Diseño de un Modelo Predictivo de Fuga de Clientes Utilizando Algoritmos Machine Learning

Diego Andrés Pinto Galindo

Universidad ECCI
Departamento de Ciencias Básicas
Bogotá, Colombia

2020

Diseño de un Modelo Predictivo de Fuga de Clientes Utilizando Algoritmos Machine Learning

Diego Andrés Pinto Galindo

Trabajo de grado presentada(o) como requisito parcial para optar al título de:

Profesional en Estadística

Director:

MSc. en Estadística, José Alexander Fuentes Montoya

Universidad ECCI

Departamento Ciencias Básicas

Bogotá, Colombia

2020

Título en español

Diseño de un modelo predictivo de fuga de clientes utilizando algoritmos machine Learning.

Title in English

Design of a Predictive Model of Customer Leakage Using Machine Learning Algorithms.

Resumen: Las empresas prestadoras de servicios de telecomunicaciones móviles contribuyen al desarrollo económico del país, ya que son necesarias para mantener el flujo de información entre diversos canales y servicios. Actualmente, en Colombia hay varias empresas nacionales e internacionales que prestan servicios móviles y la alta penetración de las mismas en el mercado ha generado se incremente la competencia por los usuarios. Por consecuente, dichas empresas tienen el gran reto de mantener la satisfacción y fidelización de los mismos. Para lograr mantener dicha fidelización se hace pertinente analizar y entender el comportamiento de los clientes antes de que se fuguen a otra empresa, para anticiparse a este comportamiento se desarrollará en el presente artículo la descripción, análisis y modelamiento por medio de técnicas de Machine Learning y se escogerá la que mejor se ajuste a los comportamientos de los usuarios.

Abstract: The companies providing mobile telecommunications services contribute to the economic development of the country, as they are necessary to maintain the flow of information between various channels and services. Now in Colombia there are several national and international companies that provide mobile services and the high penetration of this market has generated that they train in competition for users. This means that companies have the great challenge of maintaining the satisfaction and loyalty of their users. In order to achieve user loyalty, it is necessary to understand and understand the behavior of users before they request portability or flight to another company, in order to anticipate this behavior, the analysis and modeling will be developed in this work through Machine Learning techniques and the choice of that best suits user behaviors.

Palabras claves: Fidelización, Fuga de Clientes, Machine Learning, predicción.

Keywords: Loyalty, Customer Leak, Machine Learning, prediction..

Dedicatoria

Esta tesis la dedico a mi familia, en especial a mi abuela Ester que ha hecho de mí una gran persona y me ha ayudado a llegar hasta donde estoy el día de hoy, por siempre querer que "sea alguien en la vida" a base de estudio, compromiso y dedicación. También a mi madre por su amor y comprensión y a mi tía Milena por nunca desampararme.

Agradecimientos

A Dios por mi gran familia la cual siempre han estado presente y motivándome a seguir luchando por lo que se quiere lograr en la vida, a nunca rendirme en las situaciones más adversas que se tengan que afrontar.

A los profesores que durante mi carrera compartieron sus conocimientos para formarme como profesional. A Alexander Fuentes por su apoyo y compromiso.

A mis compañeros de la carrera y aquellos que quedaron por el camino muchas gracias por todos los momentos compartidos junto a ustedes, deseo enormemente que la vida les sonría a todos y cada uno de ustedes.

A mis compañeros de trabajo que influyeron en alguna parte de este trabajo y que día a día influyen en mi crecimiento profesional. En especial a Juan Manuel Peñaloza Coordinador de minería de datos y Jesús Giovanny Martínez Gerente Planeación y Control Operativo, ya que gracias a ustedes he crecido como profesional y he aportado mi conocimiento en las situaciones que lo han requerido.

Muchas Gracias!!!

Índice General

Índice de Figuras

Índice de Tablas

Introducción

Actualmente nos encontramos en una era digital que gira en torno a las nuevas tecnologías, trayendo consigo cambios profundos y transformaciones de una sociedad que se mueve en un mundo globalizado. Por ende, se tiene la gran necesidad de estar activamente conectado a una red y estar al tanto de todo lo que pasa alrededor. En poco más de una década, se duplicaron sustancialmente los usuarios de Internet, que ya alcanzaban el 50,1% de la población en 2014; hoy existen más de 700 millones de conexiones a telefonía móvil, con más de 320 millones de usuarios únicos, y muchos países de Latinoamérica se encuentran entre los que más usan las redes sociales globales. Esta es la nueva era que impactó al mundo pero también a los gobiernos, quienes han tenido que incorporar el uso de las tecnologías de la información y la comunicación en la gestión pública.

En Colombia la estimación preliminar de habitantes según estadísticas entregadas por el DANE en el censo nacional de población y vivienda del año 2018 es de un poco más de cuarenta y ocho millones de habitantes [1], y el número de líneas móviles para el año 2019 fue más de sesenta y seis millones, una cifra entregada por el Mintic (Ministerio de las Tecnologías y las Comunicaciones)[2], es decir que por cada colombiano había 1.2 líneas en el país, confirmó el Ministro de Tecnología de la Información y las Comunicaciones, David Luna. Esta cantidad es superior en comparación con la cantidad de habitantes de Colombia. Cabe recalcar que, aproximadamente un 80% de las líneas son prepago y el 20% son líneas pospago. Bajo estas estadísticas, donde es significativo el número de usuarios de servicios móviles, se hace relevante que las empresas de telecomunicaciones móviles generen estrategias de fidelización para que no exista el riesgo de fuga por parte del cliente a un mejor servicio, que alguna otra empresa de telecomunicaciones le pueda brindar.

Desde una perspectiva de inteligencia de negocios, el proceso de gestión de fuga de clientes se considera una de las tareas más importantes y se focaliza en dos actividades: la primera, en predecir una potencial fuga, y la segunda, aplicar medidas preventivas para evitar que se produzca la fuga en donde la gran cantidad de información recolectada y almacenada en las empresas, ligada a la necesidad de entender y satisfacer al cliente, ha traído consigo maneras eficientes de analizar la información, una de ella es el aprendizaje automático o más conocido en inglés como Machine Learning (ML). Hemos visto el aprendizaje automático como una palabra de moda en los últimos años, la razón de esto podría ser la gran cantidad de producción de datos por parte de las aplicaciones, el aumento de la potencia de cálculo en los últimos años y el desarrollo de mejores algoritmos. El aprendizaje automático se utiliza en cualquier lugar, desde la automatización de tareas cotidianas hasta el ofrecimiento de información inteligente, las industrias de todos los sectores intentan beneficiarse de ella. Es posible que ya esté utilizando un dispositivo que lo utiliza. Por ejemplo, un

rastreador de ejercicios portátil como Fitbit, o un asistente doméstico inteligente como Google Home.

En cuanto a las tareas que trae consigo el negocio de las telecomunicaciones se hace muy útil la implementación de técnicas Machine Learning para el estudio y entendimiento de los comportamientos de los clientes, también para la implementación de sistemas recomendadores y automatización de procesos.

1. Telecomunicaciones.

1.1. Telecomunicaciones y la Vida Cotidiana

Desde que los avances tecnológicos se han implementado en la cotidianidad de los seres humanos ha cambiado la vida y además todos y cada uno de los aspectos que la integran. Constantemente hemos de adquirir nuevos conocimientos y habilidades para seguir el ritmo impuesto por el desarrollo tecnológico. El hombre como ser social necesita comunicarse no solo con su entorno inmediato sino más allá de su cercanía próxima, esto nos lleva al uso de las telecomunicaciones, en épocas recientes el uso de estas ha aumentado debido a nuestro constante deseo de comunicarnos con personas que se encuentran cada vez más lejos del lugar donde estamos. Pero no vayamos lejos, simplemente en nuestra vida diaria utilizamos las telecomunicaciones por lo menos una vez al día cuando hacemos una llamada o enviamos un mail, todo se hace más fácil ya que antes para podernos comunicar el medio más usado era el servicio postal que tardaba aproximadamente 15 días. Los sistemas postales modernos siguieron creciendo con la aparición del ferrocarril, los vehículos de motor, los aviones y otros medios de transporte, últimamente ha surgido el correo electrónico. Sin embargo, a lo largo de los siglos siempre se han buscado medios de comunicación a larga distancia que fueran más rápidos que los convencionales. [3]

Las telecomunicaciones utilizan distintos tipos de transmisiones como son: radio, televisión, datos, audio y multimedia. Actualmente los distintos avances tecnológicos en todos los campos de la ciencia, han permitido que este tipo de transmisiones se realicen utilizando satélites artificiales de nuestro planeta, en número, funciones y rangos diversos. Uno de los medios de telecomunicación más famoso y utilizado es sin duda la televisión, esta se ha extendido por todo el mundo; los satélites de comunicaciones permiten transmitir programas de un continente a otro y enviar acontecimientos en vivo a casi cualquier parte del mundo, millones de personas se mantienen en contacto con el exterior por este medio ya que se pueden enterar de los

acontecimientos más importantes que suceden en el mundo viendo un noticiero, se pueden dar cuenta de los cambios climáticos y de los nuevos avances tecnológicos.

Otro de los medios de telecomunicación más solicitada es el teléfono celular, todas las personas tienen acceso a uno de estos y tienen diferentes usos, desde hacer una llamada hasta enviar un correo electrónico y mil cosas más que mantienen al ser humano en contacto con el mundo exterior. La computadora es otro avance que ha servido para comunicarse de un lugar a otro desde la comodidad del hogar, puedes enviar información desde la comodidad de tu hogar a cualquier parte del planeta sin la necesidad de moverte, con un solo click estas informando tu estado, enviando la tarea que te dejaron en clase o simplemente platicando con un familiar al que no ves desde hace mucho tiempo. Las telecomunicaciones se han vuelto una necesidad para las personas, no puedes estar tranquilo si no tienes el celular en las manos o si te perdiste el noticiero en la mañana, son indispensables y se volverán aún más en un futuro no muy lejano.

1.1.1. Telefonía Móvil Celular en Colombia

La telefonía móvil celular (TMC) es un servicio público de telecomunicaciones no domiciliario, de ámbito y cubrimiento nacional, que proporciona en sí mismo capacidad completa para la comunicación telefónica entre usuarios móviles y, a través de la interconexión con la red telefónica público conmutada (RTPC), entre aquellos, y usuarios fijos, haciendo uso de una red de telefonía móvil celular, en la que la parte del espectro radioeléctrico asignado constituye su elemento principal. El Servicio de Telefonía Móvil Celular se encuentra contemplado en la Ley 37 de 1993 y en las siguientes normas: Ley 442 de 1998; Decreto 741 de 1993; Decreto 2061 de1993 y Decreto 990 de 1998.

Actualmente en Colombia existen varias empresas prestadoras de servicios móviles para las cuales a mediados de 2018 se aprobó la nueva ley de telecomunicaciones y tecnologías de la información de Colombia con el objetivo de brindar las condiciones para que el país llegue al 70 por ciento de los hogares conectados a Internet. La norma crea un único regulador representado en la nueva Comisión de Regulación de las Comunicaciones (CRC) a cargo del sector de telecomunicaciones y audiovisual. Además, se liquida la Autoridad Nacional de Televisión (ANTV) para la regulación de las telcos con operaciones en Colombia. [4]. Para la operación de las empresas de telecomunicaciones en el país se debe contar con bandas de espectro el cual solo se trata del medio por el cual se transmiten las frecuencias de ondas de radio electromagnéticas que permiten las telecomunicaciones (radio, televisión, Internet, telefonía móvil, televisión digital terrestre, etc.), y son administradas y reguladas por los gobiernos de cada país antes mencionados. Claro, Avantel, Movistar, Tigo, ETB y

MVNO’s actualmente operan en Colombia con un espectro adquirido en las subastas del mismo y cada una con una participación en el mercado distribuida de diferente manera, tal como muestra la figura 1, donde Claro Colombia tiene la mayor participación en el mercado.

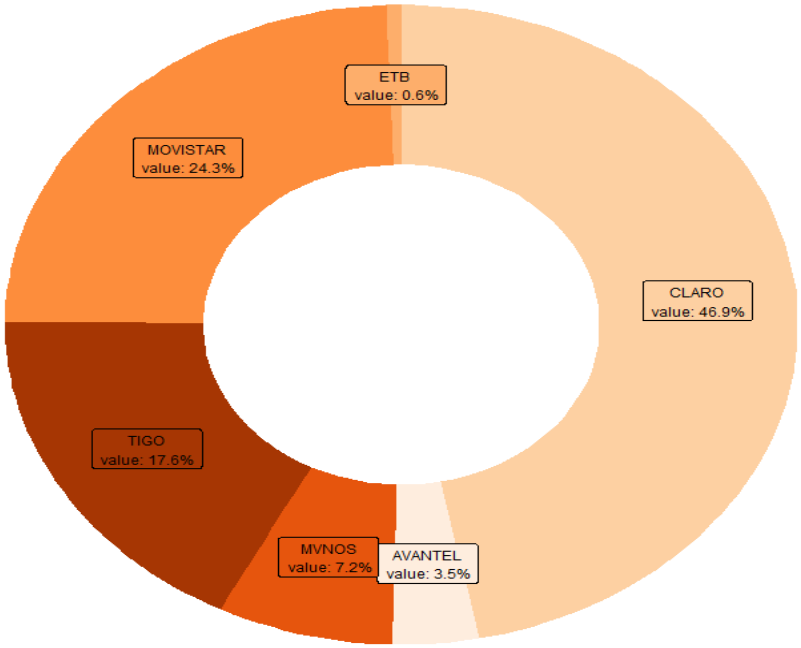

Figura 1. Participación de los operadores en el mercado móvil Colombiano.

1.1.2. TMC Prepago y Pospago

La necesidad constante de estar conectados y sentirnos cerca a las personas que nos rodean, han conllevado al uso frecuente de consumo de internet, mensajes de texto y minutos a celular disponibles para realizar una llamada telefónica. Por ende, las empresas de telecomunicaciones brindan a sus clientes dos tipos de TMC de acuerdo a las necesidades y solicitudes que tenga el cliente. La primera de estas es la TMC prepago la cual se creó para ayudarles a los clientes a evitar cargos por exceso y permitirles vivir más fácil con un presupuesto. Estos planes les permiten a los clientes elegir de una variedad de dispositivos, pero en lugar de tener que pagar tarifas de activación y mantener un plan mensual, el teléfono simplemente deja de funcionar cuando su cantidad asignada de minutos se acaba. El segundo es la TMC pospago, la cual ofrece más estabilidad y les permiten a los clientes construir su puntuación crediticia al tener un crédito con una corporación. Generalmente hay una selección más grande de dispositivos disponibles para los clientes que optan por un plan pospago. Los compradores seleccionan un plan mensual de servicio, el cual cuesta una cantidad

fija (plan pospago cerrado o con límite de consumo). Sin embargo, si el usuario termina usando más de lo que el contrato estipula, el dispositivo continuará operando y el usuario recibirá una factura más alta para ese mes (plan pospago abierto o sin límite de consumo).[5]

El incremento del uso del internet para fines recreativos, laborales y sociales, además de los diferentes avances tecnológicos ha obligado a las empresas de telefonía a innovar e ofrecer a sus clientes un mejor servicio ofreciendo una amplia variedad de planes variantes en megas de navegación, minutos todo destino, mensajes de texto y algunas apps como Facebook, Twitter, Waze sin cobro adicional. Donde el principal objetivo es lograr que el cliente este satisfecho y no vea la necesidad de cambiarse de compañía telefónica, esto se conoce como Portabilidad.

La Portabilidad se refiere a la posibilidad de que un cliente pueda mantener su número de teléfono, de línea fija o móvil, cuando decide cambiar de una compañía de teléfonos a otra compañía distinta de teléfonos o cuando se cambia de localidad o dirección. Esta transferencia de número en la mayoría de los casos está limitada por restricciones geográficas, área de cobertura o aspectos tecnológicos. Tradicionalmente, para los usuarios de servicios de telecomunicaciones ha sido necesario cambiar de número de teléfono cuando decide cambiar algunas de las condiciones mencionadas anteriormente, incluso manteniéndose en la misma empresa, ya sea por cambio de su domicilio o por necesidad de la compañía que le ofrece servicio previa notificación. La pérdida del número hace que los usuarios presenten cierta resistencia a analizar los servicios de otros actores del mercado, lo cual redunda en un estancamiento de la competitividad en este rubro.[6]

Desde una perspectiva de inteligencia para el negocio, se busca mitigar el riesgo a que suceda una portación. Por ende, se implementan metodologías que permitan entender ciertos comportamientos que tienen los clientes antes y después de que exista la portación. Para esto el uso de técnicas de Machine Learning se han convertido en un aliado ideal por su gran desempeño en el procesamiento de información y la implementación de métodos estadísticos para su evaluación.

2. Introducción a Machine Learning

2.1. ¿Qué es Machine Learning?

En 1945 cuando se desarrolló la primera computadora electrónica de uso general. Era Turing-Complete, digital y capaz de resolver una gran clase de problemas numéricos mediante la reprogramación. El nombre de la computadora era ENIAC (integrador numérico electrónico y computadora). Aunque ENIAC era una máquina destinada a realizar cálculos numéricos intensivos, la idea detrás de su desarrollo era construir una máquina para simular el pensamiento humano. Pero no pasó muchos años después del desarrollo de ENIAC y en 1950 Arthur Samuel, un pionero estadounidense en el campo de los juegos de computadora y la inteligencia artificial, acuñó el término "Aprendizaje automático" mientras estaba en IBM. También fue en ese momento que una computadora afirmó ayudar a los jugadores a mejorar su experiencia.

Al mismo tiempo, Frank Rosenblat inventó un clasificador muy simple llamado Perceptrón. El perceptrón estaba destinado a ser una máquina, en lugar de un programa. Cuando se combinó con grandes números se convirtió en un poderoso monstruo. Este monstruo en ese momento fue un verdadero avance. Luego se vio un estancamiento del aprendizaje automático y las redes neuronales debido a su dificultad para resolver algunos problemas. Gracias a las estadísticas, el aprendizaje automático se hizo muy famoso en la década de 1990. La intersección de la informática y las estadísticas dio origen a enfoques probabilísticos en IA. Esto cambió el campo aún más hacia enfoques basados en datos. Teniendo datos a gran escala disponibles, los científicos comenzaron a construir sistemas inteligentes que podían analizar y aprender de grandes cantidades de datos. Como punto culminante, el sistema Deep Blue de IBM venció al campeón mundial de ajedrez, el gran maestro Garry Kasparov. Sí, sé que Kasparov acusó a IBM de hacer trampa, pero esto es un pedazo de historia ahora y Deep Blue descansa tranquilamente en un museo. En el mundo real, estamos rodeados de humanos que pueden aprender todo de sus experiencias con su capacidad de aprendizaje, y tenemos computadoras o máquinas que funcionan según nuestras instrucciones. Pero, ¿puede una máquina también aprender de experiencias o datos pasados como lo hace un humano? Así que aquí viene el papel del aprendizaje automático**.**

El Machine Learning (ML) es un subconjunto de inteligencia artificial que se ocupa principalmente del desarrollo de algoritmos que permiten que una computadora aprenda de los datos y las experiencias pasadas por sí misma. También se puede definir tal cual como dijo Arthur Samuel en 1959 “El aprendizaje automático permite que una máquina aprenda automáticamente de los datos, mejore el rendimiento de las experiencias y prediga cosas sin ser programado explícitamente.” Con la ayuda de

datos históricos de muestra, conocidos como datos de entrenamiento**,** los algoritmos de Machine Learning crean un modelo matemático que ayuda a hacer predicciones o decisiones sin ser programado explícitamente. El Machine Learning une la informática y las estadísticas para crear modelos predictivos, donde se construye o utiliza los algoritmos que aprenden de los datos históricos. Cuanto más proporcionemos la información, mayor será el rendimiento, ya que se cree que una máquina tiene la capacidad de aprender si puede mejorar su rendimiento al obtener más datos.[7]

2.2. Clases de Algoritmos ML

Todos los algoritmos de aprendizaje automático se pueden clasificar por su tipo de aprendizaje y la tarea que se pretende realizar. Hay tres tipos de aprendizaje:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje de refuerzo

2.2.1. Aprendizaje Supervisado

Es un tipo de método de Machine Learning en el que proporcionamos datos etiquetados de muestra al sistema de aprendizaje automático para capacitarlo y, sobre esa base, predice el resultado. El sistema crea un modelo utilizando datos etiquetados para comprender los conjuntos de datos y aprender sobre cada uno de ellos, una vez que se realiza la capacitación y el procesamiento, probamos el modelo proporcionando una muestra de datos para verificar si está prediciendo el resultado exacto o no. El objetivo del aprendizaje supervisado es mapear los datos de entrada con los datos de salida. El aprendizaje supervisado se puede agrupar aún más en dos categorías de algoritmos:

Clasificación: Los algoritmos de clasificación toman datos etiquetados (porque son métodos de aprendizaje supervisados) y aprenden patrones en los datos que pueden usarse para predecir una variable categórica de salida. Esta suele ser una variable de agrupación (una variable que especifica a qué grupo pertenece un caso particular) y puede ser binomial (dos grupos) o multinomial (más de dos grupos). Los problemas de clasificación son tareas de aprendizaje automático muy comunes. ¿Spam o no? ¿Qué clientes incumplirán con sus pagos? ¿Qué pacientes sobrevivirán? ¿Qué objetos en una imagen de telescopio son estrellas, planetas o galaxias? Cuando se enfrente a problemas como estos, debe usar un algoritmo de clasificación. Ejemplo figura 2.

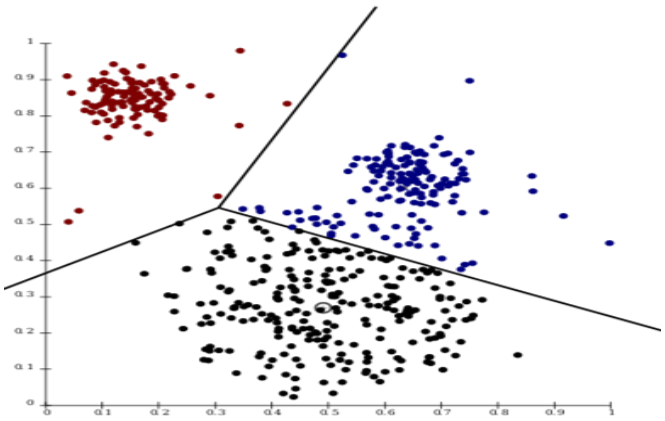

Figura 2. Objetivo de los Algoritmos de Clasificación.

Regresión: Los algoritmos de regresión toman datos etiquetados y aprenden patrones en los datos que pueden usarse para predecir una variable de salida. ¿Cuánto dióxido de carbono contribuye continuamente un hogar a la atmósfera? ¿Cuál será el precio de la acción de una empresa mañana? ¿Cuál es la concentración de insulina en la sangre de un paciente? Cuando te enfrentas a problemas como estos, debes usar un algoritmo de regresión. Ejemplo figura 3.

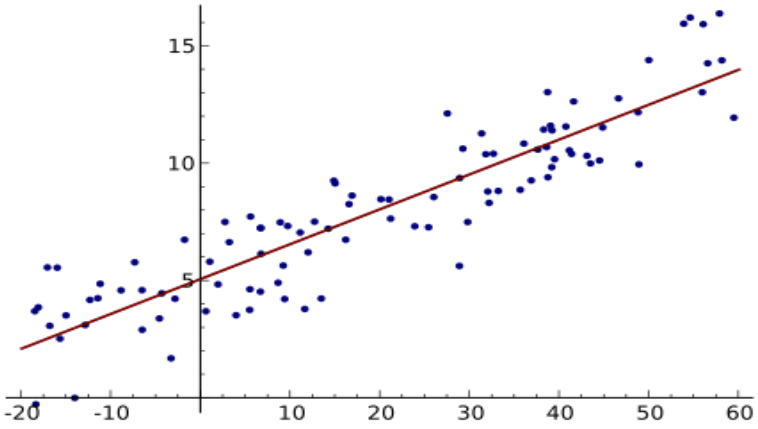

Figura 3. Objetivo de los Algoritmos de Regresión.

Estos dos tipos de aprendizaje supervisado se distinguen por el tipo de variable objetivo. En los casos de clasificación, es de tipo categórico, mientras que, en los casos de regresión, la variable objetivo es de tipo numérico. Los algoritmos más habituales que aplican para el aprendizaje supervisado son: Árboles de decisión, Clasificación de Naïve Bayes, Regresión por mínimos cuadrados, Regresión Logística, Support Vector Machines (SVM) y Métodos "Ensemble" (Conjuntos de clasificadores).[8]

La mayoría de los algoritmos de clasificación de Machine Learning son sensibles al desequilibrio en las clases de predictores. Los datos desequilibrados se refieren a problemas de clasificación en los que tenemos instancias desiguales para diferentes

clases. Un conjunto de datos desequilibrado puede conducir a resultados inexactos incluso cuando se utilizan modelos brillantes para procesar los datos. Si los datos están sesgados, los resultados también estarán sesgados.[9] por ende se hace recomendable utilizar técnicas de remuestreo las cuales implican la creación de una nueva versión transformada del conjunto de datos en el que los ejemplos seleccionados tienen una distribución de clase diferente. La estrategia más simple es elegir ejemplos para el conjunto de datos transformado al azar, llamado remuestreo aleatorio. Hay dos enfoques principales para el muestreo aleatorio para la clasificación desequilibrada los cuales son el Oversampling (Sobre-Muestreo), el UnderSampling (submuestreo) y remuestreo con igual probabilidad.

2.2.2. Aprendizaje no Supervisado.

El aprendizaje no supervisado es un método de aprendizaje en el que una máquina aprende sin supervisión. La capacitación se proporciona a la máquina con el conjunto de datos que no han sido etiquetados, clasificados o categorizados, y el algoritmo necesita actuar sobre esos datos sin ninguna supervisión. El objetivo del aprendizaje no supervisado es reestructurar los datos de entrada en nuevas características o un grupo de objetos con patrones similares. En el aprendizaje no supervisado, no tenemos un resultado predeterminado. La máquina trata de encontrar información útil a partir de la gran cantidad de datos. Puede clasificarse en dos categorías de algoritmos:

Reducción de Dimencionalidad: Los algoritmos de reducción de dimensiones toman datos sin etiquetar (porque son métodos de aprendizaje no supervisados), de alta dimensión (datos con muchas variables) y aprenden una forma de representarlos en un menor número de dimensiones. Las técnicas de reducción de dimensiones se pueden usar como una técnica exploratoria (porque es muy difícil para los humanos interpretar visualmente los datos en más de dos o tres dimensiones a la vez), o como un paso de pre procesamiento en nuestra tubería de aprendizaje automático (puede ayudar a mitigar los problemas como y colinealidad).

Agrupación: Los algoritmos de agrupamiento toman datos sin etiquetar y aprenden patrones de agrupamiento en los datos. Un grupo es una colección de observaciones que son más similares entre sí que a los puntos de datos en otros grupos. Suponemos que las observaciones en el mismo grupo comparten algunas características o identidad unificadoras que las hacen identificablemente diferentes de otros grupos. Los algoritmos de agrupación pueden usarse como una técnica exploratoria para comprender la estructura de nuestros datos, y pueden indicar una estructura de agrupación que se puede alimentar a los algoritmos de clasificación. ¿Hay subtipos de pacientes que responden en un ensayo clínico? ¿Cuántas clases de encuestados hubo

en la encuesta? ¿Existen diferentes tipos de clientes que utilizan nuestra empresa? Cuando se enfrente a problemas como estos, debe usar un algoritmo de agrupamiento.

2.2.3. Aprendizaje por Refuerzo.

El aprendizaje por refuerzo consiste en aprender a decidir, ante una situación determinada, que acción es la más adecuada para lograr un objetivo. Consta de dos componentes. Componente selectiva que involucra la selección de la mejor acción a ejecutar de entre varias opciones y la componente asociativa, en el sentido de que las alternativas encontradas se asocian a situaciones particulares en que se tomaron. El aprendizaje por refuerzo es adecuado cuando no existe un conocimiento "a priori" del entorno o este es demasiado complejo como para utilizar otros métodos.

El aprendizaje puede ser pasivo y activo. En el aprendizaje pasivo la política del agente está fijada y la tarea es aprender las utilidades de los estados (o parejas estado acción) mientras que el aprendizaje activo el agente debe aprender también que hacer. El esfuerzo activo además de recoger información del entorno utilizará ésta para tomar decisiones sobre la siguiente acción a realizar. El objeto del aprendizaje mediante el refuerzo es un comportamiento que permite resolver problemas óptimamente. Un comportamiento no es más que un conjunto de acciones que se realizan para resolver un problema y política al conjunto de acciones que se realizan en cada situación para resolver un problema.[10]

2.3. Metodología CRISP-DM para la Creación de Modelos ML

Las técnicas de Data Science o Data Analytics, que tanto interés despiertan hoy en día, en realidad surgieron en la década de los 90, cuando se usaba el término KDD (Knowledge Discovery in Databases) para referirse al amplio concepto de hallar conocimiento en los datos. En un intento de normalización de este proceso de descubrimiento de conocimiento, de forma similar a como se hace en ingeniería software para normalizar el proceso de desarrollo software, surgieron a finales de los 90 dos metodologías principales: CRISP-DM (Cross Industry Standard Process for Data Mining) y SEMMA (Sample, Explore, Modify, Model, and Assess). Ambas especifican las tareas a realizar en cada fase descrita por el proceso, asignando tareas concretas y definiendo lo que es deseable obtener tras cada fase. (Azevedo y Santos, 2008) compara ambas implementaciones y llega a la conclusión de que, aunque se puede establecer un paralelismo claro entre ellas, CRISP-DM es más completo porque tiene en cuenta la aplicación al entorno de negocio de los resultados.

CRISP-DM proporciona una descripción normalizada del ciclo de vida de un proyecto estándar de análisis de datos, de forma análoga a como se hace en la ingeniería del software con los modelos de ciclo de vida de desarrollo de software. El modelo CRISP-DM cubre las fases de un proyecto, sus tareas respectivas, y las relaciones entre estas tareas. En este nivel de descripción no es posible identificar todas las relaciones; las relaciones podrían existir entre cualquier tarea según los objetivos, el contexto, y el interés del usuario sobre los datos.

La metodología CRISP-DM contempla el proceso de análisis de datos como un proyecto profesional, estableciendo así un contexto mucho más rico que influye en la elaboración de los modelos. Este contexto tiene en cuenta la existencia de un cliente que no es parte del equipo de desarrollo, así como el hecho de que el proyecto no sólo no acaba una vez se halla el modelo idóneo (ya que después se requiere un despliegue y un mantenimiento), sino que está relacionado con otros proyectos, y es preciso documentarlo de forma exhaustiva para que otros equipos de desarrollo utilicen el conocimiento adquirido y trabajen a partir de él.

El ciclo de vida del proyecto de minería de datos consiste en seis fases como la figura 4 lo muestra.

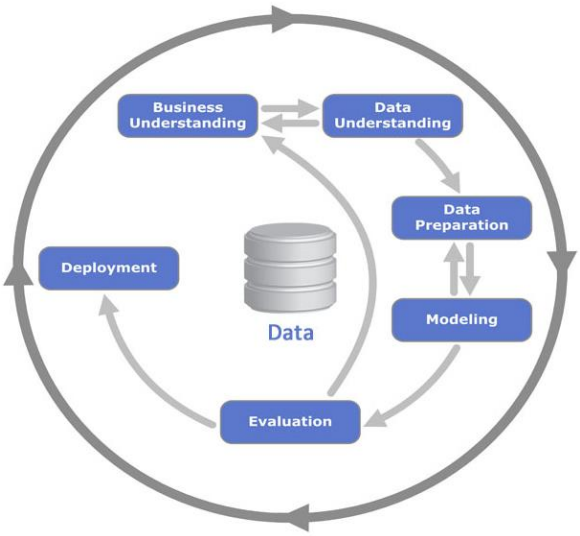

Figura 4. Fases de la Metodología CRISP-DM.

La secuencia de las fases no es rígida (se permite movimiento hacia adelante y hacia atrás entre diferentes fases). El resultado de cada fase determina qué fase, o qué tarea particular de una fase, hay que hacer después. Las flechas indican las dependencias más importantes y frecuentes.

El círculo externo en la figura simboliza la naturaleza cíclica de los proyectos de análisis de datos. Sin embargo, en esta metodología el proyecto no se termina una vez que la solución se despliega, si no que continúa en procesos de verificación y seguimiento de los resultados.[11]

2.3.1. Fases de la Metodología CRISP-DM

I. **Entendimiento del Negocio:** Esta fase inicial se enfoca en la comprensión de los objetivos de proyecto. Después se convierte este conocimiento de los datos en la definición de un problema de minería de datos y en un plan preliminar diseñado para alcanzar los objetivos.

II. **Comprensión de los Datos:** La fase de entendimiento de datos comienza con la colección de datos inicial y continúa con las actividades que permiten familiarizarse a grandes rasgos con los datos, además de poder partir de hipótesis de lo que pueda estar sucediendo.

III. **Preparación de los Datos:** La fase de preparación de datos cubre todas las actividades necesarias para construir el conjunto final de datos. Las tareas incluyen la selección de tablas, registros y atributos, así como la transformación y la limpieza de datos para las herramientas que modelan.

IV. **Modelado:** En esta fase, se seleccionan y aplican las técnicas de modelado que sean pertinentes al problema (cuantas más mejor), y se calibran sus parámetros a valores óptimos. Típicamente hay varias técnicas para el mismo tipo de problema. Algunas técnicas tienen requerimientos específicos sobre la forma de los datos. Por lo tanto, casi siempre en cualquier proyecto se acaba volviendo a la fase de preparación de datos.

V. **Evaluación:** En esta etapa en el proyecto, se han construido uno o varios modelos que parecen alcanzar calidad suficiente desde la una perspectiva de análisis de datos. Antes de proceder al despliegue final del modelo, es importante evaluarlo a fondo y revisar los pasos ejecutados, luego comparar el modelo obtenido con los objetivos de negocio.

VI. **Despliegue:** Generalmente, la creación del modelo no es el final del proyecto. Incluso si el objetivo del modelo es de aumentar el conocimiento de los datos, el conocimiento obtenido tendrá que organizarse y presentarse para que el cliente pueda usarlo. Dependiendo de los requisitos, la fase de desarrollo puede ser tan simple como la generación de un informe o tan compleja como la realización periódica y quizás automatizada de un proceso de análisis de datos en la organización.

2.4. Implementación de H2O 3 y R Studio para la Construcción de Modelos ML.

2.4.1. Que es R y R Studio?

R es un ambiente de programación formado por un conjunto de herramientas muy flexibles que pueden ampliarse fácilmente mediante paquetes, librerías o definiendo

nuestras propias funciones**.** Además es gratuito y de código abierto. R se presentó al mercado en 1993 de la mano de sus creadores Robert Gentleman y Ross Ihaka, que desarrollaron la herramienta en el Departamento de Estadística de la Universidad de Auckland. Sin embargo la base de sus orígenes se encuentra en el desarrollo del lenguaje S. R es producto de la filosofía Open Source**.** Desde sus inicios una extensa comunidad de usuarios y programadores de alto nivel contribuye a desarrollar nuevas funciones, paquetes y actualizaciones que son rápidamente accesibles a todo público de forma libre y gratuita. Esto convierte a R software en una herramienta estadística estable, confiable y a la vanguardia, ya que está sometida a una actualización permanente. La actividad de los desarrolladores de R se organiza en torno a la R Foundation, como parte oficial de la Free Software Foundation.

Ahora, por un lado, R es el lenguaje de programación, como el que hace las cuentas. Por otro lado, R Studio es un IDE (entorno de desarrollo integrado). En español, eso significa que R Studio es un programa para manejar R y utilizarlo de manera más cómoda en algunos aspectos. Cuando utilizamos R Studio tenemos ciertos paneles que nos hacen la vida más fácil, las mismas cosas se pueden hacer más rápido y con mejor organización. [12]

En R se puede utilizar muchos tipos de modelos estadísticos desde regresión lineal hasta redes neuronales y análisis de series de tiempo. R también se usa para graficar 2D y 3D, con sus librerías interactivas como lo es Plotly, que permiten al usuario una experiencia más cercana con el comportamiento de los datos.

2.4.2. Que es H2O?

H2O es una plataforma de aprendizaje automático y análisis predictivo de código abierto, en memoria, distribuida, rápida y escalable que permite crear modelos de aprendizaje automático en big data y que además proporciona una fácil producción de esos modelos en un entorno empresarial. El código principal de H2O está escrito en Java. Dentro de H2O, un almacén de clave distribuido se utiliza para acceder y hacer referencia a datos, modelos, objetos, etc., en todos los nodos y máquinas. Los algoritmos se implementan sobre el marco distribuido Map/Reduce de H2O y utilizan el marco Java Fork/Join para multiproceso. Los datos se leen en paralelo y se distribuyen por el clúster y se almacenan en memoria en un formato en columnas de forma comprimida.

El analizador de datos de H2O tiene inteligencia integrada para adivinar el esquema del conjunto de datos entrante y admite la ingesta de datos de varios orígenes en varios formatos. La API REST de H2O permite el acceso a todas las capacidades de H2O desde un programa externo o script a través de JSON a través de HTTP. La API Rest

es utilizada por la interfaz web de H2O (Flow UI), el enlace R (H2O-R) y el enlace de Python (H2O-Python). La velocidad, la calidad, la facilidad de uso y la implementación de modelos para los diversos algoritmos supervisados y no supervisados de vanguardia como Deep Learning, Tree Ensembles y GLRM hacen de H2O una API muy buscada para la ciencia de big data.

2.4.3. Como R interactúa con H2O?

El paquete H2O para R permite a los usuarios de R controlar un clúster de H2O desde un script de R. El script de R es un cliente de API REST del clúster de H2O. Los datos nunca fluyen a través de R, solo es una vía de comunicación. La siguiente secuencia de tres pasos muestra cómo un programa de R indica a un clúster de H2O que lea datos de HDFS en un fotograma H2O distribuido.

PASO 1: El usuario de R llama a la función importFile en caso que se tengan los datos desde un archivo externo como csv o xlxs; Si los datos están desde el entorno de R solo basta con la función as.h2o.

PASO 2: El cliente R le dice al clúster que lea los datos. Las flechas delgadas en la figura 5, muestran información de control.

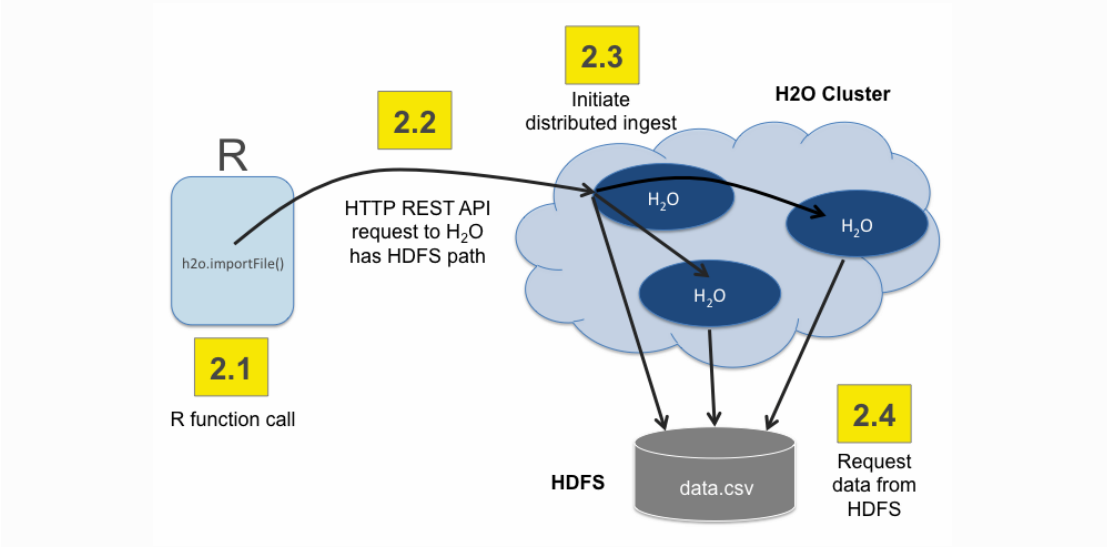

Figura 5. Lectura de los datos de R a H2O.

PASO 3: Los datos se devuelven desde HDFS a un marco H2O distribuido. Las flechas gruesas muestran los datos que se devuelven desde HDFS como se observa en la figura

6. Los bloques de datos viven en la memoria distribuida del clúster de H2O Frame. [13]

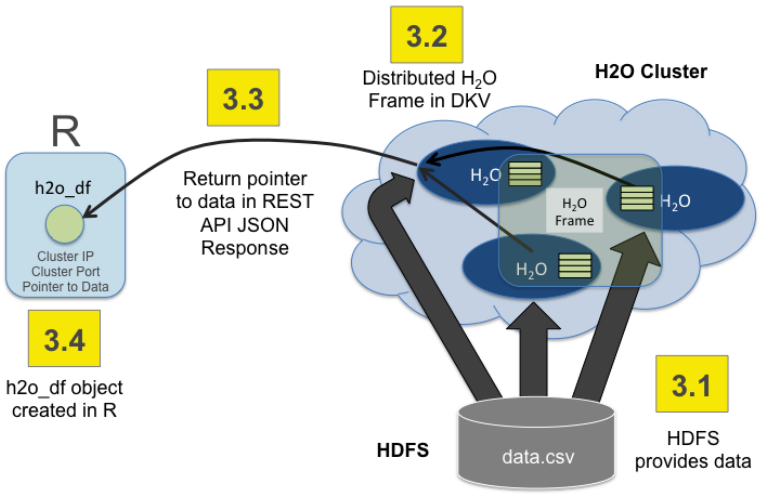

Figura 6. Lectura de datos de H2O a R.

3. Marco Teórico.

3.1. Métodos de Impulso

Los métodos de impulso son una de las ideas de aprendizaje más poderosas introducidas en los últimos veinte años. Originalmente se diseñó para problemas de clasificación, pero también puede extenderse de manera provechosa a la regresión. La motivación para impulsar estos métodos fue un procedimiento que combina los resultados de muchos clasificadores "débiles" para producir un "grupo" poderoso. Desde la implementación de la ciencia de datos en la mayoría de industrias, el hecho de impulsar algoritmos es suficiente para la mayoría de las tareas de aprendizaje predictivo, al menos para ahora ya que potentes, flexibles y se pueden interpretar bien.

AdaBoost, abreviatura de "Adaptive Boosting", es el primer algoritmo de impulso práctico propuesto por Freund y Schapire en 1996. Se centra en los problemas de clasificación y tiene como objetivo convertir un conjunto de clasificadores débiles en uno fuerte. Considere un problema de dos clases, con la variable de salida codificada como $Y \in \{-1,1\}$, un vector de variables predictoras X, y un clasificador $G(x)$, así se produce una predicción tomando uno de los dos valores $\{-1,1\}$. La tasa de error en la muestra de entrenamiento se define como

$$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I\big(y_i \neq G(x_i)\big)$$

Y la tasa de error esperada en predicciones futuras es $E_{XY}I\big(Y \neq G(X)\big)$. Un clasificador débil es aquel cuya tasa de error es solo un poco mejor que la suposición aleatoria. El propósito del refuerzo es aplicar secuencialmente el algoritmo de clasificación débil a versiones modificadas repetidamente de los datos, produciendo así una secuencia de clasificadores débiles $Gm\,(x), m = 1, 2, \ldots, M$. Las predicciones de todos ellos se combinan luego mediante un voto mayoritario ponderado para producir la predicción final:

$$G(x) = sign(\textstyle\sum_{m=1}^{M} \alpha_m G_m(x)) \qquad (3.1)$$

Aquí $\propto_1, \propto_2, \ldots, \propto_M$ se calculan mediante el algoritmo de refuerzo y pesan la contribución de cada $G_m(x)$ respectivo. Su efecto es dar una mayor influencia a los clasificadores más precisos de la secuencia.

Las modificaciones de datos en cada paso de refuerzo consisten en aplicar pesos $w_1, w_2, \dots, w_N$ a cada una de las observaciones de entrenamiento $(x_i, y_i), i = 1,2, \dots, N$. Inicialmente, todos los pesos se establecen en $w_i = 1/N$, de modo que el primer paso simplemente entrena al clasificador en los datos de la manera habitual. Para cada iteración sucesiva $m = 2, 3, \dots, M$ los pesos de observación se modifican individualmente y el algoritmo de clasificación se vuelve a aplicar a las observaciones ponderadas. En el paso m, las observaciones que fueron clasificadas erróneamente por el clasificador $G_m - 1(x)$ inducido en el paso anterior tienen sus pesos aumentados, mientras que los pesos disminuyen para aquellas que se clasificaron correctamente. Por tanto, a medida que avanzan las iteraciones, las observaciones que son difíciles de clasificar correctamente reciben una influencia cada vez mayor. Por lo tanto, cada clasificador sucesivo se ve obligado a concentrarse en aquellas observaciones de entrenamiento que se pierden por las anteriores en la secuencia. Así entonces, el algoritmo AdaBoost funciona de la siguiente manera ilustrado en 3.1

1. Inicializar los pesos de observación $w_i = 1/N,\ i = 1,2, \dots, N$.
2. Para $m = 1\ hasta\ M$

    a. Ajuste un clasificador $G_m(x)$ a los datos de entrenamiento usando pesos w_i.
    b. Calcular

$$err_m = \frac{\sum_{i=1}^{N} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

    c. Calcular $\propto_m = \log((1 - err_m)/err_m))$
    d. Establecer $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot I\left(y_i \neq G_m(x_i)\right)\right], i = 1,2, \dots, N$

3. Salida $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

El algoritmo muestra como el clasificador actual $G_m(x)$ se induce en las observaciones ponderadas en la línea 2a. La tasa de error ponderada resultante se calcula en la línea 2b. La línea 2c calcula el peso $\propto_m$ dado a $G_m(x)$ al producir el clasificador final $G(x)$. Los pesos individuales de cada una de las observaciones se actualizan para la siguiente iteración en la línea 2d. Las observaciones mal clasificadas por $G_m(x)$ tienen sus pesos escalados por un factor $\exp(\propto_m)$, aumentando su influencia relativa para inducir el siguiente clasificador $G_m + 1(x)$ en la secuencia.

3.1.1. Funciones de Pérdida y Robustez

Definimos ML como las técnicas usadas para que una máquina mejore su rendimiento en una tarea específica a partir de experiencias previas. Para esto, se requiere de un proceso iterativo en el que se busca minimizar el error de la maquina realizando dicha tarea, lo que se conoce como entrenar el modelo. Una función de pérdida J(x) mide que tan insatisfechos estamos con las predicciones de nuestro modelo con respecto a una respuesta correcta y utilizando ciertos valores de θ. Existen varias funciones de pérdida como el Error cuadrático medio, entropía cruzada, desviación exponencial o binomial y la selección de uno de ellos depende de varios factores como el algoritmo seleccionado o el nivel de confianza deseado, pero principalmente depende del objetivo de nuestro modelo.

Aunque tanto la desviación exponencial como la binomial producen la misma solución cuando se aplican a la distribución conjunta de la población, no ocurre lo mismo con conjuntos de datos finitos. Ambos criterios son funciones monótonas decrecientes del "margen" $yf(x)$. En la clasificación (con una respuesta de $-1|1$) el margen juega un papel análogo a los residuos $y - f\,(x)$ en la regresión. La regla de clasificación $G(x) = sign[f(x)]$ implica que las observaciones con margen positivo $y_i f(x_i) > 0$ se clasifican correctamente mientras que aquellas con margen negativo $y_i f(x_i) < 0$ se clasifican erróneamente. El límite de decisión está definido por $f(x) = 0$. El objetivo del algoritmo de clasificación es producir márgenes positivas con la mayor frecuencia posible. Cualquier criterio de pérdida utilizado para la clasificación debería penalizar más los márgenes negativos que los positivos, ya que las observaciones de márgenes positivos ya están correctamente clasificadas. Tanto la pérdida exponencial como la de desviación pueden verse como aproximaciones continuas y monótonas a la pérdida de clasificación errónea. Continuamente penalizan más fuertemente los valores de margen cada vez más negativos de lo que recompensan a los cada vez más positivos. La diferencia entre ellos está en grado. La penalización asociada con la desviación binomial aumenta linealmente para un gran margen cada vez más negativo, mientras que el criterio exponencial aumenta exponencialmente la influencia de tales observaciones.

En cualquier punto del proceso de entrenamiento, el criterio exponencial concentra mucha más influencia en las observaciones con grandes márgenes negativos. La desviación binomial concentra relativamente menos influencia en tales observaciones, extendiendo la influencia de manera más uniforme entre todos los datos. Por lo tanto, es mucho más robusto en entornos ruidosos donde la tasa de error de Bayes no es cercana a cero, y especialmente en situaciones donde hay una especificación incorrecta de las etiquetas de clase en los datos de entrenamiento. Se ha observado empíricamente que el rendimiento de AdaBoost se degrada drásticamente en tales situaciones. Así que el minimizador del riesgo correspondiente en la población se da de la siguiente manera

$$f*(x) = argmin\, E_{Y|x}\big(Y - f(x)\big)^2 = E(Y|x) = 2 \cdot \Pr(Y = 1|x) - 1 \qquad (3.2)$$

Como antes, la regla de clasificación es $G(x) = sign[f(x)]$. La pérdida por error al cuadrado no es un buen sustituto del error de clasificación errónea, ya que no hay una función de pérdida única para todos los algoritmos en el aprendizaje automático. Hay varios factores involucrados en la elección de una función de pérdida para un problema específico, como el tipo de algoritmo de aprendizaje automático elegido, la facilidad de cálculo de los derivados y en cierta medida, el porcentaje de valores atípicos en el conjunto de datos. En términos generales, las funciones de pérdida se pueden clasificar en dos categorías principales dependiendo del tipo de tarea de aprendizaje con la que estamos tratando: Funciones de pérdidas para regresión y clasificación.

- **Error Cuadrático Medio (MSE):** el error cuadrado medio se mide como el promedio de la diferencia cuadrada entre las predicciones y las observaciones reales. Sólo se refiere a la magnitud media del error independientemente de su dirección. Sin embargo, debido a la cuadración, las predicciones que están lejos de los valores reales se penalizan en gran medida en comparación con las predicciones menos desviadas. Además MSE tiene buenas propiedades matemáticas que hace que sea más fácil calcular degradados.

$$\text{MSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \qquad (3.3)$$

- **Error Absoluto Medio (MAE):** El error absoluto medio, por otro lado, se mide como el promedio de suma de las diferencias absolutas entre las predicciones y las observaciones reales. Al igual que MSE, esto también mide la magnitud del error sin considerar su dirección. A diferencia de MSE, MAE necesita herramientas más complicadas, como la programación lineal para calcular los degradados. Además MAE es más robusto para los valores atípicos ya que no hace uso de cuadrado.

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \qquad (3.4)$$

- **Error de Sesgo Medio (MBE):** Esto es mucho menos común en el dominio de aprendizaje automático en comparación con su contraparte. Esto es lo mismo que MSE con la única diferencia de que no tomamos valores absolutos. Claramente hay una necesidad de precaución, ya que los errores positivos y negativos podrían cancelarse entre sí. Aunque es menos preciso en la práctica, podría determinar si el modelo tiene sesgo positivo o sesgo negativo.

$$MBE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n} \qquad (3.5)$$

- **Pérdida de Huber:** La pérdida Huber combina las mejores propiedades de MSE y MAE. Es cuadrático para errores más pequeños y es lineal de otra manera (y de manera similar para su gradiente). Se identifica por su parámetro delta:

$$L_\delta = \begin{cases} \frac{1}{2}(y - f(x))^2, if\ |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, \quad en\ otro\ caso \end{cases} \qquad (3.6)$$

La pérdida de Huber es más robusta para los valores atípicos que el MSE. Se utiliza en Regresión robusta, M-estimación **y** Modelado aditivo. Una variante de Huber Loss también se utiliza en la clasificación.

- **Entropía Cruzada o Perdida de Logaritmo:** La pérdida de entropía cruzada, o pérdida de logaritmo, mide el rendimiento de un modelo de clasificación cuya salida es un valor de probabilidad entre 0 y 1. La pérdida de entropía cruzada aumenta a medida que la probabilidad pronosticada diverge de la etiqueta real. Así que predecir una probabilidad de 0.012 cuando la etiqueta de observación real es 1 sería malo y resultaría en un valor de pérdida alto. Un modelo perfecto tendría una pérdida de registro de 0. En un algoritmo de clasificación binaria como la regresión logística, el objetivo es minimizar la función de entropía cruzada.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log\big(p(y_i)\big) + (1 - y_i) \cdot \log(1 - p(y_i)) \qquad (3.7)$$

La entropía cruzada es una medida de la diferencia entre dos distribuciones de probabilidad para una determinada variable aleatoria o un conjunto de eventos según "Jasón Brownlee", en la cual si consideremos que tenemos datos de pacientes y la tarea es encontrar qué pacientes tienen cáncer. En nuestro ejemplo, como no tenemos los datos de toda la población, tratamos de predecir la probabilidad de que una persona tenga cáncer a partir de una muestra de datos. Solo necesitamos predecir la clase maligna $i.e\ P(y = 1|x) = \hat{p}$ porque la probabilidad de la clase negativa se puede derivar de ella, es decir $P(y = 0|x) = 1 - P(y = 1|x) = 1 - \hat{p}$. Un buen algoritmo de clasificación binaria debería producir un valor alto de $\hat{p}$(probabilidad de predecir la clase maligna para una muestra S) que es la estimación más cercana a P(probabilidad de predecir la clase maligna de la población total).

La idea es encontrar el máximo de la función de verosimilitud para un valor particular de θ.

$$L = \prod_{i=1}^{N} f(x_i|\theta) \qquad (3.8)$$

Encontrar un máximo de una función significa diferenciar la función $(dL/d\theta = 0)$ Como la función de verosimilitud L es un producto de la función de distribución de probabilidad de cada Xi, tenemos que usar la regla del producto en la diferenciación para diferenciar dicha función, lo que se convertirá en una tarea complicada. La aplicación de log a la función de probabilidad simplifica la expresión en una suma del logaritmo de probabilidades y no cambia la gráfica con respecto a θ. Además, diferenciar el logaritmo de la función de verosimilitud dará el mismo valor estimado θ debido a la propiedad monótona de la función logarítmica. Esta transformación de la función de verosimilitud ayuda a encontrar el valor de θ, que maximiza esta

$$Log(L) = logf(x_1|\theta) + logf(x_2|\theta) + \cdots + logf(x_N|\theta) \quad (3.9)$$

La expresión también se denomina distribución de Bernoulli. En el caso del ejemplo, la probabilidad de que el cáncer sea maligno es P. La probabilidad de que el cáncer sea benigno será $1 - P$. En una circunstancia de N observaciones, se da una función de densidad de probabilidad f como producto de funciones de densidad de probabilidad individuales. La probabilidad conjunta se define como sigue

$$f(x_i|P) = P^{x_i}(1-P)^{1-x_j} \quad (3.10)$$

$$xi = \begin{cases} 1\ Maligno \\ 0\ benigno \end{cases}$$

$$f(1|P) = P^1(1-P)^{1-1} = P \quad (3.11)$$

$$f(0|P) = P^0(1-P)^1 = 1 - P \quad (3.12)$$

$$f(x_1, x_2, x_3, \dots, x_N|P) = P^{x_1}(1-P)^{1-x}P^{x_2}(1-P)^{1-x_2} \dots P^{x_N}(1-P)^{1-x_N}$$

$$P(X_1 = x_1, X_2 = x_2, \dots, X_N = x_N) = \sum_{i=1}^{N} P^{x_i}(1-P)^{1-x_i} \quad (3.13)$$

Para la estimación de máxima verosimilitud, tenemos que calcular qué valor de P es $dL/dP = 0$, por lo que, como se discutió anteriormente; la función de verosimilitud se transforma en una función logarítmica de verosimilitud.

$$L = \sum_{i=1}^{N} P^{x_i}(1-P)^{1-x_i} \quad (3.14)$$

Como (3.14) aplicando logaritmo a ambos lados se tiene

$$Log\ L = \log[\sum_{i=1}^{N} P^{x_i}(1-P)^{1-x_i}]$$

$$= \sum_{i=1}^{N} \log(P^{x_i}(1-P)^{1-x_i})$$

$$= \sum_{i=1}^{N} x_i logP + (1-x_i)\log(1-P) \quad (3.15)$$

Como puede ver, se deriva una ecuación que es casi similar a la función de pérdida de registro / entropía cruzada solo que sin el signo negativo. En Regresión logística, el descenso de gradiente se usa para encontrar el valor óptimo en lugar del ascenso de gradiente porque se considera como un problema de minimización de pérdida, por lo que aquí es donde agregamos el signo negativo a la ecuación que da como resultado la función de pérdida de entropía cruzada binaria. .

$$M = -\big(ylog(p) + (1-y)log(1-p)\big) \quad (3.16)$$

Si es clasificación multiclase, calculamos una pérdida separada para cada etiqueta de clase por observación y sumamos el resultado. $M > 2$

$$M = -\sum_{c=1}^{M} y_{o,c} \log\big(p_{o,c}\big) \quad (3.17)$$

Siendo M el número de clases, y, el indicador binario $(0\ o\ 1)$ si la etiqueta de clase es la clasificación correcta para la observación co y p siendo la

observación de probabilidad pronosticada de la clase oc. Además, observe que maximizar la función de probabilidad logarítmica es lo mismo que minimizar la función de probabilidad logarítmica negativa. La función de pérdida calcula el error para un solo ejemplo de entrenamiento; la función de costo es el promedio de las funciones de pérdida de todo el conjunto de entrenamiento. [14]

$$= -\frac{1}{N}\sum_{i=1}^{N} x_i logP + (1 - x_i)\log(1 - P)$$

Reemplazando $x_i \; con \; y_i$.

$$= -\frac{1}{N}\sum_{i=1}^{N} ylogP + (1 - y)\log(1 - P) \qquad (3.18)$$

3.2. Arboles de decisión

Los árboles de regresión y clasificación dividen el espacio de todos los valores de las variables predictoras conjuntas en regiones disjuntas $R_j, j = 1,2,3, \dots, J$, representado por los nodos terminales del árbol. Se asigna una constante γ_j a cada región y la regla predictiva es

$$x \in R_j \Rightarrow f(x) = \gamma_j$$

Así, un árbol puede expresarse formalmente como

$$T(x;\theta) = \sum_{j=1}^{J} \gamma_j \, I(x \in R_j) \qquad (3.19)$$

Con parámetros $\theta = \{R_j, \gamma_j\}_1^J$. J se suele tratar como un meta-parámetro. Los parámetros se encuentran minimizando el riesgo empírico

$$\hat{\theta} = arg\min_{\theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j) \qquad (3.20)$$

Este es un formidable problema de optimización combinatoria, y generalmente nos conformamos con soluciones aproximadas sub-óptimas. Es útil dividir el problema de optimización en dos partes donde

- **Hallando $\boldsymbol{\gamma_j}$ dado $\boldsymbol{R_j}$:** Dado el R_j, estimar el γ_j es típicamente trivial, ya menudo $\widehat{\gamma_j} = \overline{y_j}$, la media de y_i que cae en la región R_j. Para la pérdida de clasificación errónea, $\widehat{\gamma_j}$ es la clase modal de la observación que cae en la región R_j.

- **Hallando $\boldsymbol{R_j}$:** esta es la parte difícil, para la que se encuentran soluciones aproximadas. Tenga en cuenta también que encontrar el R_j implica estimar el γ_j también. Una estrategia típica es utilizar un algoritmo de particionamiento

recursivo de arriba hacia abajo codicioso para encontrar el R_j. Además, a veces es necesario aproximar (3.20) mediante un criterio de suavizamiento y conveniente para optimizar el R_j:

$$\tilde{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} \tilde{L}(y_i, T(x_i, \theta)) \qquad (3.21)$$

Entonces, dado el $\hat{R}_j = \tilde{R}_j$, el γ_j se puede estimar con mayor precisión utilizando el criterio original. El índice de Gini reemplaza la pérdida de clasificación errónea en el crecimiento del árbol (identificando el R_j). El modelo de árbol impulsado es una suma de tales árboles,

$$f_M(x) = \sum_{m=1}^{M} T(x; \theta_M) \qquad (3.22)$$

En cada paso del procedimiento progresivo por etapas, se debe resolver

$$\tilde{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \theta_m)) \qquad (3.23)$$

Para el conjunto de regiones y constantes $\theta_m = \{R_{jm}, \gamma_{jm}\}_1^{Jm}$ del siguiente árbol, dado el modelo actual $f_m - 1(x)$.

Dadas las regiones R_{jm}, encontrar las constantes óptimas γ_{jm} en cada región suele ser sencillo:

$$\hat{\gamma}_{jm} = \arg\min_{\gamma_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm}) \qquad (3.24)$$

Encontrar las regiones es difícil e incluso más difícil que para un solo árbol. Para algunos casos especiales, el problema se simplifica. Para la pérdida de error al cuadrado, la solución a (3.23) no es más difícil que para un solo árbol. Es simplemente el árbol de regresión que mejor predice los residuos actuales $y_i - f_{m-1}(x_i)$ y $\hat{\gamma}_{jm}$ es la media de estos residuos en cada región correspondiente.

Para la clasificación de dos clases y la pérdida exponencial, este enfoque por etapas da lugar al método AdaBoost para impulsar los árboles de clasificación. En particular, si los árboles $T(x; \theta_M)$ están restringidos a ser árboles de clasificación escalados, el árbol que minimiza la tasa de error ponderada $\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq T(x_i; \theta_m)$ con pesos $w_i^{(m)} = e^{-y_i f_{m-1(x_i)}}$. Por árbol de clasificación escalado, nos referimos a $\beta_m T(x; \theta_m)$ con la restricción de que $\gamma_{jm} \in \{-1,1\}$.

Sin esta restricción, (3.23) aún se simplifica para la pérdida exponencial a un criterio exponencial ponderado para el nuevo árbol:

$$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} w_i^{(m)} \exp[-y_i T(x_i; \theta_m)] \qquad (3.25)$$

Es sencillo implementar un algoritmo codicioso de partición recursiva utilizando esta pérdida exponencial ponderada como criterio de división. Dado el R_{jm}, se puede

demostrar que la solución de (3.24) son las probabilidades logarítmicas ponderadas en cada región correspondiente

$$\hat{\gamma}_{jm} = \log \frac{\sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i=1)}{\sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i=-1)} \qquad (3.26)$$

Esto requiere un algoritmo especializado de crecimiento de árboles; en la práctica, preferimos la aproximación que utiliza un árbol de regresión de mínimos cuadrados ponderados. El uso de criterios de pérdida como el error absoluto (3.4) o la pérdida de Huber (3.6) en lugar de la pérdida por error al cuadrado (3.3) para la regresión, y la desviación en lugar de la entropía cruzada (3.7), servirá para robustecer los árboles impulsores. Desafortunadamente, a diferencia de sus contrapartes no robustas, estos criterios sólidos no dan lugar a algoritmos simples de impulso rápido. Para criterios de pérdida más generales, la solución a (3.24), dado el R_{jm}, suele ser sencilla, ya que es una estimación simple de "ubicación". Por pérdida absoluta es solo la mediana de los residuos en cada región respectiva. Para los otros criterios, existen algoritmos iterativos rápidos para resolver (3.24), y generalmente sus aproximaciones más rápidas de “un solo paso” son adecuadas. El problema es la inducción de árboles. No existen algoritmos rápidos simples para resolver (3.25) para estos criterios de pérdida más generales, y aproximaciones como (3.23) se vuelven esenciales.

3.2.1. Optimización Numérica Mediante Impulso del Gradiente.

Los algoritmos de aproximación rápida con cualquier criterio de pérdida diferenciable pueden derivarse por analogía a la optimización numérica. La pérdida al usar $f(x)$ para predecir y en los datos de entrenamiento es

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)) \qquad (3.27)$$

El objetivo es minimizar $L(f)$ con respecto a f, donde aquí $f(x)$ está restringido a ser una suma de árboles. Ignorando esta restricción, minimizar se puede ver como una optimización numérica

$$\hat{\mathrm{f}} = \arg\min_{\mathrm{f}} L(\mathrm{f}) \qquad (3.28)$$

Donde los “parámetros” $\mathrm{f} \in \mathbb{R}^N$ son los valores de la función aproximada $f(x_i)$ en cada uno de los N puntos de datos x_i:

$$\mathrm{f} = \{f(x_1), f(x_2), \ldots, f(x_N)\}$$

Los procedimientos de optimización numérica se resuelven como una suma de vectores componentes

$$\mathrm{f}_M = \sum_{m=0}^{M} \mathrm{h}_m, \quad \mathrm{h}_m \in \mathbb{R}^N,$$

Donde $\mathrm{f}_0 = \mathrm{h}_0$ es una suposición inicial, y cada fm sucesivo se induce basándose en el vector de parámetro actual f_{m-1} que es la suma de las actualizaciones inducidas previamente. Los métodos de optimización numérica difieren en sus prescripciones para calcular cada vector de incremento h_m ("paso").

3.2.1.1. Método Descenso más Pronunciado (Steepest descent)

El descenso más pronunciado elige $h_m = -\rho_m g_m$ donde ρ_m es un escalar y $g_m \in \mathbb{R}^N$ es el gradiente de $L(\mathrm{f})$ evaluado en $\mathrm{f} = \mathrm{f}_{m-1}$. Los componentes del gradiente g_m son

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)} \tag{3.29}$$

La longitud del paso ρ_m es la solución a

$$\rho_m = \arg\min_{\rho} L(\mathrm{f}_{m-1} - \rho g_m) \tag{3.30}$$

A continuación, se actualiza la solución actual

$$\mathrm{f}_m = \mathrm{f}_{m-1} - \rho_m g_m$$

Y el proceso se repite en la siguiente iteración. El descenso más pronunciado puede verse como una estrategia muy codiciosa, ya que $-g_m$ es la dirección local en $\mathbb{R}^N$ para la cual $L(\mathrm{f})$ está disminuyendo más rápidamente en $\mathrm{f}_m = \mathrm{f}_{m-1}$.

3.2.1.2. Aumento de Gradiente (Gradient Boosting)

Gradient boosting o Aumento del gradiente, es una técnica de aprendizaje automático utilizado para el análisis de la regresión y para problemas de clasificación estadística, el cual produce un modelo predictivo en forma de un conjunto de modelos de predicción débiles, típicamente árboles de decisión. Construye el modelo de forma escalonada como lo hacen otros métodos de boosting, y los generaliza permitiendo la optimización arbitraria de una función de pérdida diferenciable. La idea de la potenciación del gradiente fue originada en la observación realizada por Leo Breiman en donde el Boosting puede ser interpretado como un algoritmo de optimización en una función de coste adecuada. Posteriormente Jerome H. Friedman desarrolló algoritmos de aumento de gradiente de regresión explícita, simultáneamente con la perspectiva más general de potenciación del gradiente funcional de Llew Mason,

Jonathan Baxter, Peter Bartlett y Marcus Frean. En sus últimos dos trabajos presentaron la visión abstracta de los algoritmos de potenciación como algoritmos iterativos de descenso de gradientes funcionales. Es decir, algoritmos que optimizan una función de coste sobre el espacio de función mediante la elección iterativa de una función que apunta en la dirección del gradiente negativo. Esta visión de gradiente funcional de potenciación ha llevado al desarrollo de algoritmos de potenciación en muchas áreas del aprendizaje automático y estadísticas más allá de la regresión y la clasificación.[15]

En cada paso, el árbol de solución es el que reduce al máximo (3.25), dado el modelo actual f_{m-1} y sus ajustes $f_{m-1}(x_i)$. Por tanto, las predicciones del árbol $T(x_i; \theta_m)$ son análogas a las componentes del gradiente negativo (3.29). La principal diferencia entre ellos es que los componentes del árbol $t_m = (T(x_1; \theta_m), \dots, T(x_N; \theta_m))$ no son independientes. Están limitados a ser las predicciones de un árbol de decisión de nodo terminal J_m, mientras que el gradiente negativo es la dirección máxima de descenso sin restricciones. La solución a (3.24) en la aproximación por etapas es análoga a la búsqueda de línea en el descenso más pronunciado. La diferencia es que (3.24) realiza una búsqueda de línea separada para aquellos componentes de t_m que corresponden a cada región terminal separada $\{T(x_i; \theta_m)\}_{x_i \in R_{jm}}$.

Si el único objetivo fuera minimizar la pérdida de datos de entrenamiento (3.27), la estrategia preferida sería el descenso más pronunciado. El gradiente (3.29) es trivial de calcular para cualquier función de pérdida diferenciable $L(y, f(x))$. Lamentablemente, el gradiente (3.29) se define solo en los puntos de datos de entrenamiento x_i, mientras que el objetivo final es generalizar $f_M(x)$ a nuevos datos no representados en el conjunto de entrenamiento. Una posible solución a este dilema es inducir un árbol $T(x; \theta_m)$ en la m-ésima iteración cuyas predicciones t_m estén lo más cerca posible del gradiente negativo. Usando el error al cuadrado para medir la cercanía, esto nos lleva a

$$\tilde{\theta}_m = \arg\min_{\theta} \sum_{i=1}^{N} (-g_{im} - T(x_i; \theta))^2 \qquad (3.31)$$

Es decir, se ajusta el árbol T a los valores de gradiente negativo (3.29) por mínimos cuadrados. Existen algoritmos rápidos para la inducción del árbol de decisión de mínimos cuadrados. Aunque las regiones de solución $\tilde{R}_{jm}$ a (3.31) no serán idénticas a las regiones R_{jm} que se resuelven (3.23), generalmente es lo suficientemente similar para cumplir el mismo propósito. En cualquier caso, el procedimiento de impulso progresivo por etapas y la inducción del árbol de decisión de arriba hacia abajo son en sí mismos procedimientos de aproximación. Después de construir el árbol (3.31), las constantes correspondientes en cada región vienen dadas por (3.24).

CONFIGURACION	FUNCION DE PERDIDA	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
REGRESIÓN	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
REGRESIÓN	$\lvert y_i - f(x_i)\rvert$	$\text{Sign}[y_i - f(x_i)]$
REGRESIÓN	Huber	$y_i - f(x_i)\ para\ \lvert y_i - f(x_i) \leq \delta_m$ $\delta_m sign[y_i - f(x_i)]\ para\ \lvert y_i - f(x_i)\rvert > \delta_m$ Donde $\delta_m = \alpha th - cuantil\{\lvert y_i - f(x_i)\rvert\}$
CLASIFICACIÓN	Deviance	$componente\ kth: I(y_i = \vartheta_k) - p_k(x_i)$

Tabla 1. Gradientes para funciones de pérdida de uso común.

La tabla 3.1 resume los gradientes de las funciones de pérdida de uso común. Para la pérdida de error al cuadrado, el gradiente negativo es solo el residual ordinario $-g_{im} = y_i - f_{m-1}(x_i)$, de modo que (3.31) por sí solo es un aumento por mínimos cuadrados estándar equivalente. Con pérdida absoluta de error, el gradiente negativo es el signo del residual, por lo que en cada iteración (3.31) ajusta el árbol al signo de los residuales actuales por mínimos cuadrados. Para la regresión M de Huber, el gradiente negativo es un compromiso entre estos dos (ver la tabla). Para la clasificación, la función de pérdida es la desviación multinomial y se construyen K árboles de mínimos cuadrados en cada iteración. Cada árbol T_{km} se ajusta a su respectivo vector de gradiente negativo g_{km},

$$-g_{ikm} = \frac{\partial L(y_i, f_{1m}(x_i), \dots, f_{1m}(x_i))}{\partial f_{km}(x_i)}$$

$$= I(y_i = \vartheta_k) - p_k(x_i) \qquad (3.32)$$

Con $p_k(x)$ dado. Aunque se construyen K árboles separados en cada iteración, están relacionados. Para la clasificación binaria ($K = 2$), solo se necesita un árbol. El algoritmo 3.2 presenta el algoritmo genérico de aumento de árbol de gradiente para regresión. Los algoritmos específicos se obtienen insertando diferentes criterios de pérdida $L(y, f(x))$. La primera línea del algoritmo se inicializa en el modelo constante óptimo, que es solo un árbol de nodo terminal único. Los componentes del gradiente negativo calculados en la línea 2(a) se denominan generalizados o pseudoresiduos, r. Los gradientes para las funciones de pérdida de uso común se resumen en la tabla 3.1.

1. Inicializa $f_o(x) = \arg\min_{\gamma} \sum_{i=!}^{N} L(y_i, \gamma)$
2. Para $m = 1$ hasta *M:*

a. Para $i = 1,2,\dots,N$ calcular

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

b. Ajustar un árbol de regresión al borde de los objetivos r_{im} dando regiones terminales $R_{jm} = j = 1,2,\dots,J_m$

c. Para $j = 1,2,\dots,J_m$ calcular

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

d. Actualizar $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

3. Salida $\hat{f}(x) = f_M(x)$

El algoritmo de clasificación es similar. Las líneas 2(a) y 2(d) se repiten K veces en cada iteración m, una vez para cada clase usando (3.32). El resultado en la línea 3 es K expansiones de árbol diferentes (acopladas) $f_{km}(x), k = 1,2,\dots,K$. Estos producen probabilidades o clasifican. Dos parámetros de ajuste básicos son el número de iteraciones M y los tamaños de cada uno de los árboles constituyentes $J_m, m = 1,2,\dots,M$. La implementación original de este algoritmo se denominó MART para "árboles de regresión aditiva múltiple".

3.3. Regularizaciones

Además del tamaño de los árboles constituyentes, J, el otro metaparámetro del aumento de gradiente es el número de iteraciones de impulso M. Cada iteración generalmente reduce el riesgo de entrenamiento $L(f_M)$ de modo que para M lo suficientemente grande este riesgo puede hacerse arbitrariamente pequeña. Sin embargo, ajustar demasiado bien los datos de entrenamiento puede llevar a un sobreajuste, lo que degrada el riesgo de predicciones futuras. Por tanto, hay un número óptimo M^* que minimiza el riesgo futuro que depende de la aplicación. Una forma conveniente de estimar M^* es monitorear el riesgo de predicción como una función de M en una muestra de validación. El valor de M que minimiza este riesgo se toma como una estimación de M^*. Esto es análogo a la estrategia de detención temprana que se usa a menudo con los dos diversos tipos de algoritmos.

3.3.1. Shrinkage

Controlar el valor de M no es la única estrategia de regularización posible. Al igual que con la regresión de crestas y las redes neuronales, también se pueden emplear técnicas Shrinkage. La implementación más simple de Shrinkage en el contexto del

impulso es escalar la contribución de cada árbol en un factor $0 < y < 1$ cuando se agrega a la aproximación actual. Es decir, la línea 2(d) del algoritmo 3.2 se reemplaza por

$$f_m(x) = f_{m-1}(x) + v \cdot \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm}) \qquad (3.33)$$

Se puede considerar que el parámetro v controla la tasa de aprendizaje del procedimiento de refuerzo. Los valores más pequeños de v (más Shrinkage) dan como resultado un mayor riesgo de entrenamiento para el mismo número de iteraciones M. Por lo tanto, tanto v como M controlan el riesgo de predicción en los datos de entrenamiento. Sin embargo, estos parámetros no funcionan de forma independiente. Los valores más pequeños de v conducen a valores más grandes de M para el mismo riesgo de entrenamiento, por lo que existe una compensación entre ellos. Empíricamente se ha encontrado (Friedman 2001) que valores más pequeños de v favorecen un mejor error de prueba y requieren valores correspondientemente mayores de M. De hecho, la mejor estrategia parece ser establecer v para que sea muy pequeño $(v < 0{,}1)$ y luego elija M haciendo una parada anticipada. Esto produce mejoras dramáticas (sobre ninguna contracción $v = 1$) para la regresión y para la estimación de probabilidad. Las mejoras correspondientes en el riesgo de clasificación errónea son menores, pero aún sustanciales. El precio que se paga por estas mejoras es computacional: valores menores de v dan lugar a valores mayores de M, y el cálculo es proporcional a este último. Sin embargo, como se ve a continuación, muchas iteraciones son generalmente factibles computacionalmente incluso en conjuntos de datos muy grandes. Esto se debe en parte al hecho de que se inducen árboles pequeños en cada paso sin poda.

3.4. Problemas de clasificación con clases no balanceadas

La clasificación desequilibrada es un problema de aprendizaje supervisado en el que una clase supera a otras clases en una gran proporción. Este problema se enfrenta con más frecuencia en los problemas de clasificación binaria que los problemas de clasificación de varios niveles. El término desequilibrado se refiere a la disparidad encontrada en la variable dependiente (respuesta). Por lo tanto, un problema de clasificación desequilibrado es aquel en el que la variable dependiente tiene una proporción desequilibrado de las clases. En otras palabras, un conjunto de datos que muestra una distribución desigual entre sus clases se considera desequilibrado. Dado esto existen métodos que ayudan a el balanceo de este desequilibrio en los datos los métodos son ampliamente conocidos como 'Métodos de Muestreo'. Generalmente, estos métodos tienen como objetivo modificar los datos desequilibrados en una distribución equilibrada utilizando algún mecanismo. La modificación se produce modificando el tamaño del conjunto de datos original y proporcionando la misma

proporción de equilibrio. Estos métodos han adquirido mayor importancia después de que muchas investigaciones hayan demostrado que los datos equilibrados dan como resultado un mejor rendimiento de clasificación general en comparación con un conjunto de datos desequilibrado.

3.4.1. UnderSampling, Oversampling y Muestreo con Igual Probabilidad

El Undersampling implica la selección aleatoria de ejemplos de la clase mayoritaria para eliminar del conjunto de datos de entrenamiento figura 7. Esto tiene el efecto de reducir el número de ejemplos en la clase mayoritaria en la versión transformada del conjunto de datos de entrenamiento. Este proceso puede repetirse hasta que se logre la distribución de clase deseada, como un número igual de ejemplos para cada clase. Una limitación del Undersampling es que se eliminan ejemplos de la clase mayoritaria que pueden ser útiles, importantes o quizás críticos para ajustar un límite de decisión sólido. Dado que los ejemplos se eliminan al azar, no hay forma de detectar o preservar ejemplos " buenos " o más ricos en información de la clase mayoritaria.

Ambos enfoques pueden repetirse hasta que se logre la distribución de clase deseada en el conjunto de datos de entrenamiento, como una división equitativa entre las clases. Además se los conoce como métodos de " *muestreo ingenuo* " porque no asumen nada acerca de los datos. Esto los hace simples de implementar y rápidos de ejecutar, lo cual es deseable para conjuntos de datos muy grandes y complejos. Ambas técnicas se pueden utilizar para problemas de clasificación de dos clases (binarios) y problemas de clasificación de múltiples clases con una o más clases mayoritarias o minoritarias. Es importante destacar que el cambio en la distribución de la clase solo se aplica al conjunto de datos de entrenamiento. La intención es influir en el ajuste de los modelos. El remuestreo no se aplica al conjunto de datos de prueba o reserva utilizado para evaluar el rendimiento de un modelo. En general, estos métodos ingenuos pueden ser efectivos, aunque eso depende de los detalles del conjunto de datos y los modelos involucrados.

El Oversampling aleatorio implica duplicar al azar ejemplos de la clase minoritaria y agregarlos al conjunto de datos de entrenamiento figura 7. Los ejemplos del conjunto de datos de entrenamiento se seleccionan aleatoriamente con reemplazo. Esto significa que se pueden elegir ejemplos de la clase minoritaria y agregarlos al nuevo conjunto de datos de entrenamiento " más equilibrado " varias veces; se seleccionan del conjunto de datos de entrenamiento original, se agregan al nuevo conjunto de datos de entrenamiento y luego se devuelven o " reemplazan " en el conjunto de datos original, lo que permite que se vuelvan a seleccionar.

Esta técnica puede ser efectiva para aquellos algoritmos de aprendizaje automático que se ven afectados por una distribución sesgada y en los que múltiples ejemplos duplicados para una clase determinada pueden influir en el ajuste del modelo. Esto podría incluir algoritmos que aprendan coeficientes de forma iterativa, como redes neuronales artificiales que utilizan el descenso de gradiente estocástico. También puede afectar a los modelos que buscan buenas divisiones de datos, como máquinas de vectores de soporte y árboles de decisión.

Por otro lado el muestreo con igual probabilidad se implementan ambos métodos anteriormente mencionados en las respectivas clases. Esta metodología será denotada por "Both sampling", ya que de acuerdo a la probabilidad aplicada se reduce la clase mayoritaria y se aumenta la clase minoritaria al mismo tiempo, hasta quedar en un punto medio alcanzando la paridad.[16]

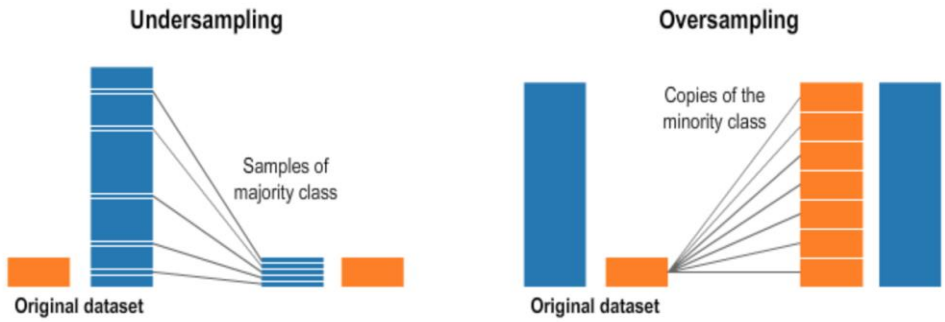

Figura 7. Métodos de balanceo.

4. Implementación de un Modelo Machine Learning

La metodología CRISP-DM mencionada en la sección 2.3. Y sus respectivas fases o etapas 2.3.1. Ayudan a dirigir un proyecto de una manera ordenada y enfocada hacia una necesidad específica de un negocio. Por esto se ha decidido implementar esta metodología para la construcción de un modelo predictivo de clasificación que logre entender, identificar y comparar ciertos comportamientos de los consumos de clientes activos e inactivos prepago, con el fin de evitar una posible portación o fuga de los clientes activos hacia la competencia.

4.1. Comprensión del negocio

El entorno competitivo de las empresas de telecomunicaciones hace que el negocio se enfoque en gran parte al cuidado y fidelización de los clientes activos de los diferentes servicios, por eso la necesidad de anticiparse a la portación de los mismos. Como indicador de cualquier compañía de telecomunicaciones se hace clave reducir la tasa de abandono o churn rate mediante diversas estrategias y aspectos claves que se deben tener en cuenta como el hecho de identificar ¿por qué se está dando el abandono?, ¿Se está cumpliendo con la expectativa del cliente? ¿Qué se puede hacer para reducir el abandono?, son preguntas que constantemente se busca atacar por áreas especializadas de analítica avanzada, en las cuales se plantean estrategias que conlleven a un beneficio dentro de la compañía con ayuda de herramientas de ciencia de datos y minería de datos que facilitan el entendimiento de la información que se tenga de una manera más sencilla y práctica, además de poderse manejar en grandes volúmenes de información.

4.2. Comprensión de los datos:

El conjunto de datos conforma registros históricos de los usuarios con líneas prepago. Para la construcción de la base de datos de entrenamiento se consolidan muestras aleatorias de la base activa de clientes, partiendo de un mes en específico y devolviéndose seis meses atrás como lo muestra la figura 8, donde se parte desde enero del año 2019 hasta noviembre del año 2018 y además también se recopilan los usuarios inactivos o Port Out durante este tiempo.

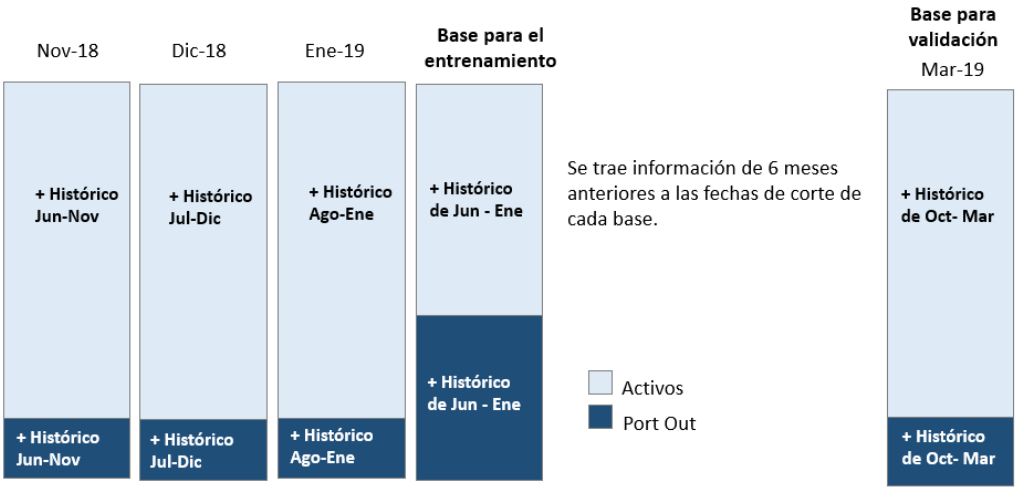

Figura 8. Construcción de la base de entrenamiento y validación.

Dado el gran volumen de la información que se tiene (más de veinte millones de registros) y pese que no se cuenta con una herramienta big data para la construcción de modelos, se realiza una muestra aleatoria simple de un millón de registros en la base activa de los tres meses de histórico y se toman cincuenta mil registros de la base inactiva para consolidar tres millones de activos y ciento cincuenta mil no activos. Dentro del conjunto de datos de entrenamiento se encuentran variables como antigüedad de la línea, tipo de equipo, promedio de ingresos por usuario en voz y datos, el número de llamadas entradas y salientes, consumo de mensajes de texto, consumo de redes sociales, tiempo de permanencia en llamadas en otras redes, entre otras; las cuales aportaron al aprendizaje del modelo. Si se quiere controlar algo, debe ser observable, y para conseguir el éxito, es esencial definir lo que se considera un éxito: ¿será precisión?, ¿exactitud?, ¿tasa de retención de clientes? esta medida debe estar directamente alineada con los objetivos de mayor nivel del negocio, y también directamente relacionada con el tipo de problema que se presenta.
Así que, teniendo conformado el conjunto de datos de entrenamiento, debe existir esa etiqueta la cual puede tener varios niveles o clases y es a la cual se debe dar respuesta. En este caso se contó con una etiqueta de interés en la que existen dos clases, "1" para referenciar al cliente inactivo o Port Out y "0" para el que no (Cliente activo).

4.3. Preparación de los datos

La selección de las variables que serán incluidas al conjunto de datos se toman de la base robusta de clientes prepago que la compañía telefónica tiene alojada en el almacén de datos (Data Warehouse), la cual cuenta con más de 200 posibles variables para seleccionar. Se tuvo en cuenta características claves de los clientes como por ejemplo:

- ✓ Ubicación geográfica por regiones.
- ✓ Ubicación departamental.
- ✓ Consumo promedio en Gigas de internet.
- ✓ Consumo en minutos fuera de la red telefónica. (A otros operadores)
- ✓ Consumo en minutos dentro de la misma red telefónica.
- ✓ Consumo promedio en Gigas de internet en apps de redes sociales.
- ✓ Cantidad de mensajes de textos enviados y entrantes.
- ✓ ARPU (Promedio de ingresos por usuario).

Esta son solo unas pocas de una lista total de 108 variables seleccionadas para el entrenamiento del modelo. Antes de comenzar a entrenar los modelos, se debe transformar los datos de una manera sencilla y entendible para el modelo ML, ya que si no se hace, muy seguramente el modelo no logre aprender de la mejor manera. Para esto se procedió a realizar la limpieza del conjunto de datos donde se realizó el respectivo análisis de valores perdidos, los cuales generalmente se representan con los indicadores "NaN" o "Null". El debido tratamiento de estos es muy importante, ya que la mayoría de algoritmos no pueden manejar esos valores faltantes y por ende omitirlos. Preferiblemente antes de entrenar el modelo, se debe identificar los valores perdidos y tratarlos de la manera más conveniente. Además, se realizó un tratamiento de datos categóricos donde es muy importante tener la misma estructura de los mismos, ya que el modelo solo aprenderá los datos que se tengan de entrada; entonces si en la predicción existe alguna otra categoría diferente, el modelo no podrá predecir y por ende la omitirá. Es fundamental tener presente que el ML sólo puede ser usado para estudiar comportamientos o patrones que están presentes en los datos de entrenamiento, por lo tanto sólo podemos reconocer lo que hemos visto antes. Cuando usamos ML estamos asumiendo que el futuro se comportará como el pasado, lo que no siempre es cierto.

Teniendo limpia y tratada nuestra base de datos, se puede proceder a realizar las diferentes estadísticas descriptivas y exploratorias, ya que describen lo que tenemos en el conjunto de datos y exponen la forma en la que se distribuyen los datos, valores atípicos, discontinuidades, etc. Por medio de un análisis descriptivo, se logró tener un previo conocimiento o hipótesis del porqué los usuarios deciden portarse a otras empresas y algunas de estas razones fueron:

- El monto de las recargas realizadas no satisfacía su necesidad y por ende necesitaban recargar más.
- El aumento en el promedio de sus consumos. (Se empezó a tener más necesidades en cuanto a internet, voz o texto).
- La cantidad de llamadas a otra red era alta, por lo que se piensa en algún tipo de influencia, entre otras.

Además, en la exploración del conjunto de datos se logró determinar que la etiqueta objetivo tiene un alto grado de desbalance, ya que tan sólo un 5% correspondió a los clientes que se fugaron y el 95% restante a los clientes activos, como se observa en la Fig. 9. Debido a que los algoritmos ML son deficientes en presencia de desbalance, por esta razón es pertinente balancear las clases existentes en la misma [17].

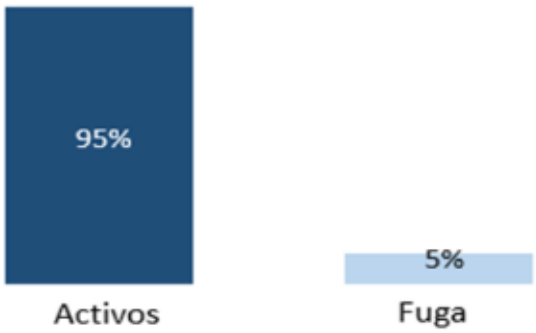

Figura 9. Estado de desbalance del conjunto de datos de entrenamiento.

Dada esta situación se procedió a utilizar técnicas de balanceo por medio de una librería en R llamada "ROSE" en la cual se encuentran funciones que generan muestras sintéticas balanceadas y por lo tanto, permiten fortalecer la estimación posterior de cualquier clasificador binario, manejando datos continuos y categóricos a partir de una estimación de densidad condicional de las dos clases. De esta librería se utilizó la función "Ovun.sample" la cual permite balancear las clases de tres diferentes maneras, tal como se había abarcado en el capítulo 3:

- **UnderSampling:** Se elimina al azar algunas de las observaciones de la clase mayoritaria para hacer coincidir los números con la clase minoritaria. Se manejaron tres escenarios diferentes en esta técnica tal como lo muestra la Fig. 10.

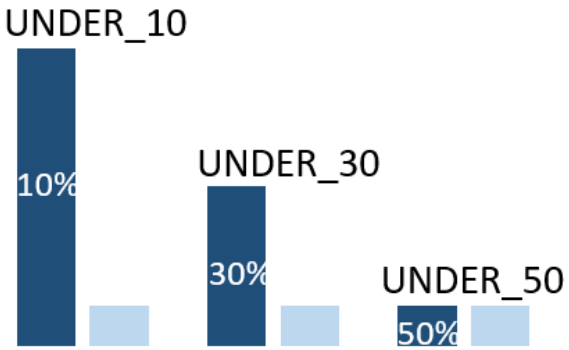

Figura 10. Escenarios implementados para el balanceo de clases por medio de la técnica UnderSampling.

- **OverSampling:** Se generan datos sintéticos aleatoriamente para una muestra de los atributos a partir de observaciones en la clase minoritaria. De la misma

manera se simularon tres escenarios para el modelo, tal como se muestra la Fig. 11.

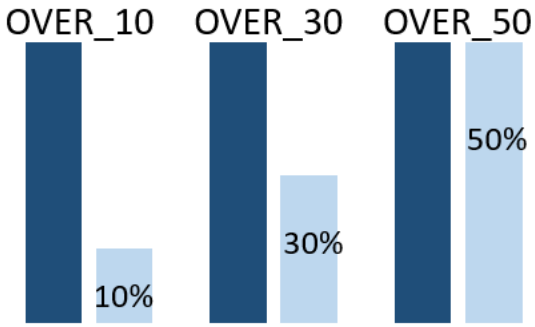

Figura 11. Escenarios implementados para el balanceo de clases por medio de la técnica OverSampling.

- **Balanceo con igual probabilidad**: En este proceso se implementan ambos métodos anteriormente mencionados en las respectivas clases, Fig. 12. Esta metodología será denotada por "BothSampling", ya que de acuerdo a la probabilidad aplicada se reduce la clase mayoritaria y se aumenta la clase minoritaria al mismo tiempo, hasta quedar en un punto medio alcanzando la paridad.

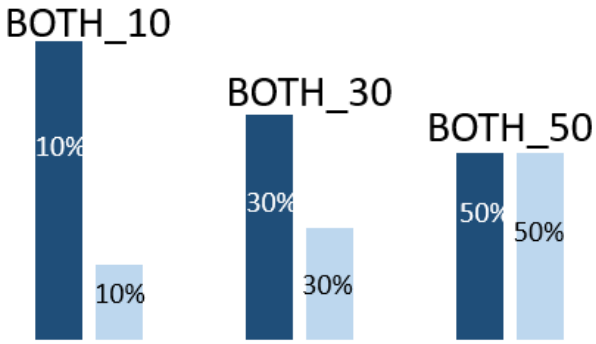

Figura 12. Escenarios implementados para el balanceo de clases por medio de la técnica BothSampling.

Es necesario dividir los datos de entrenamiento en tres subconjuntos aleatorios

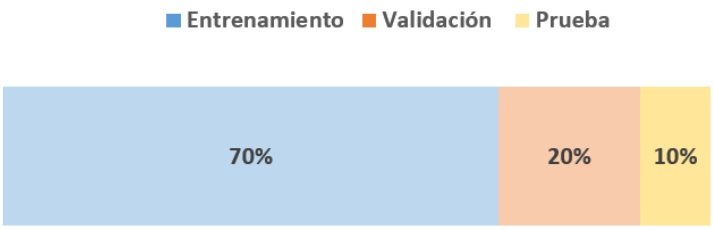

Figura 13. Partición del conjunto de datos previo al entrenamiento del modelo.

Partición de Entrenamiento: Es la primera interacción del modelo con los datos (Aquí se incluyen todos los registros de usuarios Port Out).

Partición de Validación: Cada iteración o vista que el modelo realiza a la partición de entrenamiento, será validada y puntuada por esta partición mediante validaciones cruzadas.

Partición de Prueba: Se valida el ajuste del modelo con una partición que no contiene la variable etiquetada y que el modelo desconoce completamente.

4.4. Modelado y Evaluación

El objetivo en este paso del proceso es desarrollar varios modelos de comparación con los distintos escenarios de balanceo antes mencionados, sobre el que se medirá el rendimiento y ajuste a los datos por cada uno. Así que, como se mencionó anteriormente se utilizaron los algoritmos GBM (Gradient Boosting Machine) y DRF (Distribuited Random Forest) ya que el rendimiento de ambos algoritmos en problemas de clasificación es óptimo. Se tuvo en cuenta el criterio AUC (Área Bajo la Curva "ROC")[18], para medir que tan buena clasificación de las clases existentes tuvo el modelo. Además de eso se utilizó un método común para encontrar un buen modelo por medio de la validación cruzada, en la cual se establece n número de subdivisiones en las que se fragmentara los datos y los cuales se irán puntuando en la validación del mismo.

Un algoritmo ML tiene dos tipos de parámetros, el primer tipo son los parámetros que se aprenden a través de la fase de aprendizaje, y el segundo tipo son los hiperparámetros[19] que se transfieren a los modelos, los cuales son combinación de parámetros óptimos o de mejor ajuste. Una vez identificado el modelo que se manejará, el siguiente paso es ajustar sus hiperparámetros para obtener el mayor poder predictivo posible. La forma más común de encontrar la mejor combinación de hiperparámetros se llama "Grid-Search". Esto se realizó en todos los escenarios planteados anteriormente los cuales son nueve, es decir, se realizaron 18 modelos donde 9 fueron modelados bajo el algoritmo GBM y los otros 9 bajo el algoritmo DRF, respectivamente a su escenario de balanceo.

Para esto se siguió el siguiente proceso:

- Se establece la matriz de parámetros que se evaluarán. Los parámetros fueron secuencias establecidas de acuerdo a los requerimientos de cada uno.

- Se establece el número de subdivisiones de los datos, el estado aleatorio, criterios de parada del algoritmo, máximo número de modelos que se planean se entrenen por el algoritmo o consecuentemente un tiempo límite de entrenamiento.

- Se determina una estrategia de búsqueda de cuadrícula la cual fue aleatoria discreta.

Cuando finalizó el entrenamiento, se observó cada modelo ajustado y se comparan entre ellos para así seleccionar el mejor bajo el criterio AUC, donde se estableció que el modelo que tuviera mayor AUC fuera seleccionado como el mejor. El AUC proporciona una medición agregada del rendimiento en todos los umbrales de clasificación posibles (se evalúa de cero a uno, donde uno sería una clasificación perfecta)[20]. Una forma de interpretarlo es como la probabilidad de que el modelo clasifique un ejemplo positivo aleatorio más alto que un ejemplo negativo aleatorio, se pude observar esto en la figura 13.

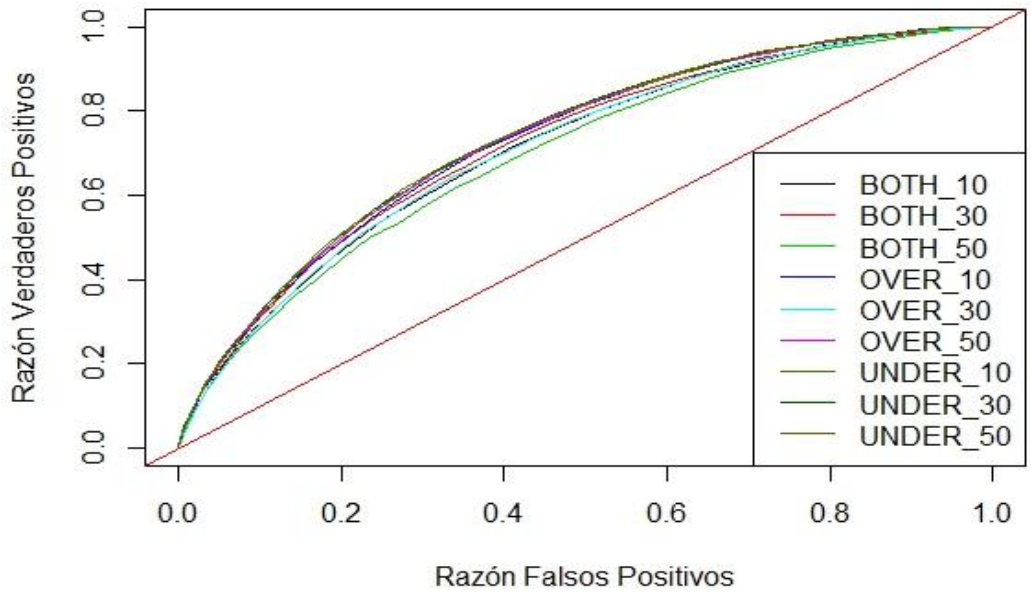

Figura 14. Validación de la métrica AUC en los nueve escenarios de balanceo en el algoritmo DRF.

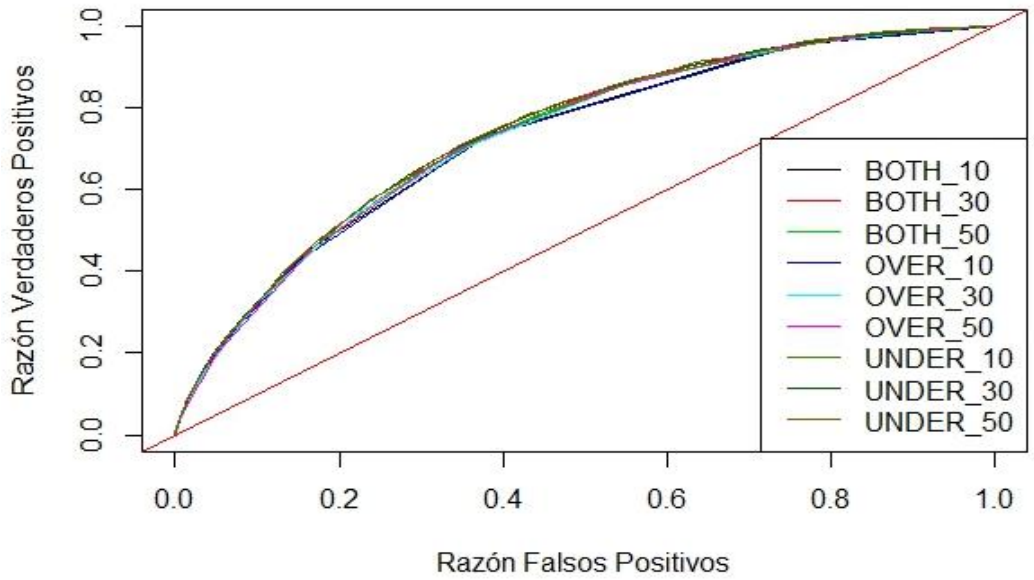

Figura 15. Validación de la métrica AUC en los nueve escenarios de balanceo en el algoritmo GBM.

Cada curva que se refleja en las figuras 14 y 15 muestra el perfomance de cada escenario de balanceo evaluado en los dos modelos. El mayor AUC fue logrado por el modelo UNDER_10 mediante el algoritmo DRF con un puntaje de 0.795, el cual es algo óptimo. Por otro lado, para el algoritmo GBM ningún escenario parece destacar de los otros.

ESCENARIOS	AUC DRF	AUC GBM
BOTH_10	0.7112	0.7385
BOTH_30	0.7205	0.7404
BOTH_50	0.6963	0.7369
OVER_10	0.7288	0.7391
OVER_30	0.7112	0.7377
OVER_50	0.7306	0.7375
UNDER_10	0.795	0.741
UNDER_30	0.7338	0.742
UNDER_50	0.733	0.742

Tabla 2. Puntaje AUC para los 9 escenarios modelados bajo los algoritmos DRF y GBM.

Además, la partición de prueba permite conocer la distribución de densidad de las predicciones. Esto quiere decir que mostrara hacia que probabilidad de predicción se agrupara para establecer puntos de cortes a nivel general. En la gráfica 16 se observa que a medida que cada escenario de balanceo estima diferente la portabilidad.

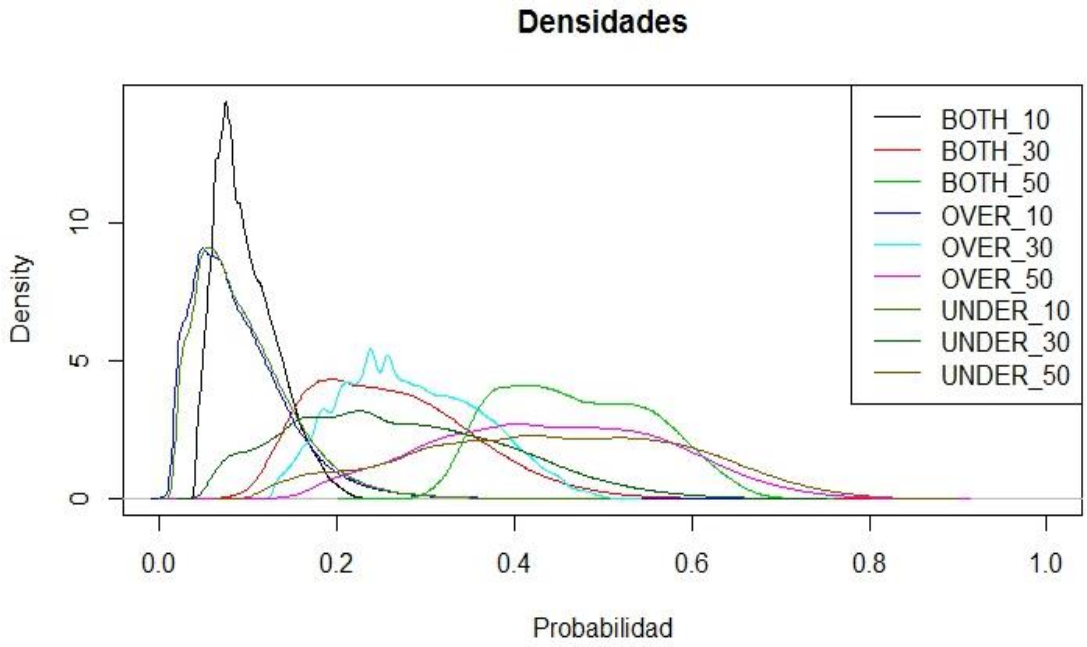

Figura 16. Densidades de probabilidad bajo los escenarios modelados DRF.

Ya que el algoritmo Random Forest es un algoritmo basado en la familia de árboles, podemos conocer las variables que influyen en la decisión de cada predicción. La importancia de la variable se determina calculando la influencia relativa de cada variable sobre el árbol para determinar si esa variable que se selecciona durante el proceso de construcción de árbol mejora (disminuyó) el error cuadrado (sobre todos los árboles) como resultado. Cada vez que H2O divide un nodo, la reducción atributiva de la entidad en el error cuadrado es la diferencia en el error cuadrado entre ese nodo y sus nodos secundarios (El error cuadrático para cada nodo individual es la reducción en la varianza del valor de respuesta dentro del mismo).

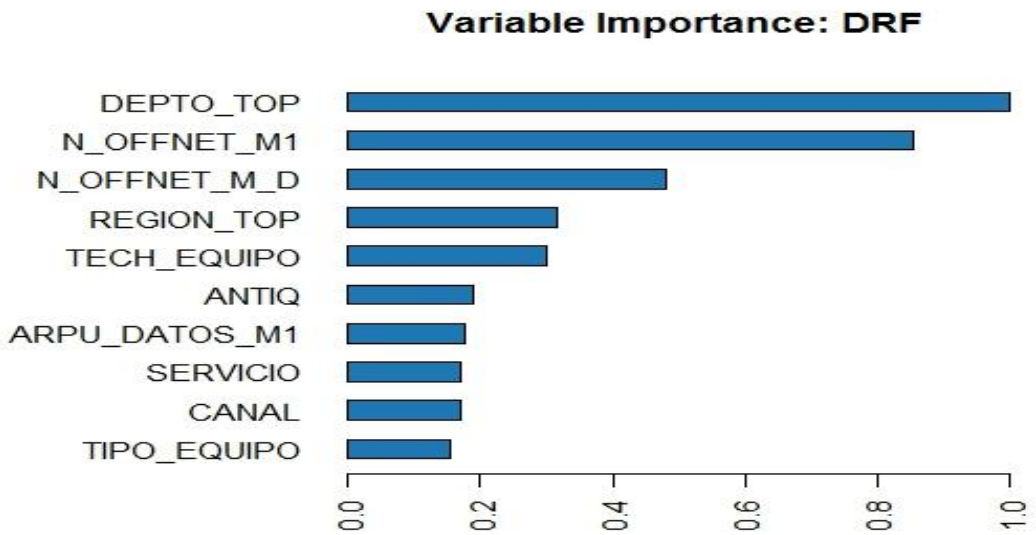

Figura 17. Variables de importancia del modelo DRF.

Dentro de las variables más importantes que caracterizan a un usuario propenso a portabilidad se encuentran: Cantidad de servicio prestado fuera de la red propia, el departamento y región de ubicación y la tecnología del equipo, el ARPU en datos, la antigüedad. El ARPU como variable importante en la toma de decisiones del modelo, conduce a pensar en varias hipótesis, ya que tal vez existen muchos usuarios que la recarga que están realizando no cumplen con las expectativas que tiene y por eso podrían portarse. Adicionalmente. El N_OFFNET que corresponde al tiempo que un usuario pasa fuera de la red propia, podría darse a pensar que el usuario puede estar siendo influenciado por algún usuario de otra red. El contraste de varias más hipótesis se toman bajo el cargo del equipo de analítica avanzada, los cuales realizaran un análisis más detallado y focalizado en los resultados e hallazgos del modelo mediante las variables que se están considerando para la toma de las decisiones de predicción,

Como ultima validación de rendimiento del modelo se utiliza el análisis de la curva LIFT (Curva de elevación) que describe un coeficiente de rendimiento sobre la proporción acumulada de una población. [21]. Con la base de prueba se predice y se segmenta las predicciones en deciles.

% of data	Lift (Model)	Lift (Random)
10	4.91	1
20	4.07	1
30	2.79	1
40	1.89	1
50	1.67	1
60	1.48	1
70	1.33	1
80	1.21	1
90	1.1	1
100	1	1

Tabla 3. Análisis Lift para el modelo UNDER_10 bajo DRF.

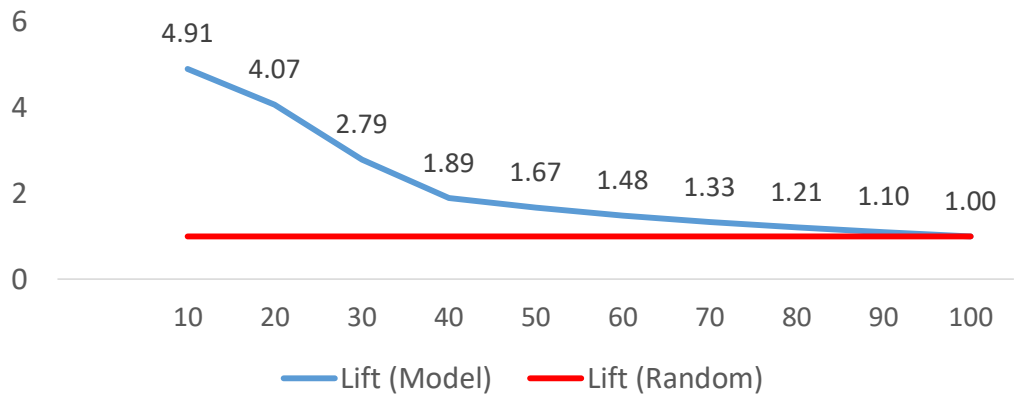

Figura 18. Curva Lift para el mejor modelo UNDER_10.

El Lift del modelo muestra que para la partición de validación, el modelo es capaz de identificar 4.91 veces el número total de objetivos versus si se tuviera un modelo aleatorio. Se rige que en los dos primeros deciles se podría encontrar 4.07 veces el número total de objetivos o los posibles clientes que se podrían portar a otra compañía de telefonía. Esto es de gran ayuda para la identificación anticipada de clientes en Port Out y poder realizar campañas de fidelización para cuidar y proteger los clientes que actualmente están activos.

5. Conclusiones

Las técnicas Machine Learning para la identificación de Portabilidad numérica en las empresas de telecomunicaciones son de bastante utilidad, ya que permiten estudiar el comportamiento de clientes Portados y la base actuales de clientes únicos y compararlos e identificar posibles clientes que entrarían en portación. Para el escenario seleccionado en el mejor algoritmo implementado, se nota que la clasificación está en un rango adecuado de confiabilidad, debido a que un 79.5% de clasificación correcta en una Dimencionalidad alta como la que se reflejó durante este trabajo, muestra el poder de identificación que logra el modelo a los posibles clientes a portarse.

Adicionalmente, el uso de la minería de datos con metodologías como la CRISP-DM y la técnica de análisis de la curva Lift, logran estudiar y entender un poco más el desempeño que logra un modelo de clasificación bajo algoritmos supervisados. El hecho de lograr encontrar en los dos primeros deciles 4 veces más la cantidad de portados que si se tuviera un modelo aleatorio implementado, da una vista de lo poderoso que puede llegar a ser la buena implementación de técnicas machine Learning y las técnicas de minería de datos. La efectividad del modelo para la gestión de usuarios y la efectividad del mismo será reflejada en los estudios posteriores de algún tipo de campaña de retención o fidelización que se realice dentro de la compañía de telecomunicaciones por las áreas que manejen este tipo de procesos.

Referencias

[1] «¿Cuántos somos?» https://www.dane.gov.co/index.php/estadisticas-por-tema/demografia-y-poblacion/censo-nacional-de-poblacion-y-vivenda-2018/cuantos-somos (accedido ago. 11, 2020).

[2] «Colombia alcanzó los 62,2 millones de líneas de celular habilitadas - Ministerio de Tecnologías de la Información y las Comunicaciones». https://www.mintic.gov.co/portal/inicio/Sala-de-Prensa/MinTIC-en-los-Medios/72964:Colombia-alcanzo-los-62-2-millones-de-lineas-de-celular-habilitadas (accedido ago. 11, 2020).

[3] «La nueva tecnología y la vida cotidiana | Opinión | EL PAÍS». https://elpais.com/diario/1983/04/03/opinion/418168811_850215.html (accedido ago. 11, 2020).

[4] «Estadísticas: telecomunicaciones en Colombia», *TeleSemana.com*. http://www.telesemana.com/panorama-de-mercado/colombia/ (accedido ago. 11, 2020).

[5] «Definición de pospago — Definicion.de», *Definición.de*. https://definicion.de/pospago/ (accedido ago. 20, 2020).

[6] «¿Qué es Portabilidad Numérica?», *Subsecretaría de Telecomunicaciones de Chile*, ene. 08, 2010. https://www.subtel.gob.cl/ique-es-portabilidad-numerica/ (accedido ago. 20, 2020).

[7] «A Brief History of Machine Learning», *Provalis Research*, jun. 22, 2017. https://provalisresearch.com/blog/brief-history-machine-learning/ (accedido abr. 14, 2020).

[8] «Machine learning: conoce qué es y las diferencias entre sus tipos», *Think Big*, nov. 16, 2017. https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/ (accedido abr. 14, 2020).

[9] «How to handle Imbalanced Data in Machine Learning», mar. 20, 2019. https://www.einfochips.com/blog/addressing-challenges-associated-with-imbalanced-datasets-in-machine-learning/ (accedido abr. 14, 2020).

[10] «Aprendizaje por Refuerzo», *Advanced Tech Computing Group UTPL*, ago. 08, 2008. https://advancedtech.wordpress.com/2008/08/08/aprendizaje-por-refuerzo/ (accedido abr. 14, 2020).

[11] «CRISP-DM: La metodología para poner orden en los proyectos», *Sngular*, ago. 02, 2016. https://www.sngular.com/es/data-science-crisp-dm-metodologia/ (accedido ago. 24, 2020).

[12] G. Grolemund y H. Wickham, *1 Introduction | R for Data Science*. .

[13] «H2O Architecture — H2O 3.30.1.1 documentation». https://docs.h2o.ai/h2o/latest-stable/h2o-docs/architecture.html (accedido ago. 25, 2020).

[14] Harshith, «Log loss function math explained», *Medium*, ene. 11, 2020. https://towardsdatascience.com/log-loss-function-math-explained-5b83cd8d9c83 (accedido sep. 07, 2020).

[15] «Boosting en Machine Learning con Python». https://relopezbriega.github.io/blog/2017/06/10/boosting-en-machine-learning-con-python/ (accedido sep. 14, 2020).

[16] W. Badr, «Having an Imbalanced Dataset? Here Is How You Can Fix It.», *Medium*, abr. 20, 2019. https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb (accedido sep. 18, 2020).

[17] A. Estabrooks, T. Jo, y N. Japkowicz, «A Multiple Resampling Method for Learning from Imbalanced Data Sets», *Computational Intelligence*, vol. 20, n.º 1, pp. 18-36, 2004, doi: 10.1111/j.0824-7935.2004.t01-1-00228.x.

[18] «Clasificación: ROC y AUC | Curso intensivo de aprendizaje automático», *Google Developers*. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419 (accedido oct. 30, 2020).

[19] «▷ Busqueda de hiperparametros. Tu modelo Perfecto», *The Machine Learners*, oct. 12, 2020. https://themachinelearners.com/busqueda-hiperparametros/ (accedido oct. 30, 2020).

[20] «Curvas ROC y Área bajo la curva (AUC)», *Aprende IA*, may 31, 2019. https://aprendeia.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning/ (accedido dic. 07, 2020).

[21] A. Serre, «Understanding Lift curve», *Medium*, feb. 20, 2020. https://medium.com/analytics-vidhya/understanding-lift-curve-b674d21e426 (accedido dic. 07, 2020).