2022)

34. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research 16, 321-357 (2002)

35. developers, T.i.-l.: SMOTEENN. (2022)

36. developers, T.i.-l.: TomekLinks. (2022)

37. https://www.kaggle.com/code/code1110/janestreet-resnet-with-autoencoder-train

38. Probst, P., Bischl, B., Boulesteix, A.-L.: Tunability: Importance of hyperparameters of machine learning algorithms. arXiv preprint arXiv:1802.09596 (2018)