UNIVERSIDAD DE COSTA RICA

SISTEMA DE ESTUDIOS DE POSGRADO

MODELO DE CLASIFICACIÓN CON ALGORTIMO TABNET PARA ABANDONO DE CLIENTES EN SERVICIOS DE TELECOMUNICACIONES DEL INSTITUTO COSTARRICENSE DE ELECTRICIDAD ENTRE AGOSTO Y OCTUBRE DE 2022.

Trabajo final de investigación aplicada sometido a la consideración de la Comisión del Programa de Estudios de Posgrado en Estadística para optar al grado y título de Maestría Profesional en Estadística.

PATRICK JOSÉ SANTAMARÍA GUZMÁN

Ciudad Universitaria Rodrigo Facio, Costa Rica

2024

Dedicatoria

Este trabajo final de investigación quiero dedicarlo a todas aquellas personas que, de una u otra forma, han contribuido a lo largo de este proceso. Pero en especial, quiero dedicarlo a mi hermana Amanda, que ha sido un gran apoyo, y, además, un pilar fundamental para sobrellevar todos los retos que se me presentaron a lo largo de este proceso.

También quiero dedicarlo a mis amigos más cercanos y familiares, por su comprensión, paciencia y por estar siempre ahí, brindándome palabras de ánimo y apoyo en los momentos más difíciles.

Agradecimientos

Quiero agradecer a mis profesores, por compartir su conocimiento, por sus valiosos consejos y por incentivar mi aprendizaje en esta rama tan apasionante de la estadística. Son muchos los conocimientos que adquirí gracias a ustedes.

A mis compañeros de clase, por los momentos compartidos, las risas, los chistes, por el apoyo mutuo y por enriquecer mi crecimiento académico con la diversidad de pensamientos y perspectivas que tenían para darme.

A mi hermana Amanda, por aconsejarme y apoyarme emocionalmente en todo este proceso e impulsarme a poder llevar a cabo esta investigación.

A mi profesor tutor y a mis lectores que me apoyaron en la búsqueda de datos y en darme los insumos necesario para lograr llevar a cabo un trabajo robusto.

Y finalmente quiero agradecer a mis amigos y a mi familia, por siempre escucharme, aún en los momentos en que me sentí más frustrado, cuando no lograba ver la luz al final del túnel, fueron fundamentales en este proceso.

Este trabajo final de investigación aplicada fue aceptado por la Comisión del Programa de Estudios de Posgrado en Estadística de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Profesional en Estadística.

LEONARDO CASTELLON RODRIGUEZ (FIRMA)

Firmado digitalmente por LEONARDO CASTELLON RODRIGUEZ (FIRMA)
Fecha: 2024.07.04 09:16:40 -06'00'

Dr. Leonardo Castellón Rodríguez
Representante de la Decana
Sistema de Estudios de Posgrado

JUAN JOSE LEITON MONTERO (FIRMA)

Firmado digitalmente por JUAN JOSE LEITON MONTERO (FIRMA)
Fecha: 2024.07.10 07:21:24 -06'00'

M.Sc. Juan José Leitón Montero
Profesor guía

HEIN JOHAN FRANCOIS VAN DUNNE (FIRMA)
PERSONA FISICA, NUP-152800088820.
Fecha declarada: 15/07/2024 07:38:39 PM
Esta es una representación gráfica únicamente, verifique la validez de la firma.

PhD. Hein Johan François van Dunné
Lector

ALEXIA PACHECO HERNANDEZ (FIRMA)
PERSONA FISICA, CPF-02-0432-0730.
Fecha declarada: 05/07/2024 01:42:24 PM
Esta es una representación gráfica únicamente, verifique la validez de la firma.

PhD. Alexia Pacheco Hernández
Lectora

GILBERT BRENES CAMACHO (FIRMA)

Firmado digitalmente por GILBERT BRENES CAMACHO (FIRMA)
Fecha: 2024.07.05 15:32:28 -06'00'

Dr. Gilbert Brenes Camacho
Director
Programa de Posgrado en Estadística

PATRICK JOSE SANTAMARIA GUZMAN (FIRMA)
PERSONA FISICA, CPF-02-0784-0681.
Fecha declarada: 07/07/2024 08:16:02 PM
Esta es una representación gráfica únicamente, verifique la validez de la firma.

Patrick José Santamaría Guzmán
Sustentante

Tabla de contenidos

Resumen

Dos de los principales objetivos de los proveedores de servicios de telecomunicación son maximizar las ganancias por el servicio brindado y mantenerse vivos en el mercado. Para lograr esto, las estrategias de retención de clientes juegan un papel fundamental, ya que el precio de retener clientes en un servicio de telecomunicación, es mucho menor que el de buscar nuevos clientes. Dada la necesidad de retención de clientes, los algoritmos de aprendizaje de máquinas suelen utilizarse en gran medida con el fin de poder aplicar estrategias de retención de clientes antes de que estos decidan abandonar el servicio de telecomunicación

Debido a esto, el principal objetivo de esta investigación, es determinar si el algoritmo TabNet, una variación de redes neuronales que promete tener un mejor desempeño en la predicción de datos tabulares que las redes neuronales convencionales, presenta mejoras respecto a algoritmos ampliamente utilizados en la predicción de datos en formato tabular, como XGBoost y LightGBM, para lo cual se utilizó un conjunto de datos proporcionado por el Instituto Costarricense de Electricidad de abandono de clientes sobre el uso del servicio prepago de sus clientes, en un periodo de estudio que va de agosto a octubre de 2022.

Para comparar el desempeño de los algoritmos se utilizan dos criterios, el primero es por métricas de desempeño, donde se compara la exactitud, sensibilidad, especificidad y ROC AUC de las predicciones realizadas por los algoritmos, y el segundo criterio es el tiempo de ejecución en minutos que tardan los algoritmos en ejecutarse.

El algoritmo TabNet logra una mejor métrica de desempeño solo cuando se compara la sensibilidad respecto a los otros dos algoritmos, logrando predecir correctamente hasta un 74,4% de los casos de abandonos de clientes, frente a XGBoost y LightGBM que lograron porcentajes de 73.2% y 72.9% respectivamente; si se compara por las demás métricas, LightGBM y XGBoost tienen mejores resultados. Por otro lado, TabNet tarda hasta 5.6 minutos más en ejecutarse que XGBoost y hasta 8.4 minutos más que LightGBM, esto cuando se comparan los algoritmos con las 3 mejores combinaciones de hiperparámetros obtenidas en la métrica de sensibilidad; si se compara por otras métricas de desempeño, estos tiempos pueden llegar a aumentarse hasta en 87 minutos si se compara contra XGBoost.

En general el algoritmo TabNet, debido a sus altos tiempos de ejecución y a la similitud de las métricas obtenidas respecto a LightGBM y XGBoost, se recomendaría solo en caso de que se busque maximizar la métrica de sensibilidad, aunque LightGBM, en general, logra métricas similares con tiempos de ejecución para el ajuste de los modelos mucho menores.

Abstract

Two of the main objectives for telecommunications service providers are to maximize profits from the service provided and to remain competitive in the market. To achieve this, customer retention strategies play a fundamental role, as the cost of retaining customers in a telecommunications service is much lower than the cost of acquiring new ones. Given the need for customer retention, machine learning algorithms are often used extensively to apply retention strategies before customers decide to leave the telecommunications service.

Due to this, the main objective of this research is to determine whether the TabNet algorithm, a variation of neural networks that promises better performance in predicting tabular data than conventional neural networks, presents improvements compared to widely used algorithms for predicting tabular data, such as XGBoost and LightGBM. For this purpose, a dataset provided by the Costa Rican Institute of Electricity on customer churn in their prepaid service was used, covering a study period from August to October 2022.

The performance of the algorithms was compared using two criteria: the first by performance metrics, where accuracy, sensitivity, specificity, and ROC AUC of the algorithm's predictions were compared; and the second criterion was the execution time in minutes for the algorithms to run.

The TabNet algorithm achieved better performance metrics only when comparing sensitivity to the other two algorithms, correctly predicting up to 74.4% of churn cases, compared to XGBoost and LightGBM, which achieved 73.2% and 72.9%, respectively. When compared using other metrics, LightGBM and XGBoost performed better. On the other hand, TabNet took up to 5.6 minutes longer to run than XGBoost and up to 8.4 minutes longer than LightGBM when comparing the algorithms with the top three hyperparameter combinations based on the sensitivity metric. When compared using other performance metrics, these times could increase by up to 87 minutes compared to XGBoost.

In general, due to its high execution times and the similarity of metrics obtained compared to LightGBM and XGBoost, the TabNet algorithm would only be recommended if the goal is to maximize the sensitivity metric, although LightGBM generally achieves similar metrics with much shorter model tuning times

Lista de cuadros

Lista de figuras

Introducción

Uno de los principales retos de las empresas que brindan servicios de telecomunicaciones es el problema de abandono de los clientes. El abandono de clientes sucede muchas veces debido a la insatisfacción del servicio brindado y/o a una mejor oferta de otros proveedores (Umayaparvathi & Iyakutti, 2016, p. 1065). Generalmente el abandono de clientes se suele clasificar en voluntario e involuntario, donde el voluntario corresponde a una decisión tomada por el cliente de cambiarse a otra compañía, mientras que el involuntario se produce debido a la reubicación de zona geográfica del cliente, caída en morosidad o impago, la muerte, entre otras causas. La mayoría de las aplicaciones de abandono de clientes en el contexto de negocio de telecomunicaciones se suele centrar en el abandono voluntario, ya que este se produce debido a factores de relación entre la compañía y el cliente (Nuñez, 2015, p. 15).

El objetivo final de los proveedores de servicios de telecomunicaciones es maximizar sus ganancias y mantenerse vivos en un mercado competitivo, en el que los clientes tienen diversas opciones y competencia a las cuales acceder, es por esto que, la detección temprana de abandono de clientes es vital en el sector de telecomunicaciones, ya que los operadores tienen que retener a sus clientes y mejorar su administración de gestión de relaciones (CRM por sus siglas en inglés) con los mismos (Ullah et al., 2019, p. 60135).

En la retención de clientes el CRM (Customer Relationship Management) juega un papel muy importante, ya que este describe la táctica que una empresa utiliza para manejar las interacciones con los clientes. Una estrategia de CRM es por ejemplo un programa de tarjetas que dan acceso a descuentos especiales cuando se utilizan, donde además esta tarjeta guarda información sobre lo que el cliente compra y le permite a la empresa crear un perfil del comportamiento del mismo, de manera que la empresa puede ofrecer programas, cupones y descuentos que los motiven a comprar sus productos (Vergara & Cardenas, 2014, p. 15).

De todas las estrategias existentes, la de retención de los clientes existentes es la menos costosa de todas, en comparación a otras, por lo que las empresas tienen que reducir

la pérdida potencial de clientes (Lalwani et al., 2022, p. 3). Se estima que el costo de nuevos clientes es mucho más alto que el de retención de los mismos (Vafeiadis et al., 2016, p. 1), de hecho predecir el abandono de clientes es alrededor de 16 veces más barato que atraer nuevos clientes y el costo de invitar a nuevos clientes es entre 5 a 6 veces más que mantener a los clientes existentes, de manera que lograr disminuir la tasa de abandono (churn) en un 5%, aumenta el beneficio del 25% al 85%, esto para el ámbito de telecomunicaciones (Ullah et al., 2019, p. 60135).

Cuando la tasa de abandono de clientes de un determinado servicio es alta eso puede tener algunas consecuencias, por ejemplo, si se pierden los clientes, se pierden los ingresos procedentes de los mismos, se comienzan a bajar los precios, lo que finalmente repercute un menor beneficio neto para la empresa, existe un riesgo para los empleados como reducción de sueldos, despidos, entre otros, la confianza de los inversores o proveedores de la empresa puede decaer, entre otras más (Vergara & Cardenas, 2014, p. 15).

Para encontrar soluciones a estas problemáticas, las empresas necesitan predecir con anticipación los clientes que corren riesgo de abandono, esto con el fin de crear estrategias para la retención de los mismos. La estrategia de retención se puede realizar de dos formas, una que es la reactiva, que es cuando las empresas esperan la solicitud de cancelación del cliente, y en consecuencia ofrecen planes atractivos para su retención, y por otro lado está la proactiva, que es cuando se predice la probabilidad de abandono, y para aquellos que tienen una mayor probabilidad de abandono, se ofrecen los planes a los clientes antes de que estos presenten la solicitud de cancelación (Lalwani et al., 2022, p. 3).

Bajo un esquema ideal, se esperaría realizar las estrategias de retención de forma proactiva, aumentando las exigencias de un modelo que sea capaz de predecir el abandono de los clientes, con el fin de proporcionar estrategias de retención anticipadas, que son vitales para la disminución de pérdidas de una empresa (Ullah et al., 2019, p. 60134).

El aprendizaje de máquinas (conocido como “machine learning” en inglés) suele ser muy utilizado en problemas de predicción para abandono de clientes en los servicios de telecomunicaciones. El aprendizaje de máquinas es la ciencia que hace que las máquinas

aprendan a partir de datos, esto quiere decir, que, en vez de programar paso a paso una solución para una necesidad planteada, se desarrollan algoritmos numéricos que sean capaces de extraer patrones de diferentes tipos de datos (Bobadilla, 2021, p. 13).

Existen varios tipos de aprendizaje de máquinas, entre ellos el aprendizaje supervisado, donde está incluidos los problemas de regresión y clasificación, que se caracterizan porque tienen una variable objetivo a predecir, luego está el aprendizaje no supervisado que involucra técnicas de agrupamiento o reducción de dimensionalidad, que se caracteriza porque no existe una variable objetivo, y finalmente están otros tipo como el aprendizaje semi-supervisado, donde una parte de los datos tienen una variable objetivo y el resto no y por último el aprendizaje por refuerzo, donde el algoritmo recibe información de un entorno real o simulado y cuando el sistema realiza una acción, el algoritmo recompensa o penaliza tal y como sucede en seres vivos (Bobadilla, 2021, p. 14-18). Para el caso particular de predicción temprana de abandono de clientes en servicios de telecomunicaciones, entraría dentro de un problema de clasificación de aprendizaje supervisado, cuya variable objetivo es si el cliente abandona el servicio o no, una respuesta binaria en este caso.

Para este tipo de modelos de abandono de clientes suele presentarse un problema muy común, que es el desbalanceo de las clases, este consiste en cuando una de las clases de la variable objetivo o respuesta está subrepresentada respecto a la otra. El problema con esto es que, al haber sobre representación de una de las clases, involuntariamente el modelo va a tender a realizar más predicciones hacia dicha clase que está sobre representada, y generalmente en este tipo de problemas la clase que más interesa es la minoritaria. Para tratar este tipo de problemas se utilizan técnica de remuestreo como el submuestreo y el sobremuestreo (Calvino, 2017, p. 5).

En un estudio realizado por Guitérrez (2020) donde se analizan una serie de estudios realizados en problemas de abandono de clientes y el tipo de algoritmo utilizado para realizar las predicciones, se determina que los más utilizados son las máquinas de soporte vectorial (utilizado 12 veces), seguido de los árboles de decisión (utilizado 11 veces), y por detrás las redes neuronales artificiales (utilizado 8 veces) y las regresiones (utilizado 7 veces), sin

embargo, los algoritmos que tuvieron mejores resultados en términos de predicción son los árboles de decisión, seguida de las regresiones, y si bien, a pesar de que las máquinas de soporte vectorial y las redes neuronales en general tienen un desempeño más bajo en este tipo de problemas, estos son menos sensibles al tamaño de muestra, ya que los árboles y las regresiones se determinó que a mayor tamaño de muestra, se reduce la precisión de las predicciones, mientras que en las máquinas de soporte y las redes neuronales no varía mucho independientemente del tamaño de muestra.

Por lo general, las redes neuronales profundas (DNN por sus siglas en inglés) no suelen ser buenas en términos de métricas de desempeño para predicción de datos en formato tabular, aunque estas si han tenido mucho éxito en la predicción de imágenes (He et al., 2016), texto (Lai et al., 2015) y audios (Amodei et al., 2015). Las DNN están compuestas por múltiples capas, estas son estudiadas en el aprendizaje profundo (conocido como Deep Learning en inglés) que es una subárea del aprendizaje de máquinas. El uso de múltiples capas en las redes neuronales permite a los modelos capturar y aprender patrones, representaciones de datos, con mayor complejidad y con múltiples niveles de abstracción que redes de una sola capa o pocas. La cantidad de capas determina la profundidad de estos modelos y su capacidad para capturar patrones más complejos, aunque a veces ajustar parámetros de tantas capas conduce a sesgos (Dinamarca, 2018, pp. 4).

A pesar de que las redes neuronales convencionales suelen ser efectivas en otro tipo de datos, el algoritmo TabNet ha demostrado ser efectivo en la clasificación de datos tabulares, superando a otras variantes de algoritmos de clasificación como XGBoost y las redes neuronales profundas convencionales, en una amplia gama de conjuntos de datos tabulares. TabNet es una arquitectura de redes neuronales profundas para datos tabulares que se entrena utilizando una optimización basada en el descenso de gradiente. El algoritmo opera bajo un enfoque iterativo, en el cual utiliza la atención secuencial para elegir las variables más importantes en cada paso de decisión, y aplicando a su vez, máscaras de atenuación aleatorias a las variables seleccionadas, esto con el fin de que evitar la dependencia excesiva de las variables y fomentar la generalización (Arik & Pfister, 2021, p. 1).

En la actualidad, la mayoría de las competencias de ciencia de datos son dominadas por los algoritmos basados en ensambles de árboles de decisión, en particular los de XGBoost y LightGBM (Arik & Pfister, 2021, p. 2). Debido a esto, el presente estudio pretende contrastar el desempeño del algoritmo TabNet en la clasificación de datos de abandono de clientes en telecomunicaciones, contra los algoritmos de ensamble de árboles de decisión de XGBoost y LightGBM. Esto permitirá determinar si este algoritmo de redes neuronales profundas para datos tabulares es útil para la detección temprana de abandonos de clientes en los servicios de telecomunicaciones de una importante empresa costarricense, para así establecer políticas de retención de clientes y disminuir las pérdidas causadas por estas razones, o bien, evitar traer consecuencias como las anteriormente mencionadas.

Objetivos de investigación

Objetivo general

Evaluar desempeño de modelo de clasificación para abandono de clientes en telecomunicaciones del Instituto Costarricense de Electricidad entre agosto y setiembre de 2022, ajustado con el algoritmo TabNet, con el fin de determinar si el algoritmo produce mejorías en el desempeño de las predicciones respecto a otros algoritmos ampliamente utilizados en la predicción de datos tabulares como XGBoost y LightGBM.

Objetivos específicos

- Ajustar modelos de clasificación para abandono de clientes en telecomunicaciones del ICE, utilizando los algoritmos de TabNet, XGBoost y LightGBM con diversas combinaciones de hiperparámetros, con el fin de que sea posible establecer comparaciones predictivas entre los tres algoritmos.
- Comparar indicadores de desempeño del modelo ajustado con el algoritmo TabNet contra algoritmos de ensamble de árboles de XGBoost y LightGBM con la finalidad de evidenciar si TabNet tiene mejor desempeño predictivo para un modelo de abandono de clientes en el ICE.
- Establecer recomendaciones acerca del uso del algoritmo de TabNet y su desempeño predictivo en el ámbito de datos tabulares, específicamente para abandono de clientes en telecomunicaciones, esto para destacar ventajas y desventajas del uso de este algoritmo en este tipo de datos.

Estado de la cuestión

Uso de algoritmos de aprendizaje de máquinas en abandono de clientes

En términos de abandono de clientes existen muchos estudios donde se implementan diversos algoritmos de clasificación, con el fin último de conocer anticipadamente el posible abandono de un cliente, y en esto está incluido el caso de las empresas de telecomunicaciones. Sin embargo, en el ajuste de dichos modelos, el reto siempre ha estado en generar mejores métricas de desempeño, y en esta área en particular influyen principalmente dos factores, el primero tiene que ver en cómo se maneja el desbalanceo de las clases, pues en la mayoría de caso las tasas de abandono son muy bajas respecto a la cantidad de clientes activos que no abandonan, y el segundo es lograr el mejor ajuste de los hiperparámetros para que los algoritmos de aprendizaje de máquinas utilizados tengan el mejor desempeño predictivo.

El mejor desempeño predictivo de estos algoritmos suele medirse utilizando métricas de desempeño, las cuales se calculan basándose en una pequeña muestra de los datos, conocida como datos de prueba, donde se comparan las clases (abandono y no abandono, por ejemplo) contra las predicciones realizadas por el algoritmo en cada uno de los registros de la muestra. Una de las métricas más utilizadas en diversos estudios a la hora de presentar resultados, es la exactitud (conocida como “accuracy” por su traducción al inglés), la cual corresponde a la proporción de casos clasificados correctamente por el algoritmo, sean positivos (abandono, por ejemplo) o negativos (no abandono, por ejemplo). Existen otras métricas utilizadas, como la sensibilidad, que es la proporción de casos positivos clasificados correctamente y la especificidad que es la proporción de casos negativos clasificados correctamente (Boehmke & Greenwell, 2019).

En la práctica, un ejemplo de aplicación de este tipo de algoritmos en datos de telecomunicaciones se hizo con datos obtenidos del sitio web de Customer DNA, que contiene datos de tráfico de 106 000 clientes de un operador de telecomunicaciones. El archivo de datos posee cifras de tráfico de 3 meses (aproximadamente 300 000 registros), y

una variable que determina si el cliente está activo o abandonó. El tipo de tráfico (saliente, entrante, vos SMS), el destino del tráfico (red, competencia), el plan de tarifas, su lealtad y comportamiento del tráfico son algunas de las variables presentes en el conjunto de datos. Se ajustaron modelos con varios algoritmos de aprendizaje de máquinas, entre ellos la regresión lineal, logística, regresión artificial, redes neuronales, k-medias, árboles de decisión y algunos modelos híbridos; mediante métricas de desempeño, se obtuvo un 0.7 de exactitud en las predicciones utilizando el algoritmo CHAID, que es una variación de árboles de decisión convencionales, este fue el que obtuvo mejor desempeño. En esta ocasión los autores emplearon sub muestreo, de forma que la cantidad de registros de la clase mayoritaria fuera aproximadamente la misma que la clase minoritaria, esto para combatir el desbalanceo de los datos, pues la tasa de abandono en este estudio era de un 5.9% (Qureshi et al., 2013).

En una tesis de maestría realizada por Nuñez (2015), se tomaron los datos de una empresa de telecomunicaciones de carácter confidencial y se trabajó con 6 archivos de datos proporcionados por la empresa de finales de junio de 2014, los cuales poseían una cantidad de registros total de entre 250 000 y 300 000, y entre 6 - 12 variables. Se ejecutaron diversos algoritmos de clasificación con el fin de obtener mejores métricas de desempeño, los algoritmos probados en este caso fueron: árboles de decisión, Naive Bayes, KNN, árboles de decisión usando Boosting y ensamble de árboles. El algoritmo de árboles de decisión tuvo métricas de desempeño superiores con valor ROC AUC de 0.768, seguido de Boosting con un 0.733 de esa misma métrica. El autor menciona que un estudio realizado con técnicas de minería de datos a Telecom para el cálculo de bajas de clientes, se analizaron 61 artículos científicos relevantes publicados en revistas como: Elsevier, IEEE Xplore, Springer Link, ScienceDirect, ACM Digital Library y Microsoft Academic; esto entre los años 2002 y 2013. En dichos artículos los árboles de decisión destacaron como la técnica más usada (25% de los artículos científicos analizados), seguido de las redes neuronales.

Una investigación realizada en el 2015 en Jordania, con datos proporcionados por Jordanian Telecommunication Company, una de las principales empresas de telecomunicaciones celulares en ese país, se comparó el rendimiento de 4 algoritmos, algunos de ellos híbridos, estos algoritmos fueron: C4.5, redes neuronales artificiales de perceptrón

multicapa (MLP-ANN), k-medias + MLP-ANN, agrupamiento jerárquico + MLP-ANN y SOM + MLP-ANN. Los resultados arronjaron que el mejor modelo, con una exactitud de 0.972 fue el híbrido que combina k-medias con MLP-ANN, mientras que los dos modelos con menor rendimiento fueron aquellos que no son híbridos, que en este caso era el C4.5 y MLP-ANN. Para la investigación el conjunto de datos poseía 11 variables y 5000 clientes asociados a un servicio prepago y seleccionados aleatoriamente en un periodo de 3 meses. Las variables comprendían estadísticas acerca de las llamadas entrantes y salientes de dichos clientes, mientras que la tasa de abandono de clientes del conjunto de datos era de 7.6%, lo que correspondía a 381 clientes absolutos (Hudaib et al., 2015). Este es un ejemplo de estudio donde las redes neuronales, en combinación con el algoritmo de k-medias, dieron excelentes resultados, sin embargo, no se aplicó ninguna corrección o técnica para controlar el desbalanceo de las clases.

Ese mismo año Dahiya y Bhatia (2015) realizaron un estudio de clasificación para abandono de clientes en telecomunicaciones comparando la regresión logística y los árboles de decisión. Para ello utilizaron datos de la Copa KDD 2019, los cuales se utilizaron para analizar la tendencia de marketing de los clientes de las grandes bases de datos de la empresa francesa de telecomunicaciones Orange. Los modelos se evaluaron con 3 distintos conjuntos de datos, uno pequeño con 10 variables numéricas y 50 registros, uno mediano con 50 variables numéricas y 200 registros, y uno grande con 100 variables (categóricas y numéricas) con 608 registros, con tasas de abandono de 30%, 31.5% y 32.9% respectivamente. La exactitud en los 3 conjuntos de datos fue superior siempre en los árboles de decisión, y las exactitudes para ambos algoritmos fueron mayores en el conjunto de datos grande, y a pesar de que no se detallan cifras, los gráficos muestran una exactitud cercana a 0.98 en el caso de los árboles de decisión y cercana a 0.96 para la regresión logística, en ambos casos para el conjunto de datos grande. El estudio no menciona nada sobre el desbalanceo de los datos, pero dado que en este caso no era tan marcado como en la mayoría de los que se han mencionado, se justifica no haber implementado alguna técnica de re muestreo.

El año siguiente, en una tesis realizada en Portugal en el 2016 para WeDo Tecnologies, que es el proveedor número uno de software de gestión de fraude y garantía de ingresos para operadores de telecomunicaciones. La empresa proporcionó datos reales para implementar modelos de aprendizaje de máquinas en abandono de clientes; en esta ocasión se pusieron a competir 6 algoritmos: KNN, Naive Bayes, J48, RandomForest, AdaBoost y ANN, donde el algoritmo Random Forest fue el que obtuvo un valor más alto de área bajo la curva ROC (0.9901) y sensibilidad (0.9050). Para esta ocasión se tenían 100 064 datos, donde un 17.5% correspondía a abandonos y el restante a no abandonos (82.5%); y con el fin de combatir el desbalanceo, utilizaron la técnica SMOTE, que es una técnica que combina sub muestreo en la clase mayoritaria y sobre muestreo en la clase minoritaria, quedando finalmente con un 42.9% de abandonos contra un 57.1% de clientes activos, y un total de 89 159 registros. Para el entrenamiento de los modelos se utilizó un 70% para datos de entrenamiento y un 30% de datos para validación, aplicando validación cruzada de 10 pliegues y 3 repeticiones en cada pliegue para cada algoritmo implementado con el fin de ajustar hiperparámetros (Esteves & Mendes, 2016). Para el caso de este estudio un área bajo la curva ROC de casi 1, para el algoritmo Random Forest, y el hecho de que se realizaran 3 repeticiones sobre cada uno de los 10 pliegues, da indicios de que podría existir algún problema de sobreajuste y adaptabilidad del algoritmo a los datos de entrenamiento/prueba.

En el 2017 para estudio realizado con datos de un operador de telecomunicaciones marroquí, se tomaron datos de 635 997 clientes, lo cuales utilizaron los servicios de telecomunicación prepago de voz, SMS y datos durante 3 meses; el conjunto de datos tenía 6 variables explicativas. Del total de clientes solo 65 573 abandonaron el servicio, es decir un 10,3%, y para combatir este desbalanceo se utilizó la validación cruzada estratificada, sin aplicar alguna técnica de submuestreo o sobremuestreo. Se ajustaron modelos de clasificación para predecir el abandono de clientes y se utilizaron los siguientes algoritmos: regresión logística, máquinas de soporte vectorial, árbol de decisión, k-vecinos más cercanos, Gradient Boosting Classiffier (GBC) y Naive Bayes; de todos el GBC fue el que obtuvo una mejor exactitud, de 0.91 en este caso, seguido del k-vecinos más cercanos (Chouiekh, 2017).

En el 2018, Sabbeh (2018) comparó diversos algoritmos de clasificación para un problema de clasificación de abandono de clientes en telecomunicaciones. Los algoritmos comparados fueron: análisis discriminante, árboles de decisión (CART), k-vecinos más cercanos, máquinas de soporte vectorial, regresión logística, modelos de ensamble (Random Forest, ADA Boosting Trees, Stocastic Gradient Boosting), Naive Bayes, y redes neuronales de perceptrón multicapa. El conjunto de datos tenía 3 333 registros y pertenecía a Telecom Company, donde un 14% correspondían a abandonos y un 86% a clientes activos. Además, el conjunto de datos estaba compuesto por 17 variables explicativas, relacionadas con el uso del servicio de los clientes durante el día en llamadas internacionales y llamadas de servicio al cliente. Para este caso, el entrenamiento se realizó por medio de una validación cruzada de 10 pliegues y utilizando 60% de entrenamiento y 40% para validación en todos los pliegues. Los algoritmos de Random Forest y ADA Boost fueron los que obtuvieron mejor métrica de exactitud, con un 0.96, seguido de las redes neuronales y el Stocastic Gradient Boosting con un 0.94. No se aplicó alguna técnica específica para controlar el desbalanceo de las clases, que, si bien en este estudio no era tan bajo, si era muy distinto a la clase mayoritaria de clientes que no abandonaron el servicio.

En otro estudio más reciente, realizado en el año 2019 para una empresa del negocio de telecomunicaciones en Colombia, se implementaron modelos de clasificación para predecir la cancelación anticipada de productos de dicha empresa, que serviría de insumo en la elaboración de campañas de prevención de abandono. En el modelado de los datos se utilizaron 4 técnicas distintas de aprendizaje de máquinas, las cuales fueron: árboles de decisión, bosques aleatorios, XGBoost y máquinas de soporte vectorial; el modelo con mejor desempeño fue el de XGBoost, con un valor ROC AUC de 0.95, seguido de los bosques aleatorios con un valor ROC AUC de 0.70 (Echeverri, 2019).

Algoritmo XGBoost en la predicción de abandonos

El algoritmo XGBoost por sí solo, desde que se creó ha dado buenos resultados cuando se trata de predicción de abandono. En una investigación llevada a cabo en el 2020, se utiliza una muestra de datos de clientes de la empresa Telkom en Indonesia, y se registra

su comportamiento registrado durante 6 meses, entre octubre de 2017 y marzo de 2018. El conjunto de datos tenía 132 columnas en total y 105 694 registros, de las cuales todas las variables predictoras eran numéricas y la variable respuesta correspondía a si el cliente abandona el servicio o no; de esta última se obtenía una tasa de abandono total de 9.56%. Se pusieron a prueba 2 algoritmos, XGBoost y la regresión logística, para lo cual el conjunto de datos se dividió en un 70% de entrenamiento y un 30% de prueba, utilizando una validación cruzada de 10 pliegues para ajustar los hiperparámetros de los algoritmos. El modelo de XGBoost obtuvo una exactitud de 0.97 mientras que el de la regresión logística de solo 0.90, a nivel de sensibilidad fueron de 0.80 y 0.06, y de especificidad de 0.998 y 0.995 respectivamente, sin embargo, existen sospechas de sobreajuste en el algoritmo XGBoost debido a la disminución de la sensibilidad desde la fase de entrenamiento, hasta la fase de prueba (Hanif, 2020).

En otro estudio, Tang et al. (2020) crean un modelo híbrido de XGBoost y redes neuronales con perceptrón multicapa (MLP por sus siglas en inglés). El estudio argumenta que las redes neuronales tradicionales no suelen lograr un buen rendimiento predictivo cuando se enfrentan a variables numéricas, mientras que, en las estructuras basadas en árboles de decisión, suelen ser muy efectivas para este tipo de problemas, por lo que el modelo híbrido podría dar mejores resultados. Utilizaron 2 conjuntos de datos disponibles en la plataforma Kaggle sobre abandono de clientes de 2 diferentes industrias, Music y Telecom. El de Music tenía un total de 992 931 registros y una tasa de abandono del 6.4%, mientras que el de Telecom, tenía solo 51 047 registros y una tasa de abandono del 28.8%, y utilizan para el entrenamiento de los modelos un 80% de datos para entrenamiento y el restante 20% de datos para validación, corriendo los algoritmos 5 veces y finalmente tomando el promedio de desempeño de los mismos. Los algoritmos que se pusieron a prueba fueron: regresión logística, MLP, XGboost y XGBoostMLP, donde, por ejemplo, para el conjunto de datos de Music, el modelo híbrido XGBoostMLP tuvo mejor desempeño, con una exactitud en las predicciones de 0.963.

Otro caso similar donde se utiliza el algoritmo XGBoost y la regresión logística como comparación en un estudio relacionado al abandono de clientes en el ámbito de

telecomunicaciones es el de Parmar y Serasiya (2021), en el cual utilizan un conjunto de datos de IBM Watson Telecom, que tiene 7 043 registros y 21 columnas, los autores mencionan que el conjunto de datos no está desequilibrado. En este caso sin aplicar validación cruzada o realizar ajuste de hiperparámetros, y simplemente dividiéndolos en entrenamiento y validación, el modelo de XGBoost obtiene una exactitud de 0.805, mientras que la regresión logística de 0.7889 al igual que en el estudio anterior de Indonesia, donde el modelo de XGBoost tenía un rendimiento superior a la regresión logística.

Falla Arango (2021), para una compañía de telecomunicaciones en Colombia, puso a prueba 8 algoritmos distintos para la predicción de abandono de cliente: regresión logística, K vecinos más cercanos, bosques aleatorios, análisis discriminante lineal, Naive Bayes, perceptrón multicapa, Gradient Boosting Machine y XGBoost, donde el algoritmo de XGBoost fue el que tuvo mejor exactitud y menor cantidad de falsos positivos y falsos negativos, con un valor del área bajo la curva de ROC de 0.7854, seguido del Gradient Boosting Machine que tuvo un valor de 0.7851. En este estudio la tasa de abandono de clientes fue de 1,8%, que correspondía a 106 501 clientes, sin embargo, el autor destaca que países operadores como Vadafone o Más Móvil en España reportan una tasa de 1.5% de abandono en el mes de mayo de 2018, y algunos como Movistar presentan una tasa de abandono significativamente menor de 0.9% (Falla, 2021, p. 31), aunque si bien, la tasa de abandono de dicho estudio no es la más baja, el autor comenta que una de las debilidades de su estudio es el desbalanceo de las clases, debido a la baja tasa de abandono de clientes que existe en los datos, y que en este caso particular no se aplicaron técnicas de submuestreo ni sobremuestreo.

Otro estudio realizado en Sri Lanka, ese mismo año, debido a un problema agravante en la industria de telecomunicaciones, se tomó un conjunto de datos de 10 000 consumidores de servicios pospago del año 2019, el conjunto de datos tenía 20 variables. Se pusieron a prueba 7 algoritmos con el fin de predecir el abandono de clientes como un problema de clasificación, estos algoritmos fueron: árboles de decisión, redes neuronales, regresión logística, bosques aleatorios, XGBoost, ADA Boost, y máquinas de soporte vectorial. Se utilizó un 70% de datos para entrenamiento y un 30% para validación, a su vez, se

implementó una validación cruzada de 5 pliegues con el fin de realizar el ajuste de los hiperparámetros y determinar la mejor combinación. El algoritmo XGBoost superó a todos los demás con una exactitud de 0.829 y con la menor tasa de error, que fue de 17.10%, seguido de la red neuronal que tuvo una exactitud de 0.824 (Senthan et al., 2021).

Uso de algoritmo LightGBM en aprendizaje de máquinas

El caso del algoritmo LightGBM, dado que es un algoritmo recientemente creado, los estudios acerca de esto en el ámbito de predicción de abandono de clientes son más limitados y con fechas más recientes. Un ejemplo de ellos es el realizado por Arai et al. (2023), donde utilizaron datos de punto de venta desde el 01 de setiembre de 2009 al 31 de diciembre de 2021, con una tasa de abandono de 42%, ya que los clientes que abandonaron fueron 28 027 y los que no abandonaron 37 359. El conjunto de datos estaba conformado por 8 variables y solo se prueba el desempeño de un algoritmo en este caso, que es el de LightGBM, el cual obtuvo un área bajo la curva ROC de 0.837.

Otro caso un poco menos reciente es el desarrollado por Wang, Nguyen y Nguyen (2020). Para este estudio utilizaron un conjunto de datos de abandono en el sector de telecomunicaciones descargado de la plataforma Kaggle, el cual poseía 99 variables y 100 000 registros, junto con la indicación de si el cliente abandonó el servicio o no. Las variables incluidas en el archivo eran referentes a 6 grandes factores: facturación, cliente (demográficos), servicio, servicio al cliente (llamadas realizadas a servicio al cliente), operaciones y producto. El conjunto de datos se divide en 70% para entrenamiento y 30% para validación y se ponen a prueba los siguientes algoritmos: árbol de decisión, bosques aleatorios, regresión logística, Stochastic Gradient Descent (SGD), máquina de soporte vectorial, Naive Bayes, redes neuronales con perceptrón multicapa, k-vecinos más cercanos, XGBoost y LightGBM. En todos los casos se realiza ajuste de hiperparámetros y validación cruzada de 10 pliegues, y en este caso el algoritmo con un mejor valor ROC AUC fue LightGBM con un valor de 0.689, seguido por XGBoost con 0.685, y luego bosques aleatorios con un valor de 0.66. El estudio recalca que se prefiere LightGBM antes que

XGBoost debido a la eficiencia computacional de LightGBM, a pesar de que los resultados son muy similares a los de XGBoost.

En otro estudio desarrollado por Malyar, Robotyshyn y Sharkadi (2022), donde se utilizaron 4 algoritmos para comparación: árbol de decisión, bosques aleatorios, LightGBM y XGBoost, el que obtuvo mejor desempeño fue el algoritmo XGBoost con un valor de 0.83 en ROC AUC, seguido de LightGBM con un 0.81 de ROC AUC. El estudio se hizo con datos del sistema Prozorro, que es un sistema de contratación pública, donde se extrajeron datos de 140 000 clientes (a veces llamados proveedores) que participan en procesos de contratación entre 2016 y 2019 en más de 2 millones de licitaciones, en estos casos es común que el cliente abandone su participación anticipadamente, y si dicha rotación se predice con anticipación, las plataformas comerciales que cobran a los clientes por el derecho de participación, pueden ofrecer un descuento y así alentar el regreso del cliente.

Otro estudio desarrollado por Sina y Amiri (2022), se pusieron a prueba diversos algoritmos de clasificación con el fin de determinar cuáles presentaban mejores métricas de desempeño. Los algoritmos implementados fueron: árboles de decisión, bosques aleatorios, máquinas de soporte vectorial, regresión logítica, XGBoost, AdaBoost, LightGBM, y dos variaciones de LightGBM, una con una optimización bayesiana (BO-LightGBM) y otro con una optimización genética (GO- LightGBM). Se utilizaron 3 conjuntos de datos para evaluar los resultados y todos eran relacionados al abandono de clientes, uno de IBM Telecom (7 043 registros, 21 columnas y 26.5% de abandono), otro de clientes bancarios (10 000 registros, 14 columnas y 20.3% de abandono) y otro de Orange (3333 registros, 20 columnas y 14.5% de abandono), todos descargados de la plataforma Kaggle. En este caso dado que el estudio se enfocaba en la prueba del algoritmo LightGBM se realizó el ajuste de hiperparámetros, donde se usaron valores de máxima profundidad del árbol entre 1 y 10, tasa de aprendizaje entre 0.01 y 1. En los 3 conjuntos de datos el modelo que tuvo mejor exactitud fue el BO-LightGBM, donde para el estudio de clientes de banco tuvo 0.869, para el de IBM un 0.804, y para el de Orange 0.958. Entre las conclusiones mencionan que LightGBM tuvo mejor exactitud y mejores resultados en términos de métricas desequilibradas. Y que XGBoost tiene

valores de ROC AUC mejores solo en el conjunto de datos de IBM, pero que, sin embargo, LightGBM resultó superior en las restantes métricas de desempeño evaluadas.

Uso de algoritmo TabNet en aprendizaje de máquinas

Respecto al uso del algoritmo TabNet, al ser tan reciente no existen muchos estudios que lo hayan implementado. Sin embargo, en un estudio realizado para una cooperativa de crédito en Perú, se tenía como objetivo predecir la Tasa Interna de Retorno (TIR) de solicitudes de crédito, utilizando los algoritmos de redes neuronales con perceptrón multicapa XGBoost y TabNet, con el fin de servir como herramienta para el análisis de crédito de los analistas en esa cooperativa. El conjunto de datos estaba compuesto por 36 402 observaciones y se ajustaron los hiperparámetros de todos los modelos, sin embargo, para este caso, el algoritmo de XGBoost fue el que obtuvo mejores métricas de desempeño (Asencios et al., 2023).

En otro caso relacionado al sector de banca, con datos proporcionados por IEEE Computational Intelligence Society (CIS), se tenían datos de transacciones y si las mismas fueron fraudulentas o no para tarjetas de crédito. Se puso a prueba el algoritmo TabNet y se comparó su desempeño contra dos algoritmos conocidos que fueron Naive Bayes y XGBoost, en este caso el algoritmo de TabNet obtuvo una mayor exactitud y valor ROC AUC, 0.884 y 0.965 respectivamente, en comparación a los otros algoritmos puestos a prueba de XGBoost y Naive Bayes (Zhang et al., 2022).

Un estudio similar del 2022 con datos de electromiograma de gestos manuales agudos, se compararon diversas técnicas de aprendizaje de máquinas, entre ellas TabNet, k-vecinos más cercanos, bosques aleatorios y otros. En este caso, TabNet obtuvo mejor métrica de exactitud, con un valor de 0.999, seguido de un algoritmo más conocido que es el k-vecinos más cercano con una exactitud de 0.978, mostrando alguna evidencia sobre la exactitud de alto nivel que tienen las redes neuronales con enfoque tabular para conjuntos de datos pequeños (Sharma et al., 2022). Sin embargo, similar a lo que ocurrió en el estudio de Esteves & Mendes (2016) con el área bajo la curva ROC, métricas de exactitud tan cercanas

a 1 generalmente tienden a poner en duda los resultados de desempeño de un algoritmo, y aquí es donde entra el llamado sobreajuste adaptativo, el cual ocurre por la reutilización del conjunto de datos de prueba, que puede causar que el algoritmo se ajuste demasiado a los datos de entrenamiento y no a la distribución de los datos como tal, conduciendo a métricas de desempeño demasiado optimistas (Roelofs et al., 2019, pp. 2); provocando que una vez que el algoritmo recibe datos nuevos para predicción, su desempeño no es tan alto como se había previsto a la hora del entrenamiento y evaluación del modelo en los resultados.

Aunque no siempre TabNet suele tener métricas de desempeño casi perfectas cuando se trata de conjuntos de datos pequeños. En otro estudio realizado con 1 030 observaciones, en los que las variables independientes son las cantidades de los diferentes componentes del diseño de la mezcla de concreto y la variable de salida es la resistencia a la compresión en diferentes edades de curado, se compararon algunos algoritmos de clasificación convencionales como: regresión logística, árboles de decisión, regresión lineal, redes neuronales, contra 3 algoritmos de impulso: XGBoost y CatBoost y TabNet. En el estudio se encontró que los algoritmos de XGBoost y CatBoost muestran errores medios significativamente más bajos entre los valores predichos y las observaciones reales de la resistencia a la compresión que las arquitecturas contemporáneas, mientras que TabNet no es tan eficiente, y en este caso atribuyen la ineficiencia del algoritmo al tamaño del conjunto de datos (Pranav et al., 2022).

Sin embargo, Arik y Pfister (2021) probaron el algoritmo TabNet en diversos conjuntos de datos para demostrar el desempeño de su algoritmo, tanto en regresión como en clasificación. En el primero de ellos, el objetivo era clasificar el tipo de cobertura forestal a partir de variables cartográficas, en este caso TabNet supera a los algoritmos basados en árboles como XGBoost, LightGBM, CatBoost y AutoML Tables, con una exactitud de 0.969, seguido de los AutoML Tables con un valor de 0.949. Otro de las pruebas consistía en una regresión dinámica inversa de un brazo robótico antropomórfico, el autor dueño de los datos mostraba que es posible un rendimiento decente con un bosque aleatorio; en este caso el algoritmo de TabNet logra un error cuadrático medio casi una unidad menor, donde TabNet obtuvo 1.25 y el bosque aleatorio 2.39. En otro conjunto de datos, la tarea era distinguir entre

un proceso de bosones de Higgs y un fondo, y en este caso TabNet logró un rendimiento mayor con una exactitud de 0.7884, con un tamaño del modelo mucho menor que la red neuronal con entrenamiento evolutivo escaso, cuyo tamaño del modelo es mucho más pesado; este último obtuvo una exactitud de 0.785.

Finalmente, son muchas las investigaciones que se han realizado relacionadas a la creación de modelos para la predicción de abandono de clientes en el ámbito de telecomunicaciones. A manera de resumen, y en acuerdo a las investigaciones revisadas, se observa que el algoritmo de árboles de decisión mostraba métricas de desempeño más altas cuando se compara con otros algoritmos, en estudios realizados antes del 2016, tal y como se demuestra en los estudios de Nuñez (2015) y Dahiya & Bhatia (2015). Sin embargo, del 2016 en adelante, la tendencia en cuanto a obtención de mejores métricas de desempeño, como lo son la exactitud y el ROC AUC, para la predicción de abandono en el ámbito de telecomunicaciones, viene a darse con mayor frecuencia en algoritmos basados en árboles de decisión que utilizan ensambles de modelos, tales como: bosques aleatorios, ADA Boost, Gradient Boosting Classiffier (GBC), XGBoost y LightGBM. Y es en el uso de estos algoritmos de ensambles de árboles donde se muestran los mejores resultados en el ámbito de las telecomunicaciones mostrados en las investigaciones revisadas previamente, tales como: Esteves y Mendes (2016), Chouiekh (2017), Sabbeh (2018), Echeverri (2019), Hanif (2020), Wang, Nguyen y Nguyen (2020), Parmar y Serasiya (2021), Falla Arango (2021), Senthan et al (2021), Sina y Amiri (2022).

En definitiva, para este tipo de datos los ensambles de árboles tienden a tener un mejor desempeño según la bibliografía consultada. Por su parte, TabNet, al ser un algoritmo que se desarrolló recientemente, no se encontraron estudios directamente relacionados al abandono de clientes, pero si algunos donde se pone a prueba su desempeño frente a otros algoritmos de aprendizaje de máquinas, teniendo valores más altos en la métrica de exactitud en estudios como los de Zhang et al. (2022) y Arik & Pfister (2021), relacionados a la predicción de transacciones fraudulentes en tarjetas de crédito, a la predicción del tipo de cobertura forestal y a la predicción de distinción entre proceso de bosones de Higgs y fondos, obteniendo exactitudes de 0.965, 0.969 y 0.788 respectivamente. Sin embargo, el algoritmo no siempre

proporciona mejores métricas de desempeño cuando se compara con otros algoritmos de ensambles de árboles, tal y como sucedió en los estudios de Asencios et al. (2023), Pranav et al. (2022), donde el algoritmo XGBoost produjo mejores resultados en términos de métricas de desempeño en modelos de aprendizaje de máquinas.

Metodología

Descripción de los datos y variables

Para este estudio se utilizarán datos proporcionados por el Instituto Costarricense de Electricidad (ICE), los cuales se derivan de registros generados automáticamente sobre el uso de los servicios de SMS, voz y datos de sus clientes en el servicio de telefonía prepago. Los datos facilitados por el ICE son datos anonimizados y sintetizados, debido a que el ICE, en virtud de la Ley de Protección de la Persona frente al tratamiento de sus datos personales (Ley Nº 8968), tiene la obligación de proteger los datos personales que recibe o gestiona de sus clientes o usuarios de los servicios finales.

Los datos se basan en el uso de tres de los servicios de telecomunicaciones para clientes de todo Costa Rica de dicha empresa, en un periodo de tres meses consecutivos, que va de agosto a octubre de 2022. A las variables de los datos originales se les aplicó transformaciones para anonimizarlas y hacerlas sintéticas. Las variables sintéticas no tienen los valores en las unidades de medida de los datos reales, pero mantienen su estructura. Las transformaciones no son del conocimiento del investigador ni se hace indicación de cual variable corresponde a qué servicio, todo lo anterior, en virtud de garantizar la protección de los datos personales como se mencionó anteriormente.

Es importante resaltar que para efectos de esta investigación no se realizará exploración previa de las variables o se darán conclusiones acerca de los datos por servicio, ya que no es el objetivo de la investigación; además, el hecho de que las variables posean transformaciones y sean sintéticas, no las hace útiles para este fin.

El algoritmo que aplica las transformaciones respectivas a las variables, es verificado internamente por la empresa de telecomunicaciones, y se alimenta de los datos registrados automáticamente por el uso de los servicios de los clientes. La calidad se asegura ya que dichos registros son los mismos que se utilizan como insumo para procesos internos, como el de facturación, además, los cálculos agrupados por cliente para conformar las variables,

son verificados internamente por la empresa, con el fin de validar que éstos son correctos para el uso de la presente investigación.

La variable dependiente y que será la que se utilizará como respuesta en los diversos algoritmos de clasificación es la condición de abandono del cliente, donde para este caso indica si el cliente abandona o no abandona el servicio, por lo que es una respuesta binaria de sí y no. Respecto a las variables explicativas o independientes, estas son:

- Tiempo de vida del cliente en la empresa. Es una variable numérica.
- Tiempo de vida utilizando los servicios de telecomunicación. Es una variable numérica.
- Categoría de crédito, esta es una variable que determina el nivel de riesgo del cliente en términos del uso en servicios de telecomunicación. Es una variable factor.
- Promedio, máximo y mínimo en diferencia de días de consumo de los servicios de telecomunicación (para cada servicio). Todas son numéricas.
- Promedio y total de consumo en pago (para cada servicio). Todas son numéricas.
- Promedio y total de consumo de los servicios (para cada servicio). Todas son numéricas.
- Promedio, máximo y mínimo en diferencia de días de recarga. Todas son numéricas.
- Promedio y total de recargas. Todas son numéricas.

El conjunto de datos está constituido por 3 134 496 registros, de los cuales, un 98,4% corresponde a registros de clientes, que, en un mes específico, no abandonaron el servicio, mientras que el restante 1,6%, si abandonaron el servicio.

Preprocesamiento de variables

A nivel de preprocesamiento de variables independientes o explicativas de los modelos se realiza lo siguiente:

1. La variable de categoría de crédito se convierte a variables dummy, de manera que quedan en columnas de 1 y 0 según la cantidad de categorías de crédito de los clientes.
2. A todas las variables se les aplica una normalización con el fin de asegurar que las unidades de medida no afecten los resultados predictivos del modelo y la importancia de las variables en los algoritmos. La transformación aplicada hace que todas las variables tengan desviación estándar 1 y media 0, para ello se aplica la ecuación (1), donde x_{est} corresponde al resultado de la variable estandarizada, x a la variable original del archivo de datos, y μ_x, σ_x corresponden a la media y la desviación estándar de la variable original respectivamente.

$$x_{est} = \frac{x - \mu_x}{\sigma_x} \quad (1)$$

Este preprocesamiento se realiza en cada pliegue de la validación cruzada, tanto en los datos de entrenamiento, como en los datos de prueba que pasarán por el modelo para finalmente realizar la predicción de si el cliente abandona o no el servicio de telecomunicación.

Algoritmos de aprendizaje de máquinas

Como se mencionó anteriormente en los objetivos, se ajustarán modelos de clasificación utilizando tres distintos algoritmos de aprendizaje de máquinas, TabNet, XGBoost y LightGBM; el funcionamiento de estos algoritmos y los parámetros que se pueden calibrar, según los paquetes que se utilizarán, se detallan en las siguientes secciones.

XGBoost

El algoritmo de XGBoost (eXtreme Gradient Boosting) fue propuesto por el Dr. Tianqi Chen, quien trabajó en la Universidad de Washington en el 2014 (Zhang & Gong, 2020, p. 220992). El algoritmo se caracteriza por reducir el riesgo de sobreajuste agregando términos regulares y usando directamente el valor de la primera y segunda derivada de la función de

pérdida. Los pasos que realiza el algoritmo para ejecutarse según Chen y Guestrin (2016) se definen a continuación:

1. Se define la función de pérdida en la ecuación (2), la cual mide la diferencia entre el valor real y el valor a predecir de la variable objetivo.

$$L(\theta) = \sum_i l(y_i, \hat{y}_i) + \sum_t \Omega(f_t) \quad (2)$$

En la ecuación (2), $\hat{y}_i$ es la predicción de salida realizada, mientras que y_i es el valor real a predecir, f_t es el t-ésimo árbol. $\sum_i l(y_i, \hat{y}_i)$ es la suma de pérdidas por muestra, donde la función de pérdida l se puede personalizar de diversas formas. $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda||w||^2$ es el término regular, donde w es el valor de peso del nodo hoja en el t-ésimo árbol, T es el número de nodos hoja en el t-ésimo árbol, γ es el término regular de penalización del t-ésimo árbol y λ es el término de penalización regular del peso del nodo hoja, que actúa como un factor de suavizado para calcular la ganancia en el proceso de división de puntos, este es el factor penalizador para la complejidad del modelo. Y en ambas partes, los términos de penalización pueden evitar el sobreajuste.

2. Se transforma la función objetivo basado en el algoritmo de Boosting (desarrollado por Freund & Schapire, 1997) y en el de árbol de decisión. Esto se realiza debido a que la ecuación (2) tiene parámetros que no pueden ser optimizados usando métodos tradicionales de optimización en espacios Euclideanos, por lo que el modelo se entrena de forma aditiva por medio del algoritmo de Boosting. En el algoritmo de Boosting, el valor final predicho es la suma de las salidas de múltiples árboles como se muestra en la ecuación (3).

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \quad (3)$$

En la ecuación (3), $L^{(t)}$ es la función objetivo en el árbol t. Por otro lado, $\hat{y}_i^{(t-1)}$ es la suma de valores de salida en los árboles $t-1$, que constituyen el valor predicho del primer árbol $t-1$, mientras que $f_t(x_i)$ es la salida del t-ésimo árbol. La

suma de estas dos anteriores constituye el último valor predicho. El término regular al final de la ecuación se convierte en el término correspondiente del árbol actual t.

3. Se entrena el primer árbol para predecir valores de muestra.
4. Se usa la derivada de segundo orden de la regla de Taylor para expandir la función de pérdida $l(y_i, \hat{y_i}^{(t-1)})$ y entrenar el t-ésimo árbol tal y como se muestra en la ecuación (4).

$$L^{(t)} \sim \sum_{i=1}^{n} \left[l(y_i, \hat{y_i}^{(t-1)}) \ + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \qquad (4)$$

En la ecuación (4), $g_i = \partial_{\hat{y_i}^{(t-1)}} l\left(y_i, \hat{y_i}^{(t-1)}\right)$ es la derivada parcial de primer orden de la función de pérdida del árbol anterior. $h_i = \partial^2_{\hat{y_i}^{(t-1)}} l\left(y_i, \hat{y_i}^{(t-1)}\right)$ es la derivada parcial de segundo orden de la función de pérdida del árbol anterior.

5. Se ignora el término constante para simplificar la función objetivo. La primera parte de la función $\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)})$ es la función de pérdida de la ronda anterior de construcción de árboles. Es una constante conocida en la ronda actual de construcción de árboles y el valor no afecta el proceso de construcción de árboles en esta ronda. Por ello se puede ignorar para así simplificar la función objetivo como se muestra en la ecuación (5).

$$\tilde{L}^{(t)} \sim \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \qquad (5)$$

6. Se agregan los residuos de los nodos hojas. Suponiendo que el árbol tiene T nodos hoja y el conjunto de muestras en el j-ésimo nodo hoja se representa como $I_j = \{i | q(x_i) = j\}$. Basado en esto $\sum_{i=1}^{n} g_i f_t(x_i) = \sum_{j=1}^{T} \left(\sum_{i \in I_j} g_i \right) w_j$. $\sum_{i \in I_j} g_i$ es la suma de las primeras derivadas correspondientes a todas las muestras que caen en el nodo hoja j. Asimismo $\sum_{i=1}^{n} \frac{1}{2} h_i f_t^2(x_i)$ puede convertirse en $\sum_{i=1}^{n} \frac{1}{2} \sum_{i \in I_j} h_j w_j^2 \, h_i$. De manera que tras la conversión y fusión de los términos regulares la ecuación (5) se convierte en la ecuación (6).

$$\tilde{L}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$
$$= \sum_{j=1}^{T}\left[\left(\sum\nolimits_{i\in I_j} g_i\right)w_j + \frac{1}{2}\left(\sum\nolimits_{i\in I_j} h_i + \lambda\right)w_j^2\right] + \gamma T \quad (6)$$

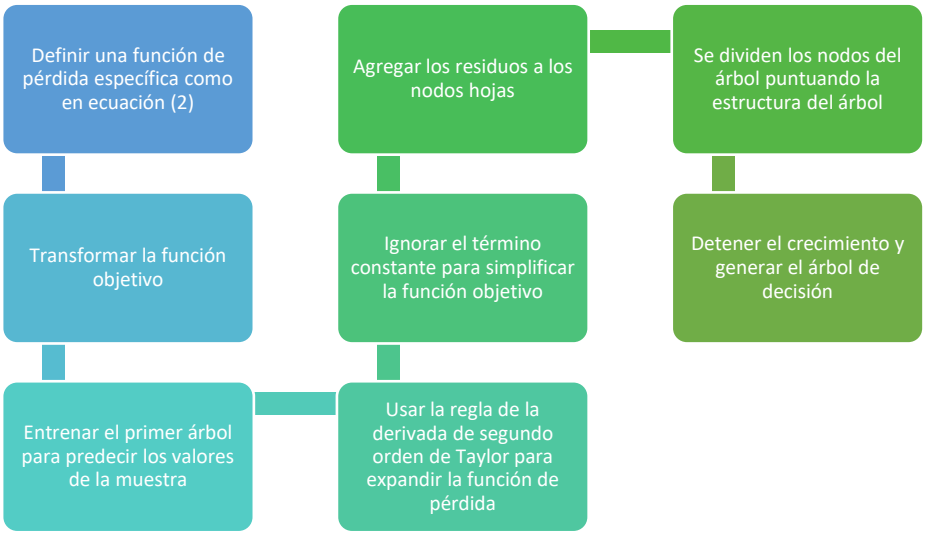
7. Se dividen los nodos del árbol puntuando la estructura del árbol usando la ecuación (7).

$$L_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda}\right] - \gamma \quad (7)$$

   Esto quiere decir, en términos más simples, que se puede dividir el árbol cuando el valor de la reducción de la función de pérdida es mayor que γ, que es el parámetro agregado al denominador mientras se calcula la ganancia para suavizarlo. γ puede ayudar a prevenir el sobreajuste, que se vuelve opcional en lugar de obligatorio.
8. Se detiene el crecimiento y se genera el árbol de decisión.

Estos pasos, además se resumen mejor en la Figura 1 donde se puede observar más simplificadamente el flujo que sigue el algoritmo para entrenar cada árbol de decisión en el algoritmo XGBoost.

Figura 1. Flujo del proceso ejecutado por algoritmo XGBoost



Fuente: tomada de "The Comparison of LightGBM and XGBoost Coupling Factor Analysis and Prediagnosis of Acute Liver Failure" de Zhang & Gong (2020)

LightGBM

LightGBM es un algoritmo potenciador de gradiente liviano, lanzado por Microsoft en el año 2017. Este algoritmo es ideal para problemas que requieren mucho tiempo de procesamiento y para muestras de datos de alta dimensión. Según Zhang y Gong (2020), el algoritmo es una versión mejorada de XGBoost por 4 aspectos nuevos que incorpora:

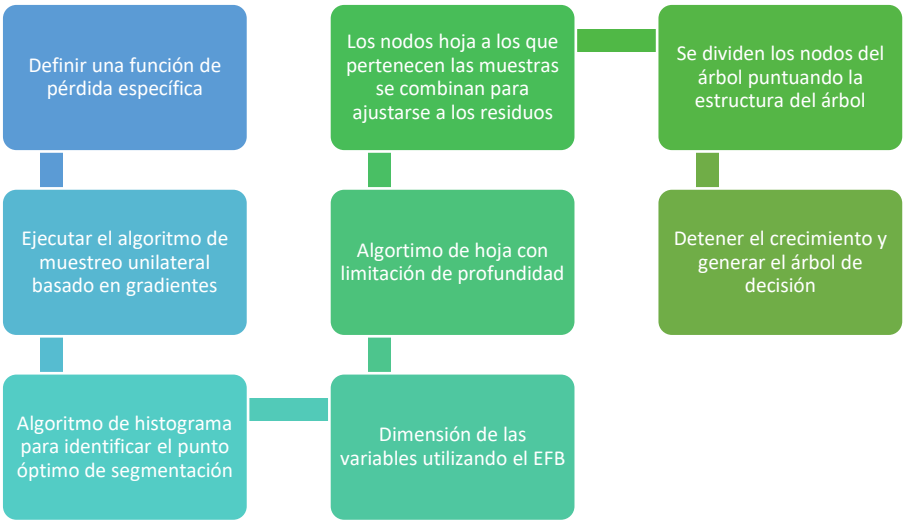
1. Incorpora el algoritmo GOSS (por sus siglas en inglés de Gradient-based One-Side Sampling) o muestreo unilateral basado en gradientes. El algoritmo logra un equilibrio entre el número de muestras y la precisión del árbol de decisión de LightGBM, ya que en el entrenamiento de los árboles se toma mayor importancia a muestras con mayores gradientes, que también tienen más influencia en la ganancia.
2. Utiliza un histograma para identificar el punto de división óptimo. El algoritmo discretiza valores propios sucesivos de punto flotante en k enteros y simultáneamente construye un histograma de ancho k. Según los valores discretizados, las estadísticas acumuladas en el histograma seleccionan como índice después de recorrer los datos,

luego se identifica el cálculo de las puntuaciones de división. El histograma sacrifica precisión por velocidad, ya que reemplaza el histograma tradicional preordenado.

3. Reduce la dimensión de las funciones hasta cierto punto EFB (por sus siglas en inglés de Exclusive Feature Bundling). Para un espacio de variables disperso, es posible reducir el número de variables válidas uniendo variables mutuamente excluyentes para formar nuevas variables.
4. Utiliza un algoritmo de hoja con limitación de profundidad en lugar del tradicional de nivel, lo que mejora en términos de precisión y sobreajuste.

Los autores también mencionan que LightGBM, a diferencia de XGBoost, usa una estrategia de hoja en lugar de una de nivel. Esto quiere decir que XGBoost divide indiscriminadamente los nodos de capa en capa, y divide nodos con ganancias muy pequeñas, lo que provoca un desperdicio de recursos, mientras que LightGBM selecciona los nodos hoja con mayor ganancia de división, lo que a su vez podría conducir fácilmente a un sobreajuste y el árbol podría crecer mucho más de lo esperado, por lo que es importante controlar la profundidad del árbol en el algoritmo LightGBM. Los pasos que sigue el algoritmo LightGBM para ejecutarse se resumen al igual que el algoritmo XGBoost en la Figura 2.

Figura 2. Flujo del proceso ejecutado por algoritmo LightGBM



Fuente: tomada de "The Comparison of LightGBM and XGBoost Coupling Factor Analysis and Prediagnosis of Acute Liver Failure" de Zhang & Gong (2020)

A nivel de hiperparámetros a calibrar, cada uno de los dos algoritmos, XGBoost y LightGBM, presentan hiperparámetros característicos de su proceso de ejecución, sin embargo, para efectos de esta investigación se calibrarán solo 6 hiperparámetros de los que están presentes en ambos algoritmos, y que están definidos como hiperparámetros generales en el paquete parsnip del software R. Según Kuhn (2023), los hiperparámetros coincidentes entre los algoritmos y que están disponibles para definición de valores son:

- mtry: es el número o proporción de predictores que serán seleccionados aleatoriamente en cada división al crear los modelos de árboles.
- trees: el número de árboles contenidos en el ensamble.
- min_n: el número mínimo de observaciones en un nodo que se requieren para que el nodo se divida aún más. Suelen probarse valores entre 1 y 10, pero si los datos tienen mucho ruido, para evitar el sobreajuste, se suelen usar valores mayores.
- tree_depth: la máxima profundidad del árbol. Este valor está comúnmente situado entre 3 y 10. Es importante controlar este parámetro en el caso de LightGBM por su tendencia al sobreajuste.

- learn_rate: representa la tasa a la que el algoritmo de impulso se adapta de iteración a iteración. Los valores usuales suelen situarse entre 0.01 y 0.3.
- loss_reduction: número que representa la reducción en la función de pérdida requerida para realizar divisiones adicionales. Los valores suelen estar en el mismo rango de learn_rate.
- sample_size: número o proporción que es la cantidad de datos que se expone el proceso de ajuste. En XGBoost el muestreo se realiza en cada iteración.
- stop_iter: el número de iteraciones sin mejora antes de detenerse.

Más adelante se mencionan cuáles de los anteriores hiperparámetros fueron seleccionados para su respectiva calibración con el fin de obtener la mejor combinación de los mismos y así obtener métricas de desempeño buenas en la predicción de abandono de clientes del servicio de telecomunicación.

TabNet

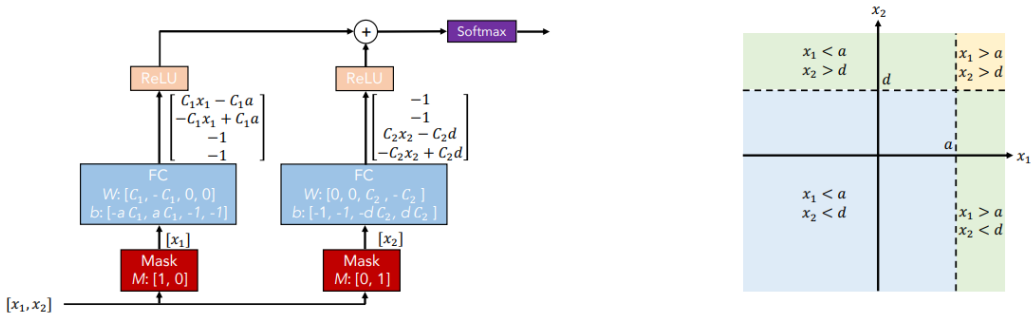
El último algoritmo que se utilizará para la predicción de abandono de clientes será el TabNet. Este una arquitectura de redes neuronales profundas para datos tabulares, que se entrena utilizando una optimización basada en el descenso de gradiente. El algoritmo opera bajo un enfoque iterativo, en el cual utiliza la atención secuencial para elegir las variables más importantes en cada paso de decisión, y aplicando a su vez, máscaras de atenuación aleatorias a las variables seleccionadas, esto con el fin de que evitar la dependencia excesiva de las variables y fomentar la generalización (Arik & Pfister, 2021, p. 1).

La Figura 3 muestra en la imagen izquierda una clasificación basada en árboles con bloques de redes neuronales convencionales. Las variables se seleccionan mediante máscaras dispersas multiplicativas en las entradas, donde posteriormente, las que se seleccionan, se transforman linealmente y después de sumar el sesgo (para representar los límites), ReLU se encarga de seleccionar las regiones poniendo cero a las mismas. La agregación de varias regiones se basa en la suma, y a medida que $C_1 y\, C_2$ se hacen más grandes, el límite de decisión se hace más nítido.

La selección de variables es crucial para obtener dichos límites de decisión en forma de hiperplano, que se puede generalizar en una combinación lineal de variables donde los coeficientes representan la proporción de cada variable. TabNet se aprovecha de las bondades de los métodos basados en árboles de decisión para la selección de sus variables, ya que:

1. Utiliza una selección de variables escasa por instancia aprendida de los datos.
2. Construye una arquitectura secuencial de múltiples pasos, donde cada paso contribuye a una parte de la decisión basada en las variables seleccionadas.
3. El algoritmo mejora la capacidad de aprendizaje a través del procesamiento no lineal de las variables seleccionadas.
4. Imita el ensamblaje a través de dimensiones más altas y más pasos (Arik & Pfister, 2021, p. 3).

Figura 3. Ilustración de clasificación basada en un enfoque de árbol de decisión usando bloques de redes neuronales profundas convencionales y su hiperplano de decisión.



Fuente: tomada de "TabNet: Attentive Interpretable Tabular Learning" de Arik & Fister (2021)

Para hacer dicha selección de variables, TabNet utiliza un codificador, el cual está compuesto por un transformador de variables, un transformador atencional y el enmascaramiento de variables. El codificador utiliza las variables numéricas sin procesar y considera el mapeo de variables categóricas con incrustaciones entrenables. No se considera ninguna normalización global de variables, sino que simplemente se aplica la normalización por lotes (BN), donde se pasan las mismas variables D-dimensionales $f \in R^{BxD}$ para cada paso de decisión, donde B el tamaño del lote. El codificador de TabNet se basa en un proceso secuencial multi pasos con N_{pasos} pasos de decisión. El i^{th} paso ingresa la información

procesada del $(i-1)^{th}$ paso para decidir cuales variables se usarán y genera la representación de la variable procesada para agregarla a la decisión general. La idea del transformador atencional, de arriba hacia abajo en forma secuencial, está inspirada en sus aplicaciones en el procesamiento de datos visuales y de texto, así como el aprendizaje por refuerzo, mientras se busca un pequeño subconjunto de información relevante en datos de entrada de gran dimensión.

Para dicha selección de variables, se utiliza una máscara de aprendizaje $M[i] \in R^{BxD}$ para la selección de las variables más destacadas. Por medio de una selección dispersa de variables más relevantes, la capacidad de aprendizaje de un paso a otro no se desperdicia en pasos irrelevantes, y el modelo se vuelve más eficiente en términos de los parámetros. El enmascaramiento de las variables es multiplicativo $M[i] \cdot f$. Se utiliza un transformador atencional (ver (d) en la Figura 4) para obtener las máscaras usando las variables procesadas el paso anterior $a[i-1]$: $M[i] = sparsemax(P[i-1] \cdot h_i\ (a[i-1]))$.

La normalización Sparsemax fomenta la dispersión al mapear la proyección Euclideana sobre el simplex probabilístico, que se observa que tiene un rendimiento superior y se alinea con el objetivo de selección de variables dispersas para explicabilidad. El objetivo del transformador atencional es aprender la importancia de las variables y decidir cuales activar o desactivar para los distintos pasos; y la normalización Sparsemax se utiliza en dicho proceso, ya que permite priorizar la activación de un subconjunto pequeño de variables (dispersión) en cada paso, lo que contribuye a la explicabilidad e interpretabilidad del modelo. $\sum_{j=1}^{D} M[i]_{b,j} = 1$. h_i es una función entrenable, que en (d) de la Figura 4 utiliza una capa FC (totalmente conectada), seguida de una BN (normalización por lotes). Por su parte $P[i]$ es el termino de escala a priori, que determina cuánto ha sido utilizada una variable particular anteriormente: $P[i] = \prod_{j=1}^{i}(\gamma - M[j])$, donde γ es un parámetro de relajación. Cuando $\gamma = 1$, la variable aplica para ser usada solo en un paso de decisión y cuando γ aumenta, se crea mayor flexibilidad para usar una variable en múltiples pasos de decisión. $P[0]$ se inicializa con unos, 1^{BxD} , sin ninguna previa sobre las variables enmascaradas. Si algunas variables no se utilizan, las entradas $P[0]$ correspondientes, se convierten en 0 para ayudar al aprendizaje del modelo (Arik & Pfister, 2021, p. 3).

Para controlar aún más la dispersión de las variables seleccionadas, se usa la regularización de la dispersión en forma de entropía $L_{sparse} = \sum_{i=1}^{N_{steps}} \sum_{b=1}^{B} \sum_{j=1}^{D} \frac{-M_{b,j}[i]\log\left(M_{b,j}[i]+\epsilon\right)}{N_{steps} \cdot B}$, donde ϵ es un número pequeño para la estabilidad numérica. Se agrega la regularización de la dispersión a la pérdida total, con un coeficiente λ_{sparse}. La dispersión proporciona un sesgo inductivo favorable para los conjuntos de datos donde la mayoría de las variables son redundantes. En síntesis, esta entropía se utiliza como una medida de dispersión de las activaciones, ya que el modelo busca activaciones con alta entropía, lo que indica una alta diversidad y se traduce en la presencia de múltiples variables activadas, que ayuda a resaltar las variables más importantes para la toma de decisiones en cada paso del codificador más eficientemente.

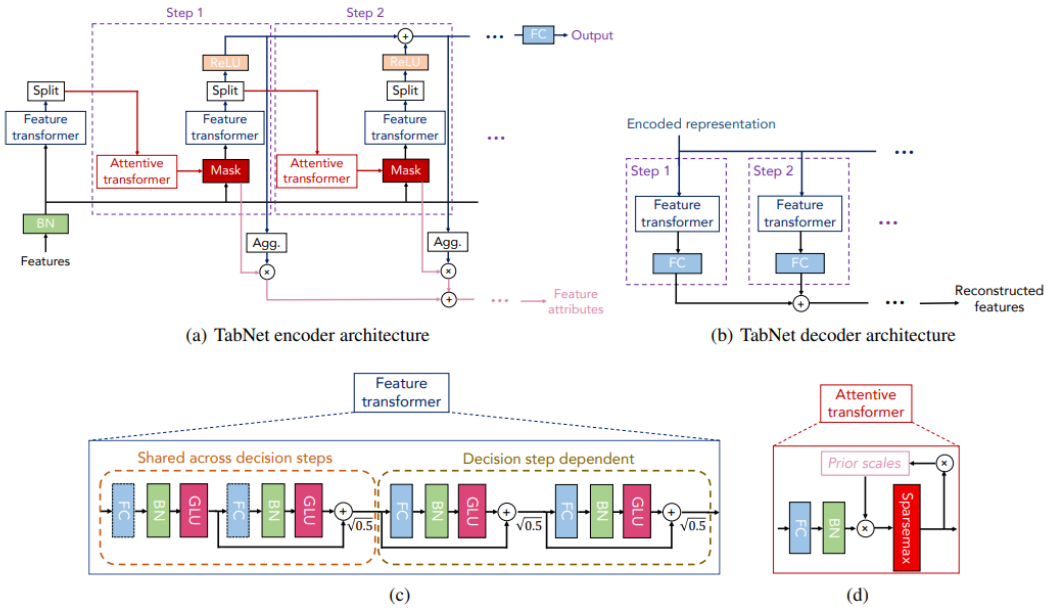
Las variables filtradas se procesan usando el transformador de variables que se muestra en (c) de la Figura 4, y luego se dividen para la salida del paso de decisión e información para el paso subsecuente, $\left[d[i], a[i]\right] = f_i(M[i] \cdot f)$, donde $d[i] \in R^{BxN_d}$ y $a[i] \in R^{BxN_a}$. Para parámetros eficientes y aprendizaje robusto de gran capacidad, el transformador de variables debe comprender capas que se comparten en todos los pasos de decisión (pues las mismas variables son entrada en los distintos pasos de decisión), así como de capas dependientes de pasos de decisión. El ejemplo (c) de la Figura 4 muestra la implementación como concatenación de dos capas compartidas y dos capas dependientes de pasos de decisión. Cada capa FC (totalmente conectada) es seguida de una BN y luego de una unidad lineal cerrada (GLU) no lineal, eventualmente conectada a una conexión residual normalizada con normalización. La normalización con $\sqrt{0.5}$ ayuda a estabilizar el aprendizaje asegurando que la variancia en toda la red no cambie drásticamente.

Para un aprendizaje más rápido, se usan lotes grandes con BN. Por lo tanto, excepto el que se aplica a las variables de entrada, se utiliza BN en modo “fantasma”, usando un B_V de tamaño de lote virtual y un impulso m_B. Para las variables de entrada, se observa el beneficio del promedio de baja variancia, por lo que se evita el BN “fantasma”. Ahora finalmente inspirándose en la agregación similar a un árbol de decisión, tal y como se

mostraba en la Figura 1, se construye la decisión general incrustando $d_{out} = \sum_{i=1}^{N_{steps}} ReLU(d[i])$. Se aplica un mapeo lineal $W_{final} d_{out}$ para obtener el mapeo de salida.

A nivel de interpretabilidad, las máscaras de selección de variables pueden arrojar luz sobre las variables seleccionadas en cada paso, y aunque en cada paso de decisión se emplea un procesamiento no lineal, los resultados se combinan en forma lineal más adelante. El objetivo es cuantificar la importancia de una variable agregada, además del análisis de cada paso de decisión. Combinar las máscaras en los diferentes pasos requiere un coeficiente que pueda pesar la importancia relativa de cada paso de decisión. Simplemente se establece $n_b[i] = \sum_{c=1}^{N_d} ReLU\left(d_{b,c}[i]\right)$ para denotar la contribución de la decisión agregada en el i-ésimo paso para la b-ésima muestra. Además, si $d_{b,c}[i] < 0$, todas las variables del paso i tienen una contribución de 0 en el paso de decisión. Mientras mas incremente este valor, juega un valor más importante en la combinación lineal general (Arik & Pfister, 2021, p. 4).

Figura 4. Arquitectura de ejecución para algoritmo TabNet.



Fuente: tomada de "TabNet: Attentive Interpretable Tabular Learning" de Arik & Fister (2021)

De esta arquitectura de TabNet existen múltiples hiperparámetros que se definen para la ejecución del modelo, en el paquete torch del software R, que se utilizará para la ejecución del algoritmo, Falbel (2023) define los siguientes hiperparámetros a modificar para ejecutar el algoritmo:

- epochs: corresponde al número de épocas de entrenamiento. Las épocas es el número de veces que el algoritmo funciona en todo el conjunto de datos de entrenamiento, cada época consta de múltiples iteraciones o pasos, donde el modelo ajusta sus parámetros utilizando un conjunto de datos de entrenamiento y actualiza sus pesos mediante el algoritmo de optimización (Brownlee, 2018, p. 3). El valor de este parámetro suele ser grande, a menudo cientos o miles, lo que permite que el algoritmo se ejecute hasta que el error del modelo se haya minimizado, existen diversos tutoriales donde se configura con valores como 10, 100, 500, 1000 y más (Brownlee, 2018, p. 4).
- penalty: es el coeficiente adicional de pérdida de dispersión, cuanto mayor sea el coeficiente, más disperso será su modelo en términos de selección de variables. Dependiendo de la complejidad, reducir este número podría ser beneficioso. Se suelen utilizar valores entre 0.01 y hasta 1.
- batch_size: es el número de ejemplos por lote, el cual se recomienda que sea grande y el valor predeterminado es 1024^2. El lote es un hiperparámetro que define la cantidad de muestras con las que se debe trabajar antes de actualizar los parámetros internos del modelo (Brownlee, 2018, p. 3).
- learn_rate: es la tasa de aprendizaje inicial para el optimizador. Cuanto menor sea la tasa el ajuste será más lento, pero más estable, por lo que se debe configurar de tal forma que se logre un balance entre velocidad de convergencia y estabilidad del modelo. Los valores comúnmente usados suelen rondar entre 0.001 hasta 0.1.
- decision_width: ancho de la capa de predicción de decisiones. Valores altos proporcionan mayor capacidad del modelo, pero se entra en riesgo de sobre ajuste, y normalmente varía entre 8 y 64.

- attention_width: el ancho de incrustación de atención para cada máscara. El valor predeterminado es 8.
- num_steps: número de pasos a realizar en la arquitectura de TabNet, usalmente está entre 3 y 10.
- feature_reusage: corresponde al coeficiente de reutilización de variables en las máscaras. Un valor cercano a 1 hace que la selección de máscaras este menos correlacionada entre las capas. Por lo general los valores oscilan entre 1 y 2.
- virtual_batch_size: es el tamaño de los mini lotes utilizados por la normalización por lotes en modo "fantasma". El valor predeterminado es 256^2.
- num_independent: número de capas independientes de Unidades Lineales Controladas (GLU por sus siglas en inglés) en cada paso. Los valores habituales rondan entre 1 y 5.
- num_shared: número de capas compartidas de Unidades Lineales Controladas (GLU por sus siglas en inglés) en cada paso. Los valores habituales rondan entre 1 y 5.
- momentum: es el momento para la normalización por lotes, normalmente oscila entre 0.01 y 0.4, pero su valor predeterminado es 0.02.

En la siguiente sección se detalla cuáles de los anteriores hiperparámetros se seleccionaron para realizar la calibración con el fin de obtener la combinación que proporcione las mejores métricas de desempeño en la predicción de abandono de clientes del servicio de telecomunicación.

Ajuste de hiperparámetros

Con el fin de que las combinaciones entre los hiperparámetros no sean demasiado distantes debido a los rangos en que se permite que estos varíen y la cantidad de combinaciones que se realizarán por tiempo y capacidad de procesamiento, se realizará la calibración solamente de 6 hiperparámetros en cada uno de los algoritmos que se van a utilizar, y los demás se dejarán con los valores predeterminados que establezca el paquete en el software utilizado de cada algoritmo. Para esta calibración, se construyen combinaciones

aleatorias de los 6 hiperparámetros con rangos específicos definidos; en los Cuadros 1 y 2 se muestran los hiperparámetros que serán calibrados en cada uno de los algoritmos, así como los rangos que se permitirán para los valores aleatorios de dichos hiperparámetros. Estos rangos son importantes ya que los valores de los hiperparámetros, al seleccionarse aleatoriamente, deberán estar en el rango que se define para cada uno de ellos.

Se construyen 70 combinaciones aleatorias de los 6 hiperparámetros en los rangos definidos en el Cuadro 1 para los algoritmos de XGBoost y LightGBM. Para el caso de TabNet, debido a que el tiempo de procesamiento del algoritmo es muy grande y la cantidad de épocas determina también que tanto tarda en ajustarse un modelo, solo se realizarán 50 combinaciones aleatorias de los hiperparámetros. Los rangos de los hiperparámetros para el algoritmo TabNet se detallan en el Cuadro 2.

Cuadro 1. Hiperparámetros y rangos de variación para calibración de hiperparámetros en algoritmos XGBoost y LightGBM.

Hiperparámetro	Rango de variación
mtry	1 – 33
trees	100 – 1500
tree_depth	3 – 10
learn_rate	0.001 – 0.4
loss_reduction	0.001 – 0.4
min_n	1 – 30

En el caso de los algoritmos de XGBoost y LightGBM el hiperparámetro de número de predictores aleatorios (mtry) se define entre 1 y 33, debido a que 33 es el número final de variables independientes que ingresan al modelo, una vez realizado el preprocesamiento de las variables, por lo que es el máximo que se podría seleccionar. La cantidad de árboles (trees) se define entre 100 y 1500 para evaluar combinaciones con pocos o con mucha cantidad de árboles en el ensamble. La profundidad del árbol (tree_depth), el paquete parsnip recomienda probar valores en el rango de 3 a 15, sin embargo, para evitar árboles demasiado profundos se limita entre 3 y 10, también debido a que Sina y Amiri (2022) utilizan el rango un rango de entre 1 y 10. En cuanto a la tasa de aprendizaje (learn_rate) y la reducción de la función

de pérdida (loss_reduction) se amplía un poco el rango sugerido por el paquete que es entre 0.01 y 0.3 a 0.001 a 0.4, con el fin de probar más valores. Finalmente, para el número mínimo de observaciones para que un nodo se divida más (min_n) se establece entre 1 y 30, la sugerencia del paquete utilizado es de 2 a 40. Los demás parámetros modificables de cada uno de los algoritmos se dejarán con sus valores predeterminados a la hora de realizar los ajustes en cada una de las 70 combinaciones.

Cuadro 2. Hiperparámetros y rangos de variación para calibración de hiperparámetros en algoritmo TabNet.

Hiperparámetro	Rango de variación
epochs	10 – 350
penalty	0.001 – 1
learn_rate	0.001 – 0.4
num_steps	3 – 10
decision_width	8 – 64
attention_width	8 – 64

Para el algoritmo TabNet según se detalla en el Cuadro 2, se ajustará el número de épocas (epochs), el cual se define entre 10 y 350 debido a que a pesar de que se suelen utilizar números hasta de 1000, los modelos ajustados con más de 450 épocas para estos datos, inicialmente daban problemas de ajuste, por lo que finalmente se decide restringir a este rango. El coeficiente adicional de pérdida de dispersión (penalty) se decide disminuir un poco el rango recomendado que era entre 0.01 y 1 a entre 0.001 y 1, debido a que se mencionaba que, para modelos complejos, tener un número bajo en este parámetro podría resultar beneficioso. La tasa de aprendizaje (learn_rate) se define el mismo rango que en el caso de LightGBM y XGBoost, entre 0.001 y 0.4. Y finalmente el número de pasos (num_steps), ancho de la capa de predicción de decisiones (decision_width) y el ancho de incrustación de atención para cada máscara (attention_width) se definen en los rangos recomendados por el paquete ya que no se encuentran referencias de otros estudios sobre que valores o rangos es común utilizar en este algoritmo.

Es importante recordar que para cada una de las 70 (en el caso de XGBoost y LightGBM) y 50 (en el caso de TabNet) combinaciones de hiperparámetros, se debe entrenar un modelo y evaluar los resultados en los datos de prueba; esto multiplicado por la cantidad de pliegues que se establecen en la validación cruzada, aumenta más el tiempo de procesamiento de los algoritmos, por lo que para disminuir los tiempos de calibración de hiperparámetros, se paralelizará el proceso utilizando el paquete future (Bengtsson, 2021) con un plan multisesión de 7 sesiones paralelas.

Validación cruzada

La validación cruzada es una técnica utilizada para evaluar los resultados del modelado y garantizar que son independientes de la partición entre datos de entrenamiento y datos prueba (Rojas, 2022). En este caso se realizará para cada una de las combinaciones de hiperparámetros anteriormente descritas una validación cruzada de 10 pliegues, con una repetición en cada pliegue, con el fin de finalmente obtener las métricas de desempeño promedio de los 10 pliegues en cada una de las combinaciones aleatorias de hiperparámetros seleccionadas. Se utilizan 10 pliegues ya que autores como Wang, Nguyen & Nguyen (2020), Hanif (2020), Sabbeh (2018), Esteves & Mendes (2016) utilizaron esta cantidad de pliegues, y cada uno de los pliegues se divide en 80% para datos de entrenamiento y 20% para datos de prueba tal y como lo hacen Wang, Nguyen & Nguyen (2020). Los pliegues se conforman a partir de todo el conjunto de datos que tiene 3 134 496 de registros, pues lo que se evaluará en esta investigación son las métricas de desempeño resultantes a través de la validación cruzada y no del ajuste final de un modelo con un algoritmo específico.

Desbalanceo de clases

Debido a que para el conjunto de datos particular de este estudio el desbalanceo es alto, ya que un 98,4% de clientes no abandonaron el servicio de telecomunicación, y un 1,6% de clientes si lo abandonaron, se hace necesario implementar alguna técnica de submuestreo

o sobre muestreo para evitar que el modelo no obtenga resultados deficientes sobre la clase minoritaria, debido a su poca representación. En este caso la clase minoritaria son los clientes que abandonan, y en particular esta es la población que más interesa detectar, con el fin de establecer estrategias de retención para estos clientes.

En la revisión bibliográfica realizada Qureshi et al. (2013) utilizaron submuestreo para combatir este problema de desbalanceo, y Esteves & Mendes (2016) utilizaron el algoritmo SMOTE, que es una técnica que combina el submuestreo de la clase mayoritaria y el sobre muestreo de la clase minoritaria. Sin embargo, cada una tiene sus ventajas y desventajas. Liu, Wu & Zhou (2008) mencionan que una de las principales críticas que se le realiza al submuestreo es que ignora muchos ejemplos de la clase mayoritaria, ya que este tiene como objetivo muestrear casos de la clase mayoritaria, con el fin de balancear el número a que sea similar a la clase minoritaria. Y por el otro lado el sobremuestreo intenta hacer que la clase minoritaria crezca hasta parecerse más a la mayoritaria, pero al ser más datos esto aumenta el tiempo de entrenamiento de los modelos, y duplica mucha información, lo cual más bien se ahorra en el caso del submuestreo.

Para este caso, debido a que se busca disminuir tiempos de procesamiento debido a que el algoritmo TabNet toma bastante más tiempo ejecutarse para cada combinación de hiperparámetros, se decide aplicar una técnica de submuestreo. Se utilizará la técnica incluida en el paquete themis de Hvitfeldt (2023), la cual se denomina: submuestreo mayoritario aleatorio con reemplazo. Esta técnica permite igualar la aparición de clases en un nivel de factor específico, que en este caso se define en 1.5. Si este nivel se define en 1 significaría que la clase mayoritaria se debe igualar a la clase minoritaria (misma cantidad de registros en ambas clases), y si, por ejemplo, tuviese un valor de 2, esto significaría que la clase mayoritaria puede tener hasta 2 veces (como máximo y aproximadamente) la cantidad de casos que la clase minoritaria; se define en 1.5 de manera que la clase minoritaria quede aproximadamente en un 40% de representación del conjunto de datos de entrenamiento y el restante 60% sea de la clase mayoritaria. Es importante recalcar que este submuestreo se aplica únicamente al conjunto de datos de entrenamiento de cada pliegue de la validación

cruzada, y no aplicará para los datos de prueba con que se obtienen las métricas en cada pliegue.

Indicadores de desempeño

Existen diversos tipos de indicadores de desempeño para evaluar un modelo de clasificación que utilice algoritmos de aprendizaje de máquinas. Los que se utilizarán en la presente investigación, para los tres algoritmos anteriormente descritos, de acuerdo al segundo objetivo específico, son cuatro, que se detallan a continuación:

- Exactitud: es la proporción de clasificaciones predichas de manera correcta sobre el total de observaciones que se sometieron a clasificación. Esta métrica es la más utilizada debido a su facilidad de cálculo y comprensión para evaluar la efectividad general de los algoritmos.
- Sensibilidad: es la proporción de casos positivos clasificados correctamente.
- Especificidad: es la proporción de casos negativos clasificados correctamente.
- ROC AUC: es el valor del área bajo la curva ROC que corresponde a la probabilidad de clasificar correctamente una clase positiva al azar más que una negativa escogida al azar (Borja et al., 2020, pp. 187).

Dichos indicadores de desempeño se obtendrán para cada combinación de parámetros en cada uno de los 10 pliegues, utilizando los datos de prueba definidos en cada apliegue específico, y finalmente para cada una de las combinaciones de parámetros, se obtendrá el promedio de las métricas en los 10 pliegues.

Evaluación de tiempos de ejecución

Para medir los tiempos de ejecución de los modelos, lo que se realizará para evitar más consumo de recursos y de memoria en la paralelización del ajuste de hiperparámetros, es que una vez que se corra el ajuste de hiperparámetros con todas las combinaciones y en

todos los pliegues de validación cruzada, según el algoritmo, se determinará la mejor combinación de hiperparámetros según cada una de las 4 métricas de desempeño, y se ajustará un modelo, en un solo pliegue, con el fin de tener una estimación del tiempo de ajuste total del modelo, en cada uno de los 3 algoritmos, con las mejores combinaciones de hiperparámetros, en cada una de las 4 métricas de desempeño. Esto pues de medirse un tiempo promedio de los 10 pliegues durante el ajuste de los hiperparámetros, y guardar esta información, se consumiría más recursos de memoria y tiempo, que son bastante limitados para esta investigación. Para realizar el ajuste y evaluación de resultados en los datos de prueba de estos modelos, no se aplicó una paralelización con plan multisesión, esto con el fin de evaluar la duración de los modelos corriendo en una única sesión utilizando el CPU de la máquina.

Software, hardware y paquetes utilizados

El software que se utilizará para construir, ajustar y medir el tiempo de todos los modelos será R (R Core Team, 2023), con el IDE RStudio (Posit team, 2023) en su versión 2023.09.0+463. Los paquetes que se utilizarán del lenguaje de programación R, según el algoritmo de cada modelo, se detallan a continuación:

- TabNet: torch (Falbel & Luraschi, 2023).
- XGBoost: xgboost (Chen et al., 2023).
- LightGBM: lightgbm (Shi et al., 2023).

Adicionalmente para el modelado se utilizarán los paquetes de la biblioteca de tidymodels (Kuhn et al., 2020) los cuales son: rsample, parsnip, recipes, tune, yardstick, workflows; para pre-procesamiento, definición del modelo, calibración de parámetros, obtención de métricas de desempeño, entre otros. Y a su vez se utilizará la biblioteca tidyverse (Wickham et al., 2019) (ggplot2, dplyr, tibble, tidyr, purrr) para preprocesamiento, transformación de los datos y visualizaciones, el paquete themis (Hvitfeldt, 2023) para

aplicar submuestreo en los datos y el paquete future (Bengtsson, 2021) para paralelizar la calibración de hiperparámetros.

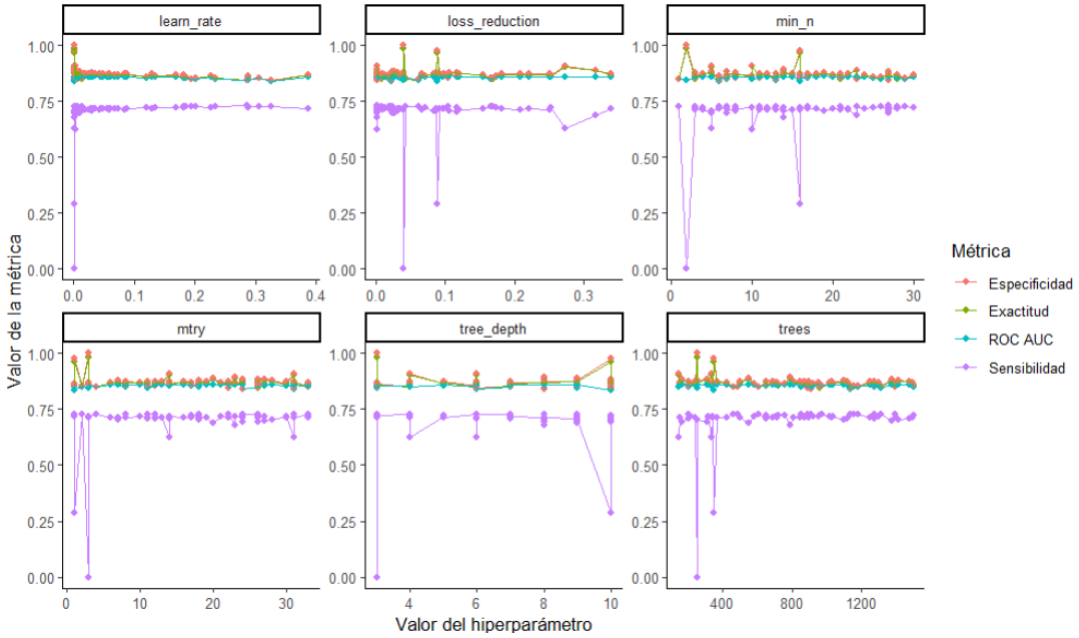
Tanto el ajuste de hiperparámetros, como la medición de los tiempos de ejecución de los algoritmos en la mejor combinación de hiperparámetros obtenida para cada métrica de desempeño en un pliegue, se realizarán en la computadora portátil del investigador, la cual es una Lenovo IdeaPad S340, que tiene un procesador Intel Core i7, con 8 núcleos y 20 GB de memoria RAM, que corre bajo el sistema operativo Windows 11 Pro; esto servirá para que los futuros lectores se den una idea de los recursos disponibles y el tiempo de ejecución de los modelos en el conjunto de datos, que en este caso está conformado por 3 134 496 registros.

Resultados

Patrones de métricas de desempeño en los valores de los hiperparámetros

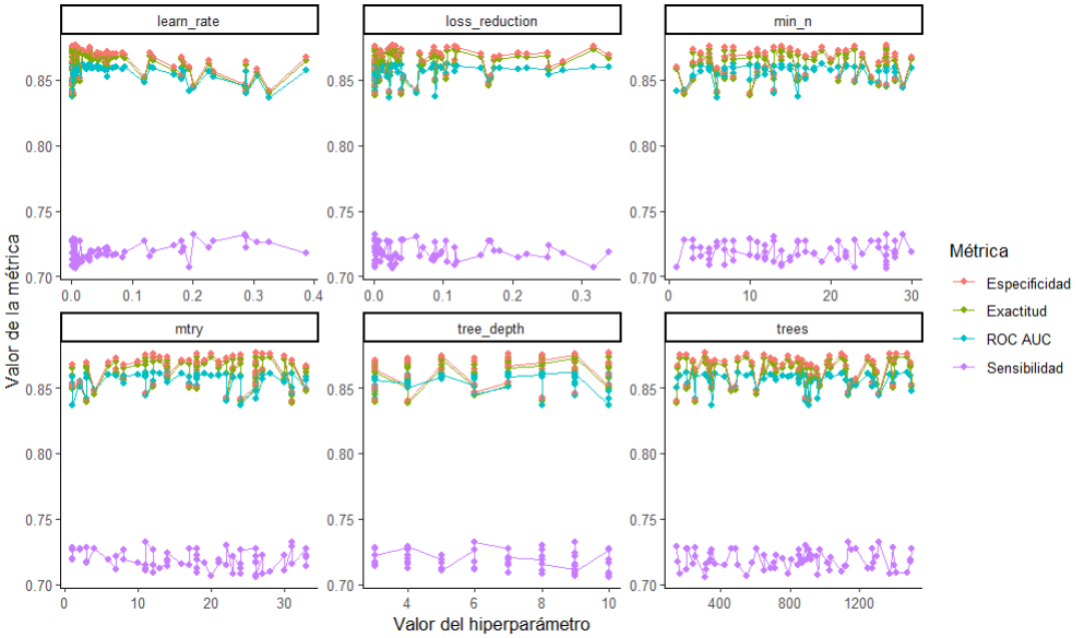
Una vez ajustados los 70 modelos utilizando el algoritmo LightGBM en los 10 pliegues de validación cruzada, la Figura 5 muestra el comportamiento de las métricas de desempeño promedio de los 10 pliegues según los valores de los parámetros que se probaron, esto con el fin de evidencia si en algún hiperparámetro existe un rango específico que proporcione mejores métricas de desempeño, a pesar de que se sabe que el desempeño del modelo depende de los 6 hiperparámetros y no solo de uno, se muestra que la tasa de aprendizaje (learn_rate) con valores por debajo de 0.05 tiende a dar mejores métricas a excepción de la sensibilidad, que más bien disminuye con valores muy bajos de tasa de aprendizaje, y que por el contrario hace que aumente la especificidad. Sin embargo, en los restantes hiperparámetros se muestra que cuando los valores de especificidad son altos, el modelo tiende a tener valores de sensibilidad muy bajos, aunque en general, para casi todos los modelos se tienden a tener sensibilidades más bajas que el restante de las métricas.

Figura 5. Métricas de desempeño según hiperparámetro y valor utilizado del hiperparámetro para el algoritmo de LightGBM.



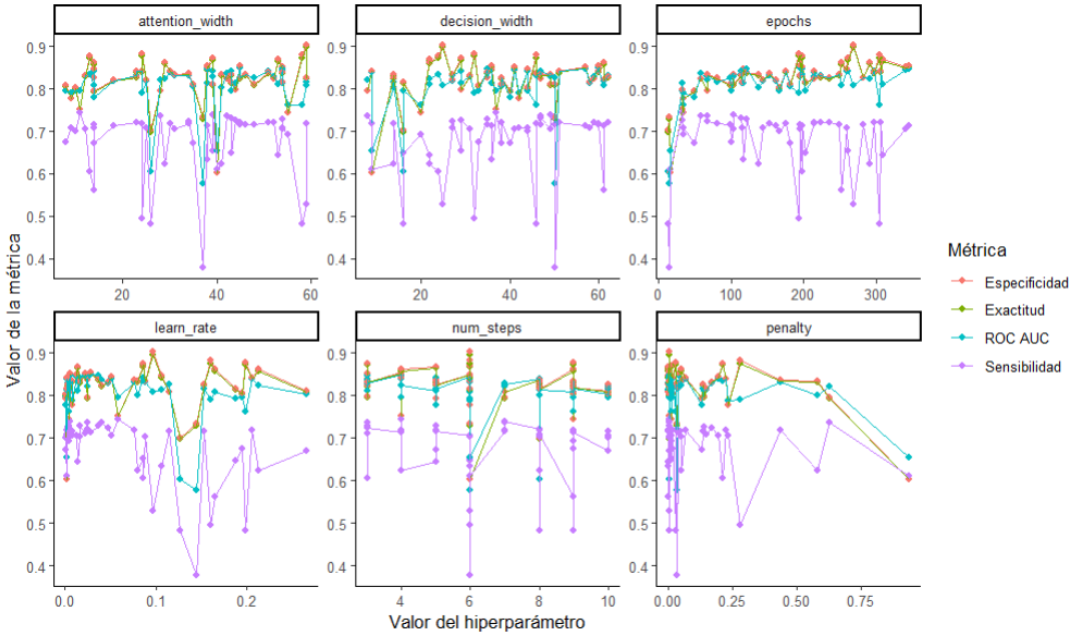
Para el caso de XGBoost la Figura 6 muestra el mismo patrón que en LightGBM, donde en todas las combinaciones probadas, los valores de sensibilidad suelen situarse con valores más bajos, entre 0.70 y 0.75. Aquí para ninguno de los parámetros se muestran rangos donde las métricas tengan un desempeño mayor, o menor, el patrón es más aleatorio que el caso de LightGBM. Sin embargo, al igual que el caso anterior, con tasas de aprendizaje (learn_rate) menores, por debajo de 0.1, si se observan mejores métricas.

Figura 6. Métricas de desempeño según hiperparámetro y valor utilizado del hiperparámetro para el algoritmo de XGBoost.



De los 3 algoritmos, TabNet es el que presenta un comportamiento más aleatorio en términos de desempeño para cada hiperparámetro específico, lo que revela la Figura 7 es que la combinación de los distintos parámetros es la que hace que se tenga un mejor desempeño y no valores específicos de un hiperparámetro, ya que no se observan rangos de ningún parámetro con patrones de métricas de desempeño altas o bajas, sino que a lo largo de todos los valores probados de los hiperparámetros, algunos modelos tuvieron mejores métricas de desempeño y en otros casos más deficientes. Recordando que para TabNet solo se tienen 50 combinaciones de hiperparámetros debido a que el ajuste de dichos modelos es mucho más demorado que el de XGBoost y LightGBM.

Figura 7. Métricas de desempeño según hiperparámetro y valor utilizado del hiperparámetro para el algoritmo de TabNet.



Métricas de desempeño obtenidas por los algoritmos

En cada una de las combinaciones de hiperparámetros de los 3 algoritmos evaluados, se obtienen las 4 métricas de desempeño promedio de los modelos ajustados en los 10 pliegues de validación cruzada, recordando que los modelos se entrenan con un 80% de los datos y se evalúan las métricas en el restante 20%, aplicando el sub muestreo en los datos de entrenamiento mencionado en el apartado metodológico, con el fin de combatir el sobreajuste de las clases.

El Cuadro 3 muestra para cada uno de los 3 algoritmos, la combinación de hiperparámetros que obtuvo el modelo con la mayor exactitud, recordando que la exactitud es la proporción de casos, que abandonan y que no abandonan el servicio de telecomunicaciones, que el modelo logró predecir correctamente. Para este caso el algoritmo LightGBM logró, con la combinación de hiperparámetros específica que se muestra, una

exactitud promedio en los 10 pliegues de 0.965, que indica que el modelo logró predecir correctamente un 96.5% de los casos de abandono y de no abandono correctamente, aunque es importante observar que para este modelo solo el 28.8% (métrica de sensibilidad) de los casos de abandono se logran predecir correctamente, lo cual es muy bajo. En segundo lugar, el algoritmo TabNet logró un modelo donde el 89.5% de los casos fueron predichos correctamente y XGBoost en último lugar con un modelo que logra predecir el 87.4% de los casos correctamente, aunque para este, un 70.6% (métrica de sensibilidad) de los casos de abandono fueron predichos correctamente, que está considerablemente mejor que TabNet y más aún que LightGBM.

Cuadro 3. Métricas de desempeño de los mejores modelos según la métrica de exactitud por algoritmo.

Algoritmo	Hiperparámetros utilizados	Exactitud	ROC AUC	Especificidad	Sensibilidad
LightGBM	mtry = 1, trees = 347, min_n = 16, tree_depth = 10, learn_rate = 0.00114, loss_reduction = 0.08917	0.965	0.837	0.976	0.288
TabNet	epochs = 268, penalty = 0.00222, learn_rate = 0.09738, decision_width = 25, attention_width = 59, num_steps = 6	0.897	0.809	0.903	0.529
XGBoost	mtry = 26, trees = 307, min_n = 27, tree_depth = 10, learn_rate = 0.00575, loss_reduction = 0.02722	0.874	0.860	0.876	0.706

Para la métrica de desempeño de ROC AUC, el Cuadro 4 muestra la misma información que el cuadro anterior, pero con las combinaciones de hiperparámetros que lograron una mejor métrica de ROC AUC. Para este caso hubo un empate entre los algoritmos LightGBM y XGboost, sin embargo, la combinación de hiperparámetros del algoritmo XGBoost logra tener una métrica de sensibilidad casi un 1% mayor, lo que indica que logra

predecir correctamente un 1% más de los casos de abandono que el algoritmo LightGBM, a pesar de que sus métricas de ROC AUC promedio resultaron iguales. TabNet por su parte tiene un valor de ROC AUC menor que los otros dos algoritmos y la única métrica que destaca en esta combinación de hiperparámetros es su sensibilidad, siendo la más alta entre los 3 algoritmos, pero, aun así, muy similar a la de XGBoost.

Cuadro 4. Métricas de desempeño de los mejores modelos según la métrica de ROC AUC por algoritmo.

Algoritmo	Hiperparámetros utilizados	ROC AUC	Exactitud	Especificidad	Sensibilidad
XGBoost	mtry = 11, trees = 202, min_n = 8, tree_depth = 9, learn_rate = 0.03076, loss_reduction = 0.08722	0.862	0.872	0.875	0.712
LightGBM	mtry = 14, trees = 1117, min_n = 3, tree_depth = 9, learn_rate = 0.00904, loss_reduction = 0.03923	0.862	0.874	0.876	0.709
TabNet	epochs = 195, penalty = 0.00275, learn_rate = 0.02809, decision_width = 60, attention_width = 45, num_steps = 4	0.849	0.852	0.854	0.715

Para la métrica de desempeño de sensibilidad, el Cuadro 5 muestra las combinaciones de hiperparámetros de los algoritmos que lograron mejor desempeño en esta métrica. Recordando que la sensibilidad indica la proporción de clientes que abandonaron el servicio de telecomunicación predicha correctamente, se obtiene una mejor métrica para el algoritmo de TabNet, donde un 74.4% de los casos de abandono real son predichos de forma correcta, sin embargo, el porcentaje de casos de no abandono predichos correctamente en esta combinación de parámetros fue de 75.1% (métrica de especificidad), que se supera en gran medida por los algoritmos de XGBoost y LightGBM que tienen valores de 84.6% y 84.4%

respectivamente en términos de especificidad. En general a pesar de que TabNet fue el algoritmo que logra un mayor porcentaje de predicción correcta en términos de clientes que abandonan, las restantes métricas son superadas por los algoritmos de XGBoost y LightGBM, donde para la predicción de casos de no abandono, estos algoritmos logran aproximadamente un 9.5% de predicciones correctas más que TabNet.

Cuadro 5. Métricas de desempeño de los mejores modelos según la métrica de sensibilidad por algoritmo.

Algoritmo	Hiperparámetros utilizados	Sensibilidad	Exactitud	ROC AUC	Especificidad
TabNet	epochs = 33, penalty = 0.00294, learn_rate = 0.05807, decision_width = 37, attention_width = 11, num_steps = 4	0.744	0.751	0.795	0.751
XGBoost	mtry = 31, trees = 1138, min_n = 27, tree_depth = 6, learn_rate = 0.28628, loss_reduction = 0.0015	0.732	0.845	0.845	0.846
LightGBM	mtry = 31, trees = 1138, min_n = 27, tree_depth = 6, learn_rate = 0.28628, loss_reduction = 0.0015	0.729	0.842	0.841	0.844

Por otra parte, el Cuadro 6 muestra las mejores combinaciones de hiperparámetros obtenidas en base a la métrica de especificidad según el algoritmo. La especificidad corresponde a la proporción de casos de clientes que no abandonan el servicio de telecomunicación predichos correctamente. Las combinaciones de hiperparámetros de los algoritmos que logran la mejor métrica de especificidad son las mismas que logran la mejor métrica de exactitud, por lo que, en este caso, el algoritmo LightGBM lidera el mejor resultado en términos de especificidad, pero recordando que dicha combinación de

hiperparámetros sacrifica la predicción en términos de sensibilidad, ya que solo logra predecir correctamente un 28.8% de los casos de clientes que abandonan el servicio de telecomunicación, lo cual es bastante deficiente en términos de negocio si lo que se busca es lograr una predicción correcta anticipada para casos de abandono con el fin de establecer campañas de retención de clientes.

Cuadro 6. Métricas de desempeño de los mejores modelos según la métrica de especificidad por algoritmo.

Algoritmo	Hiperparámetros utilizados	Especificidad	Exactitud	ROC AUC	Sensibilidad
LightGBM	mtry = 1, trees = 347, min_n = 16, tree_depth = 10, learn_rate = 0.00114, loss_reduction = 0.08917	0.976	0.965	0.837	0.288
TabNet	epochs = 268, penalty = 0.00222, learn_rate = 0.09738, decision_width = 25, attention_width = 59, num_steps = 6	0.903	0.897	0.809	0.529
XGBoost	mtry = 26, trees = 307, min_n = 27, tree_depth = 10, learn_rate = 0.00575, loss_reduction = 0.02722	0.876	0.874	0.860	0.706

Finalmente, es importante encontrar combinaciones de hiperparámetros de modelos que tengan métricas de sensibilidad aceptables sin sacrificar demasiado la especificidad, ya que si bien es importante predecir clientes que abandonarán el servicio de telecomunicación (sensibilidad), para definir cuál será la población meta a la que se debe aplicar estrategias de retención, también es relevante que el modelo sea bueno realizando predicciones de aquellos que no abandonarán el servicio (especificidad), esto pues si el modelo es deficiente en términos de especificidad, causará un aumento en la cantidad de falsos positivos, es decir,

aumentarán los casos de clientes que se predice que abandonarán el servicio, cuando en realidad no lo van a hacer, y esto puede causar que las estrategias de retención aplicadas a los clientes meta por la empresa de telecomunicación tengan un costo mayor debido a la cantidad de falsos positivos.

Para esto el Cuadro 7 muestra de los 70 modelos de LightGBM y XGBoost y los 50 de TabNet, el top 3 con el mayor valor promedio entre las métricas de especificidad y sensibilidad, esto proporciona modelos más equitativos tanto en la predicción de casos de abandonos como de no abandonos.

Aquí se logra evidenciar que tanto para los algoritmos de XGBoost y LightGBM se logra mantener sensibilidades superiores a 0.71 y especificidades de hasta 0.87, manteniendo métricas de ROC AUC y de exactitud por encima de 0.86; mientras que TabNet a pesar de que en uno de sus modelos logra una sensibilidad de hasta casi 0.73, sus métricas de exactitud y especificidad no superan el 0.84, y en todos los casos además, el valor de sus ROC AUC no supera el valor de 0.84, por lo que si se busca un modelo más equitativo en términos de todas las métricas, TabNet tiene un desempeño inferior a XGBoost y LightGBM.

Cuadro 7. Métricas de desempeño de modelos con mejor promedio entre especificidad y sensibilidad por algoritmo.

Algoritmo / Hiperparámetros utilizados	Exactitud	ROC AUC	Especificidad	Sensibilidad
LightGBM				
mtry = 19, trees = 871, min_n = 22, tree_depth = 7, learn_rate = 0.02742, loss_reduction = 0.08763	0.870	0.861	0.872	0.715
mtry = 14, trees = 1493, min_n = 8, tree_depth = 7, learn_rate = 0.03253, loss_reduction = 0.18062	0.868	0.860	0.871	0.717
mtry = 26, trees = 1278, min_n = 27, tree_depth = 3, learn_rate = 0.0675, loss_reduction = 0.15434	0.868	0.859	0.870	0.717
XGBoost				
mtry = 12, trees = 498, min_n = 11, tree_depth = 7, learn_rate = 0.03018, loss_reduction = 0.02054	0.870	0.862	0.873	0.715
mtry = 6, trees = 342, min_n = 22, tree_depth = 8, learn_rate = 0.0872, loss_reduction = 0.33872	0.867	0.860	0.869	0.719
mtry = 14, trees = 1493, min_n = 8, tree_depth = 7, learn_rate = 0.03253, loss_reduction = 0.18062	0.866	0.859	0.868	0.719
TabNet				
epochs = 195, penalty = 0.00275, learn_rate = 0.02809, decision_width = 60, attention_width = 45, num_steps = 4	0.852	0.849	0.854	0.715
epochs = 345, penalty = 0.04, learn_rate = 0.00633, decision_width = 36, attention_width = 38, num_steps = 4	0.851	0.847	0.853	0.713
epochs = 121, penalty = 0.0023, learn_rate = 0.03651, decision_width = 35, attention_width = 54, num_steps = 3	0.837	0.847	0.839	0.729

Tiempos de ejecución para ajuste de modelos

Un tema muy relevante a la hora de evaluar las métricas de desempeño y qué algoritmos utilizar, es el tiempo de ejecución que conlleva ajustar las distintas combinaciones de hiperparámetros en los distintos pliegues de validación cruzada. Esto pues en el desarrollo de modelos para la predicción de abandono de clientes, aparte de buscar una métrica con el mejor desempeño posible, normalmente el tiempo para la construcción de estos modelos es limitado, y aquí es donde toman ventaja aquellos algoritmos cuyo tiempo de ejecución es menor, ya que permiten encontrar un balance entre buenas métricas de desempeño y tiempos razonables de ejecución a la hora de ajustar hiperparámetros.

El Cuadro 8 muestra el tiempo de ejecución en minutos por algoritmo del mejor modelo ajustado según la métrica de desempeño utilizando el CPU de una sola máquina, así, por ejemplo, en los modelos con mejores métricas de exactitud, el modelo de LightGBM tardó 4.2 minutos en entrenarse y evaluar los resultados en el 20% de los datos de prueba, mientras que el algoritmo TabNet duró 93 minutos entrenándose y evaluando los resultados en los datos de prueba. A excepción de los mejores modelos con métrica ROC AUC, el algoritmo LightGBM es el que tiene menores tiempos de ejecución, mientras que TabNet en todos los casos tiene tiempos de ejecución excesivamente grandes en comparación a los restantes dos algoritmos; en menor medida en el caso del mejor modelo con la métrica de sensibilidad, que logra un tiempo de 13.5 minutos, pero en general es un algoritmo que tarda mucho tiempo en ajustarse, lo que le da una desventaja significativa respecto a los otros dos algoritmos.

Cuadro 8. Tiempo de ejecución en minutos del mejor modelo ajustado según cada métrica de desempeño.

Algoritmo	Exactitud	ROC AUC	Sensibilidad	Especificidad
LightGBM	4.2	4.4	5.1	5.1
XGBoost	5.4	3.8	7.9	5.3
TabNet	93.0	61.5	13.5	83.9

Conclusiones y recomendaciones

En relación al primero de los objetivos específicos, se cumple con el ajuste de modelos utilizando los algoritmos de TabNet, XGBoost y LightGBM, con 50 combinaciones aleatorias de 6 hiperparámetros en el caso de TabNet y 70 combinaciones aleatorias de 6 hiperparámetros en el caso de XGBoost y LightGBM, todo esto aplicando un submuestreo mayoritario aleatorio con reemplazo en los datos de entrenamiento, con el fin de balancear las clases en una proporción cercana a 60% de casos de no abandono y 40% de casos abandono. El ajuste se realizó en 10 pliegues distintos por medio de validación cruzada de una repetición por cada combinación de hiperparámetros; y en cada ajuste se utilizó el 80% como datos de entrenamiento y el 20% como datos de prueba. Esto permitió establecer comparaciones entre el desempeño de los algoritmos basado en 4 métricas de desempeño, las cuales fueron: exactitud, ROC AUC, sensibilidad y especificidad.

En cumplimiento con el segundo de los objetivos específicos, las comparaciones entre los algoritmos y las métricas de desempeño revelaron que el algoritmo TabNet solo resultó con una mejor métrica cuando se comparan los algoritmos respecto a la métrica de sensibilidad, que corresponde a la proporción de casos de clientes que abandonan el servicio de telecomunicación predichos correctamente. TabNet logró predecir correctamente, con una combinación específica de hiperparámetros, un 74.4% de los casos de abandono de clientes en los servicios de telecomunicación correctamente, pero no muy lejos de los de los porcentajes obtenidos por los algoritmos de XGBoost y LightGBM que fueron de 73.2% y 72.9% respectivamente; sin embargo, el algoritmo de TabNet, para este mismo modelo, presenta métricas más deficientes en la predicción correcta de casos de clientes que no abandonaron el servicio de telecomunicación (especificidad), con un porcentaje de 75.1%, frente a los algoritmos de XGBoost y LightGBM, que superan esta métrica en alrededor de 9%, con valores de 84.6% y 84.4% respectivamente. Si se comparan los algoritmos en los mejores modelos obtenidos con las restantes 3 métricas de desempeño, TabNet no lidera en ninguna, lo que pone en la delantera a LightGBM, con mejores métricas en exactitud y especificidad y luego a XGBoost con un mejor ROC AUC.

Respecto al desempeño mostrado por XGBoost y LightGBM en datos relacionados al abandono de clientes en el ámbito de las telecomunicaciones, el algoritmo XGBoost en términos de exactitud tuvo un valor, en la mejor combinación de hiperparámetros, de 0.874, que supera los resultados obtenidos por Parmar y Serasiya (2021) y Senthan et al. (2021), que obtuvieron exactitudes de 0.805 y 0.829, pero a su vez, la métrica obtenida en este estudio está por debajo de los resultados obtenidos por Hanif (2020) y Tang et al. (2020), que tuvieron métricas de exactitud de 0.970 y 0.963 en la predicción de abandono para empresas de telecomunicación. A nivel de la métrica de ROC AUC, en el ámbito de las telecomunicaciones, el algoritmo XGBoost tuvo un valor máximo de 0.862 en este estudio, que supera los obtenidos por Falla (2021) y Wang, Nguyen y Nguyen (2020), con valores de 0.785 y 0.685 respectivamente; pero está aún por debajo del valor obtenido por Echeverri (2019) que fue de 0.950. El algoritmo LightGBM con un ROC AUC máximo alcanzado de 0.862 en el presente estudio, supera el valor obtenido por Wang, Nguyen y Nguyen (2020) de 0.689. Si bien para el algoritmo TabNet no se encontraron aplicaciones relacionadas a modelos de predicción de abandono de clientes, en términos de exactitud, este obtuvo un valor máximo de 0.897, el cual es incluso mayor al obtenido por el algoritmo de XGBoost, que fue de 0.874, y dado que este valor de XGBoost supera los obtenidos por otros estudios, posiciona a TabNet con un buen desempeño a nivel de predicción total de casos de abandono y no abandono de clientes en el ámbito de telecomunicaciones.

Finalmente, referente al tercer objetivo específico, la medición de los tiempos de ejecución dejan el algoritmo de TabNet en desventaja, ya que el tiempo de ejecución en el ajuste de los modelos utilizando este algoritmo es mucho mayor que en los casos de XGBoost y LightGBM, donde por ejemplo, para la mejor combinación de hiperparámetros en la métrica de exactitud, TabNet tardó 93 minutos realizando el ajuste de un modelo, mientras que XGBoost y LightGBM tardaron 5.4 y 4.2 minutos respectivamente, una diferencia de 87 minutos, si se compara TabNet contra XGBoost. Sin embargo, esta diferencia no es tan grande cuando la métrica que se pone como prioridad es la de sensibilidad, donde TabNet tardó 13.5 minutos, frente a XGBoost y LightGBM que tardaron 7.9 y 5.1 minutos respectivamente; y a pesar de que en esta métrica TabNet tuvo un mejor valor, no lo demuestra para la predicción de casos de clientes que no abandonan el servicio de

telecomunicación (especificidad) como se mencionó anteriormente. En todos los casos los ajustes en cuanto a medición de tiempos de ejecución se realizaron sin paralelizar el proceso utilizando el CPU de la máquina.

Dado el desempeño en términos de métricas para el algoritmo de TabNet frente a los algoritmos comúnmente utilizados en conjuntos de datos tabulares como XGBoost y LightGBM, se recomendaría utilizarse este, solo en caso de que para el Instituto Costarricense de Electricidad sea estrictamente más relevante la predicción más acertada de casos de clientes que abandonan el servicio de telecomunicación. Lo cual también es relevante y puede ser de gran utilidad para la empresa, pues tal y como lo mencionaban Vafeiadis et al. (2016) y Ullah et al. (2019), es mucho más económico la retención de clientes por medio de predicción anticipada de abandono de los mismos, que atraer nuevos, ya que atraer nuevos clientes puede llegar a costar entre 5 y 6 veces más, cuando se habla del ámbito de las telecomunicaciones. Si más bien se busca eficiencia en el ajuste de modelos para probar con diversas combinaciones de hiperparámetros, o bien, métricas de desempeño más equilibradas, es decir, que el modelo no solo tenga un buen desempeño en términos de predicción de casos de abandono (sensibilidad), sino que también tenga un buen poder predictivo en el caso de la predicción de casos de clientes que no abandonan (especificidad) con el fin de evitar el incremento de falsos positivos, probablemente TabNet no sea el mejor algoritmo, y LightGBM o XGBoost tienen ventajas en estos dos aspectos, pues como se mostró a nivel de sensibilidad, a pesar de que estaban por debajo de TabNet por cerca de un 1%, estaban por encima de TabNet en cuanto a especificidad por más de un 9%, adicionándole a esto que los tiempo de ejecución de TabNet no mostraron ser buenos respecto a los otros dos algoritmos en ninguno de todos los escenarios.

Finalmente a nivel de limitaciones y recomendaciones para futuras investigaciones, se recomendaría realizar pruebas con otras técnicas de sub muestreo que también ofrece el paquete themis de R, como la extracción de enlaces Tomek de mayoría-minoría, o bien técnicas de sobre muestreo o híbridas como el caso de la técnica de SMOTE utilizada por Esteves & Mendes (2016), que igualmente están disponibles en el paquete themis, ya que para esta investigación por la limitación de recursos computacionales y tiempo, no fue

posible realizar pruebas con otras técnicas de remuestreo, y puede que se lograran obtener mejores resultados con algunas de estas otras técnicas. A su vez, sería importante tener una infraestructura en la nube más escalable que una computadora portátil, ya que podría ayudar a acelerar el proceso de búsqueda de las mejores combinaciones de hiperparámetros y así poder probar más combinaciones, o bien, ajustar otros hiperparámetros que se dejaron con sus valores predeterminados por los paquetes (ya que se limitó a 6 por algoritmo en este caso), sin esperar tanto tiempo para obtener resultados de desempeño en la validación cruzada.

Bibliografía

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... & Zhu, Z. (2016, Junio). Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning (pp. 173-182). PMLR.

Arai, K., Fujikawa, I., Nakagawa, Y., Momozaki, R., & Ogawa, S. (2023). Churn Customer Estimation Method based on LightGBM for Improving Sales. International Journal of Advanced Computer Science and Applications, 14(2).

Arik, S. Ö., & Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, No. 8, pp. 6679-6687).

Asencios, R., Asencios, C., & Ramos, E. (2023). Profit scoring for credit unions using the multilayer perceptron, XGBoost and TabNet algorithms: Evidence from Peru. Expert Systems with Applications, 213, 119201.

Bengtsson, H. A. (2021). Unifying Framework for Parallel and Distributed Processing in R using Futures, The R Journal (2021) 13:2, pages 208-227, doi:10.32614/RJ-2021-048

Bobadilla, J. (2021). Machine learning y deep learning: usando Python, Scikit y Keras. Ediciones de la U.

Boehmke, B., & Greenwell, B. M. (2019). Hands-on machine learning with R. Chapman and Hall/CRC.

Brownlee, J. (2018). What is the Difference Between a Batch and an Epoch in a Neural Network. Machine Learning Mastery, 20.

Calvino, H. A. A. (2017). Métodos para la mejora de predicciones en clases desbalanceadas en el estudio de bajas de clientes (CHURN). Master en Técnicas Estadísticas USC-Universidad da Coruña.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y., Yuan, J. (2023). xgboost: Extreme Gradient Boosting. R package version 1.7.5.1, https://CRAN.R-project.org/package=xgboost.

Chouiekh, A. (2017, Abril). Machine learning techniques applied to prepaid subscribers: case study on the telecom industry of Morocco. In 2017 Intelligent Systems and Computer Vision (ISCV) (pp. 1-8). IEEE.

Dahiya, K., & Bhatia, S. (2015, Setiembre). Customer churn analysis in telecom industry. In 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions) (pp. 1-6). IEEE.

Dinamarca, A. (2018). Aprendizaje y análisis de redes neuronales artificiales profundas (Doctoral dissertation, Universidad Nacional de Cuyo. Facultad de Ciencias Exactas y Naturales).

Echeverri, G. A. F. (2019). Modelo predictivo de Churn de clientes para el negocio de Telecomunicaciones.

Esteves, G., & Mendes, M. J. (2016, Setiembre). Churn prediction in the telecom business. In 2016 Eleventh International Conference on Digital Information Management (ICDIM) (pp. 254-259). IEEE.

Falbel D, Luraschi J (2023). torch: Tensors and Neural Networks with 'GPU' Acceleration. R package version 0.11.0, https://CRAN.R-project.org/package=torch.

Falbel D. (2023). tabnet: Fit 'TabNet' Models for Classification and Regression. R package version 0.4.0, https://CRAN.R-project.org/package=tabnet

Falla Arango, J. D. (2021). Predicción de abandono de clientes en telecomunicaciones mediante el aprendizaje automático.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.

Gutiérrez, G. D. (2020). Técnicas de machine learning en el análisis del churn rate.

Hanif, I. (2020). Implementing extreme gradient boosting (xgboost) classifier to improve customer churn prediction.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hudaib, A., Dannoun, R., Harfoushi, O., Obiedat, R., & Faris, H. (2015). Hybrid data mining models for predicting customer churn. International Journal of Communications, Network and System Sciences, 8(05), 91.

Hvitfeldt, E. (2023). themis: Extra Recipes Steps for Dealing with Unbalanced Data. R package version 1.0.2, https://CRAN.R-project.org/package=themis.

Kayaalp, F. (2017). Review of Customer Churn Analysis Studies in Telecommunications Industry. Karaelmas Science & Engineering Journal, 7(2).

Kuhn et al., (2020). Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles. https://www.tidymodels.org

Kuhn M, Vaughan D (2023). parsnip: A Common API to Modeling and Analysis Functions. R package version 1.1.0, https://CRAN.R-project.org/package=parsnip.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, Febrero). Recurrent convolutional neural networks for text classification. In Proceedings of the AAAI conference on artificial intelligence (Vol. 29, No. 1).

Lalwani, P., Mishra, M. K., Chadha, J. S., & Sethi, P. (2022). Customer churn prediction system: a machine learning approach. Computing, 1-24.

Liu, X. Y., Wu, J., & Zhou, Z. H. (2008). Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2), 539-550.

Malyar, M., Robotyshyn, M. M., & Sharkadi, M. (2020, Octubre). Churn Prediction Estimation Based on Machine Learning Methods. In 2020 IEEE 2nd International Conference on System Analysis & Intelligent Computing (SAIC) (pp. 1-4). IEEE.

Núñez, D. L. (2015). Modelos predictivos del churn -abandono de clientes- para operadores de telecomunicaciones.

Parmar, P., & Serasiya, S. (2021). Telecom churn prediction model using XgBoost classifier and logistic regression algorithm. Int Res J Eng Technol (IRJET), 8, 1100-1105.

Posit team (2023). RStudio: Integrated Development Environment for R. Posit Software, PBC, Boston, MA. URL http://www.posit.co/

Pranav, S., Lahoti, M., & Gopalarathnam, M. (2023). Concrete Compressive Strength Prediction Using Boosting Algorithms. In Fiber Reinforced Polymeric Materials and Sustainable Structures (pp. 307-315). Singapore: Springer Nature Singapore.

Qureshi, S. A., Rehman, A. S., Qamar, A. M., Kamal, A., & Rehman, A. (2013, Setiembre). Telecommunication subscribers' churn prediction model using machine learning. In Eighth international conference on digital information management (ICDIM 2013) (pp. 131-136). IEEE.

R Core Team (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., & Schmidt, L. (2019). A meta-analysis of overfitting in machine learning. Advances in Neural Information Processing Systems, 32.

Rojas, J. K. (2022). Ciencia de Datos para Ciencias Naturales. Recuperado de https://bookdown.org/keilor_rojas/CienciaDatos/

Sabbeh, S. F. (2018). Machine-learning techniques for customer retention: A comparative study. International Journal of advanced computer Science and applications, 9(2).

Senthan, P., Rathnayaka, R. M. K. T., Kuhaneswaran, B., & Kumara, B. T. G. S. (2021, Abril). Development of churn prediction model using xgboost-telecommunication industry in sri lanka. In 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS) (pp. 1-7). IEEE.

Sharma, J., Maheshwari, R., Khan, S., & Khan, A. A. (2022). Evaluating performance of different machine learning algorithms for the acute emg hand gesture datasets. Journal of Electronics and Informatics, 4(3), 192-201.

Shi, Y., Ke, G., Soukhavong, D., Lamb, J., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T., Titov, N. (2023). lightgbm: Light Gradient Boosting Machine. R package version 3.3.5, https://CRAN.R-project.org/package=lightgbm.

Sina, M. S. M., & Amiri, B. (2022). Model optimization analysis of customer churn prediction using machine learning algorithms with focus on feature reductions. Discrete Dynamics in Nature and Society, 2022.

Tang, Q., Xia, G., Zhang, X., & Long, F. (2020, Marzo). A customer churn prediction model based on XGBoost and MLP. In 2020 International Conference on Computer Engineering and Application (ICCEA) (pp. 608-612). IEEE.

Ullah, I., Raza, B., Malik, A. K., Imran, M., Islam, S. U., & Kim, S. W. (2019). A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector. IEEE access, 7, 60134-60149.

Umayaparvathi, V., & Iyakutti, K. (2016). A survey on customer churn prediction in telecom industry: Datasets, methods and metrics. International Research Journal of Engineering and Technology (IRJET), 3(04).

Vergara Rojas, F. C. D., & Cárdenas Roa, F. D. (2014). Estudio comparativo de las estrategias para la disminución de la tasa de cancelación de clientes (Churn), en Claro empresa prestadora del servicio de televisión en Colombia y Perú.

Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G., & Chatzisavvas, K. C. (2015). A comparison of machine learning techniques for customer churn prediction. Simulation Modelling Practice and Theory, 55, 1-9.

Wang, X., Nguyen, K., & Nguyen, B. P. (2020, Enero). Churn prediction using ensemble learning. In Proceedings of the 4th international conference on machine learning and soft computing (pp. 56-60).

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, LD., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, TL., Miller, E., Bache, SM., Müller, K., Ooms, J., Robinson, D., Seidel, DP., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., Yutani, H. (2019). "Welcome to the tidyverse.". Journal of Open Source Software, *4*(43), 1686. doi:10.21105/joss.01686, https://doi.org/10.21105/joss.01686.

Zhang, D., & Gong, Y. (2020). The comparison of LightGBM and XGBoost coupling factor analysis and prediagnosis of acute liver failure. IEEE Access, 8, 220990-221003.

Zhang, L., Ma, K., Yuan, F., & Fang, W. (2022, Enero). A Tabnet based Card Fraud detetion Algorithm with Feature Engineering. In 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE) (pp. 911-914). IEEE.