FACULTAD DE INGENIERÍA

TESIS DE MAGÍSTER:

PREDICCIÓN DE FUGA DE CLIENTES EN EMPRESAS DE TELEFONÍA MÓVIL: EL CASO DE ESTUDIO DE VIRGIN MOBILE

POR: VALENTINA CONSUELO BELTRÁN CRUZ

Tesis presentada a la Facultad de Ingeniería de la Universidad del Desarrollo para optar al grado académico de Maestría.

PROFESOR GUÍA:
Sr. MAURICIO HERRERA MARIN

A mi Abuelo Pancho,
por enseñarme a soñar en grande.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la última década la investigación en torno al problema de la fuga de clientes (Customer Churn Problem, "CCP") se ha vuelto cada vez más intensa y se ha convertido en un aspecto de alta relevancia para las empresas. En el caso particular de las compañías telefónicas, debido a la alta competitividad existente en el mercado de telefonía móvil y producto de los avances en la denominada *portabilidad numérica*, los consumidores cuentan con variadas opciones y compañías a su disposición para cambiarse, en caso de que el servicio en una de ellas no cumpla con sus expectativas. Es así que, la industria de las telecomunicaciones se enfrenta anualmente a una fuga de clientes que bordea el 30 %, estimándose además que el costo de adquirir un nuevo cliente es 5 a 10 veces superior al costo de retener uno antiguo [1].

Algunos trabajos previos [2, 3] han mostrado que los atributos de cada consumidor, sean estos demográficos o relacionados con su comportamiento de consumo histórico dentro de la empresa, son útiles en la predicción sobre su permanencia futura en la compañía. Por otro lado, las decisiones de los clientes se ven también influenciadas por su entorno y bajo esta premisa, aquellos clientes que están en contacto con otros que se han desligado de la compañía podrían eventualmente mostrar una mayor tendencia a la fuga [4].

En este trabajo el CCP se aborda desde distintas perspectivas. En primer lugar, se plantea como un problema de clasificación, en el cual los consumidores pueden ser clasificados como "fugados" o "no fugados" y se aplican distintas técnicas de aprendizaje automático (supervisados o no) para la determinación de la probabilidad de fuga condicionada al valor de los atributos individuales y sociales del cliente. En segundo lugar, es considerado como un problema de "supervivencia" ("hazard model") para el cálculo de la probabilidad de fuga dado que el cliente ha permanecido un periodo de tiempo específico en la compañía. Ambas aproximaciones permiten dar una respuesta razonablemente buena acerca de la estimación de la probabilidad de fuga de acuerdo a los atributos del cliente.

Este estudio comienza con la adquisición, procesamiento y limpieza de los datos disponibles, posteriormente se seleccionan las variables más importantes a incluir en los diferentes modelos predictivos y se evalúa el desempeño de numerosas técnicas de aprendizaje automático en busca de los patrones subyacentes en los datos. Finalmente, se construyen varios modelos de supervivencia con un estudio minucioso del rol que juega cada variable/atributo en la decisión que toma el cliente de cambiar o no de compañía.

1.1. El fenómeno de la fuga de clientes en detalles. Estudios previos

La fuga de clientes es un fenómeno que ha sido ampliamente estudiado en las últimas décadas, debido fundamentalmente al sensible rol que juegan los clientes en los resultados financieros y la rentabilidad de una compañía. El concepto de fuga se refiere a la proporción de clientes que se desliga de una compañía o proveedor de servicios durante un período de tiempo dado. Los costos de captar nuevos clientes por lo general son considerablemente mayores a los costos de

retener a los antiguos clientes, por lo que los enfoques de las empresas cada vez más se centran en generar mejores estrategias de retención, buscando relaciones comerciales fuertes [5]. Sin embargo, es común constatar que muchas compañías atribuyen las razones de la fuga de sus clientes únicamente a problemas con los precios o con el servicio al cliente, dándose a menudo el caso de que aun cuando estos aspectos son mejorados, siguen enfrentándose a altas tasas de fuga. Es por ello que comprender la complejidad y multiplicidad de causas de la fuga es vital para la supervivencia de las empresas. En este sentido, se han desarrollado métodos que utilizan datos históricos relacionados con los clientes (por ejemplo: patrones de consumo, frecuencia de compra, reclamos presentados a atención al cliente, etc.) en el desarrollo de modelos predictivos.

Por su importancia, los modelos de predicción de fuga de clientes han sido aplicados en distintos rubros (bancos, compañías de seguros, proveedores de servicios de internet, compañías de telecomunicación, etc.), convirtiéndose en parte relevante de las herramientas de negocios con que cuentan las empresas. En el caso particular del CCP para compañías telefónicas, los datos necesarios para trabajar con los modelos de predicción provienen fundamentalmente de dos fuentes, las "individuales" y las "sociales" [2, 3]. Las primeras incluyen atributos propios del cliente, por ejemplo, datos demográficos, información de planes de telefonía, antigüedad en la compañía, canales de adquisición del servicio, compañías de procedencia, etc. La segunda fuente proviene de los registros detallados de llamados telefónicos (Call Detail Record,"CDR"), que define las relaciones dentro de una red de llamadas telefónicas a través de atributos tales como: cantidad, duración y dirección de las llamadas entre usuarios de la misma compañía o con clientes de otras compañías, llamadas frecuentes, llamadas a otros clientes fugados, etc.

De acuerdo a las fuentes anteriormente indicadas, en este trabajo se definirán variables/atributos individuales del cliente en conjunto con variables/medidas que caracterizan su "posición" en una red social de llamadas telefónicas. Otra de las variables a considerar es el "momento de la fuga". En este sentido, es importante mencionar que hay dos tipos de fuga: "voluntaria" e "involuntaria" [6]. En el primer caso es el cliente quien decide poner fin al contrato y/o relación con la compañía sin intervención por parte de la empresa, mientras que en el segundo caso, es la compañía quien termina con la relación, debido a factores como deudas impagas o actos que incumplen con el contrato. El momento de fuga también dependerá del tipo de relación que tenga el cliente con la compañía. Para clientes con planes de telefonía móvil, el momento será cuando este termina el contrato, mientras que para clientes pre-pago, se establece un tiempo determinado de inactividad en el que finalmente el usuario es considerado como "fugado" [7–9].

En la literatura es posible encontrar dos enfoques distintos para el CCP. Un primer enfoque es a través de aprendizaje automático y técnicas de minería de datos, los que usan como "input", atributos individuales del consumidor (información demográfica, planes de telefonía, tiempos de llamada, formas de pago, etc.) [8]. Un marco general de cómo utilizar distintas técnicas siguiendo este enfoque es descrito en [10] y consiste básicamente en: aplicar análisis exploratorio de datos (Exploratory Data Analysis, "EDA") con el fin de identificar qué variables pueden diferenciar los distintos comportamientos de los consumidores; extraer, pre-procesar y transformar datos de las variables identificadas, crear modelos de predicción y finalmente evaluar sus desempeños [11]. Algunas de las técnicas de aprendizaje automático más utilizadas en el CCP son: Redes Neuronales [12, 13], Regresión Logística [14, 15], Support Vector Machine (SVM) [16, 17], entre otros métodos. Un estudio detallado de las distintas técnicas de clasificación aplicadas al CCP es entregado por Verbeke et al. en [4]. Un segundo enfoque busca predecir la fuga de clientes utilizando la información contenida en las redes de telecomunicación y las redes sociales. Este enfoque usa herramientas para reconocer patrones de comportamiento mediante el uso de medidas de red, que son calculadas a partir de los CDR (estas redes a menudo son denominadas Call Graphs) [2, 3, 8, 18]. Aquí, se requiere información relativa a los enlaces (es decir, los detalles de las llamadas tales como: origen, destino, duración, fecha, etc.). Algunos ejemplos de aplicación de estas herramientas son los algoritmos de propagación en [18], la clasificación basada en los links en [19], etc.

1.1.1. El rol de las redes en el CCP

Una red social (ya sea en línea o de contacto) está conformada por un grupo de individuos/organizaciones conectados entre sí bajo cierto tipo de relación (por ejemplo: familiar, amistad, trabajo, intereses en común, etc.). Esta red puede ser representada de manera abstracta mediante grafos en los que los individuos/organizaciones conforman un conjunto de "nodos", mientras que los "enlaces" representan las relaciones dadas entre los nodos [20]. Por medio de la red así representada se pueden estudiar diferentes clases de fenómenos, por ejemplo: la propagación/difusión de enfermedades, ideas, opiniones y comportamientos, la adopción de innovaciones, cascadas o "viralización" de conceptos, memes, etc. El estudio de las redes ya se ha convertido en una ciencia madura que se ha expandido a muchísimas áreas [21–23].

La estructura de la red, en particular las comunidades que se forman dentro en ella, influye en los fenómenos de propagación, sea éste de una idea innovadora, un nuevo producto, una noticia, un rumor, etc. Se le llama comunidad al conjunto de nodos que naturalmente se agrupan dentro de la red y que presentan una serie de conexiones densas dentro del grupo pero más escasas con nodos pertenecientes a otras comunidades [24,25]. Dentro de cada comunidad es posible encontrar nodos que cumplen un rol importante o más influyente respecto del resto. Identificarlos dentro de una red puede ser fundamental en el análisis de la eficiencia con la que será propagado el mensaje. Un nodo con mayor cantidad de enlaces debería poder influir de mejor manera en el resto de los nodos pertenecientes a la red.

A menudo se utiliza la terminología asociada a contagios de enfermedades, incluso en aquellos casos donde no es precisamente una enfermedad la que se propaga sobre la red. En efecto, los procesos de difusión de mensajes, cualquiera sea este, se pueden modelar en forma análoga a la propagación de una epidemia, en la que los nodos "infectados" contagian a los demás nodos "susceptibles" [26–28]. Como exponen Xia et al. en [26], una comunidad inicialmente libre de un virus puede convertirse en endémica rápidamente si un nodo influyente es infectado. Si este nodo con autoridad emite el mensaje que se desea difundir, es posible que la comunidad a la que pertenece también lo reciba y éste termine contagiando a toda la comunidad y eventualmente a toda la red.

Para identificar los nodos más influyentes dentro de la red existen diversas métricas altamente estudiadas. El *grado* ("degree centrality") es una de ellas y se define como el número de vecinos con los cuales está conectado el nodo. Esta medida nos permite identificar aquellos nodos que se encuentran más conectados dentro de la red y por ende, cuales deberían ser los más "populares" y/o que poseen mayor información. Cuando la red es dirigida se tienen dos tipos grados: "in-degree centrality" y "out-degree centrality". La primera medida cuenta el número de enlaces que "entran" al nodo (o la cantidad de nodos predecesores), mientras que la segunda medida entrega el número de enlaces que "salen" de un nodo (o el número de nodos sucesores).

Si bien el grado puede entregar información importante respecto de los nodos más conectados, para definir una red con mayor precisión se necesitan otras medidas de centralidad. La utilización exclusiva del degree centrality puede generar problemas en la caracterización de una red, ya que no siempre entrega información concluyente acerca de esta [29–31]. Tal como expone Chen et al. en [29], los miembros de comunidades más grandes tienden a tener un grado mayor en comparación a aquellos miembros de comunidades más pequeñas de una red. Por lo tanto, esta medida de centralidad puede seleccionar a dos nodos pertenecientes a una misma comunidad como "nodos influyentes", generando confusión en la propagación de la influencia o viralización.

Otras métricas definen la importancia del nodo a partir de las trayectorias o caminos que pasan por él. Algunas de las más usadas son la *intermediación central* ("betweenness centrality") y la *cercanía* ("closeness centrality") [30, 32]. La primera medida de centralidad determina el número de veces que un nodo actúa como "puente" en los caminos con distancias más cortas entre otros dos nodos cualesquiera de la red. La segunda medida indica el promedio de las distancias más cortas desde el nodo hacia todos los otros nodos de la red. Intuitivamente, la *cercanía* estima cuáles son los nodos que están mejor posicionados dentro de la red. Estas medidas en

su conjunto pueden ayudar a determinar los nodos con mayor influencia y que eventualmente pueden generar un impacto apreciable en la propagación de un mensaje.

En la propagación del contagio, es importante discutir también el concepto de *homofilia*. La homofilia se define como la inclinación que tienen los individuos a relacionarse/conectarse entre sí debido a sus características similares. Tal como menciona McPherson en [33], el contacto entre personas similares ocurre con mayor probabilidad respecto al contacto entre individuos disimiles. Es importante poder discernir entre los diferentes mecanismos de propagación del contagio producidos por un lado, debido a la influencia que ejerce un nodo sobre otro y por otro lado, debido a la homofilia. Es decir, la reacción similar ante un fenómeno o estímulo por el hecho de estar conectados debido a sus similitudes [20, 33, 34].

En el contexto del CCP, empleando los CDRs, se construye la red de llamadas, que puede ser vista como una clase de red social entre los clientes de una compañía en particular, incluyendo además la relación con clientes de otras compañías. Este tipo de red contiene información valiosa acerca de los patrones de uso del servicio telefónico y sus cambios a través del tiempo. El desafío es extraer conocimiento a partir de los datos de redes, definiendo en primer lugar variables y/o métricas apropiadas (por ejemplo, el grado, la intermediación central, cercanía, etc.) y en segundo lugar reconociendo patrones diferentes entre los clientes fugados y no fugados mediante el uso de técnicas de aprendizaje automático. Los nodos de esta red son los clientes, y los enlaces son las llamadas telefónicas entre ellos. Los nodos contienen además atributos individuales constituidos por toda la información disponible sobre los perfiles de clientes, y los enlaces adquieren atributos a partir de la duración y/o número de llamadas entre los nodos.

1.2. Descripción general del problema

La implementación de la *portabilidad numérica* ha generado grandes migraciones de clientes entre compañías que buscan un mejor servicio. De acuerdo a la página Web de la la portabilidad numérica (que se encuentra bajo la dirección de la Subintendencia de Telecomunicaciones de Chile, SUBTEL), la portabilidad numérica "*se origina en la Ley N° 20.471 y consagra el derecho de los usuarios de telefonía a cambiarse de compañía manteniendo su número. La ley estableció que todas las empresas de telefonía fija, móvil y de voz sobre Internet (VOIP), están obligadas a implementar este sistema tecnológico que entrega plena libertad a los consumidores. Los usuarios son ahora dueños de sus números telefónicos*".

Según datos del la SUBTEL, para Chile en septiembre del año 2017, Entel, Movistar y Claro poseen el 87,9 % del mercado de la telefonía móvil. Por otro lado, Virgin Mobile pasó de tener un 1,6 % en septiembre de 2016 a un 1,1 % en el mismo mes del 2017. WOM destaca por su crecimiento de un 75,2 % en el último año. Además, VTR, Virgin, Telsur, Falabella, Netline, Simple, Telestar y WOM representan un 8,9 % del mercado de prepago.

La Subtel indicó además que en noviembre de 2017 se registraron un total de 376.396 portaciones, cifra que representó un incremento de 18,9 % si se compara con las 316.474 portaciones registradas en igual mes del año pasado. La tasa de portabilidad móvil continúa creciendo, y para 2017 estimó en 19,2 %, lo que contrasta con la tasa obtenida en 2012, que fue de 3,1 %

La tabla 1.1 muestra el resumen de recepciones y "donaciones" (fugas) del año 2016 para Virgin Mobile. El valor neto se calcula tomando la diferencia de la cantidad de números portados como empresa receptora y la cantidad de números portados como empresa donante. Si bien el número de donaciones es menor al número de recepciones, la cantidad de clientes nuevos que llegan a la compañía no es considerablemente mayor al número de clientes que se fugan de esta. La figura 1.1 grafica los datos contenidos en esta tabla.

La dificultad para cuantificar la cantidad de clientes que se fugan de la empresa, los costos asociados a perder un cliente y la influencia negativa que ejerce un consumidor insatisfecho sobre otros clientes se han convertido en una preocupación para la empresa, especialmente para el área de marketing y las áreas de trabajo con clientes (servicios de atención al cliente).

Mes	Recepciones	Donaciones	Neto
Enero	19.681	4.467	15.214
Febrero	21.589	4.557	17.032
Marzo	23.704	3.136	20.568
Abril	21.669	7.459	14.210
Mayo	23.697	8.391	15.306
Junio	24.789	9.864	14.925
Julio	23.226	10.298	12.928
Agosto	22.338	10.932	11.406
Septiembre	18.812	8.279	10.533
Octubre	19.448	8.451	10.997
Noviembre	22.000	8.275	13.725
Diciembre	20.562	9.205	11.357
Total	261.515	93.314	

Tabla 1.1: Balance Virgin 2016. La tabla muestra el resumen de las recepciones y donaciones de clientes de Virgin Mobile en el año 2016. Si bien el número de recepciones tiende a aumentar con el pasar de los meses, el número de donaciones crece considerablemente, lo que queda expuesto en el valor Neto: la diferencia entre ambos valores disminuye a medida que pasan los meses. Los primeros meses del año muestran un crecimiento en las recepciones. Este fenómeno puede deberse a la aplicación de la portabilidad numérica. Sin embargo, durante el segundo semestre del años las donaciones comienzan a intensificarse.

1.3. Hipótesis

Un cliente de una compañía de telefonía móvil está influenciado por su entorno o red social y está condicionado por su perfil y patrón de consumo, por tanto, la decisión de cambiar de compañía o continuar ligado a ella sería predecible, es decir, cuantificable mediante modelos predictivos basados en datos.

Sobre la base de esta hipótesis formulamos el problema de fuga de clientes o CCP, como: *La búsqueda de modelos predictivos que permitan calcular eficientemente la probabilidad de fuga, dado el valor observado de ciertas variables/predictores que caracterizan el comportamiento del cliente como consumidor de servicios de telefonía móvil.*

A partir de aquí nos referiremos al CCP en el contexto de esta formulación. En las próximas secciones iremos desarrollando además los significados precisos de los términos usados en esta formulación. En particular ¿qué entendemos por modelos predictivos?, ¿qué significa el cálculo eficiente de la probabilidad de fuga?, ¿cuáles variables pueden describir mejor el comportamiento del cliente?, ¿cómo éstas variables pueden describir patrones subyacentes en los datos y cómo, en definitiva, permiten la predicción de la probabilidad de fuga?.

1.4. Objetivos

1.4.1. Objetivo General

El objetivo general de esta investigación es construir modelos matemáticos basados en datos para estimar la probabilidad de fuga de clientes desde compañías telefónicas condicionada al valor de sus atributos individuales y/o de red y/o dado que ha sobrevivido un tiempo dado en la compañía.

1.4.2. Objetivos Específicos

1. Recopilar datos del perfil de consumo de clientes de empresas telefónicas.

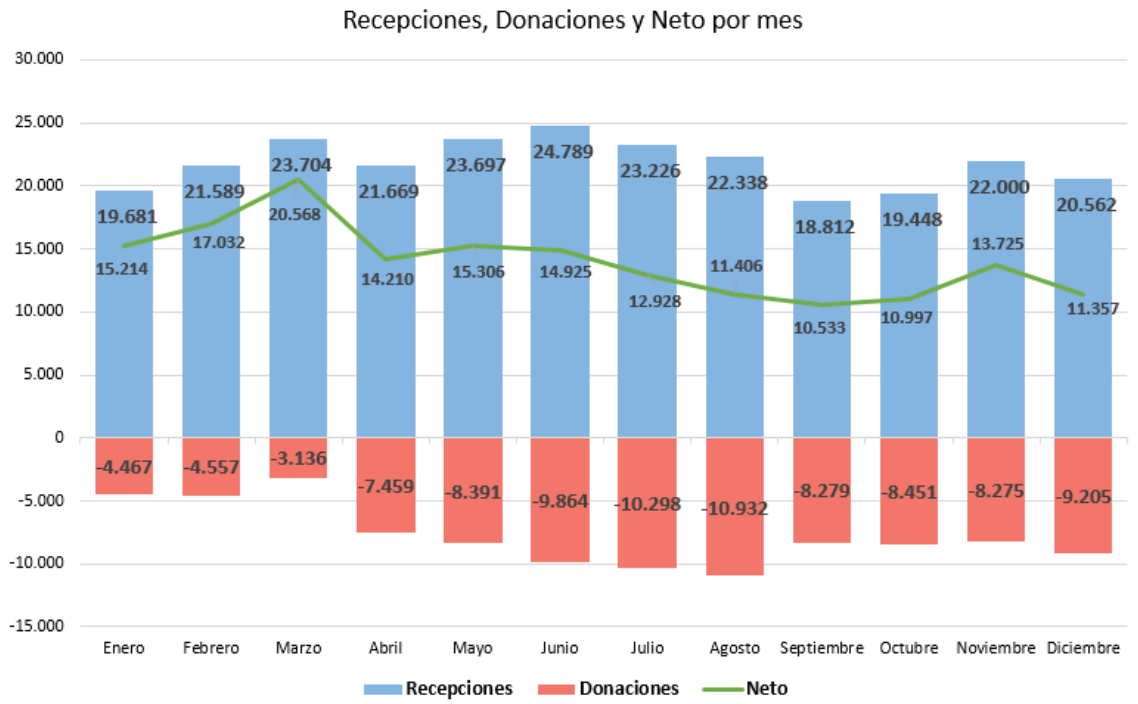


Figura 1.1: Balance Virgin 2016. Gráfico resumen del balance de Virgin para el año 2016. la línea verde indica los valores netos de cada mes. Los datos utilizados fueron recopilados de los balances de portabilidad numérica entregados por la SUBTEL para el año 2016.

2. Crear bases de datos integradas y depuradas con atributos de comportamiento en la utilización del servicio de telefonía móvil junto a datos de las redes sociales construidas a partir de los CDRs.

3. Construir modelos predictivos y de clasificación basado en los datos.

4. Validar las capacidades predictivas y el modelo sobre datos reales.

5. Crear un producto basado en el análisis predictivo de los datos.

1.5. Alcances de este estudio

Con todos los datos disponibles generados por las historias de consumo de los servicios de telefonía móvil proveniente de los clientes y registrados por la empresa (por ejemplo, atributos individuales y de consumo, CDRs, etc.), y utilizando herramientas de minería de datos y analítica predictiva se construyen modelos para la estimación de probabilidades de fuga de clientes. Estos modelos ofrecerán finalmente una ventaja competitiva a la empresa dada la posibilidad de detectar patrones característicos de fugas en el conjunto de clientes, permitiendo así a la empresa tomar decisiones oportunas, inteligentes y fundamentadas con evidencias estadísticamente confiables sobre planes y políticas de retención.

Capítulo 2

Materiales y Métodos

Las bases de datos utilizadas para el desarrollo de la investigación fueron proporcionadas por Virgin Mobile, y se dividen en información de perfiles de clientes y registro de llamadas (Call Detail Records o CDR). Para el manejo de los datos y la aplicación de modelos predictivos se empleó el software R. Para algunas visualizaciones de redes se utilizó Gephi. Dado el gran volumen de datos se usó para los cálculos el High Performance Computer HPC-UDD-Sofia[1].

2.1. Explicación de los datos

2.1.1. Datos de Perfiles de clientes

La información de los clientes utilizada en este estudio es entregada por Virgin Mobile, una compañía de telefonía móvil que opera en Chile y cuenta con datos de alrededor de 1.200.000 clientes. Esta base de datos reúne un conjunto de variables que ofrecen información relacionada con las compañías de origen y destino (en el caso de fuga) de los clientes, fechas de "fuga", la antigüedad, las etiquetas utilizadas para catalogar a los clientes en "fugado" y "no fugado", entre otras. El listado y explicación de estos atributos, que son utilizados en este estudio se muestra en la tabla 2.1.

2.1.2. Datos de Red

La información para crear el "Call Graph" fue obtenida a partir de los CDRs mensuales registrados por la compañía. En los datos de red se consideraron tres meses de observación con un registro de más de 2.000.000 de nodos (que involucran tanto a clientes de Virgin como de otras compañías) y más de 14.000.000 de enlaces (llamadas) por mes. La base de datos incluye registros de llamados realizados/recibidos por clientes que pertenecen a otras compañías de telefonía móvil, pero que interactuaron con clientes de la red Virgin. Menos de un quinto de los nodos corresponden a clientes pertenecientes a la red de Virgin y siempre se puede diferenciar si las llamadas fueron realizadas entre clientes de Virgin o con clientes que se encuentran fuera de la red de la empresa. Adicionalmente, se tiene información referente a la duración de las llamadas y el número total de llamadas realizadas y recibidas por cada nodo.

[1]Características del HPC-UDD: 392 núcleos de procesamiento de CPU (considerando 12 nodos C6320 + 2 nodos C4130 y que cada nodo tiene 28 núcleos). La memoria RAM total (considerando los 14 nodos de cómputo) es de 1344 GB. Capacidad de procesamiento sin contar GPUs de un RPeak de 11.9 TFLOPs. Considerando un 80 % de eficiencia, un RMax de 9.52 TFlops. Procesamiento de GPU: 2xTesla K40 = 2.86 GFlops2 x Intel Phi 7120P = 2.42 GFlops Capacidad de procesamiento total del clúster (CPU + GPU) en RPeak = 17.2 TFlops

Variable	Descripción
Canal	Canal de venta del chip. Puede tomar los siguientes valores: Kiosco Portabilidad, Retail, Masivo y Mayorista, WEB Virgin, Convenios, Empresas, Otros.
Portado	Variable categórica que indica si el cliente portó el número al entrar a la compañía o no. Puede tomar los valores: "Portado" o "No Portado"
Activacion	Fecha de activación del cliente
Suscrito Selfcare	Atributo binario que marca si el cliente está inscrito en la página web o no
Cia de origen	Identificador de la compañía desde donde se portó el cliente
Cia de origen txt	Compañía desde donde se portó el cliente. Puede tomar los siguientes valores: Entel, Movistar, Claro, WOM, Falabella Movil Spa, VTR movil, Otros.
Fecha portOut	Fecha cuando el cliente migra a otra compañía con su número
Fecha churn makt	Fecha cuando el cliente migra a otra compañía con su número
Cia Destino	Identificador de la compañía hacia la cual migró el cliente
Cia Destino txt	Compañía hacia la que migró el cliente. Puede tomar los siguientes valores: Entel, Movistar, Claro, WOM, Falabella Movil Spa, VTR movil, Otros.
Tenure	Variable que indica la antigüedad del cliente en años
Label	Variable binaria que indica si un cliente se fugó o no de la compañía. Puede tomar los siguientes valores: 1=Fugado, 0=No fugado
Text label	Variable categórica que puede tomar los siguientes valores: f=Fugado, nf=No fugado.

Tabla 2.1: Atributos individuales del cliente. Descripción de los datos de perfil utilizados para realizar la investigación.

2.2. Construcción de la Red

Con los atributos fijos y los datos de red se construye el grafo $G = (V, E)$ para representar la red social, donde V es el conjunto de $n \in \mathbb{N}$ nodos, conectados mediante un conjunto E de $k \in \mathbb{N}$ enlaces. Para crear la red, los datos requieren un procesamiento previo siguiendo una serie de procedimientos para evitar errores y sesgos. Así de este modo, una vez que está cargada la base de datos de perfil de clientes, se eliminan aquellos usuarios que tienen intervalos de permanencia entre la fecha de activación y fecha de fuga de la compañía (medidos con las variables: activación, fecha_churn_mkt y/o fecha_portOut) menores que 90 días. A continuación, se define la fecha de fuga como el máximo entre las fechas de salida (fecha_churn_mkt, fecha_portOut). En esta parte es necesario poner atención, ya que hay clientes fugados que tienen solo una de estas fechas definidas. Se eliminan los clientes con números reciclados, es decir, aquellos a los que la compañía les entrega el número de clientes anteriormente fugados. Para hacer esto, se eliminan los clientes (números) que tengan dos o más fechas de activación. Luego, a aquellos clientes que cuenten con una fecha de fuga (es decir, que la variable fecha_churn_mkt y/o fecha_portOut sea diferente de vacío) se les etiqueta como "f" (fugados) y al resto de los usuarios (en los que estas variables se encuentran vacías) se les entrega la etiqueta "nf" (no fugado).

Para la construcción de la red, se cargan los CDR de los clientes con actividad en un mes específico. Esta información se cruza con los datos de perfil con el fin de dotar con atributos

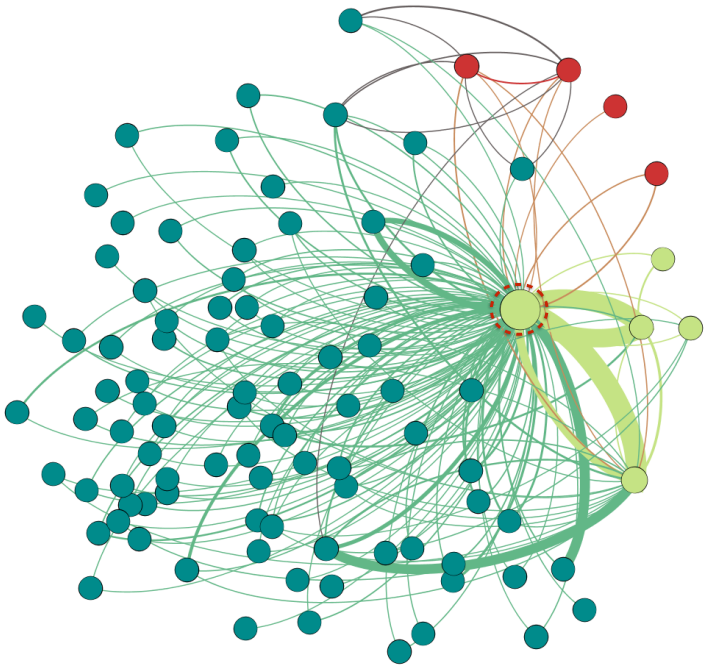
a los nodos de la red. Los datos de redes solo consideran clientes con actividad en un mes determinado, lo que elimina todos los datos de perfil correspondientes a clientes cuya fecha de fuga es anterior al comienzo del mes estudiado. Debido a que las fechas de fuga se definen con 90 días de inactividad y debido a posibles errores de origen en los datos, puede que se incluyan clientes anteriormente fugados en el mes de estudio. Como resguardo se eliminan explícitamente todos los clientes con fechas de fuga anteriores al mes y adicionalmente se eliminan todos los clientes que no registran ninguna actividad en el mes dado. Los enlaces en la red se definen a partir de la interacción entre dos clientes. Es decir, si el cliente i llama al cliente j se establece un enlace entre ambos nodos. El "peso" de cada enlace se define como la duración total de las llamadas entre dos nodos. Los atributos de enlaces se complementan con los datos de duración y cantidad de llamadas, distinción de si la llamada fue hecha o no por un cliente de Virgin, si la llamada fue hecha o no al interior de la red Virgin, la fecha en la cual se realizó la llamada, entre otros datos.

Por otro lado, para tratar la falta de datos en algunas observaciones de variables categóricas, es decir con "NA", estas se reemplazan por datos simulados de acuerdo con la distribución de probabilidades calculada a partir de los datos que si poseen información. Esto se realiza para las variables asociadas a la compañía de origen y al canal de adquisición del servicio. El objetivo de este procedimiento es trabajar con el conjunto completo de observaciones sin alterar la distribución de probabilidad de cada variable. En las variables numéricas asociadas con los enlaces (llamadas) los valores "NA" son reemplazados por "0".

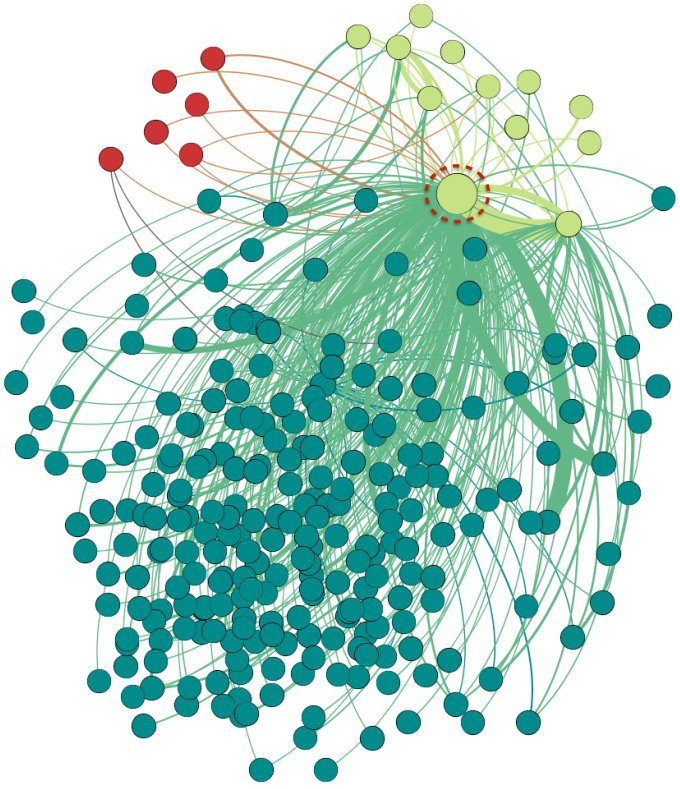
La tabla 2.2 indica las variables de red obtenidas a partir del cruce de información y que son utilizadas para hacer el estudio. En el anexo A.2 es posible encontrar los códigos empleados para generar cada variable.

Variable	Descripción
Num Vecinos	Número total de vecinos (degree del nodo)
Num Vecinos Virgin	Número de vecinos Virgin
Num Vecinos Fugados	Número de vecinos Fugados
call from Churned	Número de llamadas provenientes de fugados
call time from Churned	Duración de las llamadas provenientes de fugados
call to Churned	Número de llamadas hechas a los fugados
call time to Churned	Duración de las llamadas hechas a los fugados
tenure	Antigüedad del cliente
total call time	Duración total de llamadas
call Time In	Duración total de las llamadas recibidas
call Time Out	Duración total de las llamadas realizadas
call Out	Número total de llamadas realizadas
call In	Número total de llamadas recibidas
totall calls	Número total de llamadas
call time out on net	Tiempo de llamadas realizadas a usuarios Virgin
call out on net	Número de llamadas realizadas a usuarios Virgin
call time in on net	Tiempo de llamadas recibidas a usuarios Virgin
call in on net	Número de llamadas recibidas de usuarios Virgin

Tabla 2.2: Variables de Red. Consideramos aquí solo el vecindario de primer orden de cada nodo i de la red, que consta de todos los nodos de la red que están conectados directamente a i incluyendo al propio nodo i.



(a) Vecindad con 4 vecinos fugados



(b) Vecindad con 6 vecinos fugados

Figura 2.1: Ejemplos de vecindades para dos nodos (clientes de Virgin) con diferentes números de vecinos fugados. Los nodos de color verde claro son aquellos que pertenecen a la red de Virgin, mientras que los verde oscuros son los que no pertenecen ("NotVirg"). Los nodos rojos representan a los vecinos fugados ("f"). En la visualización, a medida que aumenta el grado del nodo, éste es representado mediante un círculo de mayor diámetro. Del mismo modo, los enlaces con mayor peso son representados con lineas más gruesas. Se puede notar la emergencia de patrones simples. En particular, al aumentar el número de vecinos fugados, también lo hace el número total de vecinos del nodo y también el número de vecinos que no son de la compañía Virgin. Es posible observar también el "núcleo" más cercano al nodo.

Para corroborar que la construcción de la red se realizó de manera correcta y tener una muestra de la complejidad de la red, es posible generar una vista o "ego-network", es decir, visualizar la vecindad de un nodo en específico de la red. Para ello, se definen distintos colores dados a los nodos catalogados como "f", "nf" y "NotVirg" (no clientes de Virgin) y se genera una visualización con Gephi (software de análisis y visualización de redes). El resultado se muestra en la figura 2.1. De esta visualización es posible observar algunos patrones simples. Por ejemplo, a medida que aumenta el número de vecinos fugados en la vecindad de un nodo específico, también aumenta el tamaño de toda la vecindad. En la figura 2.1(a) se puede contar fácilmente el número de nodos presentes, mientras que en la figura 2.1(b) este número aumenta considerablemente, haciendo más complicado este computo. Estas figuras muestran además porque las visualizaciones deben hacerse a pequeñas escalas. Es imposible observar la red completa mediante una visualización única, debido al gran número de nodos y enlaces que esta contiene.

En estas figuras el número de clientes fugados (en rojo) y el número de vecinos de la red Virgin, que no se fugan (en verde claro) son aparentemente similares. Sin embargo, se pudo constatar con múltiples visualizaciones que no siempre los nodos de la red tienen un número apreciable de fugados en su vecindad, incluso muchos de ellos no tienen nodos fugados en su vecindad. Este fenómeno, que será discutido más adelante, mostró un claro desbalance en la cantidad de vecinos fugados (clase minoritaria) respecto de la cantidad de vecinos que no se fugan (clase mayoritaria). Es conocido que este desbalance es una dificultad a tener en cuenta cuando se emplean métodos de clasificación supervisados (ver sección 3.3).

2.3. Análisis Exploratorio de los Datos (EDA) y visualizaciones

2.3.1. EDA

En una primera etapa de trabajo, para comprender los datos, hay que "explorarlos" y así detectar de manera temprana, patrones que nos ayuden a entender la información con que contamos. Uno de los objetivos más importantes del EDA es identificar aquellas variables que son más significativas en el modelado del problema. En efecto, EDA emplea una serie de técnicas y herramientas, en su mayoría gráficas, para maximizar el entendimiento de la estructura de los datos, reconocer valores atípicos, determinar la configuración óptima de variables "predictoras", entre otros objetivos. Estas técnicas por lo general son descritas como una "filosofía" en la que no existen reglas establecidas para el manejo de los datos, pero que, sin embargo, son parte esencial de cualquier análisis estadístico. Su desarrollo debe ser completo y detallado, ya que a partir de estos análisis es posible plantear preguntas que pueden dirigir la investigación y decidir la forma en que se abordará el problema. Los cuatro tipos de análisis exploratorio de datos comúnmente utilizados son:

- EDA univariado no gráfico. Su principal objetivo es apreciar y entender de mejor manera la distribución de las variables de una muestra y detectar los valores atípicos dentro del conjunto de datos. Para este estudio se puede contar con variables categóricas o cuantitativas. Una tabulación de frecuencias para el primer tipo de variables es recomendada. Para el segundo caso es posible realizar un estudio de medidas de tendencia central y dispersión, para caracterizar la distribución de la muestra a través de la media, mediana, desviación, etc.

- EDA univariado con gráficas. En este caso, la información obtenida del item anterior es visualizada de manera gráfica, para entender la distribución de cada variable. Histogramas, diagramas de torta, gráficos de densidad, diagramas de cajas y bigotes son algunas de las herramientas utilizadas para crear gráficas univariadas.

- EDA multivariados no gráficos. Esta técnica generalmente muestran la relación entre dos

o más variables extraídas de una muestra de datos. Algunas de las herramientas utilizadas son la tabulación cruzada, correlación entre variables, covarianza, etc.

- EDA multivariado con gráficas. La información rescatada del item anterior puede, por ejemplo, ser visualizada con ayuda de diagramas de cajas y bigotes agrupados (boxplots) y gráficos de dispersión, etc. Estos gráficos permiten representar de manera simultanea las distribuciones de varias de las variables predictoras y diferenciarlas respecto de la variable dependiente o de respuesta.

En esta investigación el EDA es realizado sobre los datos correspondientes a tres meses (agosto, septiembre y octubre) de registros de llamadas telefónicas de la compañía Virgin, los que en conjunto suman un total de 718.878 observaciones para el caso de la base de datos correspondiente a los atributos individuales de los clientes (ver tabla 2.1). Las primeras columnas de la tabla 2.3 muestran el número de clientes provenientes desde diferentes compañías de telefonía móvil. Las tres compañías más importantes del país (Claro, Entel y Movistar) suman un poco más del 94 % de la procedencia de los clientes que llegan a Virgin Mobile (681.672 clientes en total).

El diagrama de barras, a la izquierda de la figura 2.2, muestra la información analizada anteriormente con la distribución de las compañías de origen de los clientes que se portaron a Virgin Mobile. La imagen muestra que la mayor cantidad de clientes portados proviene de la compañía Entel, seguido de Movistar, Claro y WOM. Si bien WOM representa tan solo el 3.8 % en el origen de los clientes a nivel agregado, hay efectos dinámicos que involucran a esta compañía y que deben tenerse en cuenta.

La figura B.1, que se encuentra en el apéndice B, muestra la distribución de las compañías de origen, separados por el estatus del cliente: fugado o no fugado. Para cada uno de los casos la distribución es similar y a simple vista no se reconoce algún patrón que nos permita diferenciar entre clientes fugados y no fugados de acuerdo a esta variable.

Compañía	Clientes Portados	Proporción Portados	Clientes Fugados	Proporción Fugados
Entel	266.531	37.1 %	7.365	13.9 %
Movistar	209.804	29.2 %	11.680	22.1 %
Claro	205.337	28.6 %	12.007	22.7 %
WOM	27.507	3.8 %	20.376	38.5 %
Falabella	5.594	0,8 %	580	1.1 %
VTR	3.195	0.4 %	654	1.2 %
Otros	910	0.1 %	292	0.6 %
Total	718.878	1	52.954	1

Tabla 2.3: Cantidad de clientes portados y fugados separados por compañía de origen y destino. Tabla resumen de la cantidad de clientes provenientes de cada compañía y cantidad de clientes fugados a cada compañía. Los datos fueron extraídos de los meses de agosto, septiembre y octubre. Las primeras columnas indican que el orden de proveniencia de los clientes es desde Entel, Movistar y Claro, que en conjunto suman más del 94 % de las compañías de origen de los clientes. Las siguientes dos columnas muestran el orden de preferencia de los clientes fugados, donde se observa que WOM, Claro y Movistar son las compañías más recurrentes y que en conjunto suman más del 80 % de las preferencias de los clientes a la hora de irse de Virgin.

La dos últimas columnas de la tabla 2.3 muestran la cantidad de clientes fugados hacia diferentes compañías de telefonía. La preferencia de los clientes fugados es WOM, seguido de Claro y Movistar, que en total suman el 83.2 % (equivalente a 44.063 clientes) de las preferencias de los clientes a la hora de desligarse de Virgin Mobile. El diagrama de barras a la derecha de la figura 2.2, muestra la distribución de las compañías de destino de los clientes fugados, en la que se aprecia la tendencia de los clientes de fugarse preferiblemente hacia WOM. Como se puede observar en la figura 2.3, a partir del año 2013 se genera un incremento sostenido en la cantidad

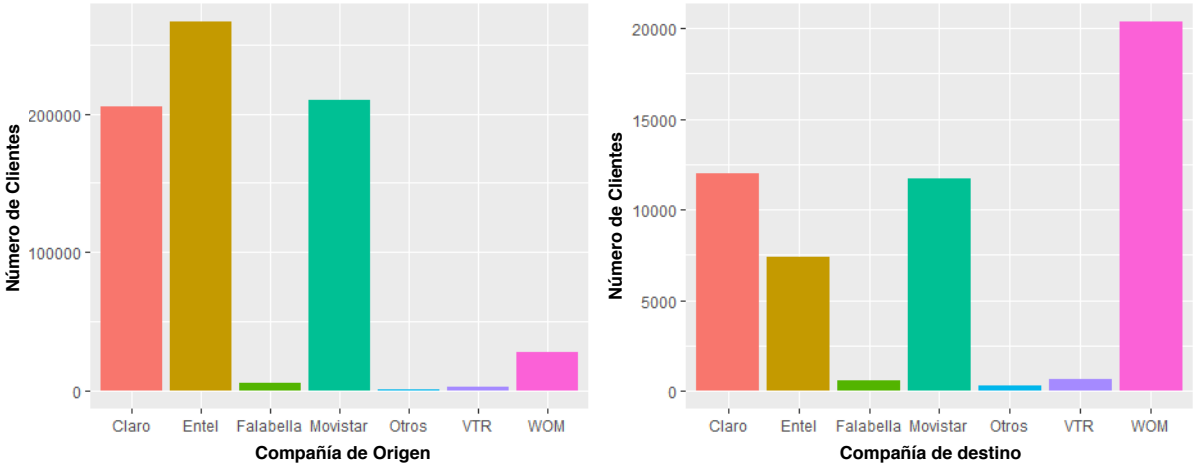


Figura 2.2: Compañías de origen y destino de clientes de Virgin Mobile. La figura de la izquierda entrega una visualización gráfica de la primera columna de la tabla 2.3. Se observa que la mayor cantidad de los clientes que llegan a Virgin Mobile provienen de las 3 compañías telefónicas más importantes del país. Por otro lado, la figura de la derecha indica las compañías de destino de clientes fugados desde Virgin Mobile. Esta vista corresponde a la visualización gráfica de la segunda parte de la tabla 2.3. Se observa que el mayor número de clientes se fuga hacia la compañía WOM, que en los últimos años ha desempeñado un rol importante dentro de las empresas de telefonía móvil en el país.

de clientes fugados. Es importante mencionar que el año 2012 se aplicó el sistema de portabilidad numérica en el país. Sin embargo, no es sino desde el 2015 en adelante que este aumento se hace considerablemente mayor. Ese año llega al país una nueva operadora de telefonía móvil, WOM, quien absorbe gran parte de las fugas de clientes de Virgin Mobile a partir de ese año.

En la figura 2.4 se muestran diagramas de cajas y bigotes correspondientes a la antigüedad de los clientes según la compañía de destino. Los valores de las medianas están ubicados en la segunda mitad del año de antigüedad. WOM expande más su influencia hasta los dos años. Alrededor de la mitad de los clientes se encuentran en el rango de dos años de antigüedad. Por esta razón, vamos a considerar a los clientes con 2 o más años de antigüedad como "antiguos" en la compañía.

La figura 2.5, muestra la antigüedad de los clientes separados por el estatus (fugado, no fugado). De la imagen podemos notar que existen diferencias entre las medianas en las antigüedades de los dos tipos de clientes. A simple vista, la antigüedad de los clientes no fugados es mayor a la antigüedad de los clientes que terminan por fugarse de Virgin Mobile. En valores concretos, la mediana de la antigüedad de los clientes fugados es de 1.005 años, mientras que la de los clientes no fugados es de 1.42 años.

La figura B.3 mostrada en el apéndice B, indica las vías de adquisición del servicio (canal) de los clientes de Virgin Mobile. La mayoría se hace cliente a través de kioskos de portabilidad (54.85 %), seguido de compras en el Retail (25.76 %) y a través de Masivos y Mayoristas (9.51 %), lo que en total suma el 90 % de los canales de adquisición del servicio. Por otro lado, la figura B.2 (que se encuentra en el apéndice B) muestra la distribución de los canales de adquisición respecto de la fecha de activación de los clientes. Es posible notar como en el año 2016, la adquisición de servicios Virgin Mobile se hace principalmente a través de convenios y kioskos de portabilidad, sin dejar de lado que el crecimiento de este último medio es sostenido a través del tiempo desde mediados del año 2013.

La figura 2.6 muestra un diagrama de flujos/tubos ("chord diagram") que visualiza los flujos entre las diferentes compañías, con la condición de que estos flujos de consumidores pasen a través de la compañía Virgin Mobile. El color y tamaño de los flujos (tubos) están en relación con el número de clientes que sale y entra de las compañías pasando por Virgin Mobile. Para obtener esta visualización es necesario contar con una matriz origen-destino que incluya el número de

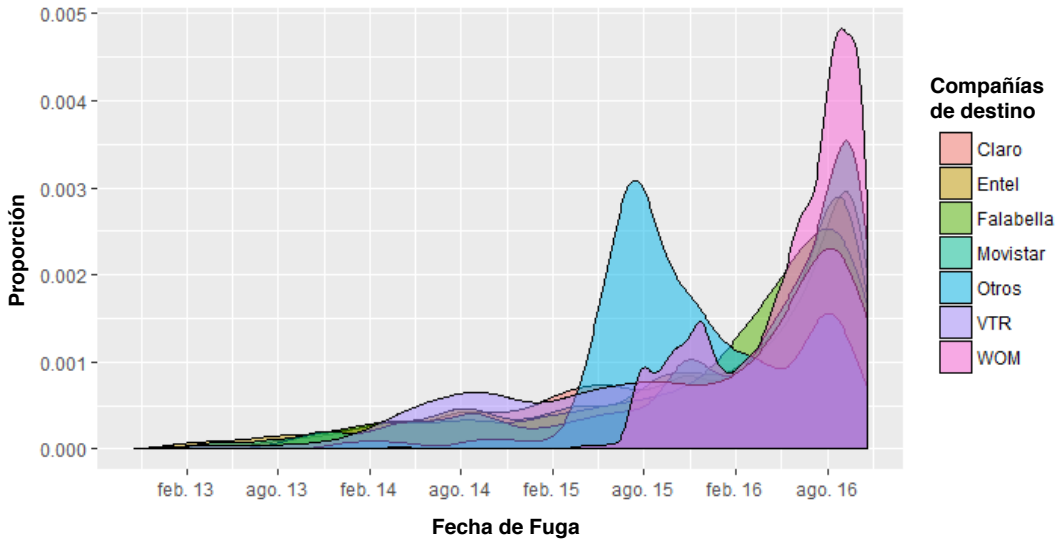


Figura 2.3: Distribución de compañías de destino de clientes fugados desde Virgin Mobile considerando sus fechas de fuga. La figura muestra la distribución de las fugas a lo largo de los últimos años, según las compañías de destino, utilizando como variable en el eje x la "fecha_port_out". Se observa que a partir de agosto del año 2015 la compañía WOM comienza a absorber gran cantidad de los clientes fugados,teniendo un crecimiento considerable a partir de febrero del 2016.

clientes que se mueve de una compañía a otra pasando por Virgin Mobile. Esta matriz es mostrada en la tabla 2.4 e indica que el flujo se presenta mayormente entre Entel, Claro y Movistar. También es posible notar que si bien WOM no tiene muchas salidas, es la compañía que recibe la mayor parte de los clientes fugados, seguido de Claro y Movistar. En la figura, los inicios de cada tubo tienen un borde recto, mientras que los finales de cada tubo son puntiagudos, para indicar hacia donde se dirige cada flujo.

	Entel	Movistar	Claro	WOM	VTR	Falabella	Otros	Total donaciones
Entel	3.564	3.590	3.555	7.690	274	203	117	18.993
Movistar	1.710	4.213	2.816	5.700	166	151	90	14.846
Claro	1.711	3.270	5.048	5.844	158	191	57	16.279
WOM	261	476	410	819	19	18	18	2.021
VTR	45	36	48	104	28	6	0	267
Falabella	64	79	113	184	2	11	7	460
Otros	10	16	17	35	7	0	3	88
Total adopciones	7.365	11.680	12.007	20.376	654	580	292	52.954

Tabla 2.4: Matriz de origen y destino de clientes. Los movimientos aquí registrados representan a los clientes que migran de una compañía a otra pasando por Vigin Mobile. WOM es la compañía que recibe la mayor cantidad de clientes fugados, seguido de Claro y Movistar. Por otro lado,Entel es la compañía que "entrega" la mayor cantidad de clientes, seguido de Claro y Movistar.

La figura 2.7 muestra la matriz de correlaciones de las diferentes variables continuas de los tres meses en estudio. Como se aprecia en el correlograma, la variable Total Calls es la que presenta mayor correlación con el resto de las variables, ya que la correlación con 7 variables es mayor a 0,6. Seguido de Total Calls se encuentra Call Out y Call Time Out, variables sobre las que hay que poner atención al momento de aplicar diferentes modelos de clasificación, ya que pueden generar problemas de multicolinealidad. El correlograma es similar en los tres meses sin son analizados de manera independiente.

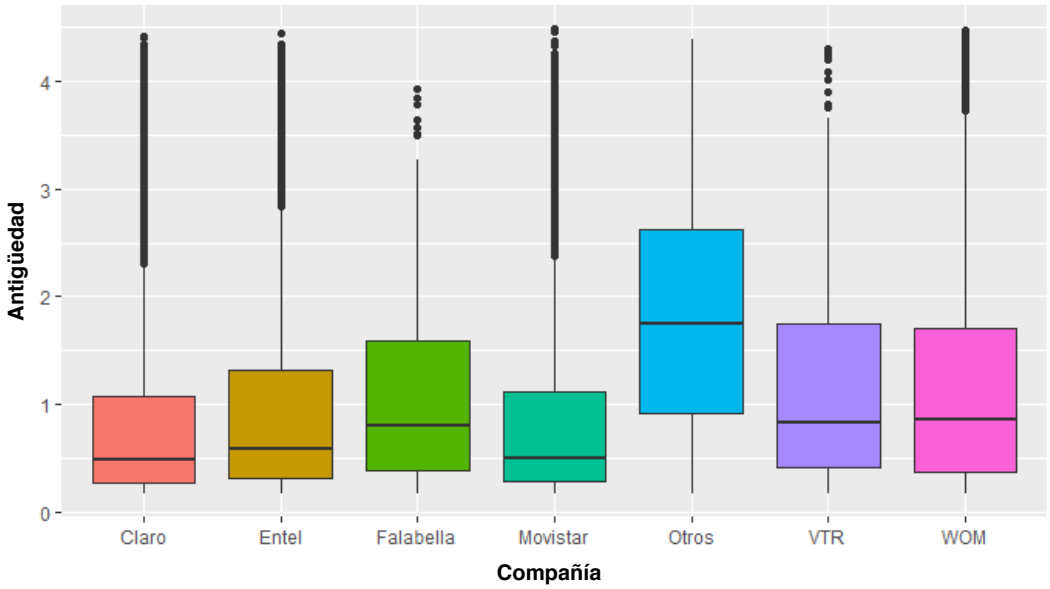


Figura 2.4: Antigüedad de los clientes según la compañía de destino. Los diagramas de cajas y bigotes indican las distribuciones y medianas de las antigüedades de clientes en dependencia de la compañía a la cual se fugan.

La figura 2.8 muestra un gráfico de Coordenadas Paralelas para los tres meses de estudio juntos. El gráfico se realizó utilizando una muestra aleatoria de 74.000 clientes aproximadamente, compuesta por el total de los clientes fugados complementada con una muestra aleatoria de 40.000 clientes no fugados. Esta gráfica busca representar cada dimensión de estudio como una escala vertical paralela. Cada observación está representada a través de una línea que une los valores de las distintas variables.

En la figura, las líneas rosadas representan a los clientes fugados, mientras que las líneas celestes representan a los clientes no fugados. Se observa que en las variables relacionadas con la cantidad y duración de las llamadas realizadas y recibidas (*CallTime_in, Call_in, CallTime_out, Call_out, TotalCalls, TotalCallTime, CallTimeOutOnNet, CallOutOnNet, CallTimeIntOnNet, CallInOnNet*), los valores de los clientes fugados son menores a los valores de los clientes no fugados. Esto indica que los clientes que se fugan muestran una actividad mucho más pasiva respecto de los clientes no fugados.

Las variables relacionadas con la influencia de clientes fugados (*CallFromChurned, CallTimeFromChurned, CallToChurned, CallTimeToChurned*), no parecen mostrar una diferencia notoria entre clientes fugados y no fugados. El mismo fenómeno se aprecia en la variable *NumVecinosFugados*. Sin embargo, es posible notar diferencia entre clases cuando se observan las variables $NumVecinos$ y $NumVecinosVirgnin$. Se aprecia que los clientes no fugados cuentan con una mayor cantidad de vecinos (tanto en el agregado como en la diferenciación "Virgin") respecto de los clientes fugados.

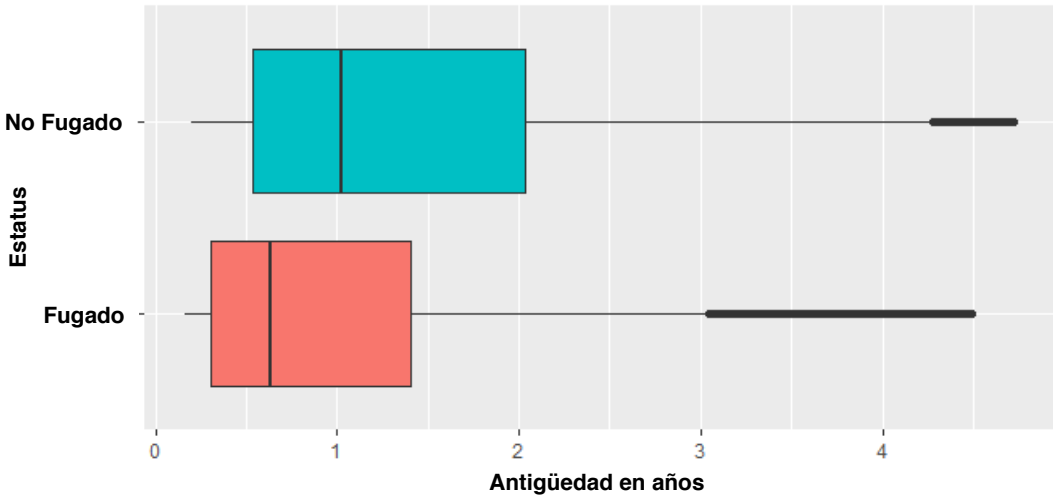


Figura 2.5: Los diagramas de cajas y bigotes muestran la distribución y las medianas de las antigüedades de los clientes separados por estatus (fugado, no fugado). Del gráfico es posible apreciar que los clientes no fugados tienen mayor antigüedad (mediana 1.42 años) que los clientes que terminan por desligarse de la compañía (que presentan una mediana de 1.005 años).

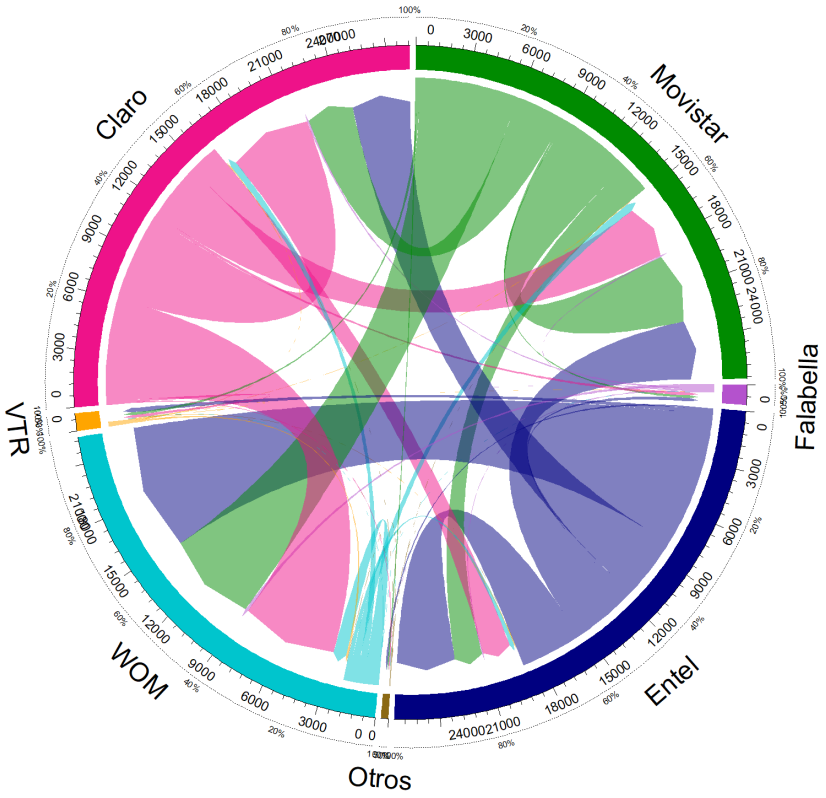


Figura 2.6: "Chord Diagram" para visualizar el flujo de clientes entre compañías de telefonía móvil que pasan por Virgin Mobile. Entel, Claro y Movistar son las compañías de procedencia con mayor número de clientes, mientras que WOM es la compañía que recibe la mayor cantidad de clientes fugados desde Virgin Mobile. Es posible notar como Entel es la compañía que mayor cantidad de clientes "entrega" en comparación a la cantidad de clientes que "recibe".

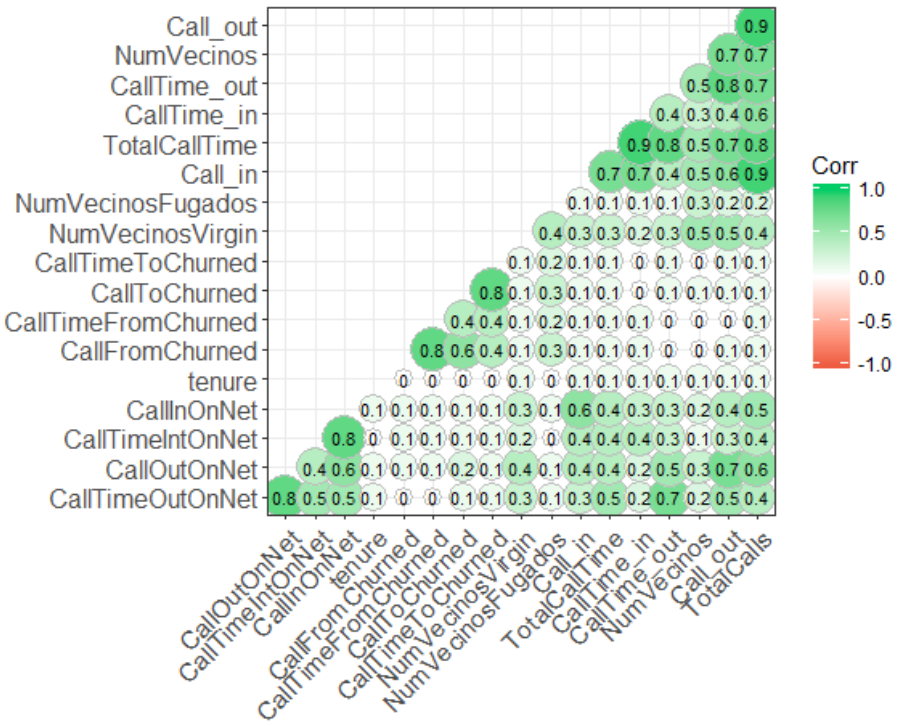


Figura 2.7: Matriz de correlación para los tres meses de estudio. La variable Total Calls es la que presenta mayor cantidad de correlaciones hartas con otras variables, por lo que podría generar el fenómeno de multicolineadidad en el caso de estudio.

Figura 2.8: Gráfico de Coordenadas Paralelas para los 3 meses de estudio. De acuerdo a este gráfico, es posible observar que aquellas variables que logran separar de mejor manera a los clientes fugados de los no fugados son las relacionadas con la duración y cantidad de llamdas (tanto realizadas como recibidas).

Las figuras 2.9, 2.10 y 2.11 muestran los patrones de llamadas de los clientes. Esta visualización es un indicador de que los usuarios en general mantienen patrones regulares de llamadas. Como es posible observar en la figura 2.9 del mes de agosto, las llamadas siguen un patrón, con peaks en ciertos días de la semana y con flujos muy bajos en otros. Los días 5, 12, 19 y 26 de agosto del año 2016 corresponden a días viernes. En estos días se ve un aumento considerable en el número de llamas realizadas por los clientes. Por otro lado, los días 7, 14, 21 y 28 corresponden a días domingo. La disminución en la cantidad de llamadas es notoria respecto del resto de los días de la semana.

Es posible observar que el día lunes 15 de agosto presenta una cantidad menor de llamadas respecto del día 14. Esto se puede deber a que el 15 de agosto es feriado en el país. Esta misma situación se aprecia en la figura 2.11, en el día lunes 10 de octubre de 2016, el cual es feriado por el día de la raza. Septiembre es más complejo de analizar, debido a las características propias del mes. Si bien los días 18 y 19 de septiembre caen domingo y lunes respectivamente (días feriados por Fiestas Patrias), es posible notar perturbaciones en los patrones de llamadas en otros días de la primera mitad del mes, que son difícilmente atribuibles a feriados.

Los patrones de llamadas observados, especialmente en los meses de agosto y octubre, permiten establecer que efectivamente existen regularidades en el comportamiento de los clientes. Estos patrones se hacen más evidentes cuando el análisis se hace de manera semanal. Es importante destacar que para esta investigación se cuenta con información de 3 meses, y por ende, con mayor cantidad de datos sería posible concluir de manera inequívoca esta hipótesis de regularidad en las llamadas. Sin embargo, los patrones aquí observados serán la base para aplicar modelos predictivos que son capaces de reconocerlos.

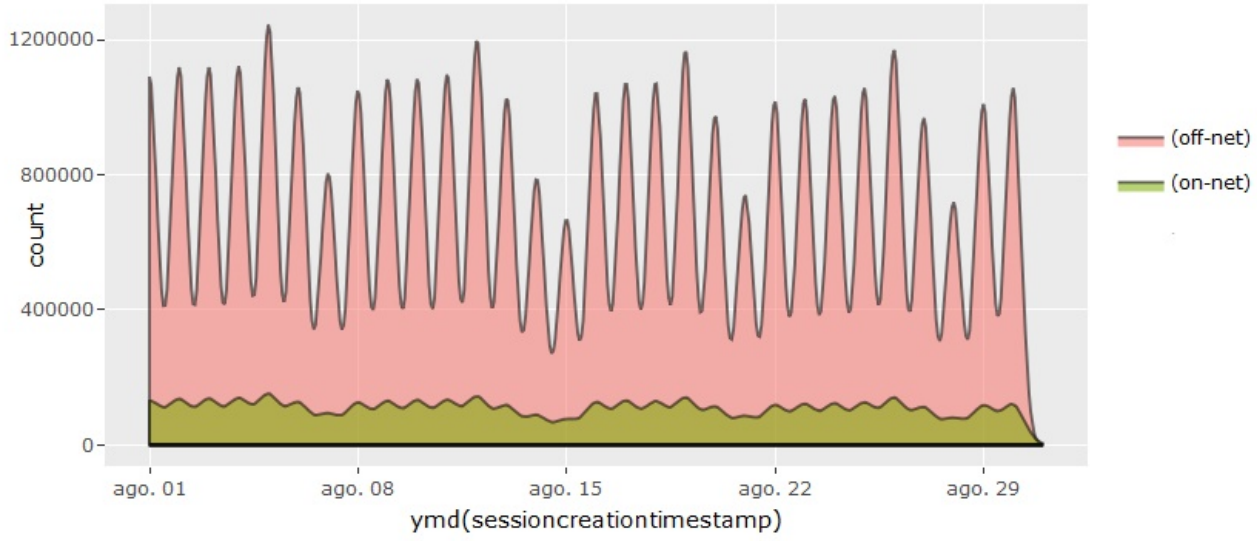


Figura 2.9: Patrón de llamadas del mes de agosto. El comportamiento de las llamadas es regular y es posible notar con claridad los fines de semana y los feriados del mes.

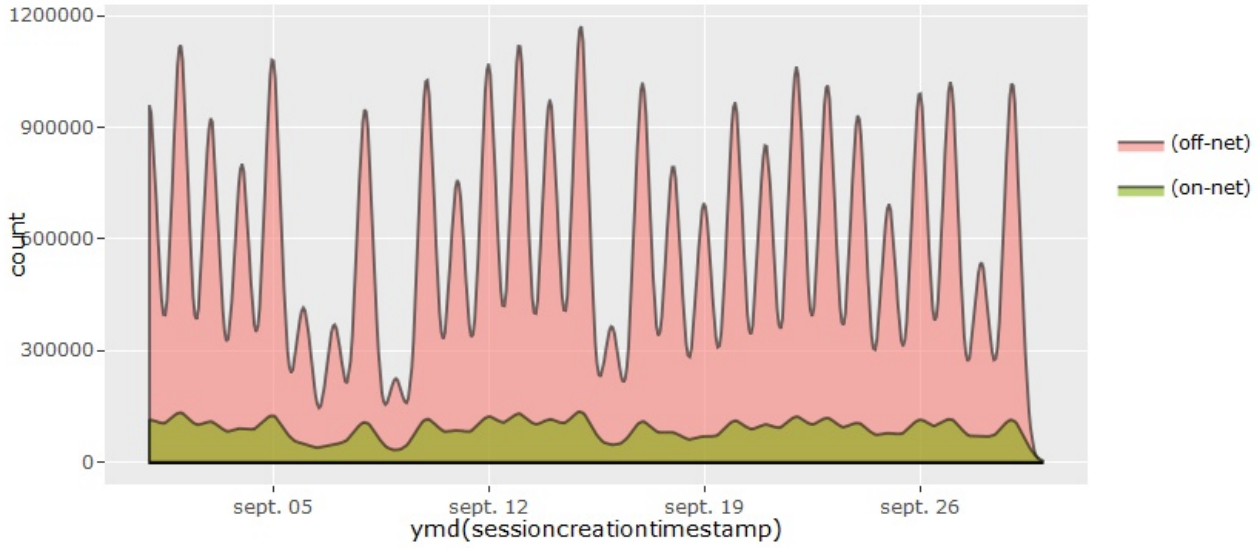


Figura 2.10: Patrón de llamadas del mes de septiembre. Este mes muestra patrones de llamadas más irregulares respecto de los otros dos meses de estudio. Septiembre se destaca por presentar un comportamiento diferente al resto de los meses en Chile. Parte de estas irregularidades se asocian a los días feriados con motivo de Fiestas Patrias.

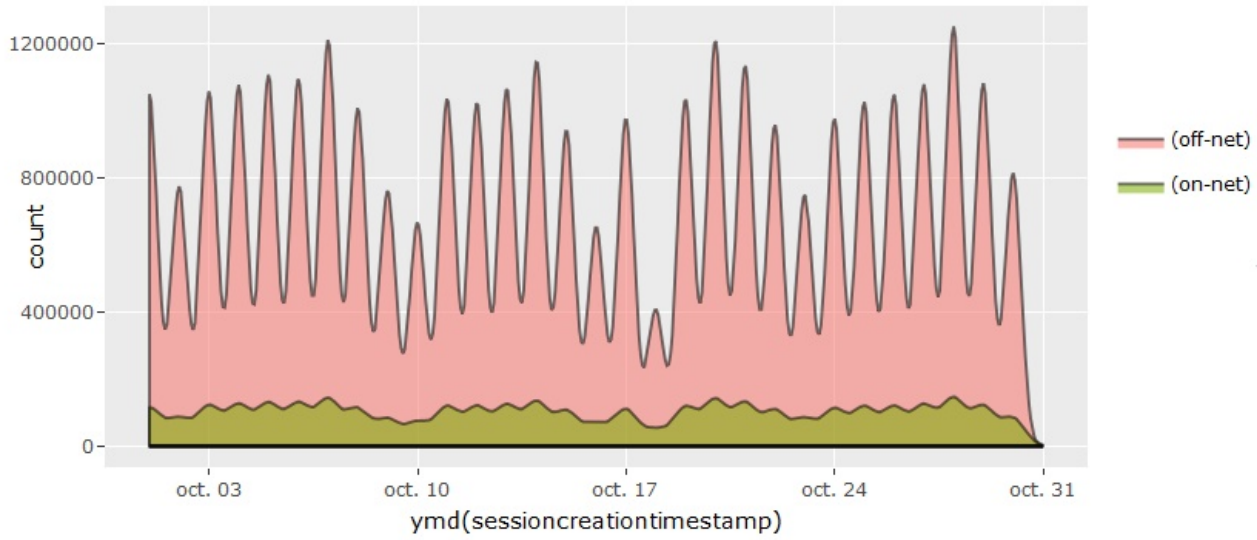


Figura 2.11: Patrón de llamadas del mes de octubre. Tal como ocurre con el mes de agosto, y en comparación al mes de septiembre, es posible notar con mayor facilidad los feriados y los diferentes días de la semana.

Capítulo 3

Modelos predictivos basados en aprendizaje automático

En este capítulo aplicaremos algunas de las metodologías y herramientas más utilizadas en el aprendizaje automático para resolver el CCP. En la sección 3.1 se describen las medidas de desempeño más comunes y que son empleadas en este trabajo para evaluar los modelos y clasificadores. En la sección 3.3 se analiza el problema del desbalance entre las clases "fugados" y "no fugados" en las muestras de datos y se proponen algunos métodos para solucionar este problema. En la sección 3.2 se revisan algunas de las herramientas de aprendizaje automático que serán utilizadas en esta investigación para generar modelos predictivos. En La sección 3.4 se resumen los resultados de la aplicación de herramientas de aprendizaje automático en el CCP, determinando los mejores modelos predictivos de acuerdo a la evaluación de su desempeño. Por último, la sección 3.5 estudia diferentes métodos para identificar las variables más relevantes en el estudio.

3.1. Medidas de desempeño

Para la selección de los mejores modelos en este estudio se requiere de criterios y medidas de desempeño, que permitan concluir de manera objetiva qué modelos son los más eficientes. Existen diversas métricas para esto, tales como la Sensibilidad ("Recall"), Precisión y Exactitud ("Accuracy"), por solo mencionar algunas.

La herramienta estadística más usual para evaluar el desempeño de los clasificadores es la denominada *Matriz de Confusión*, también conocida como matriz de error. La *matriz de confusión* es una matriz cuadrada de $n \times n$ (n es el número de clases o etiquetas de los datos) que compara los valores predichos por el modelo con los valores reales. A partir de esta matriz es posible obtener información necesaria para calcular diferentes métricas de desempeño como las nombradas anteriormente. Las columnas de esta matriz muestran las etiquetas reales de las clases en el conjunto de datos, mientras que las filas muestran las etiquetas predichas por el modelo.

Los valores que toman las clases en una clasificación binaria a menudo se dividen en "positivo" o "negativo". En este trabajo, el valor positivo o clase positiva corresponde al cliente "fugado". Se tienen 4 posibles casos: "True Positive" (tp), "True Negative" (tn), "False Positive" (fp) y "False Negative" (fn). Los casos tp y tn corresponden a las observaciones de la clase positiva y la clase negativa respectivamente, que han sido clasificadas de manera correcta por el modelo.

La Exactitud indica la fracción de los datos clasificados/etiquetados correctamente y se calcula de la siguiente manera:

$$\text{Exactitud} = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{3.1}$$

La Precisión, en general, muestra qué tan cerca están los valores predichos entre sí. Es una estimación de la probabilidad condicional de que una observación esté en la clase positiva, dado que se clasificó como positiva. Se determina de la siguiente manera:

$$\text{Precisión} = \frac{t_p}{t_p + f_p} \tag{3.2}$$

La Sensibilidad o Recall se asocia a la capacidad del modelo para encontrar todos los casos relevantes dentro de un conjunto de datos. Es la fracción de observaciones de la clase positiva que se identificó correctamente y se calcula mediante:

$$\text{Recall} = \frac{t_p}{t_p + f_n} \tag{3.3}$$

La Especificidad estima las sensibilidad para detectar la clase negativa. Es decir, la tasa de observaciones de la clase negativa identificadas correctamente:

$$\text{Especificidad} = \frac{t_n}{t_n + f_p} \tag{3.4}$$

En muchos casos se buscan clasificadores con valores elevados para la Exactitud y Precisión. Sin embargo, en ocasiones debido al desbalance de las clases existente en la muestra de los datos, el objetivo del clasificador es identificar un evento atípico. Así, en situaciones como la detección de fraude por ejemplo, el objetivo más importante es obtener un valor alto para la Sensibilidad, siempre que la Precisión y la Especificidad tengan valores razonablemente buenos.

Una representación de una matriz de confusión se muestra en la figura 3.1. En la figura además, se indican algunas de las métricas más utilizadas para medir el desempeño de los clasificadores.

<table>
<tr><td></td><td></td><td colspan="2">Valor Real</td><td></td></tr>
<tr><td></td><td></td><td>Positivo</td><td>Negativo</td><td></td></tr>
<tr><td rowspan="2">Valor Predicción</td><td>Positivo</td><td>Verdadero Positivo (TP)</td><td>Falso Positivo (FP)
(Error tipo I)</td><td>Precisión
$\frac{\sum verdadero\ positivo}{\sum predicción\ positivo}$</td></tr>
<tr><td>Negativo</td><td>Falso Negativo (FN)
(Error tipo II)</td><td>Verdadero Negativo (TN)</td><td>Valor Negativo Predictivo
$\frac{\sum verdadero\ negativo}{\sum predicción\ negativo}$</td></tr>
<tr><td></td><td></td><td>Sensibilidad
$\frac{\sum verdadero\ positivo}{\sum real\ positivo}$</td><td>Especificidad
$\frac{\sum verdadero\ negativo}{\sum real\ negativo}$</td><td>Accuracy</td></tr>
</table>

Figura 3.1: Se representa una matriz de confusión para un modelo de clasificación binaria (2×2). Esta matriz permite comparar las clases reales y las predichas por el modelo. A partir de la matriz de confusión se obtienen diferentes métricas que permiten evaluar el desempeño del modelo y que se muestran en la fila y columna anexadas a la tabla.

Otra medida frecuentemente utilizada para comparar modelos es el índice "Kappa". Kappa indica cuanto mejor o peor es un clasificador de lo que se esperaría si la clasificación se hiciera por azar. En la matriz de confusión, Kappa mide el porcentaje de valores de las observaciones que están en la diagonal principal de la tabla (las observaciones clasificadas correctamente) y los ajusta de acuerdo a la cantidad esperada debido solo al azar. En [35] se entrega un cuadro de referencia para interpretar los valores de Kappa. Según este cuadro, un valor de Kappa de

0.5 entrega una confiabilidad moderada, mientras que un valor cercano a 1 implica un mejor desempeño del clasificador.

La curva *Característica Operativa del Receptor* ("Receiver Operating Characteristic", ROC), es frecuentemente utilizada en la medición del desempeño de un modelo con clases binarias. Una curva ROC es una representación gráfica de la Sensibilidad versus la Especificidad. Más exactamente, la curva ROC es la curva generada tomando la tasa de verdaderos positivos (tpr) versus la tasa de falsos positivos (fpr) en función del umbral de decisión del modelo, es decir, el criterio para decidir en qué clase ubicar la observación. Una métrica asociada a la curva ROC de un modelo es el área bajo la curva ("Area Under the Curve", AUC), que se encuentra entre 0 y 1. Esta métricas indica un mejor rendimiento del clasificador cuando el valor de AUC esté cerca de 1.

En conjunto, estas medidas son capaces de indicar si un modelo tiene o no un buen desempeño. En general, se recomienda evitar entregar conclusiones apresuradas observando una única métrica de evaluación. Por ejemplo, un valor bueno para la Exactitud no necesariamente indica que los eventos atípicos se estén considerando correctamente.

3.2. Machine Learning y Modelos de clasificación

En el aprendizaje automático ("Machine Learning", ML), los modelos de clasificación son métodos que "aprenden" a partir de una muestra de los datos y utilizan este conocimiento para clasificar nuevas observaciones. Si bien algunos modelos de ML no requieren de grandes bases de datos para su correcto funcionamiento, contar con mayor cantidad de observaciones ayuda a mejorar su desempeño. A mayor información disponible para "entrenar" el modelo, se pueden descubrir tendencias y patrones que en conjuntos de datos más pequeños sería imposible notar, evitando así sesgos en las interpretaciones.

Tal como muestra el esquema de la figura 3.2, existen tres tipos de aprendizajes automáticos: Supervisado, No supervisado y Aprendizaje Reforzado [36–38]. El *aprendizaje supervisado* se caracteriza por trabajar con datos que ya vienen "etiquetados", por lo que el resultado de la clasificación se conoce de antemano. Los algoritmos de aprendizaje supervisado buscan modelar relaciones entre el resultado de la predicción y las características de los datos de entrada. De esta forma, los modelos son capaces de predecir nuevos valores basados en el entrenamiento con el conjunto de datos. Este tipo de aprendizajes generalmente se dividen en modelos de clasificación y regresión. Entre los métodos más empleados en ML, destacan: Árboles de Decisión, Regresión Logística, Máquinas de Vectores Soportes ("Support Vector Machine". SVM), entre muchos otros. Sus aplicaciones son muy variadas e incluyen temas tan diversos como: el pronóstico del tiempo, la detección de fraude, clasificación de imágenes, estimación de la expectativa de vida, reconocimiento de voz, etc.

Un segundo conjunto de algoritmos están basados en el *aprenizaje no supervisado*. En este caso, los datos no cuentan con etiqueta o categorías previas. El objetivo principal es clasificar las observaciones utilizando los datos de entrada para extraer reglas, detectar patrones, resumir y agrupar los datos en grupos o conglomerados (clusters). Dentro de los algoritmos no supervisados más utilizados destacan: K-vecinos más cercanos ("K-Nearest Neighbors" KNN), Análisis de Componentes Principales ("Principal Component Analysis", PCA), Mapas Auto Organizados ("Self-Ornanized Maps, SOMs), etc. Entre algunas de sus aplicaciones se encuentran: la reducción del número de variables (dimensión), la segmentación de los datos (por ejemplo, segmentación de clientes), visualización de grandes bases de datos, entre muchas otras.

Los algoritmos de Aprendizaje de Refuerzo ("Reinforcement Learning", RL) determinan automáticamente el comportamiento ideal dentro de un contexto específico, con el fin de maximizar el rendimiento. A estos modelos no se les indica qué acciones tomar, sino que deben descubrir qué acciones producen la mejor recompensa. Los algoritmos aprenden a través de prueba y error, por lo que el sistema se "refuerza" a medida que se toman una secuencia de decisiones exitosas.

Algunos de los algoritmos más comunes son "Q-Learning", "Temporal Difference" (TD), "Deep Learning", etc. Estos algoritmos han sido aplicados exitosamente en: robótica, video juegos, automóviles que se conducen por si solos, reconocimiento de voz, reconocimiento facial, etc.

En la resolución del CCP el objetivo de trabajar con múltiples modelos de clasificación es encontrar patrones en la actividad de los consumidores. Es decir, mostrar en primer lugar que estos patrones efectivamente existen y pueden ser descubiertos. Es necesario además identificar aquellos modelos que entregan un mejor desempeño en este proceso de identificar patrones y que permitan a su vez reconocer las variables (predictors) importantes para el modelado basado en datos ("data-driven model"). A continuación, incluiremos una breve introducción a los clasificadores que emplearemos en este trabajo para resolver el CCP.

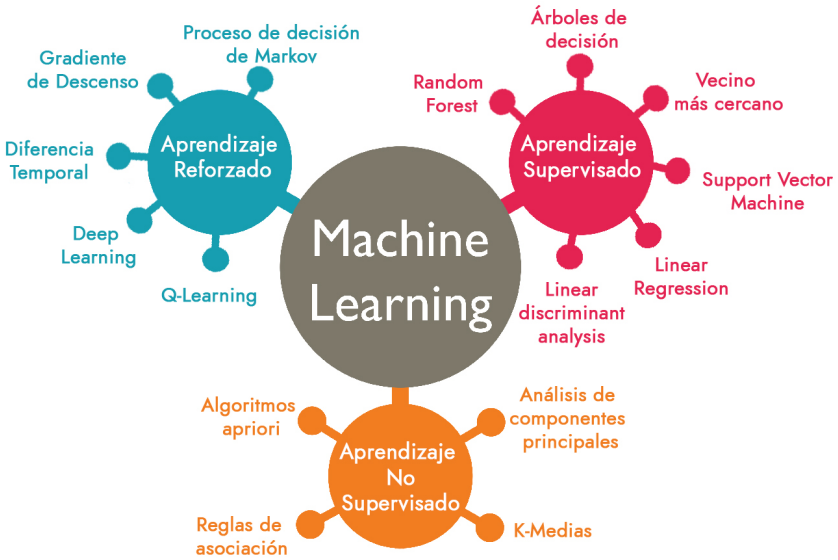


Figura 3.2: Tipos de algoritmos en ML. El esquema resume diferentes tipos de algoritmos, mostrando tres tipos: Supervisados, No-Supervisados y Aprendizaje Reforzado. Para cada uno de estos algoritmos se indican ejemplos de modelos comúnmente utilizados en la práctica.

3.2.1. El Modelo lineal Generalizado ("General linear model", GLM)

El GLM es una herramienta estadística para analizar la relación entre variables o parámetros, describiendo a la variable dependiente en función de todos sus factores contribuyentes (variables independientes). El concepto usualmente se refiere a los modelos de regresión lineal.

Un modelo lineal tiene tres componentes: *componente aleatoria, componente sistemática y función link*. La primera componente corresponde a la variable aleatoria dependiente Y. Por lo general, esta variable es binaria y sus valores se identifican como éxito y fracaso. La componente sistemática corresponde a las covariables o variables explicativas x_i. Por último, la función link especifica una función $g(\cdot)$ que relaciona las componentes aleatoria y sistemática.

Dado un conjunto de variables (predictores) continuas o categóricas, se asume que la distribución de los errores de predicción es normal. Las variables dependientes/etiquetas Y, con observaciones $(y_1, \ldots, y_n)$ donde n es el número de observaciones, se modelan como una función lineal de las variables independientes x_i con $i = 1, 2, \ldots, p-1$, donde p el número de estas variables predictoras. El modelo se escribe en la siguiente forma:

$$\alpha + \beta_1 x_1 + \ldots + \beta_p x_p \tag{3.5}$$

Donde $\overline{\beta} = (\beta_1, \ldots, \beta_p)^t$ es un vector de p parámetros que serán ajustados en el modelo y α es el

error de estimación que se distribuye según la normal. Si se denota el valor esperado de Y como $\mu = E(Y)$, entonces la función link especifica una función g$(\cdot)$ que relaciona μ con el modelo lineal:

$$g(\mu) = \alpha + \beta_1 x_1 + \ldots + \beta_p x_p \tag{3.6}$$

Si bien, los modelos lineales son sumamente útiles en el análisis de datos, éstos están limitados a los supuestos de normalidad, linealidad y homosedasticidad [39]. El GLM permite modelar datos con distribuciones diferentes a la Normal, considerando que la variable respuesta y sigue algún tipo de distribución, por ejemplo: Poisson, Binomial, Gamma o alguna de la familia de las distribuciones Exponenciales. Existen diversos tipos de modelos basados en está lógica: la Regresión Lineal, Análisis de varianza (ANOVA) y de covarianza (ANCOVA), Regresión Logística, etc. (una explicación más detallada de los distintos modelos puede ser encontrada en [40]).

Si se supone que las variables respuesta Y son variables aleatorias independientes relacionadas con los predictores x_i, las componentes del modelo quedan definidas de la siguiente manera: la componente aleatoria Y perteneciente a una familia exponencial, tiene función de densidad dada por:

$$f_Y(y, \theta, \phi) = exp\left\{\frac{y\theta - b\theta}{a(\phi)} + c(y, \phi)\right\} \tag{3.7}$$

donde θ es el parámetro canónico, ϕ es un parámetro adicional de escala o dispersión y las funciones a(), b(), y c() son conocidas. Además, se cumple que:

$$\mu = E(Y) = \overline{b}(\theta) \tag{3.8}$$

La componente sistemática se define como el vector de covariables $\overline{x} = (x_1, x_2, ..., x_p)$ que da origen al predictor lineal:

$$\eta = \sum_{j=1}^{p} x_j \beta_j = \overline{x}\overline{\beta} \tag{3.9}$$

Siendo β el vector a estimar.

Por último, la función link relaciona los componentes μ y η como

$$g(\mu) = \eta \tag{3.10}$$

Los GLM permiten modelar variables respuesta tanto continuas como categóricas. Su importancia radica en que su estructura refleja los elementos explicativos de un fenómeno por medio de relaciones funcionales probabilísticas lineales entre variables.

3.2.2. Análisis Discriminante Lineal ("Linear Discriminant Analysis", LDA)

El Análisis discriminante lineal ("Linear Discriminant Analysis", LDA) es usualmente utilizado para extraer variables relevantes y reducir la dimensión del problema. Su objetivo es descubrir relaciones lineales entre las variables que mejor separen a los objetos, con el fin de clasificar la nueva observación dentro de uno de dos grupos establecidos. El LDA clásico encuentra y proyecta la dirección de los datos, con el fin de maximizar la relación de la distancia entre clases respecto de la distancia dentro de la clase. [41, 42].

El propósito de reducir el número de variables es dejar aquellas que entregan la mayor cantidad de información a la clasificación y permiten a su vez una mejor interpretación del modelo. Algunas variables suelen estar altamente correlacionadas y por tanto es probable que gran parte de ellas no ofrezcan información nueva y relevante. Una descripción de los modelos basados en LDA puede ser encontrado en [41].

Sin embargo, el LDA no solo es utilizado como una herramienta de reducción de la dimensión, sino que también es usado como clasificador. En términos matemáticos, es necesario encontrar un nuevo espacio de variables que permita proyectar los datos con el fin de maximizar la separación

de las clases. Dicho de otra manera, el LDA hace predicciones al estimar la probabilidad de que un nuevo conjunto de observaciones pertenezca a cada clase. La clase con la mayor probabilidad para una observación dada es la clase de salida y con esto se realiza la predicción para dicha observación.

En 1988, el estadístico Ronald Fisher propone maximizar la distancia entre las medias de cada clase y minimizar la propagación dentro de la clase misma como medida del desempeño en la separación de clases del algoritmo. Sin embargo, esta medida solo es aplicable en caso de que los datos sigan una distribución normal. Por ende, un modelo LDA asumirá que los datos de entrada siguen una distribución gaussiana y que la varianza de los atributos es la misma.

La regresión logística modela la distribución condicional de la variable respuesta Y, dado el (los) predictor(es) X. En el LDA, se modela la distribución de los predictores X por separado en cada una de las clases de respuesta (es decir, dado Y) y después se usa el *Teorema de Bayes* para darle la vuelta a esta estimaciones y calcular:

$$P(X) = P(Y = k|X = x) \tag{3.11}$$

Debido a que el objetivo principal de un modelo LDA es separar los datos en clases de manera lineal, aquellos conjuntos de datos que sean linealmente separables tendrán excelentes resultados. En cambio, en aquellos datos que no sean linealmente separables, el desempeño no será óptimo, debido a que a pesar de que el modelo intentará separarlos de manera lineal, seguirán existiendo observaciones que se superpongan entre clases. En estos casos es recomendable utilizar modelos que puedan trabajar con datos no lineales.

3.2.3. K-Vecinos más Cercanos ("K-Nearest Neighbors", KNN)

El método de los K-Vecinos más Cercanos (K-Nearest Neighbors, KNN) es uno de los métodos ML más simples y puede ser utilizado tanto en problemas de regresión como de clasificación. El KNN es un procedimiento no paramétrico que asigna una etiqueta de clase a una observación en función de las etiquetas de clase de los k vecinos más cercanos [43–45]. Mediante una medida de distancia el algoritmo clasifica cada nueva observación dentro del conjunto que corresponda, según tenga k vecinos más cerca de un grupo u otro. La clasificación se realiza calculando las distancias de la nueva observación al resto de todas las observaciones existentes.

Para su aplicación se requiere definir el número de vecinos cercanos k y la distancia entre las observaciones. El valor de k puede ser definido de manera arbitraria o bien, utilizando Validación Cruzada ("Cross Validation") para encontrar un valor óptimo. Por otro lado, la *distancia* tiene diferentes formas de ser definida y su elección estará condicionada al conjunto de datos y tipo de clasificación. Las dos definiciones de distancia más utilizadas son: la distancia euclidiana y la similitud coseno. Para dos observaciones $\bar{x}_i, \bar{x}_j \in \mathcal{R}^p$, en el caso euclidiano, se tiene:

$$d(\bar{x}_i, \bar{x}_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2} \tag{3.12}$$

En el caso de la distancia mediante similitud coseno se emplea la diferencia de dirección entre dos vectores (observaciones) mediante:

$$cos(\theta) = \frac{\bar{x}_i \cdot \bar{x}_j}{\|\bar{x}_i\| \, \|\bar{x}_j\|} \tag{3.13}$$

A pesar de ser un modelo simple y efectivo, KNN presenta algunos problemas. Uno de los más comunes es la facilidad con la que el rendimiento del algoritmo se ve afectado por la presencia de datos con ruido ("noisy data"). Por otro lado, al ser un "lazy learning method", es decir un algoritmo que no crea modelos a partir del aprendizaje con datos de entrenamiento, no es apropiado en aplicaciones donde los datos se actualizan dinámicamente en tiempo real. A estos problemas se le suma su lentitud cuando el número datos de entrenamiento es muy grande, lo que deriva en una mayor necesidad de capacidad de computo y tiempo [46].

3.2.4. Árboles de Clasificación y Regresión ("Classification and Regression Trees", CART)

CART es un método no-paramétrico de segmentación binaria recursiva donde el árbol es construido dividiendo repetidamente el espacio predictivo. Si la variable respuesta es continua se habla de un árbol de regresión, en caso de ser categórica se utiliza un árbol de clasificación. CART provee la base para otros algoritmos de clasificación importantes tales como: Bagged Decision Trees, Random Forest (RF) y Boosted Decision Trees.

A diferencia de otros modelos de regresión CART no genera una función de decisión, pero en cada nodo del árbol se aplica una regla de decisión. El objetivo es segmentar las observaciones (espacio predictivo) en grupos homogéneos manteniendo el árbol razonablemente pequeño. A grandes rasgos se realizan tres pasos: 1) hacer crecer un árbol máximo utilizando segmentación recursiva, 2) podar el árbol y 3) seleccionar el árbol óptimo.

El árbol crece desde el nodo raíz que contiene todo el espacio predictivo y éste comienza a dividirse en diferentes regiones. En cada nodo/división los datos son segmentados en dos grupos mutuamente excluyentes. Eventualmente se llega a un nodo terminal o "hoja", donde se realiza la predicción. Esta predicción agrega o promedia todos las observaciones de entrenamiento que alcanzan esa hoja [47, 48].

Las divisiones se seleccionan de modo que "la impureza" de los nodos sea menor que la del nodo inicial. Seleccionando un criterio de partición se determinará la medida de impureza, la que establece el grado de homogeneidad en los grupos. La función de impureza permite determinar la calidad de un nodo. Existen varias funciones que sirven a este propósito, entre las que destacan: la entropía, el índice de Gini y el índice Towig [49].

El proceso descrito anteriormente por lo general conduce a buenas predicciones sobre el conjunto de entrenamiento, sin embargo puede producir un sobre-ajuste ("overfit") a estos datos. Es decir, puede que se adapte muy bien a los datos de entrenamiento, pero que obtenga un rendimiento deficiente cuando se use con un conjunto de prueba. El problema es que el árbol resultante obtenido por este procedimiento puede ser muy complejo. Un árbol más pequeño, con menos nodos/divisiones, podría tener una varianza más pequeña y una mejor interpretación a costa de un pequeño sesgo. La estrategia que se sigue es construir un árbol grande T_0 y después "podarlo" para obtener un sub-árbol. Intuitivamente, el objetivo es seleccionar un sub-árbol que llegue a la tasa de error más baja. Hay muchas formas de podar el árbol, una de las técnicas más empleada es el "Cost Complexity Pruning", que considera una secuencia de sub-árboles indexados por un parámetro no-negativo α. Cada valor de α corresponde a un sub-árbol $T \subset T_0$ tal que la siguiente expresión sea la mínima posible

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha \mid T \mid \tag{3.14}$$

Donde $\mid T \mid$ indica el número nodos terminales (hojas) del árbol T, R_m es el subconjunto del espacio de observaciones que corresponde al último nodo terminal m, $\hat{y}_{R_m}$ es la respuesta predicha asociada con R_m, es decir, la media de las observaciones de entrenamiento en R_m. El parámetro de ajuste α controla el trade-off entre la complejidad del sub-árbol y el ajuste a los datos de entrenamiento. Cuando $\alpha = 0$, entonces el sub-árbol T simplemente será igual a T_0. En este caso (3.14) solo mide el error de entrenamiento. Sin embargo, a medida que aumenta α, hay que pagar un precio por tener un árbol con muchos nodos terminales, por lo que (3.14) tenderá a minimizarse para un sub-árbol más pequeño.

Un árbol de clasificación es muy similar a un árbol de regresión, excepto que se usa para predecir una respuesta cualitativa en lugar de cuantitativa. Para el árbol de clasificación, se predice que una observación de prueba pertenece a la clase más común/frecuente a la que pertenecen las observaciones de entrenamiento de la región a la que esta observación pertenece. Al interpretar los resultados de un árbol de clasificación, a menudo nos interesa no solo la predicción de la clase que corresponde a una región de nodo terminal dada, sino también las

proporciones de clases entre las observaciones de entrenamiento que caen en dicha región.

Una alternativa natural al RSS (del caso del árbol de regresión) es la tasa de error de clasificación. Dado que planeamos asignar una observación en una región dada a la clase de las observaciones de entrenamiento más común en esa región, la tasa de error de clasificación es simplemente la fracción de las observaciones de entrenamiento en esa región que no pertenecen a la clase más común $E = 1 - \text{máx}_m(\hat{p}_{mk})$, donde $\hat{p}_{mk}$ es la proporción de observaciones que están en la región m, pero pertenecen a la clase k. En la práctica se usa otra medida más conveniente denominada entropía $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$, dode K es el número total de clases. Se puede mostrar que $D \approx 0$ cuando $\hat{p}_{mk}$ es cercana a 1 o 0. Por tanto, la entropía tendrá un valor pequeño si el $m - esimo$ nodo es puro/homogéneo.

3.2.5. Bosque Aleatorio ("Random Forest", RF)

El RF es un clasificador que utiliza un conjunto de árboles de decisión para predecir [50]. Al conjunto de árboles creados se les denomina comúnmente bosque (de ahí su nombre "Bosque Aleatorio"). Para clasificar una nueva observación, el RF decide en base a las predicciones realizadas en cada árbol perteneciente al bosque. El procedimiento de un RF es hacer crecer una gran cantidad de árboles y permitirles votar por la clase más popular. Por lo tanto, el bosque elige la clasificación con más votos sobre todos los árboles en el bosque y clasifica la nueva entrada [51, 52].

RF es una mejora del modelo CART con respecto a la Exactitud, ya que compite con los mejores métodos ML al combinar resultados de muchos árboles de decisión y a la Inestabilidad, debido a que los arboles de decisión tienden a presentar problemas de sobre-ajuste. El RF es relativamente estable al ser una combinación de muchos árboles de decisión.

El RF puede ser utilizado en regresiones y clasificaciones, es computacionalmente simple y fácil de ajustar. Además, permite el manejo de variables categóricas y puede trabajar con problemas no lineales. Sus principales dificultades radican en su aplicación a las regresiones, donde los valores extremos a menudo no se predicen con precisión (subestima los máximos y sobre-estima los mínimos), además es menos interpretable que los árboles de decisión.

3.2.6. Máquina Vectorial de Soporte ("Suport Vector Machine", SVM)

Dado un conjunto de N observaciones con sus respectivas etiquetas o clases (x_i, y_i), $i = 1, \ldots, N$, donde $x_i \in \Re^p$, p es el número de variables predictoras, $y_i \in \{1, -1\}$ (1 representa la clase positiva, -1 la clase negativa), el SVM consiste en buscar el hiperplano $H(x) = w^T x + b$ que maximice el margen de separación entre subconjuntos de los datos etiquetados de manera distinta [53] a través de la resolución del siguiente Problema de Programación Cuadrática ("Quadratic Programming Problem", QPP):

$$\begin{aligned} \text{mín } \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i \\ y_i(w^T x + b) \geq 1 - \xi_i, \forall i = 1, 2, ..., N \\ \xi_i \geq 0, \forall i = 1, 2, ..., N \end{aligned} \tag{3.15}$$

Donde ξ_i es el margen de error de la i-ésima observación de entrenamiento en la forma de una variable de holgura y $C > 0$ es un parámetro de regulación o costo, w es la normal a dicho hiperplano. La distancia desde el hiperplano al origen está dada por $\frac{b}{||w||}$, donde $||w||$ representa la norma Euclidiana.

En los casos donde las observaciones no son linealmente separables mediante el hiperplano anteriormente descrito, es posible obtener un clasificador no lineal utilizando $Kernels$ [54]. La formulación dual de QPP permite el uso de Kernels en un espacio de Hilbert de mayor dimensión,

donde se puede construir un hiperplano con un margen máximo. Se tiene la siguiente formulación de SVM basada en kernels [55]:

$$\begin{aligned} \text{máx} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \forall i = 1, 2, ..., N \end{aligned} \tag{3.16}$$

Donde α son las variables duales (multiplicadores de Lagrange) correspondientes a las restricciones del QQP original. En este caso la función de decisión es:

$$H(x) = \sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) \tag{3.17}$$

Las α_i^*, $i = 1, 2, \ldots, n$ son las soluciones del QQP (3.16).

Para fines de clasificación, una opción común es utilizar el Kernel Gaussiano:

$$K(x_i, x_j) = \exp\left\{ -\frac{\| x_i - x_j \|^2}{2\sigma^2} \right\} \tag{3.18}$$

Para la clasificación SVM con Kernel Gaussiano, se deben estimar dos parámetros: C que regula el costo o penalización de violar el margen entre las clases y σ que es un parámetro propio del Kernel. No existe un procedimiento único para estimar estos dos parámetros.

Otros posibles Kernels utilizados para construir clasificadores son detallados en [16]. Algunos de los Kernels más utilizados son:

- Kernel Lineal:
$$K(x, x_i) = (x \cdot x_i)$$
- Kernel Polinómico:
$$K(x, x_i) = (\gamma x \cdot x_i + r)^d$$

3.2.7. Aprendizaje Profundo (“Deep Learning”, DL)

El DL es una metodología de Aprendizaje de Representación (“Representation Learning”). Este tipo de aprendizaje está constituido por un conjunto de métodos que permiten que una máquina reciba datos sin procesar y descubra automáticamente las representaciones necesarias para la detección o clasificación a través del uso de algoritmos de propósito general y múltiples capas no lineales [56]. Los métodos basados en DL poseen múltiples niveles de representación, que se obtienen mediante la composición de módulos simples pero no lineales cada uno de los cuales transforma una representación desde un nivel dado (comenzando con la entrada en bruto) a una representación en un nivel superior más abstracta. Con la composición de suficientes transformaciones de este tipo, se pueden aprender funciones muy complejas. Para las tareas de clasificación, las capas más altas de representación amplifican aspectos de la entrada que son importantes para la discriminación y suprimen las variaciones irrelevantes. Un aspecto clave del DL es que estas capas no están diseñadas por ingenieros humanos sino que se aprenden a partir de los datos mediante un procedimiento de aprendizaje de propósito general.

Las redes Deep Feedforward, a menudo también llamadas feedforward neural networks o perceptrones de múltiples capas (“Multi- layer Perceptrons”, MLP), son los modelos DL por excelencia. Un ejemplo muy utilizado de este tipo de red es la “Convolutional Neural Network” (CNN) diseñada para procesar datos que vienen en forma de arreglos múltiples. Las arquitecturas

recientes de las CNN tienen de 10 a 20 capas, cientos de millones de parámetros (pesos) y miles de millones de conexiones entre unidades.

El objetivo de una red Deep Feedforward es aproximar una función f dada. Por ejemplo, para un clasificador, $y = f^*(\mathbf{x})$ asigna a una entrada $\mathbf{x}$ una categoría y. Esta red define la asignación $y = f(\mathbf{x}, \theta)$ mediante el aprendizaje del valor de los parámetros θ, resultando en la mejor aproximación posible para la función f.

Estos modelos se denominan *Feedforward* porque la información fluye a través de la función que se evalúa desde $\mathbf{x}$, a través de cálculos intermedios utilizados para definir f y finalmente a la salida y. Se denominan *redes* porque, por lo general, éstas representan la composición de múltiples funciones mediante un gráfico acíclico dirigido. Por ejemplo, dadas tres funciones $f^{(1)}$, $f^{(2)}$ y $f^{(3)}$ éstas son conectadas en una cadena, para formar $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Estas estructuras de cadena son muy utilizadas en las redes neuronales. En este caso, $f^{(1)}$ se le llama primera capa de la red, $f^{(2)}$ es la segunda capa y así sucesivamente. La longitud total de la cadena define la *profundidad* del modelo. Es a partir de esta terminología que surge el nombre de "aprendizaje profundo". A estos modelos se les llama neuronales porque utilizan estructuras lógicas que se asemejan a la organización del sistema nervioso.

Otro tipo de redes DL son las "Recurrent Neural Networks" (RNNs). Estas redes son entrenadas empleando el método de "backpropagation" y se emplean fundamentalmente en tareas que involucran entradas secuenciales, como el habla y el lenguaje. Las RNN procesan una secuencia de entrada un elemento a la vez, manteniendo en sus unidades ocultas un "vector de estado" que contiene implícitamente información sobre el historial de todos los elementos pasados de la secuencia.

3.3. Balance de clases en los datos

Cuando las clases o etiquetas de los datos están distribuidas de manera desigual se plantea que la base de datos está *desbalaceada*. Estos desbalances pueden llegar a ser desde 100:1 hasta 1.000.000:1, en los que una clase está claramente mejor representada que la otra. Al usar datos desbalanceados, un algoritmo, basado en aprendizaje automático supervisado, podría no obtener información necesaria sobre la clase minoritaria para hacer una predicción precisa. Este fenómeno acarrea una serie de problemas que a la larga producen predicciones incorrectas y sesgo, comprometiendo significativamente el rendimiento de la mayoría de los algoritmos supervisados del aprendizaje automático. Es por ello, que en los últimos años se han multiplicado las investigaciones en esta línea [57–59].

Existen diferentes métodos para balancear datos. Estos métodos son conocidos como Métodos de Muestreo ("Sampling Methods") y buscan modificar el desbalance alterando el tamaño del conjunto de datos original creando una proporción similar en las clases [60]. Cuando el tamaño de las muestras se reduce, se habla de Métodos de *Submuestreo* ("Undersampling"). En el caso contrario, cuando el tamaño de la muestra aumenta se habla de *Sobremuestreo* ("Oversampling"). Otra manera de balancear los datos es a través del "Synthetic Data Generation", un método que genera un sobremuestreo con datos ficticios a partir de la clase minoritaria.

Uno de los métodos más ampliamente utilizado es la *Técnica de Sobremuestreo de Minoría Sintética* ("Synthetic Minority Over-sampling Technique". SMOTE). Este método genera un conjunto sintético de observaciones correspondientes a la clase minoritaria buscando cambiar el sesgo en el aprendizaje del clasificador hacia la clase minoritaria. Para crear este conjunto de observaciones, SMOTE primero determina los k-vecinos más cercanos ("K-Nearest Neighbors", KNN), generando una muestra sintética, calcula la diferencia entre la muestra minoritaria y su vecino más cercano, luego se multiplica esta diferencia por un número aleatorio entre 0 y 1 y finalmente, se agrega a la muestra minoritaria original. Este método se usa para evitar el sobreajuste al agregar réplicas exactas de instancias minoritarias al conjunto de datos principal. Otro método utilizado es la técnica de *Ejemplos Aleatorios de Sobre-Muestreo* ("Random Over-

Sampling Examples", ROSE). ROSE utiliza un arranque suave (smoothed bootstrap) para la generación de nuevos ejemplos artificiales de la clase minoritaria [61].

En el contexto del CCP, la distribución de las clases "fugado" y "no fugado" por lo general está desbalanceda, siendo "fugado" la clase minoritaria. La tabla 3.1 muestra la cantidad de clientes fugados y no fugados por mes desde la compañía Virgin Mobile. Se observa que octubre presenta una menor proporción de fugados. En la literatura se considera que las clases están desbalanceadas cuando la clase minoritaria representa alrededor del 1 % de la distribución total. En el caso del mes de octubre, por ejemplo, la proporción de clientes fugados es cercana a un 3 %, por lo que emplearemos las técnicas de balance de datos antes mencionadas, previamente al trabajo con los modelos.

	Fugados	No fugados	Proporción fugados
Agosto	30.164	222.356	0,1194
Septiembre	16.918	222.428	0,0706
Octubre	6.690	220.340	0,0294

Tabla 3.1: Número de clientes Fugados y No Fugados por mes. El desbalance en la distribución de clases es más notoria en el mes de octubre.

3.3.1. Aplicación de las técnicas de balance de datos en el CCP

En R existen una serie de paquetes que permiten balancear los datos, tales como `unbalanced, DMwR, ROSE`. Sin embargo, el paquete `caret` ofrece incorporar fácilmente técnicas de sobremuestreo, submuestreo, ROSE y SMOTE con remuestreo de validación cruzada, utilizando las librerías de los paquetes mencionados anteriormente. Para tener una primera aproximación de los efectos de balancear datos, compararemos la predicción de los modelos utilizando las diferentes formas de balanceo que ofrece `caret`. Es importante destacar que debido al volumen de datos y la complejidad de algunos clasificadores, los cálculos tanto de esta sección, como de las secciones siguientes (específicamente en 3.4.1, 3.4.2, 3.4.3 y 3.4.4), fueron realizados en el HPC-UDD.

En este caso, entrenamos un modelo "glmnet" (para mayor información respecto del modelo, revisar sección 3.2.1) de la siguiente manera:

```
ctrl = trainControl(method = "repeatedcv", number=10, repeats=3,
orig_fit = train(StatusFugado ~ ., data = train_data, method="glmnet",
                 preProcess = c("scale", "center"),trControl=ctrl)
```

El paquete `caret` cuenta con `trainControl()`, que permite hacer diferentes validaciones cruzadas. Para realizar una k-validación cruzada (k-fold cross-validation), se define el método en "repeatedcv", en este caso repetida 3 veces. La medida de desempeño utilizada para cada modelo es la Exactitud (Accuracy). Los clasificadores seleccionados para este análisis corresponden a GLMNET, SVM (Radial), LDA y KNN (ver sección 3.2 para mayor información sobre estos modelos) y se entrenan sobre un conjunto de datos "train" correspondiente al 70 % de la muestra originalmente seleccionada (muestra aleatoria de 10.000 observaciones para cada mes). Para probar las distintas metodologías para el balance de clases, el parámetro "data" cambia de acuerdo a lo que queramos probar: "down_train", "up_train", "smote_train" y "rose_train", que son comparados con los datos originales "train_data" para tener la referencia de comparación.

Una vez entrenados los modelos, podemos compararlos tal como que se muestra en la tabla 3.2. Cada uno de los parámetros de estos modelos es definido de forma automática por `caret`. De la tabla observamos que si bien los valores de Accuracy son mejores en los entrenamientos con los datos originales, el resto de las medidas de desempeño son muy malas. Esto, debido a que el clasificador es muy eficiente clasificando la clase negativa, que es mayoritaria, pero no así con respecto a la clase positiva minoritaria. Por ende, el modelo clasifica una gran mayoría de las observaciones como "no fugado", perdiendo precisión y sensibilidad en su desempeño.

Por otro lado, si bien en los entrenamientos con diferentes tipos de balance se genera una pérdida en precisión, si se observa que el resto de indicadores mejoran sus resultados. Estos resultados se aprecian también al aplicar los modelos sobre el conjunto de datos test (que corresponde al 30 % restante de las 10.000 observaciones seleccionadas de manera aleatoria del mes de octubre). De esta manera, es posible ilustrar el uso de las técnicas de balance de datos para el caso del mes de octubre. Los resultados no son replicables en los meses de agosto y septiembre, debido a que la diferencia en la distribución de clases no genera en principio mayores problemas en el desempeño de los modelos.

Clasificador	Balance	Train			Test			
		Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity	Balanced Accuracy
KNN	Original	0,970	0,000	1,000	0,969	0,000	1,000	0,500
	DOWN	0,674	0,750	0,598	0,589	0,714	0,585	0,649
	UP	0,933	1,000	0,776	0,866	0,429	0,880	0,654
	SMOTE	0,816	0,756	0,853	0,838	0,308	0,855	0,581
	ROSE	0,722	0,406	0,999	0,278	0,956	0,257	0,607
GLMNET	Original	0,970	0,005	1,000	0,970	0,000	1,000	0,500
	DOWN	0,738	0,811	0,664	0,654	0,769	0,651	0,710
	UP	0,741	0,806	0,675	0,661	0,780	0,658	0,719
	SMOTE	0,794	0,779	0,804	0,769	0,505	0,778	0,642
	ROSE	0,711	0,613	0,804	0,616	0,846	0,609	0,728
LDA	Original	0,968	0,014	0,998	0,966	0,000	0,996	0,498
	DOWN	0,738	0,811	0,664	0,654	0,769	0,651	0,710
	UP	0,738	0,845	0,631	0,624	0,791	0,618	0,705
	SMOTE	0,794	0,779	0,804	0,769	0,505	0,778	0,642
	ROSE	0,702	0,593	0,807	0,594	0,846	0,586	0,716
SVM	Original	0,969	0,000	1,000	0,970	0,000	1,000	0,500
	DOWN	0,687	0,668	0,701	0,687	0,703	0,686	0,695
	UP	0,851	0,916	0,794	0,747	0,670	0,749	0,710
	SMOTE	0,830	0,787	0,861	0,837	0,407	0,850	0,628
	ROSE	0,932	0,929	0,953	0,302	0,945	0,282	0,613

Tabla 3.2: Medidas de desempeño para muestras balanceadas con diferentes métodos y modelos. El análisis se realizó sobre el conjunto de datos train y test del mes de octubre. Como es posible apreciar, para todos los modelos se ve una mejora en el desempeño al momento de balancear los datos utilizando Up Sample, SMOTE y ROSE.

3.4. Aplicación de los clasificadores

Debido a la cantidad de observaciones con la que se entrena a los modelos predictivos (sobre 200.000 datos con más de 20 variables por cada mes), la aplicación de los clasificadores se divide en 4 partes. La primera parte consiste en la aplicación de los modelos comúnmente utilizados de acuerdo a la literatura estudiada y que son más simples de entrenar con el paquete `caret`. La segunda parte entrena modelos predictivos que en pruebas de menor tamaño entregaron mejores desempeños, que corresponden a SVM y Deep Learning. En la tercera etapa se aplica una combinación de modelos predictivos con el fin de mejorar el desempeño en la predicción de fuga de clientes. La etapa final corresponde a la selección de los mejores clasificadores y el ajuste de parámetros para cada uno de ellos, con el fin de realizar un estudio acabado en cuanto al desempeño de cada modelo. Las métricas de evaluación y comparación de modelos se explican de manera previa en la sección 3.1.

3.4.1. Etapa 1: Aplicación de clasificadores con `caret`

La primera etapa consiste en aplicar los modelos LDA, GLM, GLMNET, KNN, CART y RF. Estos modelos se entrenaron de manera básica utilizando el paquete `caret` de R. La función

utilizada para entrenar los clasificadores es `train()`. Esta función puede ser usada para varios procedimiento: evaluar mediante re-muestreo el efecto de los parámetros de ajuste del modelo en el rendimiento; elegir el modelo óptimo a partir de los parámetros obtenidos anteriormente; o bien, determinar el desempeño de los modelos a través de un conjunto de datos de testeo. Es esta última característica la que nos interesa tener en cuenta, ya que se quiere definir el mejor modelo. Los modelos se aplicarán inicialmente sobre datos de entrenamiento (train), con el objetivo de determinar de manera preliminar aquellos clasificadores que tengan mejor desempeño.

Para este estudio se han establecido diferentes casos que serán evaluados y comparados. Cada uno de estos casos se aplica en cada mes por separado y luego, se entrenan los modelos sobre los tres meses unidos. El detalle de cada caso se explica en la tabla 3.3

Número	Caso	Explicación
1	Variables de red, todos los datos, sin balance	Se utilizan solo las variables de red (incluyendo tenure, que corresponde a una variable fija). Todas las observaciones contenidas en la base son utilizadas, dividiendo el conjunto en train y test. Los datos no son balanceados.
2	Variables sin correlación, todos los datos, sin balance	Del conjunto de predictores con los que se cuenta, se seleccionan solo aquellos que posean una correlación menor a 0.8. Además, el total de observaciones es utilizado separando los datos de manera aleatoria en dos sets: train y test (70 % y 30 % respectivamente). A pesar del desbalance en los datos, no se aplica la técnica de SMOTE para equilibrar las clases de datos.
3	Todas las variables, todos los datos, sin balance	Se utilizan todas las variables con las que se cuenta para hacer el estudio (es decir, variables de red y variables fijas). Se utiliza el conjunto completo de observaciones divididos en datos train y test. Los datos no se balancean.
4	Todas las variables, todos los datos, con balance	Se utilizan todas las variables con las que se cuenta para hacer el estudio. Se ocupa el set de datos completo, separando los datos de manera aleatoria en dos sets: train y test. Se aplica la técnica de SMOTE para equilibrar las clases de datos.
5	Todas las variables, variables de reciprocidad, todos los datos, con balance	Se utilizan todas las variables con las que se cuenta para hacer el estudio y se incluyen 3 variables adicionales relacionadas con la reciprocidad de la red. Se ocupa el set de datos completo, separando los datos de manera aleatoria en dos sets: train y test. Se aplica la técnica de SMOTE para equilibrar las clases de datos.

Tabla 3.3: Resumen de los casos analizados. En esta primera etapa se analizan los resultados de 5 casos diferentes, con el objetivo de hacer un estudio preliminar pero más detallado respecto de qué clasificadores mustran de manera general un mejor desempeño.

Las variables de reciprocidad mencionadas en la tabla 3.3 son: “Número de Vecinos Fugados Cercanos”, “Número Vecinos Virgin Cercanos” y “Reciprocidad”. Estas variables se obtienen utilizando las variables de red pero con ciertas restricciones. Para entender esto, se muestran las líneas de código utilizadas para definir las restricciones. Las variables originales son las que no utilizan ninguna restricción. El código para crear el grafo dirigido asociado a la red es el siguiente:

```
g=graph_from_edgelist(el,directed=T)
M=as_adjacency_matrix(g,attr="weight")
g=graph_from_adjacency_matrix(M,mode='plus',weighted=TRUE)
```

Donde *el* corresponde a la matriz de llamadas (que contiene las variables “from” y “to”, equivalente al origen y destino de cada llamada). La restricción necesaria para crear el grafo que dará origen a las variables de reciprocidad, es la siguiente:

```
g1=subgraph.edges(g,E(g)[which_mutual(g)],delete.vertices = T)
```

De esta manera se consideran solo aquellos nodos en los que se tiene registro de llamadas mutuas, es decir, el nodo "i" llama a nodo al "j" y viceversa. Con este nuevo subgrafo $g1$ se crea la matriz de adyacencia y el grafo asociado. Con los datos obtenidos a partir del grafo con restricción se crea una nueva variable denominada "Reciprocidad", cuyo valor es igual 1 para todos los nodos en el grafo gl. Para considerar la reciprocidad, es necesario cruzar la base de datos original y la base de datos con los nodos que tienen reciprocidad (a las cuales llamaremos X e Y respectivamente). Para esto, seleccionamos algunas variables de la base Y de la siguiente forma:

```
Y1=Y[,.(ID=ID,NumVecinosFugadosR=NumVecinosFugados, NumVecinosVirginR=
NumVecinosVirgin, Reciproco =1))]
```

Con esta nueva información se hace el cruce de ambas bases de datos de la siguiente manera:

```
X=X[Y1,on=c(ID='ID'),nomatch=NA]
```

En línea anterior la opción "nomatch=NA" indica que en los casos donde no hay coincidencia se reemplaza por NA. Luego, para evitar problemas con estos valores NA, se aplica el siguiente código:

```
Z= mutate(X, Reciprocidad = ifelse(is.na(Reciproco),0,Reciproco),
NumVecinosFugadosCercanos = ifelse(is.na(NumVecinosFugadosR),0,
NumVecinosFugadosR), NumVecinosVirginCercanos =
ifelse(is.na(NumVecinosVirginR),0,NumVecinosVirginR))
```

De esta manera, los valores inexistentes son reemplazados por 0. Con estos códigos es posible generar las nuevas variables que serán utilizadas en el estudio. Estas variables permiten suponer que los nodos con enlaces recíprocos son nodos "cercanos".

Antes de aplicar los diferentes clasificadores es importante considerar que el tamaño de las muestras utilizadas varía en cada mes, tal como se indica en la tabla 3.1. También varía la proporción de clientes fugados en cada mes. Sin embargo, la proporción para dividir los conjuntos de datos en train y test es la misma: 70 % y 30 %. Una muestra de los resultados obtenidos a partir de la aplicación de los modelos predictivos se presenta de la tabla 3.4:

	Accuracy					
	Min	1st Qu	Median	Mean	3rd Qu	Max
LDA	0,8784	0,8789	0,8791	0,8791	0,8792	0,8796
GLM	0,8801	0,8802	0,8803	0,8804	0,8805	0,8812
GLMNET	0,8800	0,8803	0,8804	0,8804	0,8805	0,8807
CART	0,9193	0,9202	0,9212	0,9210	0,9217	0,9229
KNN	0,8949	0,8954	0,8957	0,8959	0,8963	0,8973
RF	0,9298	0,9309	0,9320	0,9317	0,9322	0,9332

Tabla 3.4: Muestra de resultados para la métrica Accuracy de diferentes clasificadores.

La tabla 3.4 muestra los valores de Accuracy del caso 3 (ver tabla 3.3) para el mes de agosto. Esta misma tabla se obtiene para el resto de las métricas utilizadas: Kappa, ROC, Sensibilidad y Especificidad. Esto quiere decir que en total es posible conseguir 75 tablas, si se consideran los

3 meses por separado, los 5 casos definidos y las 5 métricas mencionadas. La tabla 3.5 resume los resultados.

Como se puede observar en la tabla 3.5, los valores de Accuracy empeoran cuando se aplica balance en los datos (escenario 4 y 5). Lo mismo ocurre en el caso de la Especificidad. Estos fenómenos se aprecian en todos los meses y modelos estudiados. Recordemos que la Accuracy mide la proporción de eventos clasificados de manera correcta. Por ende, dependiendo de cuan balanceada están las clases, no siempre es una métrica apropiada por si sola. Dicho de otra manera, puede que el valor de la Accuracy sea alto, sin embargo esto puede deberse a que se están clasificando de manera correcta sólo los clientes no fugados, sin considerar la clasificación de los clientes fugados.

Al contrario de lo que ocurre con las medidas comentadas, el valor de ROC y de la Sensibilidad mejora al balancear los datos. Este efecto se aprecia en todos los clasificadores a excepción del modelo CART. Debido a que la proporción de clientes fugados es muy pequeña y el objetivo es diferenciar a estos clientes del resto de la población, estas medidas pasan a cumplir un rol más importante a la hora de decidir qué modelo presenta un mejor desempeño.

De forma general, es posible concluir que el clasificador que presenta mejores resultados es el RF. Si bien en los casos en los que no existe balance de datos el desempeño de CART parece ser mejor, una vez que las clases se balancean utilizando SMOTE, el método del KNN parece entregar un mejor desempeño.

Los casos aquí analizados también fueron aplicados sobre una base de datos que contiene los 3 meses. Este estudio fue realizado con el objetivo de determinar si existen patrones más claros al momento de agregar la información. El resumen de los resultados se observa en la tabla 3.6.

En esta ocasión el caso 3, que corresponde a la eliminación de las variables correlacionadas, no será considerado. Esto debido a que al realizar el estudio de variables correlacionadas se obtenían 15 variables con correlación mayor a 0.8, por lo que el problema reducía en exceso su dimensión. Al aumentar el valor de correlación (hasta 0.99999) la cantidad de variables seguía siendo muy alta (8 variables con ese valor de correlación).

Considerando los otros 4 escenarios y tal como ocurre en los 3 meses por separado, se reconocen los mismos patrones en las métricas de evaluación: tanto la Accuracy como la Especificidad disminuyen su valor al momento de balancear las muestras, mientras que los valores de ROC, Kappa y Sensibilidad mejoran. De acuerdo a los valores de ROC y Sensibilidad el mejor modelo dentro de los aquí estudiados sigue siendo RF, quedando en segundo lugar el KNN. Es debido a estos resultados, que se analizarán en mayor profundidad estos clasificadores, optimizando sus parámetros y comparándolos con los métodos estudiados en la siguiente sección.

		AGOSTO					SEPTIEMBRE					OCTUBRE				
Modelo	Caso	Accuracy	Kappa	ROC	Sens	Spec	Accuracy	Kappa	ROC	Sens	Spec	Accuracy	Kappa	ROC	Sens	Spec
LDA	1	0,879	0,041	0,694	0,030	0,995	0,927	0,042	0,700	0,029	0,996	0,969	0,046	0,782	0,030	0,997
	2	0,879	0,041	0,694	0,030	0,995	0,927	0,044	0,701	0,029	0,996	0,968	0,053	0,782	0,035	0,997
	3	0,879	0,043	0,660	0,031	0,995	0,929	0,042	0,704	0,027	0,997	0,968	0,048	0,786	0,031	0,997
	4	0,726	0,418	0,767	0,532	0,871	0,728	0,422	0,773	0,525	0,880	0,774	0,533	0,845	0,691	0,835
	5	0,718	0,401	0,760	0,523	0,864	0,726	0,420	0,767	0,542	0,863	0,754	0,489	0,826	0,652	0,829
GLM	1	0,880	0,015	0,708	0,010	0,999	0,929	0,005	0,722	0,004	1,000	0,971	0,009	0,826	0,005	1,000
	2	0,880	0,015	0,708	0,010	0,999	0,929	0,005	0,724	0,004	0,999	0,970	0,008	0,827	0,005	1,000
	3	0,880	0,024	0,717	0,016	0,998	0,930	0,007	0,726	0,004	0,999	0,971	0,015	0,829	0,008	1,000
	4	0,739	0,452	0,776	0,583	0,856	0,745	0,466	0,785	0,599	0,855	0,785	0,562	0,862	0,760	0,803
	5	0,728	0,426	0,764	0,556	0,856	0,731	0,434	0,770	0,575	0,848	0,773	0,538	0,849	0,746	0,794
GLMNET	1	0,880	0,015	0,708	0,010	0,999	0,929	0,001	0,722	0,003	1,000	0,971	0,008	0,826	0,005	1,000
	2	0,880	0,015	0,708	0,010	0,999	0,929	0,002	0,723	0,003	0,999	0,970	0,009	0,827	0,004	1,000
	3	0,880	0,024	0,717	0,015	0,998	0,930	0,002	0,726	0,004	1,000	0,970	0,013	0,829	0,007	1,000
	4	0,740	0,452	0,776	0,581	0,858	0,746	0,465	0,785	0,597	0,857	0,785	0,561	0,862	0,760	0,803
	5	0,728	0,426	0,764	0,552	0,859	0,733	0,438	0,771	0,571	0,851	0,774	0,540	0,849	0,748	0,794
CART	1	0,922	0,524	0,750	0,416	0,991	0,946	0,428	0,716	0,314	0,995	0,975	0,292	0,700	0,173	0,999
	2	0,922	0,524	0,765	0,419	0,991	0,947	0,446	0,751	0,337	0,994	0,974	0,280	0,659	0,145	0,999
	3	0,921	0,498	0,730	0,375	0,996	0,946	0,398	0,719	0,276	0,997	0,975	0,324	0,708	0,206	0,999
	4	0,748	0,458	0,727	0,509	0,924	0,742	0,453	0,723	0,548	0,882	0,774	0,521	0,780	0,592	0,913
	5	0,751	0,473	0,738	0,558	0,888	0,737	0,448	0,741	0,606	0,852	0,765	0,523	0,798	0,732	0,799
KNN	1	0,904	0,373	0,747	0,281	0,989	0,937	0,280	0,726	0,192	0,994	0,972	0,172	0,721	0,092	0,998
	2	0,905	0,381	0,747	0,288	0,989	0,937	0,282	0,731	0,192	0,994	0,972	0,183	0,727	0,098	0,998
	3	0,896	0,288	0,730	0,209	0,989	0,933	0,193	0,700	0,113	0,996	0,971	0,079	0,704	0,047	0,999
	4	0,783	0,549	0,841	0,679	0,861	0,777	0,538	0,832	0,681	0,849	0,800	0,590	0,870	0,749	0,838
	5	0,764	0,510	0,820	0,652	0,849	0,759	0,501	0,812	0,654	0,839	0,785	0,558	0,853	0,730	0,824
RF	1	0,929	0,583	0,825	0,479	0,990	0,951	0,505	0,810	0,387	0,994	0,976	0,379	0,842	0,260	0,998
	2	0,930	0,587	0,825	0,479	0,990	0,951	0,506	0,807	0,383	0,995	0,976	0,377	0,847	0,256	0,998
	3	0,932	0,598	0,840	0,490	0,992	0,952	0,516	0,819	0,392	0,995	0,976	0,397	0,856	0,276	0,998
	4	0,898	0,788	0,953	0,811	0,964	0,886	0,762	0,938	0,796	0,952	0,885	0,763	0,946	0,815	0,939
	5	0,892	0,775	0,945	0,800	0,961	0,880	0,751	0,929	0,786	0,951	0,875	0,741	0,934	0,794	0,938

Tabla 3.5: Resumen de métricas para los 3 meses de estudio en los 5 escenarios. La tabla muestra el resumen de los valores promedio obtenidos para cada casa y en cada mes.

Modelo	Caso	Accuracy	Kappa	ROC	Sens	Spec
LDA	1	0,8847	0,0727	0,7247	0,0499	0,9941
	2	0,8856	0,0717	0,7294	0,0487	0,9949
	4	0,7356	0,4426	0,7905	0,5653	0,8635
	5	0,7462	0,4678	0,8013	0,5988	0,8566
GLM	1	0,8861	0,0660	0,7681	0,0427	0,9966
	2	0,8884	0,0881	0,7737	0,0556	0,9972
	4	0,7584	0,4986	0,8104	0,6530	0,8374
	5	0,7599	0,5026	0,8132	0,6618	0,8332
GLMNET	1	0,8862	0,0642	0,7680	0,0413	0,9968
	2	0,8883	0,0852	0,7737	0,0536	0,9973
	4	0,7592	0,4991	0,8105	0,6516	0,8388
	5	0,7623	0,5068	0,8133	0,6609	0,8347
CART	1	0,8993	0,2645	0,6984	0,1969	0,9915
	2	0,9024	0,3769	0,6982	0,3042	0,9804
	4	0,7636	0,5085	0,7585	0,6469	0,8499
	5	0,7621	0,5048	0,7606	0,6554	0,8464
KNN	1	0,8938	0,2855	0,7316	0,2240	0,9815
	2	0,8963	0,3004	0,7336	0,2358	0,9825
	4	0,7906	0,5678	0,8496	0,7128	0,8495
	5	0,7923	0,5704	0,8497	0,7052	0,8565
RF	1	0,9033	0,3797	0,7975	0,3105	0,9805
	2	0,9113	0,4352	0,8140	0,3540	0,9838
	4	0,8825	0,7558	0,9434	0,7936	0,9490
	5	0,8849	0,7611	0,9453	0,7991	0,9497

Tabla 3.6: Resumen de métricas para los 3 meses agregados

3.4.2. Etapa 2: Aplicación de SVM y Deep Learning

Como se mencionó con anterioridad, los clasificadores SVM y Deep Learning fueron probados en muestras más pequeñas y en general presentaron un buen desempeño en la clasificación. Estos modelos pueden ser desarrollados en paquetes diferentes a `caret`: `e1071` para el SVM y `h2o` para Deep Learning [1]. Por estas razones, estos modelos se analizan separados de los clasificadores anteriormente estudiados.

Tal como se describe en la sección 3.2.6, el SVM busca separar las clases a través de hiperplanos. Aquellos problemas que son linealmente separables, pueden utilizar un Kernel lineal. Sin embargo, en esta ocasión, al tener un problema que no puede ser separado linealmente se utiliza un Kernel radial o gaussiano. Los parámetros iniciales en este caso son: $C = 1$ y $\gamma = 0{,}05$. Es importante destacar que debido a que en esta etapa queremos obtener un margen de referencia, sólo se utilizará esta combinación de parámetros. En la sección 3.4.4 se probarán diferentes combinaciones de parámetros.

Utilizamos la totalidad de los datos para cada mes. Los datos son divididos aleatoriamente en conjuntos train y test, correspondiente al 70 % y 30 % de las observaciones respectivamente. Los escenarios a estudiar son los siguientes:

1. Todas las variables de red más la variable “tenure”.

[1] `h2o` es un programa para ML escrito en *Java*. R se conecta a través de una API con h2o. Para mayor información de `h2o`, revisar http://docs.h2o.ai/

2. Todas las variables de red más la variable "tenure" balanceando los datos con SMOTE.
3. Todas las variables disponibles.
4. Todas las variables disponibles balanceando los datos con SMOTE.
5. Todas las variables disponibles considerando 3 nuevas variables de reciprocidad.
6. Todas las variables disponibles incluyendo las de reciprocidad y balanceando los datos con SMOTE.

Los diferentes escenarios fueron entrenados con el conjunto de datos train. La tabla 3.7 muestra el resultado obtenido de la clasificación con SVM sobre el conjunto de datos test. En este caso se cuenta con 5 métricas de evaluación: Exactitud, Precisión, Especificidad, Sensibilidad y Exactitud Balanceada.

Mes	Escenario	Accuracy	Precision	Specificity	Sensitivity	Balanced Acc.
Agosto	1	0,8859	0,8603	0,9991	0,0393	0,5192
	2	0,7746	0,5084	0,8226	0,6137	0,7182
	3	0,8855	0,6844	0,9971	0,0477	0,5224
	4	0,8247	0,3102	0,8811	0,4014	0,6412
	5	0,8861	0,7321	0,9976	0,0494	0,5235
	6	0,8272	0,3190	0,8822	0,4141	0,6482
Septiembre	1	0,9312	0,8906	0,9998	0,0226	0,5112
	2	0,7784	0,1850	0,7893	0,6339	0,7116
	3	0,9295	0,6400	0,9999	0,0032	0,5015
	4	0,8427	0,1918	0,8776	0,3827	0,6301
	5	0,9297	0,6613	0,9997	0,0081	0,5039
	6	0,8470	0,1935	0,8833	0,3689	0,6261
Octubre	1	0,9710	1,0000	1,0000	0,0050	0,5025
	2	0,7989	0,0978	0,8013	0,7181	0,7597
	3	0,9705	0,5000	1,0000	0,0010	0,5005
	4	0,8410	0,1002	0,8498	0,5500	0,6999
	5	0,9705	0,5000	1,0000	0,0005	0,5002
	6	0,8413	0,1011	0,8503	0,5545	0,7023

Tabla 3.7: Clasificación SVM. Los resultados fueron obtenidos a sobre los datos test utilizando los modelos entrenados con el 70 % de los datos de cada mes por separado. En general se observa que la Sensibilidad de los modelos aumenta cuando las clases son balanceadas con SMOTE (escenario 2, 4 y 6), pero esta mejora se refleja en desmedro de la Precisión de los modelos.

De la tabla se observa que para el mes de agosto, los mejores resultados de Exactitud, Precisión y Especificidad se dan en los casos en los que no existe balance de datos (utilización de SMOTE). Sin embargo, los valores de Sensibilidad son considerablemente bajos (bajo el 5 %). La Sensibilidad puede ser considerada como la capacidad de detectar a la clase positiva, es decir, a los fugados. Si se observan los casos en los que se aplica balance de clases, los valores de Sensibilidad aumentan considerablemente. Sin embargo, esta mejora se produce a costa de una disminución en las otras métricas.

Para el mes de octubre se observa con mayor claridad los fenómenos mencionados. Recordemos que el mes de agosto cuenta con una proporción de fugados del 12 %, que es bastante superior a la proporción del mes de octubre que se acerca a un 3 %. La Exactitud muestra un desempeño considerablemente mejor a los otros dos meses, sin embargo, como ya se ha discutido anteriormente, esto puede deberse a que la clase negativa está siendo detectada con mucha

facilidad debido al desbalance en los datos. Por otro lado, los resultados en la Precisión, la Especificidad y Sensibilidad mejoran en comparación a lo obtenido en agosto y septiembre. En los casos donde no se utiliza SMOTE la Precisión llega a un 50 % pero la Sensibilidad alcanza sus peores valores, llegando a un 5 % y 10 %. Si analizamos lo que ocurre al utilizar SMOTE, vemos que la precisión disminuye de manera abrupta, bajando a un 10 %, pero la Sensibilidad de los modelos aumenta considerablemente hasta llegar a un 55 %.

Una conclusión importante que podemos rescatar, es que en general, al comparar los escenarios en los que no se utiliza balance de datos, el mejor desempeño se obtiene en el primer caso (utilizando solo las variables de red más la variable tenure). Sin embargo, la Sensibilidad sigue mostrando valores demasiado bajos para el objetivo del estudio. En este sentido, se justifica la utilización de SMOTE como técnica de balance, con el fin de mejorar la capacidad de detectar observaciones de la clase positiva. Tal como vemos en el escenario 2, en el cual se utilizan las variables de red y se utiliza SMOTE, el valor de la Sensibilidad aumenta considerablemente, pero a costa de una disminución importante en el valor de la Precisión.

A pesar de que los desempeños no son considerablemente mejores respecto de los clasificadores estudiados en la sección anterior, es importante destacar que aquí solo se está utilizando una combinación de parámetros para el SVM (costo igual a 1 y gamma igual a 0.05). Es posible que al ajustar los parámetros se obtenga mejor desempeño en todas las métricas consideradas como importantes.

Describimos ahora los resultados con el uso de Deep Learning (ver sección 3.2.7). En el análisis de utilizaron todas las observaciones disponibles para cada mes, separando los datos en conjuntos train y test que corresponden a 70 % y 30 % de los datos respectivamente. También se realizó el análisis reuniendo los 3 meses. Los escenarios analizados son los mismos planteados para el análisis con SVM.

Las siguientes líneas muestran la forma genérica del código R empleado para clasificar con DL:

```
m <- h2o.deeplearning(x, y, train)
p <- h2o.predict(m, test)
```

Donde x corresponde a los predictores, y representa la variable de salida "Status Fugado", *train* es el conjunto de datos de entrenamiento, *test* es el conjunto de testeo y p es el vector de las predicciones realizadas.

Los resultados obtenidos con la aplicación Deep Learning se muestran en la tabla 3.8. Como se puede apreciar, el desempeño de Deep Learning es considerablemente superior al desempeño del resto de los clasificadores aquí estudiados. Los valores para la Exactitud se encuentran mayoritariamente sobre un 75 %, y los resultados obtenidos para la Precisión y la Especificidad (sobre todo para esta última métrica) son bastante mejores en comparación a lo estudiado anteriormente.

Se observa que los valores obtenidos para la exactitud son mejores cuando los modelos son entrenados sin balancear los datos. En general el valor de esta métrica se encuentra sobre el 90 %, pero su desempeño disminuye bastante al utilizar SMOTE. Este mismo fenómeno se replica en lo que ocurre con la especificidad. Sin embargo, para la precisión y la sensibilidad se aprecia una mejora en los valores cuando se aplica el balance de clases. Si bien en el mes de agosto los resultados no son muy decidores, para el caso de septiembre y octubre el uso de SMOTE genera una mejora considerable en la capacidad de los modelos para detectar a los fugados.

De acuerdo a los resultados mostrados en las tablas 3.7 y 3.8, hay dos combinaciones de variables que destacan como relevantes. En el caso del SVM, son las variables de red más la variable tenure las que entregan el mejor desempeño. Estos valores se ven mejorados cuando las clases han sido balanceadas utilizando SMOTE. Por otro lado, para Deep Learning, los mejores resultados se obtienen cuando se han utilizado todas las variables disponibles más las variables de reciprocidad (considerando además que los datos se balancean).

Mes	Escenario	Accuracy	Precisión	Especificidad	Sensibilidad	Acc. Balanceada
Agosto	1	0,9110	0,5396	0,9390	0,6547	0,7969
	2	0,7279	0,4192	0,6920	0,8682	0,7801
	3	0,9189	0,3781	0,9208	0,8844	0,9026
	4	0,7714	0,5473	0,7356	0,8687	0,8022
	5	0,9173	0,3924	0,9212	0,8515	0,8864
	6	0,7597	0,5582	0,7325	0,8260	0,7793
Septiembre	1	0,9146	0,4744	0,9598	0,4068	0,6833
	2	0,6315	0,1811	0,6083	0,8627	0,7355
	3	0,9439	0,2764	0,9484	0,7702	0,8592
	4	0,7452	0,5132	0,7147	0,8295	0,7721
	5	0,9409	0,2457	0,9457	0,7445	0,8451
	6	0,7599	0,5108	0,7209	0,8774	0,7992
Octubre	1	0,9549	0,3432	0,9814	0,2562	0,6188
	2	0,7084	0,4903	0,6932	0,7470	0,7201
	3	0,9699	0,0391	0,9706	0,6000	0,7853
	4	0,7900	0,6366	0,7649	0,8411	0,8030
	5	0,9746	0,0601	0,9751	0,7272	0,8512
	6	0,8012	0,6414	0,7744	0,8581	0,8162
3 meses	1	0,8951	0,1975	0,9015	0,7078	0,8047
	2	0,6286	0,1975	0,6146	0,7375	0,6760
	3	0,8933	0,8997	0,8997	0,7130	0,8064
	4	0,7676	0,6408	0,7592	0,7830	0,7711
	5	0,9051	0,3177	0,9141	0,7374	0,8257
	6	0,7830	0,5879	0,7511	0,8602	0,8056

Tabla 3.8: Resultado obtenido de la clasificación con Deep Learning. Los diferentes escenarios fueron estudiados utilizando la totalidad de los datos disponibles en cada mes. Como es posible apreciar, el escenario 6 (correspondiente a la utilización de todas las variables disponibles, incluyendo las de reciprocidad, aplicando SMOTE) es el que entrega los mejores desempeños, sobre todo en las métricas de precisión y sensibilidad.

3.4.3. Etapa 3: Combinación de modelos predictivos

Otra técnica importante a considerar son los métodos de conjuntos. Los métodos de conjunto ("ensemble methods") son una técnica de aprendizaje automático que combina una serie de modelos predictivos en uno solo con el fin de mejorar el desempeño inicial, ya sea disminuyendo la varianza ("bagging"), el sesgo ("boosting") o mejorando las predicciones ("stacking"). Estos métodos usan como entradas un algoritmo de clasificación, denominado *modelo base*, sobre el cual adjuntan una serie de otros algoritmos. Para que los métodos de conjunto presenten mejor desempeño que cualquiera de sus miembros individuales, los clasificadores deben ser lo más preciso y diversos posible.

Tal como se mencionó anteriormente, existen tres técnicas que son las más utilizadas para combinar clasificadores:

1. *Bagging* Viene de la combinación de palabras en inglés: "bootstrap" y "aggregation". Esta técnica implica tomar múltiples muestras del conjunto de datos de entrenamiento (con reemplazo) y entrenar un modelo para cada muestra. Generalmente se construyen modelos del mismo tipo. En otros términos, se crean diferentes conjuntos de datos para entrenar una serie de modelos distintos y luego estos modelos son "agregados". La clasificación se realiza utilizando "votación" en el caso de modelos de clasificación (es decir, se elige la clasificación que fue entregada por la mayor cantidad de modelos diferentes) o "promedio"

para obtener el resultado de una regresión.

2. *Boosting* hace referencia a una serie de algoritmos de aprendizaje que son débiles pero que se pueden convertir en clasificadores robustos. Al igual que en bagging, boosting crea múltiples modelos en secuencia (que generalmente son del mismo tipo). Cada uno de estos modelos es capaz de aprender a corregir los errores del modelo anterior. En este caso se utiliza la misma base de datos, pero los pesos de las instancias se ajustan de acuerdo con el error de la última predicción. El objetivo es forzar a los modelos a enfocarse en las instancias que son difíciles de predecir.

3. *Stacking* es una técnica que combina múltiples modelos de clasificación o regresión a través de un meta-clasificador o un meta-regresor. A diferencia de las técnicas anteriores, los modelos creados son por lo general diferentes entre sí. El objetivo de esta técnica es que el modelo "supervisor" sea capaz de aprender cómo combinar mejor las predicciones del resto de los modelos "primarios". En este caso, cada modelo primario es entrenado con el mismo conjunto de datos original, mientras que el modelo supervisor se alimenta y entrena con los resultados obtenidos de los modelos primarios.

Las figuras 3.3, 3.4 y 3.5 muestran las diferencias de los distintos métodos de conjunto descritos anteriormente. Bagging y Boosting tienen N nuevos modelos a través de la generación de nuevos datos en la etapa de entrenamiento. En el caso del Bagging, todos los nuevos elementos tienen la misma probabilidad de aparecer en el nuevo conjunto de datos, mientras que para el caso de Boosting, algunas observaciones tienen mayor peso, y por ende, estas observaciones aparecerán con mayor frecuencia en el nuevo conjunto de datos. Estos nuevos conjuntos de datos son representados en la figura 3.3 como los círculos que contienen una serie de puntos en su interior.

La etapa de entrenamiento es diferente en ambos métodos: para Bagging el entrenamiento de hace de manera paralela (ya que cada modelo es construido de manera independiente), mientras que para Boosting el entrenamiento se realiza de manera secuencial (ya que cada nuevo algoritmo toma en consideración el desempeño del algoritmo anterior).

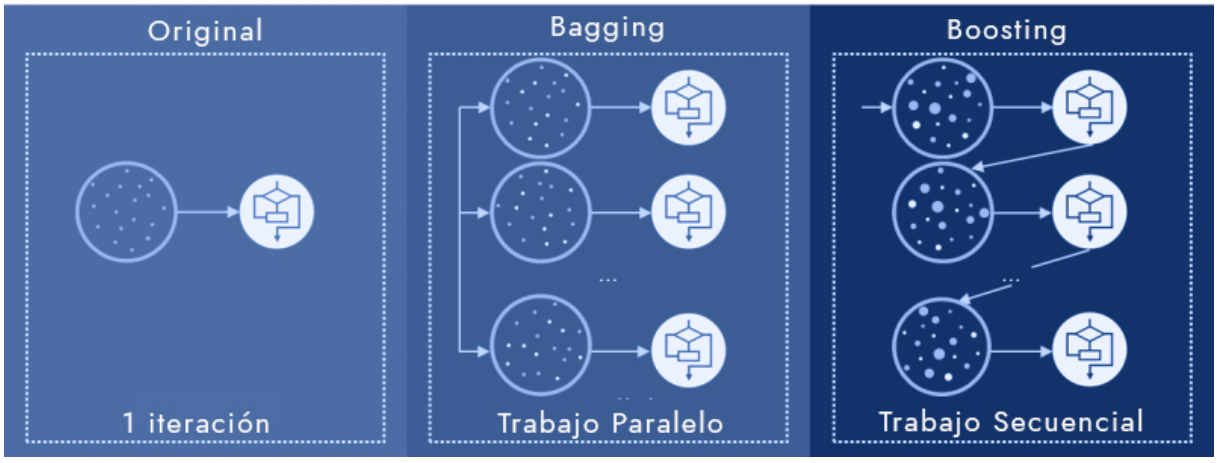


Figura 3.3: Descripción gráfica de la etapa de entrenamiento de ensemble methods. En el caso de bagging el entrenamiento de hace de manera paralela, mientras que para boosting el entrenamiento de realiza de manera secuencial.

La figura 3.4 muestra la diferencia en la etapa de clasificación de cada modelo. Como ya se mencionó anteriormente, Bagging utiliza el promedio de los modelos para clasificar una observación. Por otro lado, Boosting considera diferentes pesos, para calcular un promedio ponderado (los pesos se asignan de acuerdo a los resultados obtenidos en la etapa de entrenamiento, donde aquellos modelos que presentaron mejor desempeño, tendrán una mayor ponderación). Por lo

tanto, Bagging entrena y almacena los resultados de todos los modelos creados, mientras que Boosting entrena y evalúa los resultados, con el fin de seguir el rastro de los errores obtenidos en modelos anteriores (revisar figura 3.5).

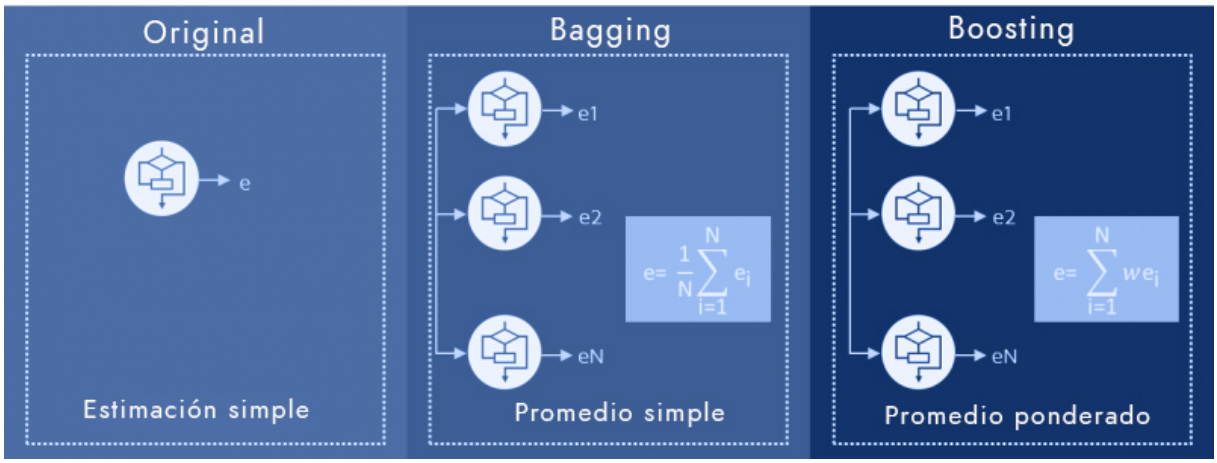


Figura 3.4: Descripción gráfica de la etapa de clasificación de "ensemble methods". *Bagging* considera un promedio simple de los resultados de todos los modelos entrenados, mientras que *Boosting* calcula un promedio ponderado.

Figura 3.5: Descripción gráfica de la etapa de evaluación de "ensemble methods". Al igual que en el entrenamiento de algoritmos de aprendizaje supervisado, para *Bagging* cada resultado es almacenado y considerado de manera posterior. Por otro lado, para *Boosting* se consideran los errores de los modelos anteriores, por lo que cada nuevo resultado debe ser evaluado.

El paquete `Caret` cuenta con una serie de herramientas para ejecutar métodos de conjunto y que simplifican su aplicación. Para esta parte del estudio se aplicarán los modelos de conjunto sobre todas las variables disponibles (atributos fijos, variables de red y de reciprocidad), considerando una muestra de datos compuesta por el total de clientes fugados complementada con 40.000 clientes no fugados elegidos de manera aleatoria. El 70 % de los datos de la muestra serán el conjunto de datos train, mientras que el 30 % restante serán los datos de testeo. En el caso de Boosting y Bagging, se compararán los siguientes métodos: "C5.0", "Stochastic Gradient Boosting" (GBM), "Boosted Logistic Regression" (LogitBoost) y "Bagged CART" (Treebag) (los tres primeros corresponden a método Boosting y los últimos corresponden a Bagging).

3.4.3.1. C5.0

C5.0 es una extensión del modelo C4.5, que a su vez es el sucesor de ID3. Este último algoritmo crea un árbol de múltiples vías, y encuentra para cada nodo la característica categórica

que producirá la mayor ganancia de información para los objetivos categóricos. C4.5 elimina la restricción de que las características deben ser categóricas mediante la definición dinámica de un atributo discreto (basado en variables numéricas) que divide el valor del atributo continuo en un conjunto discreto de intervalos. Este método convierte los árboles entrenados (es decir, la salida del algoritmo ID3) en conjuntos de reglas "si-entonces". Dentro de las mejoras del modelo se encuentra: (1) maneja bases de datos incompletas; (2) resuelve el problema de sobre-ajuste mediante una técnica conocida como "poda"; y (3) se pueden aplicar diferentes pesos a las características que comprenden los datos de entrenamiento. C5.0 es la última versión del algoritmo C4.5, el cual ocupa menor memoria y crea conjuntos de reglas más pequeños.

3.4.3.2. Stochastic Gradient Boosting

Gradient Boosting es una técnica de aprendizaje automático, que utiliza Boosting para combinar una serie de algoritmos débiles para formar un modelo fuerte, por lo general arboles de decisión. Los árboles son utilizados como modelo base, y cada árbol subsiguiente se basa en los errores calculados por el árbol anterior. Intuitivamente, GBM aprovecha los patrones de error y fortalece un modelo de bajo desempeño. El objetivo de GBM, como el de cualquier algoritmo de aprendizaje supervisado, es en definitiva establecer una función de pérdida y minimizar los errores asociados. Una modificación a este modelo permite incorporar aleatoriedad al algoritmo. En cada iteración se extrae una sub-muestra aleatoria del conjunto completo de datos de entrenamiento (sin reemplazo). Esta muestra seleccionada al azar luego se usa, en lugar de la muestra completa, para entrenar al modelo base. El sub-muestreo es una fracción f (menor a 1) del conjunto completo de entrenamiento. A medida que el valor de f disminuye, se introduce aleatoriedad en el algoritmo, se previenen problemas de sobre-ajuste y el modelo se vuelve más rápido. Según resultados obtenidos por Friedman en [62], los valores óptimos de f se encuentran entre 0.5 y 0.8.

3.4.3.3. Boosted Logistic Regression

Como ya se ha definido anteriormente, Boosting busca combinar diferentes clasificadores débiles para crear uno que sea más robusto en la predicción. Un modelo altamente estudiado y utilizado es AdaBoost (la abreviación de "Adaptative Boosting"), que propone entrenar un conjunto de clasificadores débiles de manera iterativa, en la que se ponga mayor énfasis en aquellas observaciones que fueron mal clasificadas por los algoritmos predecesores. Si bien los resultados de AdaBoost son bastante buenos al clasificar nuevos datos, el algoritmo sufre de sobre-ajuste cuando se trabaja con bases de datos con "ruidos". LogitBoost aparece como una solución a este problema, ya que puede reducir los errores de entrenamiento de forma lineal y por tanto, producir una mejor generalización. Tal como se menciona en [63], LogitBoost utiliza considerablemente menos datos que otros modelos y por lo tanto es más rápido en su ejecución.

3.4.3.4. Bagged CART

Los árboles de clasificación y regresión (CART) suelen ser algoritmos que presentan una alta varianza, por lo que las técnicas de Bagging se utilizan para reducirla. Por lo general, estos árboles de decisión son sensibles a los datos con los cuales son entrenados, pero al momento de ingresar nueva información, el resultado puede variar y ser muy diferente de lo esperado. Más detalles del modelo CART pueden ser revisados en la sección 3.2.4.

Modelo	Caso	Agosto					Septiembre					Octubre				
		Accuracy	Kappa	ROC	Sens	Spec	Accuracy	Kappa	ROC	Sens	Spec	Accuracy	Kappa	ROC	Sens	Spec
c5,0	1	0,9178	0,4771	0,7992	0,3558	0,9943	0,9439	0,3494	0,7804	0,2341	0,9976	0,9730	0,1878	0,8191	0,1007	0,9994
	2	0,9176	0,4593	0,7979	0,3334	0,9960	0,9434	0,3354	0,7831	0,2127	0,9984	0,9728	0,1637	0,8026	0,0882	0,9995
	3	0,9223	0,5114	0,8169	0,3919	0,9943	0,9448	0,3855	0,7916	0,2664	0,9965	0,9732	0,1719	0,8154	0,1259	0,9985
	4	0,7720	0,5182	0,8161	0,6177	0,8851	0,8081	0,4837	0,7939	0,4818	0,9459	0,8891	0,4105	0,8297	0,3549	0,9782
	5	0,7793	0,5344	0,8256	0,6071	0,9082	0,8145	0,5018	0,7997	0,5004	0,9496	0,8915	0,4298	0,8306	0,3508	0,9819
gbm	1	0,9174	0,4660	0,7980	0,3440	0,9957	0,9437	0,3437	0,7864	0,2258	0,9981	0,9729	0,1866	0,8362	0,1082	0,9992
	2	0,9172	0,4516	0,7996	0,3260	0,9973	0,9431	0,3249	0,7870	0,2110	0,9985	0,9728	0,1785	0,8233	0,1019	0,9993
	3	0,9212	0,5023	0,8122	0,3843	0,9939	0,9445	0,3732	0,7947	0,2519	0,9974	0,9735	0,1968	0,8345	0,1169	0,9993
	4	0,7685	0,5079	0,8129	0,5687	0,9192	0,8056	0,4707	0,7934	0,4595	0,9526	0,8894	0,4205	0,8392	0,3490	0,9803
	5	0,7715	0,5148	0,8171	0,5758	0,9194	0,8083	0,4796	0,7964	0,4663	0,9554	0,8906	0,4193	0,8393	0,3415	0,9829
LogitBoost	1	0,9136	0,4149	0,7192	0,2889	0,9990	0,9417	0,2809	0,6653	0,1741	0,9999	0,9701	0,1583	0,7716	0,0925	0,9981
	2	0,9142	0,4129	0,7203	0,2875	0,9993	0,9414	0,2762	0,6654	0,1708	0,9999	0,9722	0,1156	0,7704	0,0611	0,9998
	3	0,9143	0,4138	0,7216	0,2976	0,9971	0,9410	0,2764	0,6647	0,1709	0,9999	0,9726	0,1282	0,7654	0,0799	0,9988
	4	0,7360	0,4353	0,7355	0,5197	0,8980	0,7642	0,4090	0,7150	0,5251	0,8665	0,8697	0,2208	0,7380	0,1662	0,9870
	5	0,7394	0,4456	0,7328	0,5411	0,8815	0,7697	0,4048	0,7141	0,5125	0,8737	0,8660	0,1727	0,7247	0,1398	0,9864
cart	1	0,9146	0,4773	0,7722	0,3801	0,9877	0,9419	0,3720	0,7448	0,2696	0,9928	0,9716	0,2025	0,7421	0,1242	0,9974
	2	0,9133	0,4711	0,7684	0,3799	0,9857	0,9402	0,3581	0,7395	0,2653	0,9914	0,9705	0,1948	0,7102	0,1303	0,9961
	3	0,9194	0,5113	0,7885	0,4140	0,9878	0,9433	0,4024	0,7596	0,2974	0,9926	0,9726	0,2295	0,7485	0,1486	0,9975
	4	0,7602	0,4946	0,7978	0,5937	0,8853	0,7999	0,4639	0,7694	0,4776	0,9377	0,8819	0,4066	0,7956	0,3609	0,9697
	5	0,7664	0,5086	0,8048	0,6027	0,8889	0,8062	0,4867	0,7771	0,4960	0,9373	0,8831	0,4133	0,7984	0,3695	0,9701

Tabla 3.9: Resultados obtenidos para “Ensemble Methods” sobre el conjunto de datos train. Si bien los resultados mostrados son en general buenos, la Sensibilidad de los algoritmos es muy baja, sobre todo en el mes de octubre. Junto a esto, la Exactitud para el mes de agosto, es baja respecto de lo que podría esperarse, al ser el mes con mayor cantidad de información disponible.

La tabla 3.9 muestra el resultado de entrenar los modelos de conjunto. Estos clasificadores fueron entrenados en 5 escenarios diferentes, los cuales son descritos en la tabla 3.3. Los modelos muestran un muy buen desempeño sobre los datos de testeo. Al observar la medida de Exactitud podemos notar que los valores para los 3 primeros casos se encuentran sobre 90 % en todos los modelos y meses de estudio. Sin embargo, esta medida decrece considerablemente al aplicar los modelos que consideran todas las variables (incluidas las de reciprocidad). Este mismo fenómeno es posible observarlo en la Especificidad de los modelos. De todos modos, es importante destacar el alto desempeño que presenta la Especificidad en todos los modelos estudiados. Para el resto de las medidas utilizadas, la aplicación de todas las variables disponibles mejora sus valores. no obstante, esta mejora no es suficiente a la hora de evaluar los modelos, ya que la Sensibilidad de éstos se encuentra bajo el 60 % en todos los casos. Este mal desempeño se ve con mayor claridad en el mes de octubre. Así mismo, los valores de ROC no se pueden catalogar como completamente satisfactorios.

Los datos de testeo, mostrados en la tabla 3.10 se comportan de manera similar a la analizada en el caso de la muestra de entrenamiento. Si bien la Exactitud, la Especificidad y la Precisión de los modelos disminuye al utilizar todas las variables disponibles, el valor de la Sensibilidad y Kappa aumenta considerablemente. En general, estos modelos presentan un muy buen desempeño a la hora de clasificar nuevas observaciones. Este fenómeno se aprecia con claridad en el mes de agosto, sin embargo, en el mes de octubre las medidas no generan buenos resultados.

Como ya se ha mencionado con anterioridad, al estar frente a un problema de desbalance de datos, las medidas de desempeño más importantes son la Sensibilidad y la Precisión. Si bien los valores de esta última medida son relativamente buenos, la Sensibilidad de los clasificadores es muy baja, lo que quiere decir que son incapaces de detectar clientes fugados, pero cuando los detectan lo hacen de muy buena manera.

A partir de las tablas analizadas en esta sección, podemos concluir que los mejores “Ensemble Methods” corresponden a C5.0 y GBM. Sin embargo, la sensibilidad de estos algoritmos es muy baja considerando que el objetivo del estudio es lograr determinar aquellos clientes que terminarán por fugarse de la compañía.

Modelo	Caso	Agosto					Septiembre					Octubre				
		Accuracy	Kappa	Sens	Spec	Precision	Accuracy	Kappa	Sens	Spec	Precision	Accuracy	Kappa	Sens	Spec	Precision
c5,0	1	0,9206	0,4722	0,3466	0,9968	0,9347	0,9424	0,3231	0,2092	0,9986	0,9208	0,9731	0,1769	0,1028	0,9992	0,8032
	2	0,9177	0,4580	0,3344	0,9967	0,9312	0,9430	0,3360	0,2199	0,9984	0,9130	0,9735	0,1856	0,1066	0,9997	0,9025
	3	0,9221	0,5156	0,4013	0,9928	0,8828	0,9462	0,3898	0,2671	0,9972	0,8788	0,9729	0,1944	0,1122	0,9997	0,9091
	4	0,7727	0,5209	0,6028	0,9012	0,8220	0,8072	0,4839	0,5009	0,9346	0,7614	0,8853	0,3800	0,3041	0,9824	0,7430
	5	0,7838	0,5433	0,6093	0,9154	0,8445	0,8176	0,4981	0,4844	0,9543	0,8131	0,8924	0,4283	0,3448	0,9838	0,7808
gbm	1	0,9199	0,4715	0,3496	0,9957	0,9143	0,9423	0,3249	0,2117	0,9983	0,9047	0,9730	0,1705	0,0988	0,9992	0,7967
	2	0,9179	0,4553	0,3295	0,9975	0,9468	0,9431	0,3321	0,2152	0,9989	0,9353	0,9734	0,1949	0,1136	0,9994	0,8502
	3	0,9208	0,4999	0,3823	0,9939	0,8945	0,9455	0,3726	0,2512	0,9977	0,8912	0,9730	0,2118	0,1249	0,9993	0,8533
	4	0,7720	0,5160	0,5733	0,9224	0,8482	0,8072	0,4687	0,4565	0,9532	0,8025	0,8869	0,4063	0,3365	0,9789	0,7274
	5	0,7755	0,5229	0,5766	0,9254	0,8536	0,8084	0,4708	0,4629	0,9502	0,7922	0,8930	0,4353	0,3533	0,9831	0,7772
LogitBoost	1	0,9163	0,4174	0,2902	0,9995	0,9862	0,9406	0,2720	0,1679	0,9999	0,9885	0,9660	0,2077	0,1678	0,9899	0,3330
	2	0,9148	0,4161	0,2901	0,9993	0,9813	0,9412	0,2830	0,1756	0,9999	0,9912	0,9728	0,1310	0,0721	1,0000	1,0000
	3	0,9146	0,4158	0,2893	0,9995	0,9868	0,9422	0,2827	0,1755	0,9998	0,9855	0,9718	0,1150	0,0629	1,0000	0,9773
	4	0,7248	0,4042	0,4417	0,9390	0,8457	0,7657	0,4076	0,5206	0,8677	0,6210	0,8694	0,1996	0,1416	0,9911	0,7263
	5	0,7518	0,4675	0,5120	0,9325	0,8512	0,7671	0,4034	0,5086	0,8732	0,6219	0,8654	0,1250	0,0823	0,9961	0,7783
CART	1	0,9178	0,4860	0,3871	0,9883	0,8147	0,9401	0,3547	0,2564	0,9925	0,7242	0,9719	0,2030	0,1295	0,9971	0,5762
	2	0,9146	0,4746	0,3805	0,9869	0,7970	0,9393	0,3569	0,2637	0,9911	0,6933	0,9712	0,2170	0,1441	0,9962	0,5314
	3	0,9197	0,5141	0,4159	0,9880	0,8249	0,9444	0,4055	0,2986	0,9930	0,7617	0,9718	0,2382	0,1537	0,9972	0,6313
	4	0,7610	0,4965	0,5912	0,8895	0,8019	0,8041	0,4686	0,4752	0,9410	0,7703	0,8806	0,3983	0,3534	0,9687	0,6535
	5	0,7676	0,5104	0,6017	0,8927	0,8087	0,8061	0,4801	0,5039	0,9301	0,7472	0,8871	0,4379	0,3882	0,9704	0,6867

Tabla 3.10: Resultados obtenidos de los Ensemble Methods para muestra de testeo. Como es posible apreciar, la Especificidad y la Precisión de los modelos es relativamente buena para los tres meses, sin embargo, la Sensibilidad disminuye considerablemente su valor. Junto a esto, el valor de la Exactitud es bajo en comparación a otros modelos que ya han sido estudiados.

3.4.4. Etapa 4: Selección de clasificadores y ajuste de parámetros

Después de revisar y comparar los resultados de diferentes clasificadores, nos centraremos en 4 modelos que parecen generar las mejores predicciones: RF, KNN, SVM y DL. El siguiente paso es determinar los parámetros óptimos para cada modelo predictivo, con el fin de mejorar los resultados obtenidos en el estudio previo.

En el caso del KNN, se cuenta con solo un posible parámetro para ajustar: el número de vecinos k. Después del re-muestreo realizado con validación cruzada, el proceso produce un perfil de medidas de rendimiento disponible para guiar al usuario en cuanto a los valores de los parámetros de ajuste que se deben elegir. Por defecto, la función elige automáticamente los parámetros de ajuste asociados con el mejor valor, es decir, `caret` al momento de entrenar el modelo elige los parámetros que optimizan el valor de la Exactitud Dicho esto, no aplicaremos técnicas de ajuste de parámetros para este clasificador, ya que los valores anteriormente entregados son el mejor desempeño posible de este método.

RF por otra parte, `caret` permite definir dos parámetros para ajustar el modelo: *mtry* que corresponde al número de variables tomadas al azar como candidatos en cada división y *ntree* que es el número de árboles para crecer. El valor predeterminado para *mtry* son diferentes para la clasificación ($\sqrt{p}$ donde p es el número de predictores) y la regresión ($p/3$). El valor predeterminado para *ntree* es igual a 500.

Con `caret` creamos una función que nos permite definir parámetros óptimos para el RF. Con esta función se prueban diferentes combinaciones de valores para *mtry* y *ntree*. Específicamente se estudian 30 combinaciones de valores, que se definen con la siguiente línea de código,

```
tunegrid <- expand.grid(.mtry=c(1:10), .ntree=c(1000, 1500, 2000))
```

Es decir, para *mtry* se prueban los valores desde 1 hasta 10, mientras que para *ntree* se utilizan los valores 1.000, 1.500 y 2.000.

Es importante destacar que, con el fin de mejorar el desempeño del modelo, el conjunto de datos que fue utilizado para entrenar el RF fue previamente "balanceado" de manera manual. Este proceso se realizó seleccionando el total de clientes fugados con los que se cuenta en un mes determinado (por ejemplo, los 30.164 clientes fugados de agosto) y complementándolo con una muestra aleatoria de 40.000 clientes no fugados de dicho mes. De esta manera, por ejemplo para el mes de agosto, el modelo se entrenó con 49.115 observaciones (que equivalen al 70 % del total de datos utilizados) y se evaluó utilizando 21.049 clientes (correspondientes al 30 % restante de la información).

El resultado de la búsqueda de los parámetros óptimos se muestra en la tabla 3.11.

Mes	*mtry*	*ntree*
Agosto	7	2.000
Septiembre	10	1.500
Octubre	6	2.000

Tabla 3.11: La tabla muestra el valor de los parámetros ajustados para los meses de agosto, septiembre y octubre. Los modelos fueron entrenados con el 70 % de los datos balanceados de manera manual (este conjunto de datos se creó considerando todos los fugados de cada mes y agregando una muestra aleatoria de 40.000 no fugados).

Para establecer un marco de comparación es necesario tener los resultados de aplicar el RF con y sin ajuste de parámetros. En la tabla 3.12 se muestran los resultados de aplicar el RF sobre los datos de testeo de cada mes, considerando un modelo entrenado de manera básica y otro modelo con los parámetros ajustados de manera óptima. Si bien el resultado de aplicar el modelo con parámetros óptimos no es tajante en cuanto a la mejora del desempeño, si muestra un incremento en el valor de Kappa, de la Sensibilidad y sobre todo en la Precisión de los resultados. Considerando que estas dos últimas medidas son importantes a la hora de detectar fugados, podemos concluir que un ajuste en los parámetros del RF (con los valores que

se muestran en la tabla 3.11) nos permiten determinar de mejor manera aquellos clientes que terminan por fugarse de la compañía.

Medida	Agosto		Septiembre		Octubre	
	Sin Ajuste	Con Ajuste	Sin Ajuste	Con Ajuste	Sin Ajuste	Con Ajuste
Accuracy	0,8891	0,7779	0,9047	0,8084	0,9154	0,8895
Kappa	0,4694	0,5283	0,3583	0,4869	0,2175	0,4225
Sensibilidad	0,5252	0,5864	0,4718	0,4823	0,4785	0,3507
Especificidad	0,9388	0,9214	0,9372	0,9492	0,9288	0,9793
Precisión	0,5395	0,8484	0,3611	0,8038	0,1709	0,7389
Acc Balanceada	0,7320	0,7539	0,7045	0,7158	0,7037	0,6650

Tabla 3.12: La tabla muestra los resultados de aplicar los modelos RF entrenados con y sin ajuste de parámetros. Los modelos sin ajuste de parámetros fueron entrenados con el 70 % de los datos de cada mes sobre los cuales se les aplicó SMOTE. La tabla resume los resultados obtenidos al aplicar los modelos sobre el conjunto de datos test de cada mes. Como es posible ver, el valor de las métricas en general mejora cuando se testean los datos sobre el modelo ajustado, sobre todo en el caso de la Precisión, que aumenta notablemente su valor.

Es importante destacar que se cuenta con menor información para entrenar a los modelos con ajuste de parámetros en comparación con los modelos sin ajuste de parámetros. Esto debido a que se han utilizado conjuntos de datos balanceados de manera manual que representan menos del 70 % de la totalidad de los datos. Más específicamente y como se mencionó anteriormente, para el mes de agosto se cuenta con 70.164 observaciones que corresponde al 27,8 % de la información total del mes. En el caso de septiembre se utilizan 56.917 datos (equivalente al 23,8 % de los datos totales), mientras que en octubre el conjunto de datos ocupado es de 46.690 (que es el 20,6 % de los datos del mes). De estos conjuntos se extraen las muestras aleatorias para entrenar y testear los modelos (que representan el 70 % y 30 %, tal como se ha hecho en los análisis anteriores). Es decir, los clasificadores están siendo entrenados con 49.114, 39.842 y 32.683 observaciones, que equivalen al 19,5 %, 16,7 % y 14,4 % de la información de cada mes respectivamente. A partir de lo anterior podemos indicar que lo resultados obtenidos muestran un desempeño bastante bueno y robusto considerando que, en comparación a los modelos originales, se cuenta con muy poca información para entrenarlos.

En el caso del SVM, como fue mencionado en la sección 3.2.6, es posible ajustar 3 parámetros: tipo de $Kernel$, valor de la penalización C (cuyo valor por defecto es 1) y γ que es el parámetro necesario para todos los Kernels excepto para el lineal (por defecto se define como 1/(dimensión de datos)). Como también se mencionó con anterioridad, el tipo de datos con el que se trabaja en este estudio no permite la utilización de un Kernel lineal, ya que los datos no son linealmente separables. Por ende, solo vamos probar entre un Kernel polinomial y uno radial.

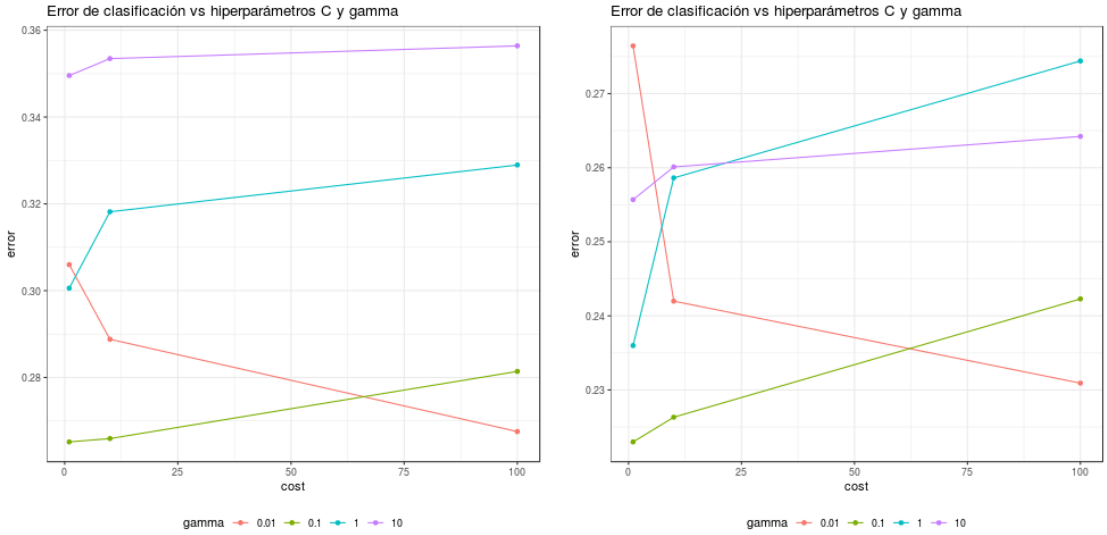
Un Kernel polinomial de orden n entrega todas las funciones cuyas derivadas de orden $(n+1)$ son constantes y por lo tanto todas las derivadas de orden $(n+2)$ o superiores son cero. Por otro lado, un Kernel radial da acceso a todas las funciones (es decir, todas las funciones infinitamente diferenciables). Por lo tanto, en cierto sentido, se puede decir que un Kernel radial (o gaussiano) es tan poderoso como un Kernel polinomial de orden infinito. Por lo general, los Kernesl radiales son utilizados en métodos no paramétricos, en los que la utilización de mayor cantidad de datos genera mejores resultados. Un Kernel radial permite mayor flexibilidad y en general entrega mejores resultados en comparación a los otros dos tipos de Kernels, tal como se expone en [64].

Teniendo en cuenta lo anterior, en este estudio solo se trabajará con el Kernel radial, ajustando los otros dos parámetros disponibles: Costo y γ. El paquete `e1071` contiene la función **tune** que permite ajustar estos parámetros. En esta ocasión se probarán los valores 1, 10, y 100 para el costo, mientras que para γ se considerarán los valores de 0.01, 0.1, 1 y 10. Las líneas de código utilizadas son las que se muestran a continuación:

```
svm_tune <- tune("svm", StatusFugado~ ., data = train, kernel = "radial",
```

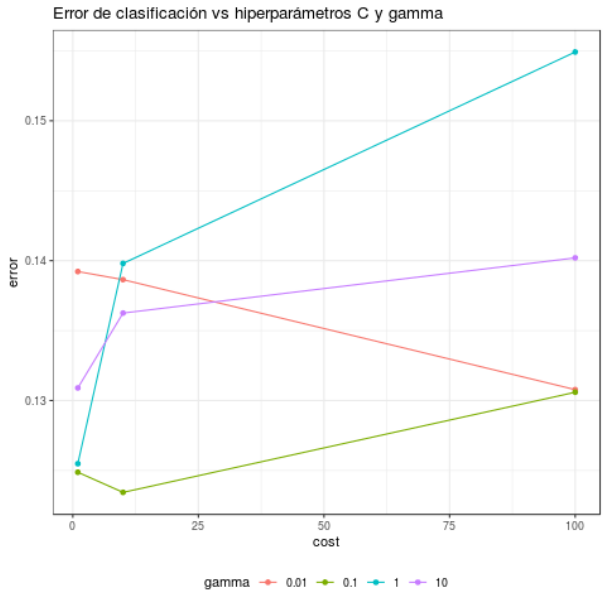
```
scale = T, ranges = list(cost=c(1, 10, 100),
gamma = c(0.01, 0.1, 1, 10)))
```

Es importante indicar que el conjunto de datos train utilizado para entrenar el modelo corresponde a datos balanceados de manera manual, tal como se hizo para el caso del RF. A partir de los resultados obtenidos del código anterior es posible conseguir los gráficos que se muestran en la figura 3.6. Los gráficos muestran las curvas de error versus el costo asociado, utilizando los diferentes valores de γ. Vemos que el valor de γ igual a 0.1 es el que entrega los menores errores de clasificación. La tabla 3.13 resume los parámetros óptimos asociados a cada mes.



(a) Tune Agosto

(b) Tune Septiembre



(c) Tune Octubre

Figura 3.6: Ajuste de parámetros del SVM para cada mes. En las imágenes es posible apreciar el comportamiento de los modelos considerando diferentes valores para Costo y γ. Los modelos fueron entrenados con el 70 % de los datos balanceados de manera manual.

Para la aplicación y comparación de los modelos se ha utilizado el mismo procedimiento que en el caso del RF. Esto quiere decir que los modelos sin ajuste de parámetros se entrenaron con el 70 % de los datos balanceados con SMOTE y fueron testeados con el 30 % de los datos restantes.

Mes	Costo	Gamma
Agosto	1	0,1
Septiembre	1	0,1
Octubre	10	0,1

Tabla 3.13: Resumen de parámetros óptimos para la aplicación del clasificador SVM sobre los datos mensuales.

Mientras que los modelos con ajuste fueron entrenados con el 70 % de los datos balanceados de manera manual, lo que al igual que en el caso anterior, equivale a menos del 70 % de los datos totales contenidos en la base de datos de cada mes.

La tabla 3.14 resume los resultados obtenidos de aplicar los modelos SVM con y sin ajuste de parámetros. Los modelos ajustados presentan una mejora considerable en las métricas de evaluación. El valor de Kappa mejora en casi 20 puntos en cada mes, mientras que la Precisión mejora en hasta un casi 60 % (por ejemplo, para el mes de octubre). Si bien el valor de la Exactitud se ve comprometido en agosto y septiembre, es posible notar la mejora en la Sensibilidad para estos meses.

Métricas	Agosto		Septiembre		Octubre	
	Sin Ajuste	Con Ajuste	Sin Ajuste	Con Ajuste	Sin Ajuste	Con Ajuste
Accuracy	0,8272	0,7374	0,8470	0,7733	0,8413	0,8722
Kappa	0,2624	0,4469	0,1777	0,3832	0,1275	0,3157
Sensibilidad	0,4141	0,5715	0,3689	0,4000	0,5545	0,2493
Especificidad	0,8822	0,8618	0,8833	0,9344	0,8501	0,9819
Precisión	0,3190	0,7561	0,1935	0,7247	0,1011	0,6969
Acc Balanceada	0,6482	0,7166	0,6261	0,6672	0,7023	0,6156

Tabla 3.14: Comparación del clasificador SVM con y sin ajuste de parámetros.

Finalmente, DL es uno de los clasificadores que cuenta con la mayor cantidad de parámetros para optimizar. Dentro de los más utilizados se encuentran: *epochs* (que corresponde al número de ciclos de entrenamiento que realiza el algoritmo), *stoppingmetric* (cómo decidir si el modelo está mejorando o no), *stoppingtolerance* (valor asociado al mejoramiento de la métrica elegida anteriormente), *hidden* (número de neuronas en cada capa oculta), entre otros 25 parámetros que es posible ajustar utilizando `h2o`.

Los algoritmos DL están entrenados para aprender progresivamente usando datos. Se necesitan grandes conjuntos de datos para asegurarse de que la máquina proporciona los resultados deseados. La red neuronal artificial de un DL requiere una gran cantidad de datos. Cuanto más poderosa sea la abstracción que desea, más parámetros deben ajustarse y más parámetros requieren mayor cantidad de datos.

Se realizaron algunas pruebas con el objetivo de mejorar el desempeño en la predicción ajustando diversos parámetros, sin embargo, no fue posible determinar una combinación óptima de valores que permitieran mejorar los resultados. Como se menciona en la sección 3.2.7, DL es una especie de "caja negra", en la que se ingresan los datos y se obtiene un resultado, pero se desconoce el procedimiento del algoritmo. Debido a esto, junto a la dificultad para determinar qué parámetros es necesario ajustar y el tiempo acotado para desarrollar el estudio, es que se decidió utilizar los valores predeterminados para hacer las predicciones.

Luego de revisar detalladamente diferentes modelos de clasificación y predicción, es importante destacar algunos aspectos. Gran parte de los cálculos fueron realizados con el computador de alto rendimiento HPC-UDD-Sofia. Su uso se debe principalmente al alto volumen de datos

con el que se trabajó. Lo estudiado y obtenido aquí nos permite generar una serie de conclusiones respecto al uso de clasificadores en el CCP.

La metodología utilizada para revisar los distintos clasificadores consistió en utilizar el total de observaciones de cada mes y dividirla en dos sets: train y test. Los clasificadores fueron entrenados con el conjunto train (correspondiente al 70 % de los datos) mientras que el conjunto de datos de test se utilizó para medir el desempeño de los modelos. Cada modelo fue evaluado de acuerdo a una serie de medidas de desempeño, con las cuales se efectuaron las comparaciones entre algoritmos. Estos clasificadores fueron evaluados en diferentes escenarios, que variaban de acuerdo a la cantidad y tipo de variables que contenían.

El estudio de ocho clasificadores diferentes (GLM, LDA, GLMNET, KNN, CART, RF, SVM y DL) nos indica que uno de los problemas más importantes en la investigación es el desbalance de clases. Para contrarrestar los efectos de este fenómeno y generar mayor cantidad de observaciones de la clase positiva, se utilizó la técnica SMOTE. Si bien SMOTE puede ser utilizado en cualquier problema con desbalance de clases, existe muy poca literatura respecto de su combinación con clasificadores para el problema de fuga de clientes. Más aun, no se han encontrado registros de la utilización conjunta de SMOTE y Deep Learning.

En términos generales, de acuerdo a los cálculos para este CCP, SMOTE permite mejorar los resultados de sensibilidad y precisión, en desmedro de la exactitud y la especificidad de los modelos. Si se vuelve a revisar la tabla 3.5, observamos que para los modelos "más simples", como lo son LDA, GLM y GLMNET, aplicar SMOTE permite mejorar las medidas de desempeño del mes de octubre por sobre los otros dos meses (recordar que octubre es el mes más desbalanceado).

La comparación de diferentes modelos predictivos fue realizada utilizando al menos seis medidas de desempeño diferentes. Si bien los motivos han sido mencionados a lo largo del estudio, es importante recalcar que, al encontrarnos ante un problema desbalanceado, hay clasificadores que podrán etiquetar muy bien a la clase negativa, pero no así a la clase positiva. De este modo, el objetivo era contar con la mayor cantidad de información posible para decidir respecto de qué modelos eran los más efectivos en nuestro problema en particular.

Esta parte del estudio nos permite concluir que aquellos clasificadores más complejos (SVM y DL), permiten encontrar más eficientemente las relaciones entre las variables. Sus desempeños son buenos en muestras de datos acotados a un mes de estudio. Por ende, compañías con características similares a las de Virgin Mobile, y que cuentan con mayor cantidad de información, podrían basar sus estudios de CCP en estos clasificadores. Este aumento en la cantidad de información puede corresponder a horizontes de tiempos más largos (por ejemplo, semestres o años) o bien, a la división de los meses en semanas. Una mayor cantidad de datos permite una mejora en la detección de patrones de comportamiento de los clientes.

Si bien encontramos clasificadores que permiten clasificar de manera eficiente aquellos clientes que terminarán por fugarse, existe otro algoritmo que permite modelar el tiempo de sobrevivencia de los individuos a partir de sus características individuales. Estos algoritmos se conocen como análisis de supervivencia y Modelo de Regresión de Cox, los cuales se revisarán en mayor detalle en el capítulo siguiente.

3.5. Variables relevantes

Para esta investigación a partir de los datos de perfil y CDRs se construyó un conjunto de variables predictoras con el objeto de resolver el CCP. Con este propósito se calcularon diferentes métricas de redes y se generaron algoritmos para determinar los atributos del consumidor relacionados con su "posición" en la red y su actividad en ella. La hipótesis de este trabajo es precisamente la declaración de que con este conjunto de variables es posible dar una solución adecuada al CCP. En este sentido, es necesario validar un grupo de estas variables indicando su relevancia en la resolución del CCP. En las secciones anteriores se probó con un conjunto de clasificadores que es posible encontrar patrones en los valores de estas variables que permiten

predecir con cierta confianza si un cliente se fugará o no de la compañía en un futuro próximo. Sin embargo, aún no queda claro qué variables en particular son realmente importantes y deben ser consideradas en un modelo más eficiente. Un modelo que incorpora un número grande de variables puede tener un mejor ajuste a los datos, pero podría a la vez aumentar la varianza del modelo y propiciar el sobre-ajuste. Por otro lado, un modelo con insuficiente número de variables relevantes, puede tener menos varianza pero el ajuste y la descripción de los datos también puede empeorar.

Existen algunos procedimientos útiles en ML para la validación y selección de variables importantes en modelos predictivos. Algunos de ellos, que serán revisados en esta sección, son: Eliminación Recursiva de Características, Splines de Regresión Adaptativa Multivariante, RF y Boruta. Estas herramientas permiten reducir el número de variables del modelo dejando aquellas que son más importantes. Algunas de las razones por las cuales esto debe hacerse son:

- Eliminar una variable redundante ayuda a mejorar la Precisión del modelo. Del mismo modo, la inclusión de una variable relevante mejora la Precisión de este.
- Tener demasiadas variables pueden resultar en un sobre-ajuste, lo que a su vez implica que el modelo sea inservible al ser aplicado a datos nuevos.
- Demasiadas variables en el modelo implica el incremento de recursos computacionales
- Demasiadas variables dificultan la interpretación del modelo .

El paquete `caret` en R permite obtener información valiosa para decidir qué variables son importantes dentro del modelo. Una primera etapa antes de comenzar a buscar variables importantes es crear una base de datos que sea posible de manejar en este tipo de modelos. El pre-procesamiento de los datos permite obtener una base de datos limpia y útil para probar los modelos de clasificación. Como primera etapa, se simulan los datos de perfil que se encuentran vacíos o con NA, de modo tal que no se alteren las distribuciones de probabilidad de los datos presentes. Para esto, primero se determinan las probabilidad a partir de los datos de la muestra y luego se generan los datos simulados. En el caso de las variables de redes, se reemplazan los datos que contengan NA con un 0 con la siguiente función:

```
f=function(t)ifelse(is.na(t),0,t)
```

Las variables utilizadas para hacer esta parte del estudio son las mencionadas en la tabla 2.2 y junto a estas se suman algunas características del perfil de consumidor: “canal”, “compañía de origen”, “compañía de destino” y “suscrito selfcare”. Debido a que las tres primeras variables son categóricas (ver tabla 2.1 para revisar las categorías de cada variable), es necesario convertirlas a variables numéricas para poder utilizarlas en los algoritmos de aprendizaje automático. A este tipo de variables no es posible asignarles un número aleatorio para que sean reconocidas en los modelos de regresión, debido a que no existe una relación numérica entre ellas y por ende, no es posible realizar operaciones lógicas y/o matemáticas. Las variables“dummy” o variables ficticias se crean en esta situación para poder aplicar algoritmos de regresión.

La función `dummyVars()` permite convertir la variable categórica en N variables binarias como categorías (con N igual al número de niveles que posea la variable categórica). Por ejemplo, como la variable canal tiene 7 niveles (ver tabla 2.1 para revisar las categorías en la variable Canal), tendríamos que reemplazarla con 7 nuevas variables ficticias. Por ende, si un cliente se subscribió a la compañía mediante Retail, solo la nueva variable ficticia “Canal.Retail” toma el valor de 1, mientras que las demás variables asociadas al canal toman valor 0.

3.5.1. Correlación de variables

La función `findCorrelation()` busca en una matriz de correlación y devuelve aquellas variables para eliminar y reducir las correlaciones de pares, dicho de otra manera, intentar eliminar

el problema de multicolinealidad. Este fenómeno causa que los estimados de los coeficientes en los modelos lineales sean muy inestables y deja sin sentido la determinación de las variables importantes. Aplicando `findCorrelation()` obtenemos que los atributos que están altamente correlacionados para el mes de octubre son "CallInOnNet", "CallOutOnNet", "CallTimeInOnNet", "TotalCalls", "CallOut" y "CallFromChurned". Estas variables podrían generar problemas de multicolinealidad. Por lo tanto, es necesario mantener a vista lo que ocurre con estas variables a lo largo del estudio.

3.5.2. Variables importantes analizadas con el paquete `caret`

La función `varImp()` calcula la importancia de las variables para objetos producidos por métodos `train()` y otros métodos específicos. El código utilizado para determinar las variables importantes es el siguiente:

```
control = trainControl(method="repeatedcv", number=10, repeats=3)
model = train(StatusFugado∼., data=x, method="glmnet",
              preProcess="scale", trControl=control)
importance = varImp(model, scale=FALSE)
```

Se construye un modelo de "glmnet". x corresponde al conjunto de 5.000 datos seleccionados de manera aleatoria del mes de octubre sobre el cual se entrenará el modelo. Una vez entrenado el modelo, podemos obtener el listado de las variables más importantes aplicando la función `varImp()`. La importancia de las variables se puede estimar a partir de los datos mediante la construcción de un modelo. Para algunos algoritmos, esto se puede estimar usando un análisis de curva ROC, realizado para cada atributo. Para la mayoría de los modelos de clasificación, cada predictor tendrá una importancia variable diferenciada para cada clase. Además, las medidas de importancia se escalan para tener un valor máximo de 100, a menos que el argumento de escala de `varImp()` esté establecido en FALSO. En este caso, "scale" se ha establecido en falso y se obtienen la importancia de cada variable usando la curva ROC.

Los resultados para el mes de octubre se muestran en la tabla 3.15 (para cada uno de los meses es posible conseguir la misma tabla). De acuerdo a lo obtenido, las 5 variables más importantes dentro del modelo para el mes de octubre son "Call in" , "Total Calls", "Tenure", "Número de Vecinos" y "Call to Churned". Para los meses de agosto y septiembre los resultados son un poco diferentes. Mientras que para el mes de agosto las 5 primeras variables más importantes son "tenure", "suscrito selfcare", "Call To Churned", "Número de Vecinos" y "Call Out On Net", para el mes de septiembre las variables más importantes son "Número Vecinos", "tenure", "Call In On Net", "Call To Churned" y "Total Calls". En los tres meses se cuenta con 3 variables importantes en común: *Número de Vecinos*, *Antigüedad* (Tenure), y *Llamadas hacia fugados*, por lo que sobre estas variables hay que poner especial atención.

3.5.3. Variables importantes con otros paquetes

Fuera de las funciones entregadas por el paquete `caret`, existen otras formas de definir variables importantes dentro de un modelo. Una forma es hacerlo con Random Forest [65, 66].

El paquete `party` cuenta con la función `cforest()` que permite construir un RF con la siguiente línea de código:

```
cf=cforest(StatusFugado∼., data= x,
                control = cforest_unbiased(mtry=2, ntree=50))
```

Los parámetros del RF son definidos con "control". En este caso, "ntree" indica la cantidad de árboles (se ha establecido en 50), "mtry" representa el número de variables de entrada que son muestreadas aleatoriamente como candidatas en cada nodo (se ha definido igual a 2).

La tabla 3.16 muestra las 12 variables más importantes obtenidas luego de aplicar la función `varimp` sobre el modelo "cf". Esta función permite calcular la importancia de la variable para un "cforest", siguiendo el principio "disminución media en la precisión" al permutar las variables.

Variable	Overall
Call in	1,009
Total Calls	0,200
Tenure	0,167
Num Vecinos	0,118
Call To Churned	0,073
Portado Portado	0,000
Ciaorigen_Movistar	0,000
Canal_WEBVirgin	0,000
Call Time In On Net	0,000
Call Out On Net	0,000

Tabla 3.15: Cuadro resumen de las 10 variables más importantes dentro del modelo para el mes de octubre. Estas variables fueron obtenidas al entrenar el clasificador GLM sobre una muestra aleatoria de 50.000 observaciones del mes de octubre.

Todos los resultados del RF están sujetos a variación aleatoria, por lo que antes de interpretar la importancia de las variables, es importante verificar si se logra la misma clasificación con una semilla aleatoria diferente o cambiando la cantidad de árboles. En este caso, se probó el `cforest()` con "ntree" igual a 50, 80, 100 y 120 y en todos los casos el orden de las variables importantes era el mismo.

Como es posible observar en la tabla 3.16, las variables más importantes definidas con este método son "Call in", "Tenure", "Número de Vecinos", "Total Calls" y "Call out". En esta ocasión vuelven a aparecer la antigüedad y el número de vecinos como variables importantes. Sin embargo, con esta función no se consideran como importantes las variables asociadas a la fuga (en contraste a lo que se había obtenido con el método anterior). Para los meses de agosto y septiembre los resultados son relativamente similares. En el caso de agosto las 5 variables más importantes son "Tenure", "Número de Vecinos", "suscrito selfcare", "Call out" y "Total Calls", mientras que para el caso de septiembre la variable "suscrito selfcare" cambia por "Call out on Net" y la variable "Total Calls" se reemplaza con "Call in on Net". Por lo tanto, vuelven a destacar la antigüedad y el número de vecinos como variables importantes, y aparece una nueva variable "Call Out".

Variables	varimp	varimpAUC
Call in	0,00199	0,07029
Tenure	0,00344	0,04305
Num Vecinos	0,00771	0,03985
Total Calls	0,00359	0,03180
Call out	0,00348	0,02896
Call In On Net	0,00110	0,02397
Portado	0,00054	0,01582
Call Out On Net	0,00094	0,01577
Num Vecinos Virgin	0,00404	0,01100
Total Call Time	0,00035	0,01014
Call Time out	0,00086	0,00983
Suscrito selfcare	0,00108	0,00637

Tabla 3.16: Variables importantes utilizando cforest para el mes de octubre. Estos resultados fueron obtenidos aplicando la función cforest del paquete `party`. Se entregan los valores de importancia basado en disminución media de la precisión y la importancia de las variables calculadas en base al área bajo la curva AUC.

Otra forma de seleccionar variables importantes es a través de Splines de Regresión Adaptativa Multivariante (MARS). Los métodos splines suelen emplearse para la estimación de la

función de regresión. Lo que hace MARS es básicamente construir modelos flexibles ajustando regresiones lineales por partes; es decir, la no linealidad de un modelo se aproxima mediante el uso de pendientes de regresión separadas en intervalos distintos del espacio variable independiente. Esta técnica permite modelar relaciones complejas entre predictores y la variable de respuesta, ya que combina la robustez de los arboles de regresión y las splines [67–69].

El paquete `earth` construye modelos de regresión usando las técnicas explicadas en los artículos de Friedman [69], denominadas "Multivariate Adaptive Regression Splines" y "Fast MARS". El modelo MARS que se utilizará queda definido por la siguiente línea de código:

```
marsModel <- earth(StatusFugado  ., data=x, nfold=10, degree=2, keepxy=T)
```

La función `earth` tiene una funcionalidad de validación cruzada incorporada que se puede trazar y usar para elegir un número óptimo de términos para un modelo MARS. También es posible agregar términos de orden superior, es decir, posibles interacciones entre las variables. Para esto, es necesario cambiar dos términos de la función (descritos de forma respectiva): "nfold" que será igual a 10 y "degree" que se iguala a 2. Con la función se obtiene que las 5 variables más importantes son "Tenure", "Call Time out", "Número de Vecinos", "Call Out On Net" y "Número de Vecinos Virgin". En esta ocasión se vuelven a repetir las variables antigüedad y número de vecinos como las variables más importantes.

La función `evimp()` usa tres criterios para estimar la importancia de la variable en un modelo MARS. El criterio "nsubsets" cuenta el número de subconjuntos de modelos que incluyen la variable. El criterio "rss" (suma de cuadrados residual) primero calcula la disminución en el RSS para cada subconjunto relativo al subconjunto anterior. Luego, para cada variable, suma estas disminuciones en todos los subconjuntos que incluyen la variable. El "gcv" corresponde a la validación cruzada generalizada. Los resultados se muestran en la tabla 3.17.

Variable	nsubsets	gcv	rss
Tenure	32	100	100
Call Time out	32	100	100
Num Vecinos	31	84,5	84,7
Call Out On Net	31	84,5	84,7
Num Vecinos Virgin	26	53,9	54,7
Portado Portado	23	44,4	45,3
Suscrito selfcare	22	40,9	41,9
Call in	22	40,9	41,9
Call out	19	32,9	34,1
Total Calls	18	32,5	33,6
Call Time In On Net	16	29	30,1
Call Time To Churned	15	26,6	27,8
Total Call Time	13	22,1	23,3
Call To Churned	11	18,4	19,7

Tabla 3.17: Selección de variables con MARS

Una tercera opción para definir variables importantes, es mediante el método **Boruta**. Boruta se basa en la misma idea que forma la base de los RF, es decir, agregar aleatoriedad al sistema y recolectar resultados del conjunto de muestras aleatorias. El algoritmo se puede describir de la siguiente manera: (1) generar una mezcla de valores de los predictores y unirlos con los predictores originales; (2) crear un RF en el conjunto de datos fusionado; (3) hacer una comparación de las variables originales con las variables aleatorias para medir la importancia de la variable; finalmente, (4) solo se consideran importantes las variables que tienen mayor importancia que las variables aleatorias. Para más detalles respecto del funcionamiento de este algoritmo, revisar [70].

La principal diferencia entre un RF y el método Boruta, es que RF entrega un puntaje Z

(que se calcula dividiendo la pérdida promedio de exactitud por su desviación estándar) que no está directamente relacionado con la significancia estadística de la importancia de la variable mientras que Boruta ejecuta un RF tanto en atributos originales como en aleatorios y calcula la importancia de todas las variables, con el fin de obtener resultados estadísticamente sólidos.

El paquete `Boruta` permite obtener el listado de las variables más importantes. Al igual que los métodos anteriores, Boruta es capaz de trabajar con variables categóricas, por lo que no es necesario crear variables ficticias. El resultado de aplicar este modelo sobre una muestra de 50.000 observaciones se muestra en la tabla 3.18. De un total de 22 variables, 21 atributos se confirman como importantes y 1 se confirma como No importante. La variable más importante dentro del estudio es "Tenure", seguido de "Número de Vecinos", "Número de Vecinos Virgin", "Call in" y "Call Time out".

Variable	meanImp	medianImp	decision
Tenure	78,38	78,20	Confirmed
Num Vecinos	53,73	53,55	Confirmed
Num Vecinos Virgin	35,51	35,47	Confirmed
Call in	33,30	33,72	Confirmed
Call Time out	29,42	28,89	Confirmed
Call Time in	28,30	28,02	Confirmed
Total Calls	26,47	26,59	Confirmed
Suscrito selfcare	26,10	25,96	Confirmed
Call out	25,20	25,09	Confirmed
Total Call Time	23,11	23,10	Confirmed
Portado	21,54	21,51	Confirmed
Call Time To Churned	19,44	19,71	Confirmed
Call To Churned	18,96	18,90	Confirmed
Call Time Out On Net	16,67	16,85	Confirmed
Call Time In On Net	15,04	15,11	Confirmed
Call Time From Churned	14,27	13,90	Confirmed
Call From Churned	13,23	12,83	Confirmed
Call In On Net	12,30	12,21	Confirmed
Canal	12,23	12,45	Confirmed
Call Out On Net	12,00	11,60	Confirmed
Num Vecinos Fugados	8,19	8,08	Confirmed
Cia origen	0,49	0,35	Rejected

Tabla 3.18: Variables seleccionadas con **Boruta**. La variable "Número de Vecino" aparece como la más importante dentro del estudio, seguida de el "Número total de llamadas". En este caso, la variable categórica "Compañía de origen" es rechazada de acuerdo a Boruta, debido a su baja importancia.

Después de analizar, mediante diferentes métodos, la importancia del conjunto de variables usadas en este estudio (detalladas en las tablas 2.1 y 2.2) en la resolución del CCP, algunos comentarios son necesarios. Al tomar el CCP como un problema de clasificación y probar con distintos clasificadores hemos encontrado que el conjunto de variables usado permite en general detectar la presencia de patrones en los datos que anticipan la fuga. Algunas de las variables más significativas empleadas por varios de estos clasificadores están relacionadas con la antigüedad del cliente en la compañía y con su actividad en la red. En efecto, el número de vecinos, o más exactamente el número de personas con las que el cliente tiene contacto telefónico es una medida de la actividad de éste en la red. También lo son, el número total de llamadas y el número de llamadas al interior de la red Virgin. En el caso de la antigüedad, aquellos clientes que llevan más tiempo en la compañía muestran patrones de mayor estabilidad respecto a la posibilidad de fuga, tal vez porque están satisfechos con la compañía o simplemente sus conductas son más conservadoras.

Estos resultados, sin embargo, hay que tomarlos con cautela en el sentido de que la interrelación entre las variables es compleja y las interpretaciones no siempre son claras y directas. Por ejemplo, el considerar que el número de llamadas al interior de la red Virgin es una variable con cierta importancia, no nos permite decir que aquellos clientes que solo tienen contactos con otros clientes dentro de la red Virgin, no se fugan. La relación puede ser más sutil, e indicar por ejemplo que los clientes Virgin tienen su núcleo familiar también en la misma compañía ya que por lo general es uno de los miembros de la familia quien adquiere el servicio para el resto. En este caso, la fuga de un familiar de la compañía implicaría que el resto lo seguiría y por tanto hay cierta barrera al cambio, dado que éste es mayor que en el caso de un cliente único. De acuerdo con esto, podemos declarar con cierta seguridad que los patrones de antigüedad y actividad (consumo del servicio) del cliente son importantes en la predicción de fuga. Más aún, podemos construir *funciones de decisión* basados en varios clasificadores de modo tal que sea factible estimar la probabilidad de esta fuga. Pero, no podemos indicar las razones individuales por las que un cliente en particular se fuga de la compañía. La factibilidad de la predicción radica en lo masivo de los datos no en el estudio de casos individuales.

Para profundizar más en las causas de la fuga, en el próximo capítulo aplicaremos modelos de supervivencia con el fin de estimar la probabilidad de fuga, condicionada a que el cliente a "sobrevivido" o tiene cierta antigüedad en la compañía. La idea, es explotar los resultados obtenidos a partir del uso de clasificadores, es decir, el hecho de que los cambios en los patrones de actividad en conjunto con ciertas atributos individuales del cliente permiten anticipar la fuga y estimar la probabilidad de ésta.

Capítulo 4

Análisis de Supervivencia

4.1. Análisis de Supervivencia

El *Análisis de Supervivencia* (también se le conoce como "Hazard Analysis") es el estudio del tiempo de "supervivencia" o el tiempo para que ocurra un determinado suceso y de los factores que influyen en él [71]. Una característica clave en los datos de supervivencia es que la variable respuesta es una variable aleatoria discreta o continua no negativa, que representa el tiempo desde un origen bien definido hasta un evento bien definido. Frecuentemente, debido al tipo de aplicaciones que ha tenido el análisis de supervivencia, al tiempo se le denomina "tiempo de falla" y al evento final simplemente "falla". Este tipo de análisis permite generalizar las predicciones con respuesta del tipo binaria (sí/no; fallecido/vivo) al incluir el tiempo de seguimiento, es decir, el tiempo que ha transcurrido hasta producirse la respuesta o bien, hasta el final del estudio, si es que la respuesta no se ha producido.

Cuando el tiempo de seguimiento termina antes de producirse el fenómeno estudiado o antes de completar el período de observación se habla de un caso *censurado por la derecha.* Formalmente, si T es una variable aleatoria que representa el tiempo de falla y U es una variable aleatoria que representa el tiempo para un evento censurado, lo que observamos es $T = \text{mín}(T; U)$ y un indicador de censura $\delta = I[T < U]$. Es decir, δ es 0 o 1 según si T es un tiempo censurado o un tiempo de falla observado, respectivamente. Con menos frecuencia, ocurre la *censura por la izquierda*, donde se sabe que los eventos ocurrieron antes de un cierto tiempo, o la *censura de intervalos*, donde se sabe que el tiempo de falla solo ocurrió dentro de un intervalo de tiempo específico.

Para ordenar los conceptos, tenemos las siguientes definiciones [72]:

- Tiempo de seguimiento. Corresponde al tiempo que transcurre entre la entrada del individuo al estudio hasta la fecha en la que se registra la última observación.
- Tiempo de supervivencia o falla. Corresponde al tiempo de aquellos individuos que si presentaron la respuesta esperada del estudio. En el caso de los estudios de cáncer por ejemplo, corresponde al tiempo de vida de un paciente hasta su fallecimiento durante del tiempo de seguimiento.
- Tiempos censurados. Corresponde al tiempo asociado a aquellos individuos que durante el seguimiento no han presentado la respuesta esperada. Estas observaciones, censuradas por la derecha, igual aportan información para estimar las probabilidades de supervivencia.
- Individuos "perdidos", son aquellos sujetos que por diferentes razones abandonan el estudio antes de la fecha de término y no presentan el fenómeno analizado.

Los métodos de análisis de supervivencia dependen de la distribución de supervivencia. Hay dos formas fundamentales de definirla: mediante la *función de supervivencia* $S(t)$ y mediante la

función de riesgo ("Hazard") $h(t)$. La función de supervivencia $S(t)$ proporciona la probabilidad de sobrevivir al menos hasta el tiempo t. Es una función decreciente, con valor 1 al inicio del seguimiento y que tiende a 0 (o permanece constante) al aumentar el tiempo de seguimiento. Formalmente:

Suponemos que T es una variable aleatoria continua con función de distribución acumulativa $F(t) = P(T < t)$

$$S(t) = P(T > t); 0 < t < \infty \tag{4.1}$$

En el caso discreto la función de supervivencia se escribe como el producto de probabilidades condicionadas, donde si:

$$p_i = P(t > t_i | t > t_{i-1}) \tag{4.2}$$

Entonces

$$S(t_i) = p_1 \cdot p_2 \cdot ... \cdot p_i \tag{4.3}$$

La función de supervivencia a menudo se define en términos de la función de riesgo, que es la tasa de falla instantánea. Es la probabilidad de que, dado que un sujeto ha sobrevivido hasta el tiempo t, falla en el siguiente intervalo de tiempo, dividido por la longitud de ese intervalo. Formalmente, esto se puede expresar mediante:

$$h\left(t\right) = \lim_{\delta \to 0} \frac{P\left(t < T < t + \delta | T > t\right)}{\delta} \tag{4.4}$$

La función de riesgo $h(t)$ (también se le conoce como la *función de intensidad* o *fuerza de mortalidad*) es una medida de la probabilidad que se produzca el suceso estudiado en un individuo que aún no ha sufrido dicho suceso. El numerador es la probabilidad condicional de que el evento ocurre en el intervalo $[t, t + \delta)$ dado que no ha ocurrido antes, mientras que el denominador es el intervalo de tiempo. Con esto se obtiene la tasa de ocurrencia de un evento por unidad de tiempo.

La función de riesgo $h(t)$ se relaciona con $S(t)$ y $F(t)$ de la siguiente manera:

$$h(t) = \frac{f(t)}{S(t)} = \frac{f(t)}{1 - F(t)} \tag{4.5}$$

Donde $S(t)$ equivale a la probabilidad de sobrevivir al menos hasta el tiempo t, definida como:

$$S(t) = P(T \geq t) = 1 - F(t) = \int_t^\infty f(x)dx \tag{4.6}$$

La forma de la función de riesgo puede ser muy distinta en dependencia de los datos que se estén analizando. Esta función puede ser creciente, decreciente, constante o cualquier combinación de estas por partes dentro del dominio de t.

La función de riesgo acumulativo $H(t)$ se define como el área bajo la función de riesgo hasta el tiempo t, es decir,

$$H\left(t\right) = \int_0^t h\left(u\right) \, \mathrm{d}u \tag{4.7}$$

La función de supervivencia se puede definir en términos de la función de riesgo mediante:

$$S\left(t\right) = \mathrm{e}^{-\int_0^t h(u) \, \mathrm{d}u} = \mathrm{e}^{-H(t)} \tag{4.8}$$

La supervivencia media es el valor esperado del tiempo de supervivencia:

$$\mu = E(T) = \int_0^\infty t f\left(t\right) \, \mathrm{d}t \tag{4.9}$$

Usando integración por partes y el hecho que $f(t) = -\frac{dS}{dt}$ de puede escribir:

$$\mu = \int_0^\infty S\left(t\right)\,\mathrm{d}t \tag{4.10}$$

El tiempo medio de supervivencia solo se define si $S(\infty) = 0$, es decir, si todos los sujetos finalmente fallan. Si este no es el caso y $S(\infty) = c$, el área bajo la curva es infinita. En este caso, una solución alternativa es especificar un tiempo de supervivencia máximo posible, de modo que la integral se convierta en finita.

La mediana del tiempo de supervivencia se define como el tiempo t tal que $S(t) = \frac{1}{2}$. Si la curva de supervivencia no es continua en $t = \frac{1}{2}$ (si la función de supervivencia es una función escalonada, por ejemplo), se toma la mediana como el valor de t más pequeño, de manera que $S(t) \leq \frac{1}{2}$.

La función de distribución $f(t)$ puede ser aproximada mediante modelos de distribución conocidos, por ejemplo, la Exponencial, Poisson, Weibull, etc. En estos casos los parámetros del modelo elegido deben ajustarse con las observaciones.

Supongamos que tenemos una serie de observaciones en los instantes $t_1, t_2 \ldots, t_n$; de cierta distribución con parámetros desconocidos. De acuerdo a la teoría de estimación de máxima verosimilitud el parámetro puede ser estimado maximizando la función de verosimilitud:

$$L\left(\lambda, t_1, t_2, \ldots, t_n\right) = \prod_{i=1}^{n} f\left(t_i, \lambda\right) \tag{4.11}$$

Si se censuran algunas observaciones, tenemos que hacer un ajuste a esta expresión. Para una observación de falla, escribimos la función f como se hizp en (4.11). Pero, para una observación censurada por la derecha, usamos la función de supervivencia, lo que indica que se sabe que la observación solo supera un valor particular. La verosimilitud en general toma la forma:

$$L\left(\lambda, t_1, t_2, \ldots, t_n\right) = \prod_{i=1}^{n} \left(f\left(t_i, \lambda\right)\right)^{\delta_i} \left(S\left(t_i, \lambda\right)\right)^{1-\delta_i} = \prod_{i=1}^{n} \left(h\left(t_i, \lambda\right)\right)^{\delta_i} S\left(t_i, \lambda\right) \tag{4.12}$$

Esta expresión significa que cuando t_i es una falla observada, el indicador de censura es $\delta_i = 1$ y usamos un factor con f. Cuando t_i es una observación censurada, tenemos $\delta_i = 0$ y se utiliza un factor de supervivencia. Alternativamente, podemos ingresar un factor de riesgo para cada observación censurada y un factor de supervivencia para cada observación censurada o no.

Existen múltiples modelos paramétricos para aproximar la función de distribución (o equivalentemente, la función de riesgo). Pero, ¿qué modelo paramétrico se debe usar para una aplicación en particular? Al modelar la supervivencia en casos reales, es difícil saber qué familia paramétrica elegir, y con frecuencia ninguna de las familias disponibles tiene la flexibilidad suficiente para modelar la forma real de la distribución. Por lo tanto, en las aplicaciones prácticas, los métodos no paramétricos resultan ser más eficientes. En el análisis de supervivencia, los métodos de estimación no paramétricos más utilizados son el Modelo Actuarial y el modelo del límite-producto de Kaplan-Meier. Ambos se basan en el cálculo de probabilidades condicionadas.

El método actuarial calcula la probabilidad de sobrevivir en intervalos de tiempos definidos. La estimación de la función de supervivencia en el tiempo t_k viene dado por:

$$\widehat{S}(t_k) = \prod_{i=1}^{k} \left(\frac{n_i - \frac{m_i}{2} - d_i}{n_i - \frac{m_i}{2}}\right) \tag{4.13}$$

Donde d_i corresponde el número de fallas en el intervalo $[t_{i-1}, t_i)$, n_i corresponde al número de individuos sujetos a riesgo al inicio del intervalo $[t_{i-1}, t_i)$ y m_i corresponde al número de censuras en el intervalo $[t_{i-1}, t_i)$.

El modelo Kaplan-Meier es un estimador de la función de supervivencia, en presencia de datos censurados u observaciones incompletas. A diferencia del método actuarial, en el modelo del

límite-producto no se pre-definen intervalos de tiempo. La función de supervivencia es estimada en cada momento de ocurrencia del evento. La estimación de la función de supervivencia del modelo Kaplan-Meier viene dada por:

$$\widehat{S}(t) = \prod_{t_i \leq t} \left(\frac{n_i - d_i}{n_i} \right), \forall t \geq t_1 \tag{4.14}$$

La representación gráfica de la probabilidad de supervivencia acumulada a través del tiempo se denomina *curva de supervivencia*, que se obtiene a partir de la aplicación del estimador de Kaplan-Meier (4.14). Es posible graficar curvas diferentes que, por ejemplo, podrían corresponder a tratamientos diferentes en enfermos de cáncer y compararlas para estimar la efectividad o no del tratamiento.

Es posible determinar si existe diferencia significativa entre diferentes curvas de supervivencia utilizando el estadígrafo *logrank*. Este estadígrafo calcula para cada instante de tiempo el número de eventos que se esperan. Se obtiene un valor final que puede compararse mediante una distribución de χ^2 y que puede dar un valor-p que permita valorar la existencia de diferencias estadísticas significativas entre las curvas. Para comprobar si la diferencia entre dos curvas es estadísticamente significativa, es necesario plantear la hipótesis y comprobarla con el estadígrafo *logrank*. La hipótesis nula plantea que no existen diferencias entre las funciones de supervivencia, mientras que la hipótesis alternativa plantea que existe diferencia entre las curvas de supervivencia.

4.2. Modelo de Regresión de Cox

Los modelos paramétricos requieren fuertes suposiciones sobre la forma de la distribución de supervivencia subyacente y sus parámetros son estimados a partir de la función de verosimilitud general (4.12). Una manera de relajar estas suposiciones es considerando la *función de verosimilitud parcial* para generar un modelo semiparamétrico. El uso de una función de verosimilitud parcial permite utilizar una distribución de supervivencia base no especificada $h_0(t)$ que contiene toda la dependencia temporal y trabajar independientemente con las distribuciones de supervivencia de los sujetos en función de sus variables predictoras. La verosimilitud parcial difiere de la función de verosimilitud general de dos maneras. Primero, es un producto de expresiones, una para cada tiempo de falla, donde los tiempos de censura no contribuyen con ningún factor. Segundo, los factores de la verosimilitud parcial son probabilidades condicionales.

Definimos la tasa de riesgo (en relación con el riesgo de referencia) para el sujeto i por $\psi_i = \exp X_i\beta$. Donde X_i es un vector de valores de las variables predictoras para el sujeto i y β es un vector de coeficientes, uno para cada variable predictora.

Justo antes del primer tiempo de fallo, todos los sujetos están "en riesgo" de fallar y entre estos, uno va a fallar. El "conjunto de riesgo" es el conjunto de todos los individuos en riesgo de fallar en el instante de tiempo t_j y se denota por R_j. La verosimilitud parcial es un producto de términos, uno por cada tiempo de falla. Para cada factor, (es decir, para cada i), el denominador es la suma de todos los riesgos en el conjunto de riesgo R_i (denotado por "$k \in R_j$") $h_k = h_0\psi_k$ y el numerador es la función de riesgo $h_i = h_0\psi_i$ para un individuo en el conjunto de riesgo R_i. La función de riesgo base $h_0(t_j)$ se cancela de todos los términos, de la siguiente manera:

$$L(\beta) = \prod_{j=1}^{D} \left(\frac{h_0(t_j)\,\psi_j}{\sum_{k \in R_j} h_0(t_k)\,\psi_k} \right) = \prod_{j=1}^{D} \left(\frac{\psi_j}{\sum_{k \in R_j} \psi_k} \right) \tag{4.15}$$

Donde D es el número de fallas. Esta función se denomina verosimilitud parcial porque carece de factores para las observaciones censuradas.

El modelo de regresión de Cox (también conocido como modelo proporcional de Cox) es una técnica semiparamétrica que permite modelar la relación entre la supervivencia (función

de riesgo) y un conjunto de variables predictoras. Este modelo estima la función de riesgo condicionada al valor de las variables predictoras y emplea la verosimilitud parcial (4.15) para estimar los parámetros β asociados con estas variables.

Si se considera una respuesta binaria en el análisis de supervivencia, se podría asumir que una regresión logística sería suficiente para modelar. Sin embargo, estas regresiones evalúan la información en un instante de tiempo determinado y no podrían, por ejemplo, comparar las curvas de supervivencia que se dan a través del tiempo. La regresión de Cox en cambio, modela la función de riesgo $h(t)$, es decir, la tasa de riesgo que corresponde al número de nuevos casos de falla por unidad de tiempo. Dicho de otra manera, modela la probabilidad condicional de que dado que un individuo sobrevive en un instante de tiempo t con ciertos valores de las variables predictoras, éste experimentará la falla en el siguiente instante.

Para este modelo se asume que la proporción de riesgo para cada individuo con variables predictoras $X = (x_1, \ldots, x_p)$, toma la forma semiparamétrica:

$$h(t \mid X) = h_0(t) \exp(\sum_p \beta_p X_p) \tag{4.16}$$

Donde $h_0(t)$ es la tasa de riesgo base (sin parámetros) definida como una función del tiempo. La expresión $\psi = \exp(\sum_p \beta_p X_p)$ describe la dependencia de la función de riesgo respecto a las variables predictoras. El modelo proporcional de Cox puede considerarse como una función del riesgo relativo y el aumento o la reducción del riesgo es el mismo en todos los instantes de tiempo t. Dicho en otros términos, la expresión $h_0(t)$ es la única parte del modelo de Cox dependiente del tiempo, mientras que la parte $\exp(\sum_p \beta_p X_p)$, solo depende del vector de variables predictoras. Esta aproximación a la función de riesgo se considera un método semiparamétrico debido a que:

- Considera una parte paramétrica asociada a la expresión $\exp(\sum_p \beta_p X_p)$, con la cual se estiman los parámetros $\widehat{\beta} = (\widehat{\beta}_1, ..., \widehat{\beta}_p)$ de la regresión.
- Comprende una parte no paramétrica $h_0(t)$. Ésta es una función arbitraria y no especificada del tiempo, que se estima en un segundo estadio, condicionada al valor hallado para los parámetros de la regresión $\widehat{\beta} = (\widehat{\beta}_1, ..., \widehat{\beta}_p)$.

4.3. Aplicación modelo de supervivencia al CCP

En la aplicación del análisis de supervivencia al CCP, vamos a considerar el *tiempo de supervivencia o falla* como el tiempo desde que el cliente adquirió el servicio hasta que termina por fugarse de la compañía, es decir, la antigüedad de éste cliente en Virgin Mobile. El evento que produce la "falla" es el acto de la fuga. Algunos de los clientes considerados en el tiempo de seguimiento de este estudio no se fugan, por lo que no se conoce con certeza su tiempo de supervivencia. Se plantea entonces, que las observaciones correspondientes a estos casos están censuradas por la derecha. .

Es importante tener en cuenta que: (1) los individuos no entran al estudio (fecha en que adquieren el servicio) al mismo tiempo, (2) no todos los clientes se fugan (evento de "falla") durante el tiempo de seguimiento (3) algunos clientes se "pierden" mientras son estudiados, es decir eventualmente tienen nula actividad [73]. La gran ventaja de los modelos de supervivencia es que este tipo de casos son considerados de manera apropiada en el análisis.

Para el análisis de supervivencia emplearemos R. El paquete `eha` ("event history analysis") se centra en los modelos proporcionales de Cox y se considera como una extensión del paquete `survival`, el cual contiene los comandos básicos para realizar un análisis de supervivencia, incluida la definición de objetos *Surv*, las curvas de Kaplan - Meier, modelos de Cox, etc. Por otro lado, el paquete `survMisc` ayuda en el análisis de supervivencia de datos que están censurados por la derecha, mientras que `survminer` permite generar gráficos detallados de las curvas de supervivencia.

Inicialmente, consideramos el modelado no-paramétrico de la función de supervivencia empleando el estimador (4.14). La mediana del tiempo de fuga se define como $t_{\text{med}} = \text{ínf}\left\{t : S(t) \leq \frac{1}{2}\right\}$. En este caso particular, este tiempo coincide con la mediana de la antigüedad ("tenure") de los clientes. El valor de t_{med} junto al intervalo de confianza puede calcularse mediante:

```
result <- survfit(Surv(tenure, StatusFugado) ~ 1, data=y,
                  conf.type="log-log",type='fh')
```

En el comando anterior, el tiempo de fuga es "tenure" y se utiliza la variable binaria "StatusFugado" como el indicador de falla o resultado del evento al final del periodo de seguimiento. Es decir, en el caso de que el cliente se haya fugado StatusFugado = 1, en caso contrario StatusFugado = 0. Este último caso, se toma como una observación censurada por la derecha. Los resultados obtenidos son los siguientes:

N	Eventos	Mediana	0.95LCL	0.95UCL
854852	583622	0.737	0.734	0.74

Lo que indica que la mediana del "tiempo de vida" de los clientes de la compañía Virgin Mobile es de 0.737 años. La figura 4.1 representa la curva de supervivencia, es decir, la probabilidad empírica de supervivencia para 854.852 clientes de Virgin entre los años 2012 y 2016.

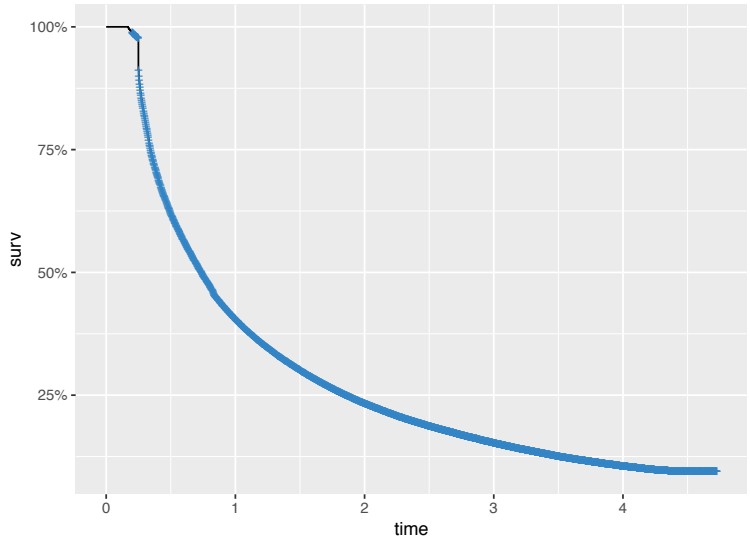
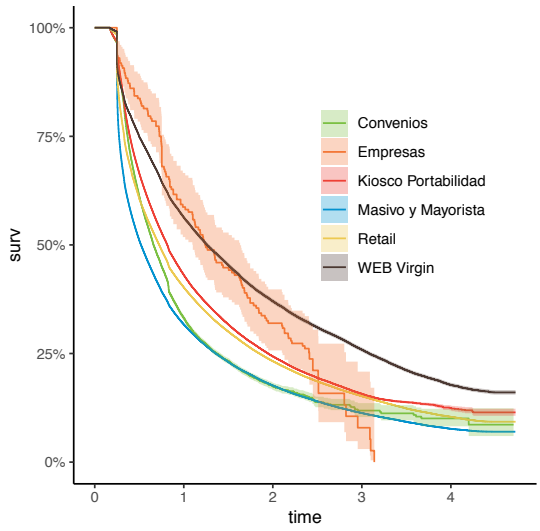
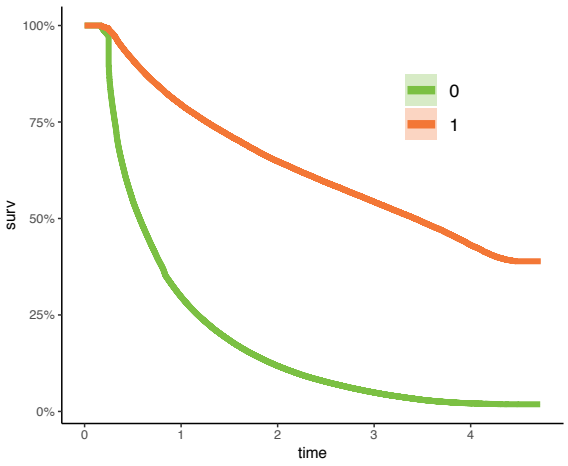


Figura 4.1: El gráfico muestra la curva de supervivencia para clientes de Virgin Mobile dede "2012-04-11" al "2016-10-19". Se observa que la probabilidad de supervivencia disminuye prácticamente de manera exponencial. Poe ejemplo, La probabilidad de tener un año de antigüedad en Virgin Mobil es cercana al 40 %, mientras que la probabilidad de llegar a los 3 años disminuye a un 15 % aproximadamente.

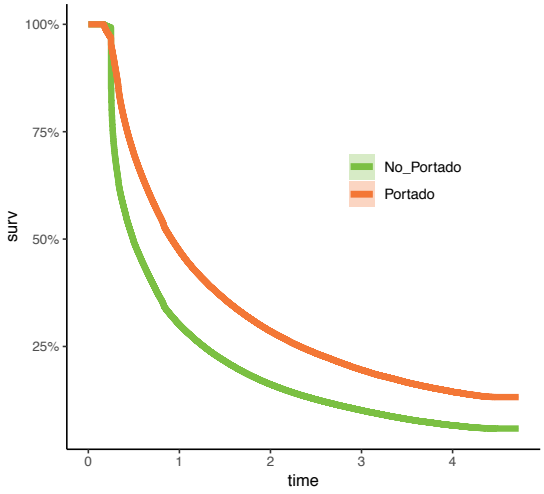
La figura 4.2 muestra las curvas de supervivencia empíricas, obtenidas a partir del estimador de Kaplan-Meier (4.14) estratificadas según las variables "Canal", "Portado" y "Suscrito selfcare", utilizando como tiempo de falla la variable tenure. En la figura 4.2(a) se aprecia la estratificación de los datos a partir de la variable Canal. La curvas en general tienen un comportamiento similar, salvo por dos diferencias que saltan a la vista: 1) la gran variabilidad del Canal "Empresas" y 2) la mayor tasa de supervivencia del Canal "Web Virgin". De la figura 4.2(b) concluimos que aquellos clientes que no se encuentran inscritos en la página Web tienen menor tasa de supervivencia respecto de los que sí lo están. Por último, la figura 4.2(c) indica que los clientes que se no se han "portado" a Virgin tienen mayor probabilidad de fugarse de la compañía que aquellos que si lo han hecho.



(a) Curva de supervivencia estratificada por Canal



(b) Curva de supervivencia estratificada por Suscrito selfcare



(c) Curva de supervivencia estratificada por Portado

Figura 4.2: Curvas de supervivencia empíricas estratificadas por diferentes atributos fijos. Se utilizaron datos de 850.000 clientes de la compañía desde "2012-04-11" al "2016-10-19".

4.3.1. Análisis de intervalos de tiempo específicos para el seguimiento

Para analizar el efecto de las variables de red (ver tabla 2.2), que son calculadas en varios meses (agosto, septiembre, octubre) cambiamos el marco de tiempo en el análisis de supervivencia, limitando el intervalo de seguimiento a un mes o periodo de tiempo en particular. Sea f_{ini} la fecha de inicio de la observación (por ejemplo, f_{ini} ="2016-08-01"), f_{end} fecha final del seguimiento (por ejemplo, f_{end} ="2016-08-31"), f_{a} la fecha de activación del cliente (fecha de ingreso a la compañía, por ejemplo f_{a} ="2012-07-03") y f_{f} la fecha de fuga si el cliente se fugó en el tiempo de seguimiento. Definimos además los tiempos: $t_1 = [f_{\text{ini}}, f_{\text{end}}]$, $t_2 = [f_{\text{a}}, f_{\text{end}}]$, $t_3 = [f_{\text{a}}, f_{\text{f}}]$ y $t_4 = [f_{\text{ini}}, f_{\text{f}}]$ El nuevo tiempo de seguimiento t_{obs} se calcula entonces de la siguiente manera:

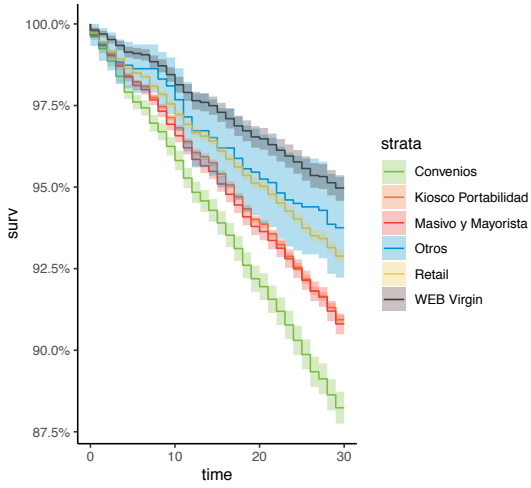
Para los clientes no fugados:

$$t_{\text{obs}} = \text{mín}(t_1, t_2), \text{con la condición } f_{\text{a}} < f_{\text{end}}$$

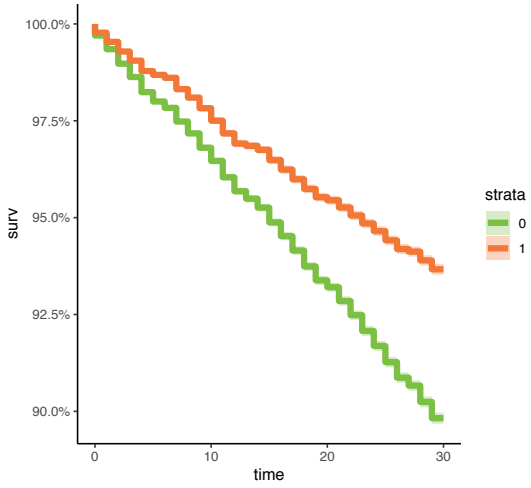
En el caso de los clientes fugados:

$$t_{\text{obs}} = \text{mín}(t_1, t_2) \quad \text{si} \quad f_{\text{f}} > f_{\text{end}}$$
$$t_{\text{obs}} = \text{mín}(t_3, t_4) \quad \text{si} \quad f_{\text{ini}} < f_{\text{f}} \leq f_{\text{end}}$$

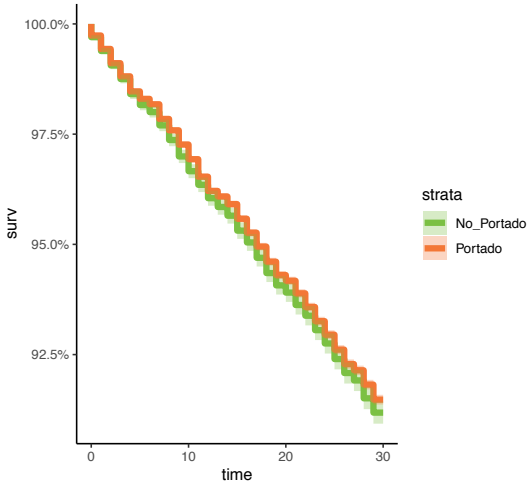
Utilizando el tiempo de observación t_{obs} se puede localizar el efecto de las variables predictoras a un intervalo dado. Por ejemplo, en las figuras 4.3 se muestran los efectos de los atributos fijos "Canal", "Suscrito selfcare" y "Portado" para el mes de agosto de 2016. Se puede observar que la probabilidad de supervivencia decae a una tasa constante. La figura 4.3(a) muestra que el valor "Web Virgin" de la variable "Canal" presenta una mayor probabilidad de supervivencia en comparación con el resto. De la figura 4.3(c), podemos constatar que la variable "Portado" no es una variable que distingue el comportamiento de manera clara en este mes en particular.



(a) Curva de supervivencia estratificada por Canal



(b) Curva de supervivencia estratificada por Suscrito selfcare



(c) Curva de supervivencia estratificada por Portado

Figura 4.3: Curvas de supervivencia empíricas estratificadas por diferentes atributos fijos. Se utilizaron datos de clientes de la compañía durante el mes de agosto de 2016.

La figura 4.4 muestra la curva de supervivencia estratificada por número de vecinos fugados. Estás curvas evidencian comportamientos distintos de acuerdo a este número. La probabilidad de supervivencia cae a una tasa más alta en aquellos clientes que tienen al menos un vecino fugado en su vecindad. El efecto en las diferencia de tasas se observa con cierto retraso, es decir, un tiempo después de iniciado el periodo de seguimiento. Este comportamiento puede ser

interpretado como una evidencia de posible "contagio y/o homofilia" entre clientes. Dicho de otro modo, la probabilidad de fuga crece si en la vecindad del cliente hay otros que se fugan.

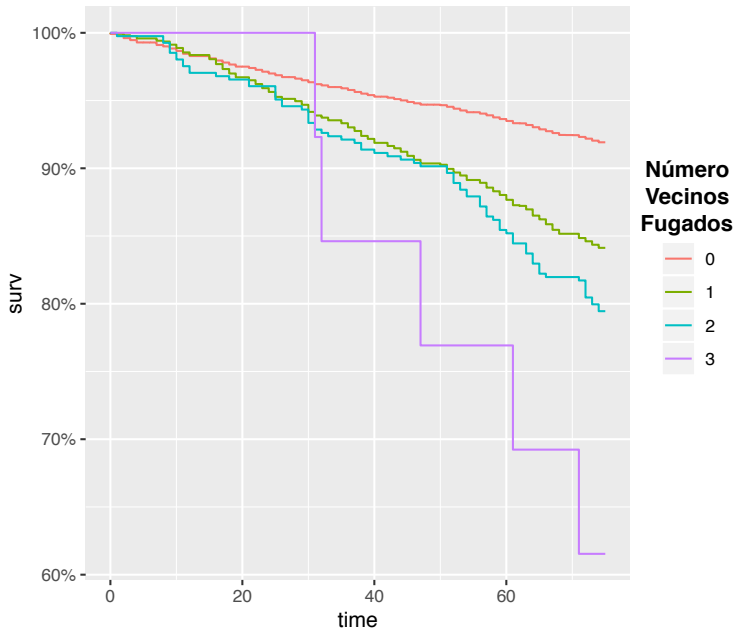


Figura 4.4: La figura muestra la curva de supervivencia estratificada por número de vecinos fugados para clientes de Virgin Mobile dede "2016-08-01" al "2016-10-15". Las curvas muestran evidencias de un efecto de "contagio", que se acentúa a lo largo del tiempo. La diferencia de comportamiento de los que tienen al menos un vecino fugado respecto a los que no lo tienen es más evidente.

La función `survdiff` determina si existe diferencia significativa entre dos o más curvas de supervivencia, o bien para una curva única respecto de una alternativa conocida. La función entrega la estadística de χ^2 y el valor-p para la prueba de log-rank, con hipótesis nula de que no hay diferencia entre las dos curvas de supervivencia. Como planteamos en el párrafo anterior, la figura 4.4 deja entrever que aquellos clientes que están "expuestos" a clientes fugados, tienden a presentar un tiempo de supervivencia menor que aquellos que no tienen este contacto. Para determinar si esta hallazgo es confiable, podemos realizar una prueba de log-rank mediante:

```
survdiff(Surv(tenure, StatusFugado) ~ NumVecinosFugados,data=y)
```

Obtenemos los siguientes resultados:

	N	Observado	Esperado	(O-E)^2/E	(O-E)^2/V
NumVecinosFugados=0	223050	17275	18367.9	65.0	788.7
NumVecinosFugados=1	18475	2404	1501.1	543.0	587.9
NumVecinosFugados=2	1728	301	138.4	191.2	192.8
NumVecinosFugados=3	174	41	13.6	55.2	55.3

El valor de χ^2 es igual a 856 mientras que el valor-p es menor a 2e-16. Esto indica que existe evidencia significativa para rechazar la hipótesis nula, por lo que los tiempos de supervivencia son diferentes en cada uno de los grupos.

4.3.2. Aplicación del modelo semi-paramétrico proporcional de Cox al CCP.

La función `coxreg` del paquete `eha` permite ajustar el modelo semi-paramétrico proporcional de Cox (4.16) mediante:

```
fit <- coxreg(Surv(tm_obs,StatusFugado)~ x,data = y)
```

Inicialmente consideramos el efecto del número de vecinos fugados, el número de vecinos en general y la antigüedad sobre la función de riesgo $h(t)$. Tal como se hizo para generar la figura 4.4, se definió un tiempo de seguimiento, que considera el intervalo entre las fechas "2016-08-01" y "2016-10-15" (75 días).

```
fit <- coxreg(Surv(tm_obs,StatusFugado)~ NumVecinosFugado,data = y)

Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
NumVecinosFugado    0.091     0.479     1.614     0.017     0.000

Events                    20021
Total time at risk        17164460
Max. log. likelihood      -246770
LR test statistic         690.08
Degrees of freedom        1
Overall p-value           0

fit <- coxreg(Surv(tm_obs,StatusFugado)~ NumVecinos,data = y)

Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
NumVecinos         17.821    -0.006     0.994     0.001     0.000

Events                    20021
Total time at risk        17164460
Max. log. likelihood      -247045
LR test statistic         140.87
Degrees of freedom        1
Overall p-value           0
```

Estos resultados indican que el número de vecinos fugados ("NumVecinosFugado") efectivamente contribuye al incremento del riesgo, mientras que el número de vecinos en general ("NumVecinos', que también puede ser interpretado como el grado del nodo/cliente en la red de llamadas) contribuye a disminuir la probabilidad de fuga.

Al considerar la antigüedad del cliente en la compañía tenemos:

```
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
tenure              1.458    -0.566     0.568     0.010     0.000

Events                    20021
Total time at risk        17164460
Max. log. likelihood      -244800
LR test statistic         4630.62
Degrees of freedom        1
Overall p-value           0
```

Lo que indica, que a mayor antigüedad del cliente menor es la probabilidad de fuga.

Con las tres variables incluidas simultáneamente en el modelo obtenemos:

```
fit <- coxreg(Surv(tm_obs,StatusFugado)~ NumVecinos + NumVecinosFugado+
              tenure, data = y)

Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
```

```
tenure              1.458    -0.561     0.571     0.010     0.000
NumVecinosFugado    0.091     0.540     1.717     0.017     0.000
NumVecinos         17.821    -0.005     0.995     0.001     0.000

Events                    20021
Total time at risk        17164460
Max. log. likelihood      -244378
LR test statistic         5474.50
Degrees of freedom        3
Overall p-value           0
```

Otro de los aspectos de interés en este estudio es determinar cómo los patrones de actividad y consumo del servicio por parte del cliente contribuyen a la función de riesgo. Con este fin, consideramos las variables "TotalCalls" y "CallFromChurned".

```
fit <- coxreg(Surv(tm_obs,StatusFugado)~ TotalCalls + CallFromChurned,data = y)
```

Este resultado muestra que una mayor actividad del cliente ("TotalCalls") contribuye a la disminución de la probabilidad de fuga. Por otro lado, si esta actividad está vinculada con clientes que se fugan, entonces la función de riesgo crece.

```
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
CallFromChurned     0.051     0.843     2.322     0.019     0.000
TotalCalls         17.731    -0.007     0.993     0.001     0.000

Events                    18212
Total time at risk        15109199
Max. log. likelihood      -221725
LR test statistic         1495.73
Degrees of freedom        2
Overall p-value           0
```

Considerando en el modelo la actividad general y la interrelación con vecinos fugados de manera simultanea se tiene:

```
Covariate           Mean       Coef     Rel.Risk   S.E.    Wald p
CallFromChurned     0.051     0.758     2.134     0.019     0.000
TotalCalls         17.731    -0.006     0.994     0.001     0.000
tenure              1.459    -0.570     0.566     0.010     0.000
NumVecinosFugado    0.088     0.505     1.657     0.018     0.000

Events                    18212
Total time at risk        15109199
Max. log. likelihood      -219343
LR test statistic         6259.46
Degrees of freedom        4
Overall p-value           0
```

Algunas de las variables predictoras en la base de datos están altamente correlaciondas (ver figura 2.7 y sección 3.5.1). Por tanto, al considerar un modelo con todas las variables debe tenerse en cuenta la posibilidad de muticolinealidades que pueden afectar la interpretación correcta del rol de las variables en el modelo. Sin perder de vista este hecho, construimos un modelo con todas las variables:

```
fit <- coxreg(Surv(tm_obs,StatusFugado)~ . ,data = y)
```

Covariate	Mean	Coef	Rel.Risk	S.E.	Wald p
tenure	1,567	-0,224	0,799	0,018	0
NumVecinos	19,494	0,004	1,004	0,003	0,201
NumVecinosVirgin	0,633	-0,015	0,985	0,03	0,606
NumVecinosFugado	0,098	-0,011	0,989	0,063	0,857
CallFromChurned	0,044	0,084	1,087	0,059	0,155
CallTimeFromChurned	0,074	-0,03	0,971	0,02	0,145
CallTimeToChurned	0,164	0,02	1,02	0,007	0,002
TotalCallTime	193,220	0	1	0	0,304
CallTime_in	33,192	0,001	1,001	0,001	0,619
CallTimeOutOnNet	3,948	-0,002	0,998	0,003	0,576
CallOutOnNet	2,659	0,009	1,009	0,01	0,358
Reciprocidad	0,823	0,05	1,052	0,046	0,274
NumVecinosFugado2	0,040	0,186	1,204	0,092	0,043
NumVecinosVirgin2	0,303	-0,058	0,943	0,048	0,225
NumVecinos2	17,826	0,026	1,026	0,003	0
NumVecinosVirgin2	0,555	0,005	1,005	0,031	0,876
NumVecinosFugado2	0,056	0,077	1,08	0,071	0,278
CallFromChurned2	0,023	0,086	1,09	0,069	0,209
CallTimeFromChurned2	0,040	0,009	1,009	0,014	0,514
CallTimeToChurned2	0,096	0,007	1,007	0,014	0,641
TotalCallTime2	157,797	0,001	1,001	0	0
CallTime_in2	30,587	-0,002	0,998	0,001	0,015
CallTimeOutOnNet2	3,565	0,001	1,001	0,003	0,646
CallOutOnNet2	2,399	0,009	1,009	0,01	0,385
Reciprocidad2	0,791	0,079	1,082	0,044	0,075
NumVecinosFugado3	0,022	0,056	1,058	0,103	0,583
NumVecinosVirgin3	0,262	-0,028	0,973	0,052	0,596
NumVecinos3	19,019	-0,068	0,934	0,004	0
NumVecinosVirgin3	0,524	0,108	1,114	0,033	0,001
NumVecinosFugado3	0,024	-0,107	0,898	0,113	0,342
CallFromChurned3	0,010	0,185	1,203	0,092	0,044
CallTimeFromChurned3	0,017	-0,031	0,97	0,031	0,32
CallTimeToChurned3	0,040	-0,019	0,981	0,03	0,521
TotalCallTime3	183,385	-0,002	0,998	0	0
CallTime_in3	33,155	0	1	0,001	0,931
CallTimeOutOnNet3	3,767	-0,001	0,999	0,005	0,804
CallOutOnNet3	2,524	-0,014	0,986	0,016	0,372
Reciprocidad3	0,767	0,133	1,142	0,038	0
NumVecinosFugado3	0,010	0,151	1,163	0,152	0,320
NumVecinosVirgin3	0,254	-0,001	0,999	0,054	0,980

Tabla 4.1: Aplicación Modelo de Cox. Las variables de los tres meses de estudio fueron fusionadas en un solo set de datos. El número 2 indica el mes de septiembre, mientras que el número 3 corresponde al mes de octubre (las variables sin número están asociadas al mes de agosto). La columna Coef representa los valores de los coeficientes β del modelo. Es posible notar que aquellas variables asociadas al contacto con clientes fugados tienen un efecto positivo en el modelo de Cox, lo que indica que tienden a disminuir la probabilidad de supervivencia de los clientes.

La tabla 4.1 muestra los resultados de la aplicación del modelo con todas las variables. Para su construcción, una vez considerado el tiempo de seguimiento se agregaron/fusionaron en una

sola base de datos, todas las variables predictoras (atributos y datos de redes) correspondiente a este periodo de observación. De esta manera, se contó con una mayor cantidad de variables para el estudio. El número 2 y 3 al final de los nombres de las variables, indican a qué mes pertenecen (septiembre u octubre, respectivamente).

La columna Coef indica los valores de los coeficientes β_p de la función 4.16. Un valor de β_p positivo, o equivalentemente una proporción de riesgo mayor que uno, indica que a medida que aumenta el valor de la variable predictora p, aumenta el riesgo del evento y, por lo tanto, la duración de la supervivencia disminuye. Dicho de otra manera, una proporción de riesgo por encima de 1 indica que una variable predictora está asociada positivamente con la probabilidad del evento y negativamente con la supervivencia. Junto a la estimación de la función de riesgo, la regresión permite valorar el efecto y la significancia de las variables en la probabilidad de supervivencia.

De la tabla es posible observar que los coeficientes relacionados a las variables asociadas a "influencia" tienen en general coeficientes positivos, es decir, disminuyen la probabilidad de supervivencia (aumentan el riesgo). Más específicamente estas variables son: "CallFromChurned", "CallTimeToChurned", "NumVecinosFugado" y "CallTimeFromChurned". Mientras que la variables asociadas a consumo activo poseen en general coeficientes negativos. Algunas de esta variables son: "CallTimeIn", "CallOutOnNet","Reciprocidad","NumVecinosVirgin".

De acuerdo al valor Wald p del test para evaluar si el coeficiente β correspondiente a una variable dada es significativamente diferente de 0 se puede concluir que las variables "NumVecinos", "NumVecinosFugado", "CallFromChurned", "TotalCallTime", "Reciprocidad", "NumVecinosFugado" y "NumVecinosVirgin" poseen coeficientes significativamente distintos de cero.

El modelo considerando todas las variables, entrega los siguientes resultados:

```
Events                    4379
Total time at risk        14003028
Max. log. likelihood      -35464
LR test statistic         34854.20
Degrees of freedom        65
Overall p-value           0
```

Lo que nos indica que el modelo es significativo. Si se considera el valor del test LR (34.854), obtenemos que su valor-p asociado es 0.001, lo que indica que existe evidencia estadísticamente significativa para rechazar la hipótesis nula.

4.3.3. Análisis de supervivencia dinámico y el Modelo Aditivo de Aalen.

En el modelo de Cox (4.16), los coeficientes son constantes a lo largo del tiempo. Más aún, la tasa de riesgo $\psi_i = \exp X_i\beta$ o riesgo relativo es constante e igual para todos los individuos. Dicho de otra manera, las funciones de riesgo de dos individuos son proporcionales y solo se diferencian por una función base $h_0(t)$ que contiene toda la información del transcurso del tiempo. Por lo tanto, el análisis estándar de Cox no proporciona información sobre los efectos de las variables predictoras sobre el riesgo a través del tiempo. Considerar esta información es relevante dado que podría darse el caso, por ejemplo, de que un tratamiento médico sea efectivo solo al inicio o en cierto intervalo de tiempo fuera del cuál no tendría efecto alguno. Un modelo lineal con coeficientes variables en el tiempo para datos de análisis de supervivencia fue desarrollado por Aalen en 1989. En este modelo [74, 75] la intensidad $h(t)$ (función de riesgo), se le denomina función Aditiva de Aalen y es escrita en la forma:

$$h(t|X) = \beta_0(t) + \beta^T(t) \cdot X \tag{4.17}$$

Con este modelo, la función de riesgo depende linealmente de las variables predictoras con coeficientes que pueden variar en el tiempo. El vector $\beta(t)$, contiene la información de la regresión.

El primer elemento $\beta_0(t)$ es una función de base o de intersección que define el comportamiento del riesgo cuando no actúan las variables predictoras, mientras que al resto de los coeficientes, se les denomina funciones de regresión. En la práctica, resulta más fácil estimar las funciones de regresión acumulativas, que son definidas mediante:

$$A(t) = \int_0^t \beta(s)ds \tag{4.18}$$

Donde $t_1 < t_2 < \cdots < t_k$ corresponden a los distintos tiempos de falla ordenados cronológicamente y donde los componentes del vector $A(t)$ son funciones del tiempo. Las pendientes de las curvas definidas por $A(t)$ proporcionan información sobre los coeficientes β y por tanto de los efectos de las variables predictoras correspondientes.

El estimador de mínimos cuadrados de Aalen, está dado por:

$$A^*(t) = \sum_{t_k \leq t} Z(t_k) I_k \tag{4.19}$$

Donde I_k es una función indicadora que toma el valor de 1 solo para aquellos individuos cuyo evento o falla ocurre en el tiempo t_k y cero si esto no ocurre, y $Z(t)$ es la inversa generalizada de $Y(t)$. $Y(t)$ se construye de la siguiente manera: si el evento considerado no ha ocurrido para el i-ésimo individuo y no está censurado, entonces la i-ésima línea de $Y(t)$ es el vector $x_i(t) = (1, x_{i1}(t), x_{i2}(t), ..., x_{ip}(t))'$. En caso contrario, si el individuo no está bajo riesgo en el tiempo t, entonces la línea correspondiente de $Y(t)$ contiene sólo ceros.

Los componentes de $A^*(t)$ convergen asintóticamente, bajo condiciones apropiadas, para procesos gaussianos. En este caso, un estimador de la matriz de covarianza de $A^*(t)$ está dado por:

$$\Omega^*(t) = \sum_{t_k \leq t} Z(t_k) I_k^D Z(t_k)' \tag{4.20}$$

Donde I_D^k es una matriz diagonal con I_k como diagonal principal. Se puede estimar el riesgo acumulado y la función de supervivencia correspondiente dados los valores de las variables predictoras. Sea $x(t) = (1, x_1(t), x_2(t), ..., x_p(t))'$ el conjunto de variables en el tiempo t. El estimador del riesgo acumulado queda definido como:

$$H^*(t) = A^*(t)' x(t) \tag{4.21}$$

Y la función de supervivencia queda definida por:

$$S^*(t) = \exp(-H^*(t)) \tag{4.22}$$

Independientemente de la extensa utilización de los modelos de riesgos proporcionales, los modelos aditivos permiten obtener información adicional respecto de la influencia temporal que tienen las variables predictoras en el riesgo. En la próxima sección empleamos el modelo aditivo de Aalen en el CCP, para capturar posibles patrones de influencias cambiantes, a lo largo del tiempo de seguimiento, de las variables predictoras en el riesgo de fuga.

4.3.4. Aplicación Modelo Aditivo de Aalen al CCP

El paquete `survival` cuenta con la función `aareg` para ajustar el modelo de Aalen 4.17. Las interpretaciones del modelo ajustado deben hacerse con cuidado dado que las estimaciones podrían eventualmente volverse inestables cerca de la parte final de un conjunto de datos, dado que el incremento de β en el tiempo t está basado en aquellos sujetos que aún se encuentran en riesgo en el tiempo t y los parámetros de tolerancia pueden actuar para truncar la estimación antes de la última falla.

La aplicación del modelo 4.17 se realiza sobre los valores de las variables predictoras calculados para el mes de agosto. Con estos valores así fijados, tomamos el tiempo de seguimiento definido entre las fechas f_{ini} ="2016-08-01" y f_{end} ="2016-10-15". De esta manera, intentamos capturar el efecto de las variables predictoras sobre el riesgo de fuga en el mes de agosto y en parte de los próximos dos meses.

Para el análisis, consideramos inicialmente las variables predictoras que están relacionadas con posibles influencias y/o contagios de vecinos fugados.

```
Call:
aareg(formula = Surv(tm_obs, StatusFugado) ~ NumVecinosFugados +
NumVecinosVirgin + NumVecinosFugadosCercanos + NumVecinosVirginCercanos,
data = y)

n= 243469
75 out of 75 unique event times used

slope                        coef           se(coef)       z        p
Intercept                    0.001310  4.51e-06 3.48e-08 130.0 0.00e+00
NumVecinosFugados            0.000722  2.49e-06 1.64e-07  15.2 6.41e-52
NumVecinosVirgin            -0.000148 -4.62e-07 4.01e-08 -11.5 1.16e-30
NumVecinosFugadosCercanos    0.000777  3.27e-06 2.86e-07  11.5 2.33e-30
NumVecinosVirginCercanos    -0.000260 -9.11e-07 6.35e-08 -14.3 1.20e-46

Chisq=1663.6 on 4 df, p=<2e-16; test weights=aalen
```

La figura 4.5 muestra los valores acumulados de los coeficientes de regresión 4.18 del modelo de Aalen, correspondientes a las variables "NumVecinosFugados", "NumVecinosVirgin", "NumVecinosFugadosCercanos" y "NumVecinosVirginCercanos". Es posible apreciar que el coeficiente asociado a la variable "NumVecinosFugados" tiene una pendiente positiva, que es prácticamente constante a través del tiempo de seguimiento, incrementando con ello el riesgo de fuga. Esto quiere decir, que la exposición continua a clientes fugados disminuye la probabilidad de supervivencia.

La variable "NumVecinosVirginCercanos" ejerce una influencia prácticamente nula (pendiente constante de la curva) en los primeros 10 días del mes de agosto. Pero, luego influye de manera positiva en el riesgo de fuga. Por otro lado, la variable "NumVecinosVirgin" contribuye negativamente al riesgo de fuga, tal como lo muestra la pendiente negativa en la curva.

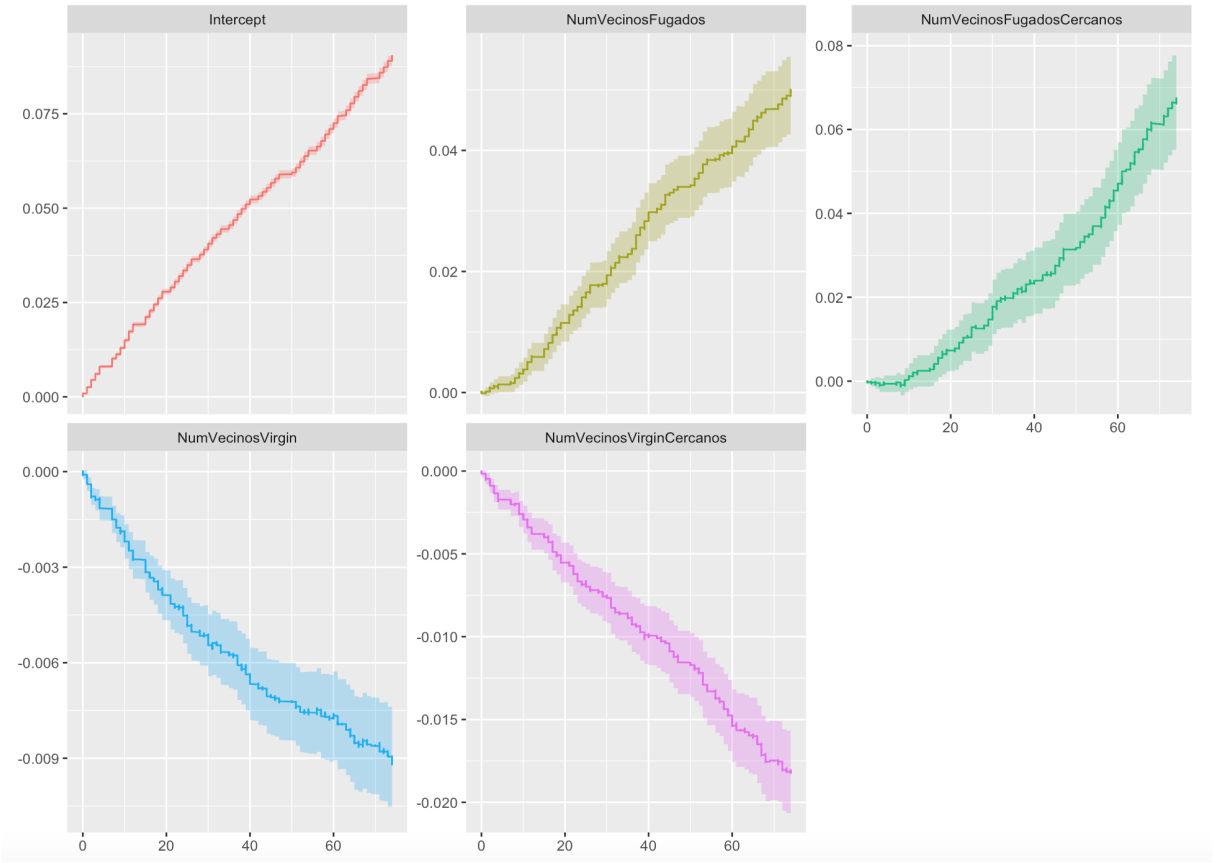


Figura 4.5: Modelo Aditivo de Aalen con variables de influencia. Las gráficas fueron construidas utilizando valores calculados para el mes de agosto, definiendo el tiempo de seguimiento desde el 1 de agosto hasta el 15 de octubre. La gráficas muestran el efecto prácticamente constante, si grandes variaciones a través del tiempo de estas variables. Por ejemplo, después de un efecto nulo en los primeros 10 días del mes de agosto, el Número de Vecinos Fugados Cercanos ejerce una influencia positiva (pendiente positiva de la curva) en el riesgo de fuga. Por otro lado, el número de vecinos en la red Virgin tiene un efecto negativo en el riesgo de fuga. Las curvas mostradas para los valores acumulados de los coeficientes de regresión, corroboran el "contagio".

En una segunda combinación de variables analizamos la actividad/interacción de los clientes con otros clientes fugados.

```
aareg(formula = Surv(tm_obs, StatusFugado) ~ CallFromChurned +
CallTimeFromChurned + CallToChurned + CallTimeToChurned,
data = y)

n=202156 (41313 observations deleted due to missingness)
64 out of 64 unique event times used

slope                    coef         se(coef)       z         p
Intercept            1.10e-03 4.59e-06 4.03e-08 114.000  0.00e+00
CallFromChurned      1.72e-03 6.93e-06 3.09e-07  22.400 2.71e-111
CallTimeFromChurned 3.71e-05 6.24e-08 8.45e-08   0.738  4.61e-01
CallToChurned        6.04e-04 2.82e-06 1.77e-07  15.900  8.97e-57
CallTimeToChurned    5.93e-06 3.61e-08 5.95e-08   0.606  5.45e-01

Chisq=1034.64 on 4 df, p=<2e-16; test weights=aalen
```

La figura 4.6, muestra que la cantidad de llamadas desde y hacia fugados influye de manera positiva en el riesgo de fuga. El efecto del tiempo total de estas llamadas parece no tener

mayor impacto en el riesgo de fuga de los clientes. El número de estas llamadas resulta ser más importante que el tiempo agregado de estas llamadas. Las gráficas muestran que la interacción con clientes fugados genera un mayor riesgo de fuga.

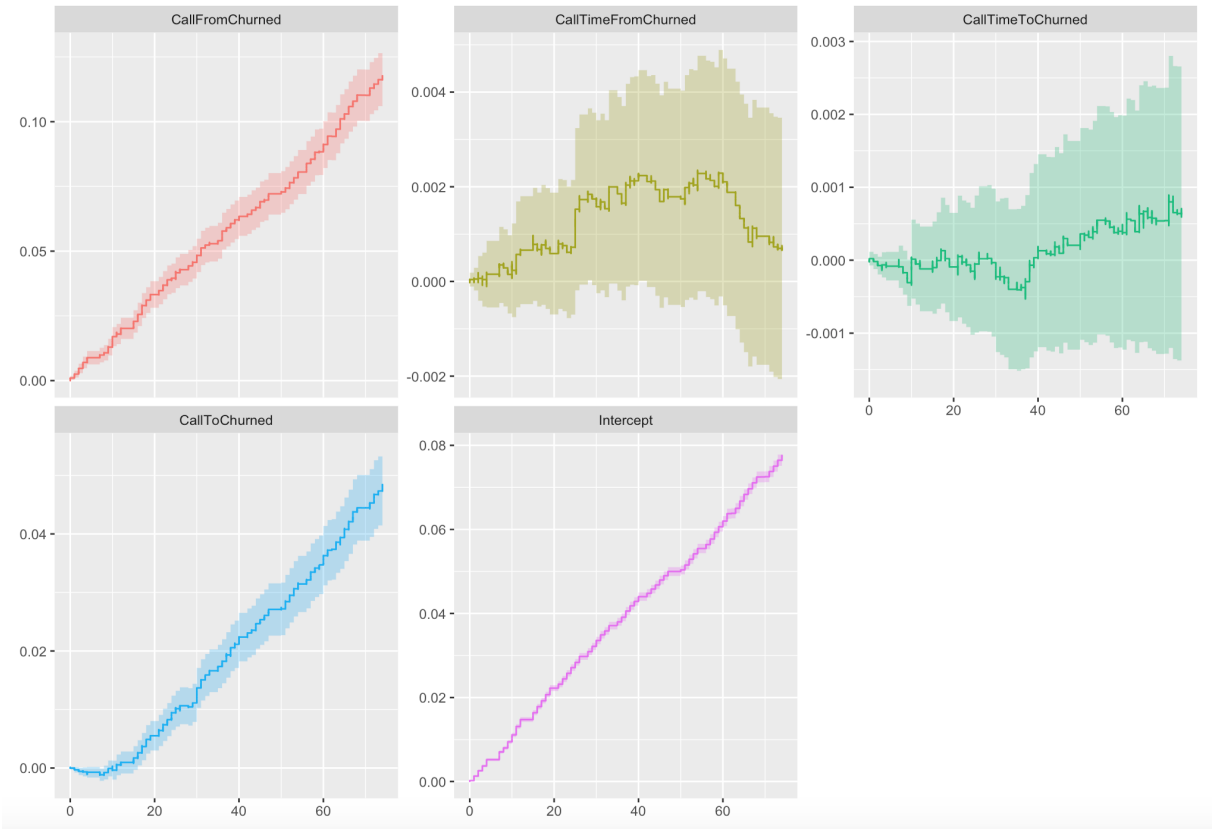


Figura 4.6: La figura muestra los valores acumulados de los coeficientes de regresión para el Modelo Aditivo de Aalen usando variables predictoras de interacción con fugados. Al igual que en la figura 4.5, el tiempo de seguimiento comprende desde el 1 de agosto hasta el 15 de octubre. Las gráficas muestran el efecto positivo en el riesgo de las variables relacionadas con la cantidad de llamadas realizadas y recibidas con clientes fugados. Sin embargo, el agregado de la duración de dichas llamadas no parece ejercer mayor influencia en el modelo.

Podemos concluir, que en general, las variaciones o efectos dinámicos no son muy apreciables para el conjunto de variables usados en los análisis anteriores, basados en modelos Aditivos de Aalen 4.17. Estos modelos, sin embargo, confirman algunos hallazgos realizados previamente, pero con el punto adicional de que los efectos sobre el riesgo de fuga producidos por estas variables predictoras son basicamente constantes en el tiempo. En particular, el "contagio" o influencia de los fugados y la interacción con estos vecinos es *siempre* una influencia que incrementa el riesgo de fuga, mientras que la actividad mayormente con otros vecinos no fugados favorece *siempre* la disminución del riesgo.

Capítulo 5

Resultados y Discusión

5.1. Clasificación supervisada en el CCP.

La hipótesis de esta investigación (ver sección 1.3) hace referencia a la posibilidad de predecir la fuga de clientes desde una compañía telefónica (Virgin Mobile, en este caso) mediante modelos predictivos basados en datos. Estos datos registran la actividad del cliente en el consumo del servicio telefónico, la relación con otros clientes de la compañía y fuera de ella, así como sus atributos o perfiles. Con el objeto de validar esta hipótesis, en esta investigación se analizaron bases de datos de la compañía Virgin Mobile correspondientes a los perfiles de todos los clientes de la compañía en el transcurso de 4 años (alrededor de 1.200.000 clientes), en conjunto con los CDR de tres meses consecutivos (alrededor de 14.000.000 de llamadas por mes). Este gran volumen de datos es requerido para detectar regularidades y patrones más allá de la aparente aleatoriedad en el comportamiento y características de un consumidor individual. Es interesante constatar, por ejemplo, la notable regularidad en el número y periodicidad de las llamadas que hacen los clientes de la compañía durante las semanas del estudio (ver figuras 2.9 - 2.11). Con escalas tan grandes en los volúmenes de datos, es posible observar patrones regulares de comportamiento de los clientes, de tal modo que incluso se podría identificar el día de la semana, feriados, fines de semana, etc. observando tan solo la periodicidad de las llamadas hechas por los clientes.

Los clasificadores lineales solo pueden dividir su espacio de entrada en regiones muy simples, es decir, semi - espacios separados por un hiperplano. Los clasificadores no lineales permiten en cambio, separar espacios más intricados. Pero, problemas tales como descubrir/analizar patrones subyacentes en los registros de llamadas telefónicas, en la cantidad, duración y origen/destino de estas llamadas requieren que la función de entrada-salida no sea sensible a las variaciones irrelevantes de la entrada, por ejemplo, las variaciones individuales, los casos atípicos, circunstancias temporales específicas, etc. Al mismo tiempo requiere que esta función sea capaz de captar regularidades sutiles, globales y significativas en los comportamientos de los clientes fugados y los no-fugados. Esta es la razón por la cual los clasificadores superficiales (para diferenciarlos de los clasificadores basados en aprendizaje profundo) requieren un buen extractor de características que resuelva el dilema de selectividad-invariancia. Es decir, uno que produzca representaciones selectivas que discriminen diferentes comportamientos de los clientes, pero que sean invariantes a aspectos irrelevantes, tales como las variaciones individuales entre clientes que pertenezcan a una misma clase. Este hecho explica el porque de las dificultades encontradas en esta investigación para obtener un clasificador con buena sensibilidad en la detección de la clase minoritaria y por otro lado, el mejor desempeño del clasificador DL empleado en la sección 3.4.2.

Esta investigación tuvo como objetivo principal la resolución del problema de fuga de clientes o CCP desde compañías de telefonía. Más exactamente, la búsqueda de modelos predictivos para el cálculo de la probabilidad de fuga, dado el valor observado de ciertas variables predictoras que caracterizan el comportamiento del cliente de telefonía móvil. Para ello, se utilizó una

metodología cercana a las empleadas en proyectos de ciencias de datos con las siguientes etapas:

1. Recopilación y limpieza de los datos

2. Definición de las variables predictoras y cálculo de sus valores a partir de los datos disponibles (perfiles de clientes y CDRs mensuales). Las variables predictoras se dividieron en dos grandes grupos: los *atributos fijos* es decir aquellos que no varían a lo largo del tiempo (Canales de adquisición del servicio, compañías de origen, si el cliente portó su número al ingresar a la compañía, etc.) y las *variables de red* que permiten caracterizar la "ego-red"del cliente (número de vecinos o grado del nodo/cliente, cantidad de llamadas realizadas y recibidas, duración de las llamadas, etc.).

3. Análisis Exploratorio de los Datos o EDA. Este análisis permitió encontrar de manera temprana algunos patrones en el comportamiento de los clientes. En términos generales con el uso de EDA sobre los atributos fijos y de perfil, se descubrió que aquellos clientes que no se fugaban tenían una antigüedad promedio de 1.42 años, mientras que los clientes fugados presentaban una antigüedad promedio menor al año (fig. 2.5). La mayor cantidad de clientes portados hacia Virgin Mobile provenían de la compañía Entel (fig. 2.2) y que gran parte de los clientes fugados se dirigían principalmente a la compañía WOM (fig. 2.3). En el caso de las variables de red, tal y como muestra el gráfico de Coordenadas Paralelas (fig. 2.8), las variables que logran separar de mejor manera las características de los clientes fugados respecto de los no fugados son las relacionadas con la duración y la cantidad de llamadas. Los clientes que contaban con mayor número de vecinos y a su vez, con mayor cantidad de llamadas realizadas y recibidas, tendían a fugarse menos que aquellos clientes que contaban con menor actividad. Por otro lado, se encontraron patrones de periodicidad reconocibles en cada mes (fig. 2.9 - 2.11). Este echo indicó regularidades en el comportamiento que apoyaban la hipótesis de que estos patrones cuantificables y medibles podrían ser usados en la resolución del CCP.

4. Construcción y entrenamiento de clasificadores para el descubrimiento de patrones subyacentes en los datos y su uso en la resolución del CCP. En una primera instancia se probó con una batería de diferentes métodos y algoritmos implementados en R a través de múltiples paquetes. El desafío fundamental consistió en el manejo de grandes volúmenes de datos, lo que requirió de gran capacidad de cálculo y computación de alto rendimiento. Los clasificadores seleccionados para esta primera etapa fueron: Modelo Lineal Generalizado (GLM), Análisis Discriminante Lineal (LDA), K-Vecinos más Cercanos (KNN), Árboles de Clasificación y Regresión (CART) y Bosque Aleatorio (RF). Estos modelos fueron entrenados con el 70 % de los datos disponibles para cada mes y testeados sobre una muestra aleatoria correspondiente al otro 30 % de las observaciones. Una de las dificultades en este tipo de clasificación es la poca representatividad de la clase minoritaria, en este caso "fugados". En la base de datos de la compañía este desbalance entre las clases "fugados", "no fugados" se hizo más notorio en el mes de octubre, donde la clase minoritaria alcanzaba cerca de un 3 % del total de los datos. Los modelos fueron entrenados en 5 escenarios diferentes, en los que se consideraron distintas combinaciones de variables y la utilización de diferentes técnicas (Up Sample, SMOTE y ROSE) para balancear las clases "fugado" y "no fugado". De este estudio se obtuvo que RF y KNN son los métodos que entregan el mejor desempeño. Si bien estos modelos produjeron predicciones del orden del 85 % de acierto, la sensibilidad por lo general era baja. Así, en una segunda etapa de análisis se buscaron otros clasificadores con mejores desempeños.

   La segunda etapa consistió en aplicar los clasificadores Support Vector Machine (SVM) y Deep Learning (DL). Los resultados obtenidos con el uso de SVM con kernel gaussiano mejoraron la exactitud (en muchos casos superior al 90 % de acierto). Sin embargo, para incrementar la sensibilidad se necesitó de un ajuste intenso de los parámetros del modelo

y grandes dificultades para evitar el "overfit". Por otro lado, el desempeño mostrado por DL fue notoriamente mejor que el resto de los modelos estudiados. En general, las medidas de desempeño mostraron valores de sensibilidad por sobre el 70 % . La determinación de los parámetros óptimos del SVM y RF permitieron obtener mejoras considerables de estos modelos, sin embargo no superaron el desempeño del DL. Por otro lado, el ajuste del DL es sumamente complejo, ya que cuenta con más de 25 parámetros diferentes. Junto a esto, se suma el hecho de que este método es una "caja negra" que dificulta la interpretación de las predicciones. Por ende, la decisión respecto de qué parámetros utilizar y cómo optimizarlos es compleja y requiere de un estudio más detallado.

La tercera etapa fue aplicar técnicas de combinación de modelos predictivos. Los resultados obtenidos demuestran que, si bien la exactitud y la especificidad de los modelos es muy buena en todos los meses, los valores asociados a la sensibilidad y la precisión no son óptimos, sobretodo en el mes de octubre. Es importante destacar que, al igual que el resto de los clasificadores, la combinación de modelos predictivos fue probada en diferentes escenarios, los cuales incluían la utilización de SMOTE. A pesar de la aplicación de dicha técnica de balance de datos, los resultados en la sensibilidad para el mes de octubre en su mayoría no supera el 15 %. Lo mismo ocurre con Kappa que, en el mismo mes, no sobrepasa el 20 % (salvo ciertas excepciones) y la precisión, que si bien entrega algunos resultados buenos, no genera respuestas homogéneas en cuanto a su comportamiento. Los mejores modelos en esta sección fueron C5.0 y GBM.

5. Estudio de variables predictoras relevantes. Con el empleo de diferentes métodos (Eliminación Recursiva de Características, Splines de Regresión Adaptativa Multivariante, RF y Boruta) se procedió a reducir el número de variables del modelo de clasificación, buscando aquellas que son más importantes o que permitieran una mejor clasificación. Si bien estos métodos, basados en distintos algoritmos, tenían ciertas diferencias en sus elecciones de variables importantes y además, su aplicación a distintos intervalos de tiempo (meses) producía también algunas diferencias, por lo general tenían en común las variables relacionadas con el número de vecinos, la antigüedad, las llamadas hacia/desde fugados y el número de llamadas. Estas variables destacan precisamente tres aspectos de relevancia en este estudio: a) la actividad/inactividad del cliente en el consumo del servicio, b) la antigüedad (variable relacionada con la fidelidad a la compañía) y c) el posible "contagio" o influencia producido por otros clientes fugados.

La metodología descrita anteriormente busca resolver el CCP, llevándolo a un problema de clasificación supervisada. Después del análisis con estos clasificadores se cuenta con modelos ajustados ya entrenados y con desempeños razonablemente buenos para predecir probabilidades de fuga. En principio, se requiere analizar uno o varios meses de comportamiento del cliente junto a algunos de sus atributos y sobre esta base es posible dar una probabilidad condicionada a estos valores, sobre su futura permanencia o no en la compañía. Más aún, es posible distinguir qué variables/indicadores son más relevantes y por tanto donde la compañía requiere poner mayor atención si desea conservar al cliente. Una importante conclusión que se deriva del hecho de tener estos clasificadores disponibles, es que efectivamente existen patrones detectables de actividad, influencia y fidelidad del cliente que pueden hacer la diferencia entre la decisión de fugarse o mantenerse en la compañía.

5.2. El análisis de supervivencia en el CCP

Otra manera de abordar el CCP es modelando el tiempo de supervivencia del cliente en la compañía. Más exactamente, construyendo un modelo para el potencial de fuga en un instante de tiempo dado, por unidad de tiempo y condicionado a que se ha sobrevivido hasta ese instante. Este potencial o función de riesgo fue analizado en esta investigación mediante modelos no

paramétricos (estimador de Kaplan-Meier) y semi-paramétricos (modelo proporcional de Cox), considerando dos opciones para el tiempo de seguimiento. En la primera, tomamos la antigüedad medida en años como el tiempo de supervivencia o de "falla" y en la segunda variante tomamos el inicio del tiempo de seguimiento a partir del primer día del mes con datos de actividad registrados en la base de datos (1 agosto de 2016). A partir de este análisis encontramos que la mediana de la antigüedad de los clientes de la compañía Virgin Mobile es de 0.737 años. Los clientes cuyo canal de entrada es a través de la Web Virgin o que portan sus números a la compañía tienen menor tasa de fuga. El número de vecinos fugados favorece el incremento del riesgo de fuga, mientras que el número de vecinos en general contribuye a disminuir la probabilidad de fuga. Una mayor actividad del cliente medida por el número total de llamadas contribuye a disminuir el potencial de fuga. Por otro lado, si esta actividad está relacionada con clientes fugados, entonces el riesgo crece. Los clientes más antiguos tienen menor riesgo de fuga.

El empleo de modelos no - paramétricos no permite determinar el efecto de las variables predictoras sobre la tasa o probabilidad de fuga. En el caso de los modelos semi-paramétricos como el de Cox, por otra parte, se considera que el riesgo relativo (dependiente de las variables predictoras) es constante e igual para todos los individuos y a través del tiempo. Ésta es una restricción/limitante fuerte en general de este modelo. Para considerar los efectos temporales (no constantes) de las variables predictoras sobre el riesgo, a través del tiempo de seguimiento, se usó el modelo aditivo de Aalen. Este modelo se aplicó usando valores de las variables predictoras calculados para un mes en particular y se estimó su efecto a través del tiempo en el propio mes y más adelante. Se pudo constatar, sin embargo, que las variaciones o efectos dinámicos no son apreciables. Por otro lado, se confirmaron resultados de modelo previos, pero esta vez con el conocimiento adicional de que el efecto es realmente constante en todo el tiempo de seguimiento. En particular, el "contagio" o influencia de los fugados y la interacción con estos vecinos es *siempre* una influencia que incrementa el riesgo de fuga, mientras que la actividad mayormente con otros vecinos no fugados favorece *siempre* la disminución del riesgo.

A partir de los modelos de supervivencia construidos es posible predecir la probabilidad de fuga en función del tiempo de observación. Es decir, dado un cliente con ciertos atributos y cierto consumo en un instante de tiempo dado, medido a partir de un punto de referencia bien definido es posible determinar su riesgo de fuga. Esto es un instrumento fundamental para la compañía que podría contar con un indicador de riesgo para cada cliente a partir del cual puede tomar decisiones.

5.3. Limitaciones y trabajo futuro

En esta investigación se utilizaron bases de datos correspondientes a los atributos fijos de los clientes (Customer Profile) en conjunto con los CDRs, es decir la información detallada del servicio de telefonía (llamadas, sesiones de datos, SMS). Si bien los resultados, con esta información entregada por la compañía y discutidos en las secciones anteriores, ofrecen resultados razonablemente buenos en la predicción de fuga, la inclusión de mayor información y otros tipos de datos que son registrados de manera regular por la compañía podría eventualmente mejorar aún más el desempeño de los clasificadores.

Estas otras bases de datos detallan el consumo del cliente, es decir, contienen información sobre los productos adquiridos (la compañía los denomina bolsas y anti-planes), cargas, re-cargas, minutos, cantidad de megas/gigas de datos, entre otra información que podría ser relevante (la compañía denomina Recharge a esta base datos), junto con la información de Facturación de Ingreso (datos del consumo diario por cliente y por producto valorada en dinero). Este detalle de consumo, en combinación con otros métodos estadísticos avanzados[1] permitiría además

[1] R proporciona varios paquetes para el análisis del comportamiento de clientes, uno de los más importantes es el **BTYD** (Buy 'Til You Die) que utiliza registros históricos de transacciones para ajustar modelos probabilísticos con el objeto de describir y predecir el consumo de los clientes.

segmentar el conjunto de consumidores de acuerdo a diferentes métricas (Customer Lifetime Value - CLV, Customer Equity, P(alive), etc.) con el objeto de identificar clientes valiosos de la compañía y optimizar las posibles estrategias de retención desarrolladas por la compañía, reduciendo así los costos en la implementación de tales estrategias (por ejemplo, el costo del incentivo, el costo de contacto, etc.).

En gran parte de este trabajo, se utilizaron clasificadores superficiales y solo una pequeña parte de la investigación empleó herramientas de DL (sección 3.4.2) con mejores resultados. En un futuro trabajo, se requerirá el uso más intensivo de DL para obtener mejores clasificadores que capturen la relación compleja e intricada entre las variables predictoras y su capacidad para predecir la fuga. En este sentido, el trabajo se pretende realizar con herramientas "state-of-the-art" en DL, tales como: *Keras* y *TensorFlow*.

Otra de de las posibilidades para proporcionar predicciones más sensibles/precisas es usar herramientas del aprendizaje relacional en redes tales como: clasificadores relacionales y métodos de inferencia colectiva [76]. Estas herramientas permiten usar la existencia y peso de los enlaces en la red para inferir el comportamiento de un nodo/cliente basado en la similitud de éste con sus nodos vecinos. Hipotéticamente, un cliente sería más propenso a la fuga si posee "similitudes" con otro cliente fugado.

En el análisis de supervivencia se utilizó el modelo aditivo de Aalen 4.17, que considera que los coeficientes asociados a las variables predictoras pueden depender del tiempo. Una evidente generalización de este modelo es considerar que no solo los coeficientes dependen del tiempo, sino que también las variables predictivas $X = X(t)$ pueden depender del tiempo $h(t|X) = \beta_0(t) + \beta^T(t) \cdot X(t)$. Es decir, las variables pueden tomar diferentes valores y éstos cambian a lo largo del periodo de seguimiento. En este trabajo, solo empleamos valores fijos para estas variables, correspondientes a un mes en particular. La idea entonces, es utilizar intervalos de tiempos semanales para el calculo de los valores de las variables predictoras y estimar los efectos dinámicos a partir de este nuevo modelo [77].

Capítulo 6

Conclusiones

El problema de la fuga de clientes (CCP) desde distintas compañías ha sido investigado intensivamente en distintos rubros durante la última década. Este problema es particularmente relevante en telefonía móvil, donde su resolución es vital para que la compañía sobreviva y prospere en este mercado ferozmente competitivo. El interés de las compañías en este tema radica fundamentalmente en que es relativamente fácil para los clientes descontentos cambiar de proveedor, mientras que para la empresa, la adquisición de nuevos clientes es más costosa que conservar a los actuales. Más aún, los clientes felices pueden atraer a otros. Es por ello que las estrategias de retención y fidelización son cada vez más importantes dentro del conjunto de las estrategias de negocio aplicadas por las compañías de telefonía.

Para que las campañas de retención de clientes sean eficientes y efectivas un punto relevante es detectar potenciales clientes fugados, antes de que la fuga tenga lugar. Afortunadamente, las empresas de telefonías recopilan una gran cantidad de datos sobre sus clientes, incluyendo información demográfica, consumo de los servicios, registros detallados de llamadas, etc. Todo este volumen de datos debe usarse inteligentemente para extraer el conocimiento necesario y abordar el CCP. La hipótesis planteada, es que a partir de estos datos se puede predecir las eventuales fugas de clientes. El reconocer los patrones subyacentes en esta abundancia de datos y poder evaluar el potencial de fuga es el tema que fue tratado en este trabajo. Efectivamente, la presente investigación abordó el CCP a través de dos enfoques: el primero de ellos fue la utilización de diferentes modelos predictivos de la minería de datos para la clasificación de clientes y la segunda, el modelado del CCP utilizando análisis de supervivencia ("hazard analysis").

Los modelos predictivos construidos en este trabajo, entregan desempeños relativamente buenos al predecir la fuga. Entre estos modelos destaca el clasificador basado en la metodología *Deep Learning*, el que en la práctica ha generado los mejores resultados. Ésto, debido fundamentalmente a que la compleja interrelación entre las variables predictoras es quizás mejor reconocida por una representación multicapas. La transformación de los datos de entrada en tan solo uno o dos espacios de representación, con el uso de clasificadores "superficiales" requiere de un trabajo previo arduo, por ejemplo, para hallar las variables más importantes o relevantes, para balancear las clases, disminuir la dimensionalidad, etc. Todo este trabajo de *ingeniería de características* se realizó en esta investigación, con el objetivo adicional de comprender la naturaleza del problema de la fuga a partir de variables predictoras asociadas a los atributos/perfiles de clientes y datos de redes basados en los CDRs. La metodología DL, en cambio, automatiza completamente este paso simplificado los flujos de trabajo, pero a costa del ajuste con un gran número de parámetros (más de 20) y de cierta perdida de la compresión del problema modelado.

Ligado al uso de clasificadores superficiales se utilizaron diferentes técnicas para identificar algunas variables relevantes en la clasificación. Éstas son: el Número de Vecinos, la Antigüedad en la compañía y las llamadas realizadas/recibidas con clientes fugados. Estás variables permiten diferenciar claramente tres instancias relevantes en el CCP: 1) La actividad del cliente (diferenciada entre clientes potenciales fugados y no-fugados) registrada a partir del número de

vecinos que posee el cliente, 2) la fidelidad del cliente a partir de su antigüedad y 3) los efectos de contagio a partir de su interacción con clientes fugados. En este último punto, los resultados obtenidos permiten indicar que el efecto de contagio es plenamente observable y cuantificable. De acuerdo a lo obtenido en este estudio, aquellos clientes clasificados como fugados tienden a tener mayor exposición a otros clientes fugados.

Mediante el análisis de supervivencia se calculó la función de riesgo de fuga, más exactamente, la probabilidad de que, dado que un cliente ha sobrevivido hasta el tiempo t en la compañía, se fuga en el siguiente intervalo de tiempo, dividido por la longitud de ese intervalo. Empleando esta función, se reafirma el resultado anterior con mayor claridad, concluyendo que los clientes que tienen mayor contacto con otros fugados, aumentan su riesgo de fuga.

Este análisis permitió además mostrar que una mayor actividad/consumo del cliente (por ejemplo, a partir de la variable "TotalCalls") contribuye a la disminución de la probabilidad de fuga. Pero, por otro lado, si esta actividad está dierctamente vinculada con clientes que se fugan (por ejemplo, medida a partir de la variable "CallFromChurned"), entonces la función de riesgo crece. En general, el consumo activo del servicio con otros clientes dentro de la misma compañía (medido, por ejemplo, con la variable "NumVecinosVirgin"), disminuye la probabilidad de fuga. Algunos atributos fijos de los clientes, relacionados de alguna manera con la fidelidad, también fuero hallados relevantes mediante este estudio. Por ejemplo, los clientes que no portaron su número a Virgin, que no se encuentran suscritos a la página Web o que no adquirieron el servicio a través de convenios, tienen mayor riesgo de fuga. A partir de modelos Aditivos de Aalen, también se confirman algunos hallazgos realizados previamente, pero con la certeza de que los efectos sobre el riesgo de fuga producidos por las variables predictoras son constantes a través del tiempo. En particular, el contagio o influencia de los fugados y la interacción con ellos es *siempre* una influencia que incrementa el riesgo de fuga, mientras que la actividad mayormente con otros vecinos no fugados favorece *siempre* la disminución del riesgo.

Los resultados obtenidos con ambas metodologías junto a los modelos ya entrenados y disponibles a partir de este estudio pueden ser utilizados por la compañía para generar la capacidad de detectar a tiempo a clientes con mayor potencial de fuga, convirtiéndose así en una herramienta para la toma de decisiones. Estos métodos fueron capaces de reconocer patrones en los comportamientos de los clientes y definir, en base a sus características personales y de red, las probabilidades de fuga. Al contar con estos modelos predictivos robustos y eficientes, la compañía puede incorporar herramientas para: mejorar sus estados financieros (al retener a clientes valiosos dentro de la empresa); focalizar las estrategias de retención en los segmentos de clientes más riesgosos; y entregar mejores servicios a los clientes que se encuentran insatisfechos y que eventualmente podrían en breve tomar la decisión de desligarse de la compañía.

Bibliografía

[1] J. Lu. Predicting customer churn in the telecommunications industry — an application of survival analysis modeling using sas. *SAS User Group International (SUGI27) Online Proceedings*, pages 114–27, 2002.

[2] W. Verbeke, D. Martens, and B. Baesens. Social network analysis for customer churn prediction. *Applied Soft Computing*, 14:431–446, 2014.

[3] M. Óskarsdóttir, C. Bravo, W. Verbeke, C. Sarraute, B. Baesens, and J. Vanthienen. A comparative study of social network classifiers for predicting churn in the telecommunication industry. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 1151–1158. IEEE, 2016.

[4] W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens. New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European Journal of Operational Research*, 218(1):211–229, 2012.

[5] D. L. García, À. Nebot, and A. Vellido. Intelligent data analysis approaches to churn as a business problem: a survey. *Knowledge and Information Systems*, 51(3):719–774, 2017.

[6] J. Miranda, P. Rey, and R. Weber. Predicción de fugas de clientes para una institución financiera mediante support vector machines. *Revista Ingeniería de Sistemas Volumen XIX*, 2005.

[7] A. Backiel, B. Baesens, and G. Claeskens. Predicting time-to-churn of prepaid mobile telephone customers using social network analysis. *Journal of the Operational Research Society*, 67(9):0, 2016.

[8] A. Backiel, Y. Verbinnen, B. Baesens, and G. Claeskens. Combining local and social network classifiers to improve churn prediction. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 651–658. ACM, 2015.

[9] F. Castanedo, G. Valverde, J. Zaratiegui, and A. Vazquez. Using deep learning to predict customer churn in a mobile telecommunication network, 2014.

[10] S.Y. Hung, D. C Yen, and H.Y. Wang. Applying data mining to telecom churn management. *Expert Systems with Applications*, 31(3):515–524, 2006.

[11] Q. Han and P. Ferreira. Peer influence and subcriber churn in wireless networks. 2016.

[12] P. Datta, B. Masand, DR Mani, and B. Li. Automated cellular modeling and prediction on a large scale. *Artificial Intelligence Review*, 14(6):485–502, 2000.

[13] W.H. Au, K. CC Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE transactions on evolutionary computation*, 7(6):532–545, 2003.

[14] W. Buckinx and D. Van den Poel. Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual fmcg retail setting. *European Journal of Operational Research*, 164(1):252–268, 2005.

[15] G. Nie, W. Rowe, L. Zhang, Y. Tian, and Y. Shi. Credit card churn forecasting by logistic regression and decision tree. *Expert Systems with Applications*, 38(12):15273–15285, 2011.

[16] K. Coussement and D. Van den Poel. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert systems with applications*, 34(1):313–327, 2008.

[17] M. A. H. Farquad, V. Ravi, and S. B. Raju. Churn prediction using comprehensible support vector machine: An analytical crm application. *Applied Soft Computing*, 19:31–40, 2014.

[18] K. Dasgupta, R. Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. A Nanavati, and A. Joshi. Social ties and their relevance to churn in mobile telecom networks. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 668–677. ACM, 2008.

[19] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503, 2003.

[20] L. Weng, F. Menczer, and Y. Y. Ahn. Virality prediction and community structure in social networks. *Scientific reports*, 3:2522, 2013.

[21] Bernardo A Huberman, Daniel M Romero, and Fang Wu. Social networks that matter: Twitter under the microscope. *arXiv preprint arXiv:0812.1045*, 2008.

[22] Dunia López-Pintado. Diffusion in complex social networks. *Games and Economic Behavior*, 62(2):573–590, 2008.

[23] Nicholas A Christakis and James H Fowler. The collective dynamics of smoking in a large social network. *New England journal of medicine*, 358(21):2249–2258, 2008.

[24] M. Girvan and M. EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[25] B. Ball, B. Karrer, and M. EJ Newman. Efficient and principled method for detecting communities in networks. *Physical Review E*, 84(3):036103, 2011.

[26] C. Xia, S. Sun, F. Rao, J. Sun, J. Wang, and Z. Chen. Sis model of epidemic spreading on dynamical networks with community. *Frontiers of Computer Science in China*, 3(3):361–365, 2009.

[27] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.

[28] A. Grabowski and RA Kosiński. Epidemic spreading in a hierarchical social network. *Physical Review E*, 70(3):031908, 2004.

[29] Y.C. Chen, W.C. Peng, and S.Y. Lee. Efficient algorithms for influence maximization in social networks. *Knowledge and information systems*, 33(3):577–601, 2012.

[30] D. Chen, L. Lü, M.S. Shang, Y.C. Zhang, and T. Zhou. Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications*, 391(4):1777–1787, 2012.

[31] M. Kitsak, L. K Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888, 2010.

[32] P. Wang, C. Tian, and J. Lu. Identifying influential spreaders in artificial complex networks. *Journal of Systems Science and Complexity*, 27(4):650–665, 2014.

[33] M. McPherson, L. Smith-Lovin, and J.M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[34] S. Aral, L. Muchnik, and A. Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549, 2009.

[35] A. J Viera, J. M Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363, 2005.

[36] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[37] S. B Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[38] T. O. Ayodele. Types of machine learning algorithms. In *New advances in machine learning*. InTech, 2010.

[39] U. Olsson. Generalized linear models. *An applied approach. Studentlitteratur, Lund*, 18, 2002.

[40] A. Agresti. *Categorical data analysis*. John Wiley & Sons, 2013.

[41] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *Advances in neural information processing systems*, pages 1569–1576, 2005.

[42] Z. Qiao, L. Zhou, and J. Z. Huang. Effective linear discriminant analysis for high dimensional, low sample size data. In *Proceeding of the world congress on engineering*, volume 2, pages 2–4. Citeseer, 2008.

[43] J. M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4):580–585, 1985.

[44] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[45] P. Langley and W. Iba. Average-case analysis of a nearest neighbor algorithm. In *IJCAI*, pages 889–894. Citeseer, 1993.

[46] S. Jiang, G. Pang, M. Wu, and L. Kuang. An improved k-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, 39(1):1503–1509, 2012.

[47] W.Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[48] P. Breheny. Classification and regression trees. 1984.

[49] Breiman Leo, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. *Wadsworth International Group*, 1984.

[50] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[51] K.L. Du and M. NS Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013.

[52] A. D Kulkarni and B. Lowe. Random forest algorithm for land cover classification. 2016.

[53] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[54] B. Huang, M. T. Kechadi, and B. Buckley. Customer churn prediction in telecommunications. *Expert Systems with Applications*, 39(1):1414–1425, 2012.

[55] Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[56] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[57] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[58] M. Kubat, R. C Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[59] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme. A new evaluation measure for learning from imbalanced data. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 537–542. IEEE, 2011.

[60] Varsha S Babar and Roshani Ade. A review on imbalanced learning methods.

[61] N. Lunardon, G. Menardi, and N. Torelli. Rose: A package for binary imbalanced learning. *R Journal*, 6(1), 2014.

[62] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[63] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[64] B Yekkehkhany, A Safari, S Homayouni, and M Hasanlou. A comparison study of different kernel functions for svm-based classification of multi-temporal polarimetry sar data. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(2):281, 2014.

[65] M. Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

[66] N. Horning et al. Random forests: An algorithm for image classification and generation of continuous fields data sets. In *Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, Osaka, Japan*, volume 911, 2010.

[67] JR. Leathwick, J. Elith, and T. Hastie. Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions. *Ecological modelling*, 199(2):188–196, 2006.

[68] E. J. Wegman and I. W. Wright. Splines in statistics. *Journal of the American Statistical Association*, 78(382):351–365, 1983.

[69] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[70] Miron B Kursa, Witold R Rudnicki, et al. Feature selection with the boruta package. *J Stat Softw*, 36(11):1–13, 2010.

[71] David Roxbee Cox. *Analysis of survival data*. Routledge, 2018.

[72] David G Kleinbaum. Survival analysis, a self-learning text. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 40(1):107–108, 1998.

[73] D. R Cox and D. Oakes. Analysis of survival data. 1984. *Chapman&Hall, London*, 1984.

[74] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. *Survival and event history analysis: a process point of view*. Springer Science & Business Media, 2008.

[75] Odd O Aalen. A linear regression model for the analysis of life times. *Statistics in medicine*, 8(8):907–925, 1989.

[76] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*, volume 1. MIT press Cambridge, 2007.

[77] Scheike T. H. Martinussen T. *Dynamic Regression Models for Survival Data*. Springer Science & Business Media, 2006.

Apéndice A

Construcción y definición de variables predictoras.

A.1. Atributos fijos y perfil del consumidor.

Los atributos fijos de los clientes son determinados a partir del set de datos "Customer Profile" facilitado por Virgin Mobile. A continuación, el código utilizado para la construcción de variables fijas.

```
datos_profiles=read.csv('20161020 SNA Customer Profile.csv',header=T,sep=',',
                        dec='.')
```

Con las siguientes líneas aplicamos una limpieza preliminar a los datos. De esta manera es posible descartar posibles errores básicos en los datos.

```
datos1=subset(datos_profiles,!datos_profiles$fecha_portOut=="")
datos2=subset(datos_profiles,!datos_profiles$fecha_churn_mkt=="")
A=datos1[which(interval(ymd(datos1$activacion),
               ymd(datos1$fecha_portOut))/ddays(1)<=60
B=datos2[which(interval(ymd(datos2$activacion),
               ymd(datos2$fecha_churn_mkt))/ddays(1)<=90),]
C=datos2[which(interval(ymd(datos2$fecha_portOut),
               ymd(datos2$fecha_churn_mkt))/ddays(1)<=0),]
D=datos1[which(interval(ymd(datos1$fecha_portOut),
               ymd(datos1$fecha_churn_mkt))/ddays(1)<=0),]
datos3=subset(datos_profiles,!rownames(datos_profiles)%in%rownames(A))
datos4=subset(datos3,!rownames(datos3)%in%rownames(B))
datos5=subset(datos4,!rownames(datos4)%in%rownames(C))
datos6=subset(datos5,!rownames(datos5)%in%rownames(D))
H=datos6
```

A continuación eliminamos los ID repetidos. La empresa puede reciclar números de clientes que han sido daods de baja. De esta manera, suponemos que un ID se repite si tiene varias fechas de activación.

```
H1=count(H,ID)%>%setDT()
H=data.table(H)
H2=H[H1,on=c(ID='ID'),nomatch=0]
H3=filter(H2,n==1)
H3$n=NULL
H=H3
```

Consideramos que aquellos clientes que presentan una "fecha_portOut" o "fecha_churn_mkt" distinto de *vacío* son clientes que se han fugado. Por lo taanto, colocamos las etiquetas de fugados y no fugados de la siguiente manera:

```
p=!(H$fecha_portOut=='')
q=!(H$fecha_churn_mkt=='')
H4=mutate(H,StatusFugado=ifelse(p|q,'fugado','no_fugado'))
H=H4
```

Agregamos la variable tenure (antigüedad) medida en años. La última fecha considerada para hacer el cálculo es "2016-12-31". Para el caso de los clientes no fugados la antigüedad será el intervalo de tiempo entre la fecha de activación, mientras que para los clientes fugados, el tenure será el intervalo de tiempo medido entre la fecha de activación y la fecha_portOut/fecha_churn_mkt.

```
H1=as.data.table(H)
H2=H1[,.(t1=ifelse(StatusFugado=='no_fugado',interval(ymd(activacion),
                ymd("2016-12-31"))/dyears(1),interval(ymd(activacion),
                ymd(fecha_portOut))/dyears(1)))]
H1$t1=H2$t1
H2=H1[,.(t2=ifelse(is.na(t1),interval(ymd(activacion),
            ymd(fecha_churn_mkt))/dyears(1),t1))]
H$tenure=H2$t2
```

Reemplazamos los códigos de números por los nombres de las compañías en cia_origen y cia_destino y la colocamos en una nueva columna.

```
subs_f=function(t){
  t=gsub('219','Claro',t)
  t=gsub('220','Entel',t)
  t=gsub('215','Movistar',t)
  t=gsub('225','WOM',t)
  t=gsub('240','Falabella_Movil_Spa',t)
  t=gsub('235','VTR_Movil',t)
  t=gsub('242','Otros',t)
  t=gsub('212','Otros',t)
  t=gsub('210','Otros',t)
  t=gsub('216','Otros',t)
  t=gsub('214','Otros',t)
  t=gsub('315','Otros',t)
  t=gsub('316','Otros',t)
  t=gsub('317','Otros',t)
  t=gsub('319','Otros',t)
  t=gsub('322','Otros',t)
  t=gsub('323','Otros',t)
  t=gsub('326','Otros',t)
  t=gsub('331','Otros',t)
  t=gsub('320','Otros',t)
}
H$nombre_cia_origen=subs_f(H$cia_origen)
H$nombre_cia_destino=subs_f(H$cia_destino)
```

A.2. Atributos/variables de red.

Consideraremos el mes de agosto (el mismo procedimiento se utiliza para el resto de los meses). Primero determinamos a los fugados con fecha_portOut después del primero de agosto (si se fugaron antes de esta fecha, no tendrán actividad en este mes). A continuación, incluimos una condición que considera solo a los clientes que se activaron antes del 31 de agosto. La tercera línea incluye a los no fugados. De esta manera excluimos a los clientes no fugados que se activaron después del 31 de agosto.

```
a=(H$StatusFugado=='fugado')&(ymd(H$fecha_portOut)>=ymd('2016-08-01'))
b=(ymd(H$activacion)<=ymd('2016-08-31'))
c=(H$StatusFugado=='no_fugado')
H1=filter(H,(b&c)|(b&a))
```

```
H=H1
```

A continuación cargamos el CDR del mes de Agosto.

```
datos=read.csv('20161124 August2016 Calls CDR.csv',sep=',',dec='.', header=T)
```

Con esta base de datos podemos generar una visualización de los patrones de llamadas. Este gráfio es un indicador de que los usuarios en general mantienen patrones regulares de llamadas.

```
p=qplot(ymd(sessioncreationtimestamp), ..count.., data=datos, alpha=0.3,
              geom="density", fill=destinationzone)
p
ggplotly(p)
```

Hacemos los cálculos considerando una red no dirigida. El grafo g lo creamos de la siguiente manera:

```
el=matrix(c(as.character(datos$from),as.character(datos$to)),nc = 2,
              byrow = F)
g=graph_from_edgelist(el,directed=FALSE)
```

Le añadimos el atributo duración de la llamada a cada enlaces y comprobamos que efectivamente el grafo g se convirtió en un grafo con peso. Además, definimos las variables TotalCallTime y el NumVecinos, como el peso de los enlaces y el grado del nodo, respectivamente.

```
E(g)$weight=datos$weight
E(g)$destinationzone=as.character(datos$destinationzone)
M=as_adjacency_matrix(g,attr="weight")
g=graph_from_adjacency_matrix(M,mode='plus',weighted=TRUE)
h=as_ids(V(g))
(l1=length(h))
df=data.frame(ID=h)
df$TotalCallTime=graph.strength(g, mode="total")
df$NumVecinos=degree(g, v = V(g), mode = c("total"),loops = FALSE,
                     normalized = FALSE)
```

Con esto es posible identificar a quienes pertenecen a Virgin, incorporando la variable Status Virgin. Esto lo hacemos comprobando que el listado de ID del grafo g (almacenado en h) se encuentra en el listado de ID del set de datos H (almacenado en k). Aquellos casos en los que se cumple esta condición, se clasifican con la variable Virgin Status igual a 1.

```
k=H$ID
P=h %in% k
A=data.table(p=P,h=h)
f=function(t){ifelse(t,1,-1)}
C=A[,.(Virgin_status=f(p))]
df$Virgin_status=C$Virgin_status
X=data.table(df)
status=data.frame(ID=k,StatusFugado=H$StatusFugado) %>% setDT
gdt.e = igraph::as_data_frame(g, what = "edges") %>% setDT()
status_to = gdt.e[status, on = c(to = "ID"), nomatch = 0]
virgin_to=gdt.e[X, on = c(to = "ID"), nomatch = 0]
s0=filter(virgin_to,Virgin_status==1)
s01=count(s0,from) %>% setDT()
s1=filter(status_to,StatusFugado=='fugado')
s2=count(s1,from) %>% setDT()
names(s2)[2]='m'
Y=s2[X,on=c(from='ID'),nomatch=NA]
Y1=s01[Y,on=c(from='from'),nomatch=NA]
```

Definimos las variables TotalCallTime, NumVecinosFugados, NumVecinos y NumVecinosVirgin, reemplazando además los NA por ceros.

```
Z=Y1[,.(ID=from,TotalCallTime=ifelse(is.na(TotalCallTime),0,TotalCallTime),
              NumVecinosFugados=ifelse(is.na(m),0,m),
              NumVecinos=ifelse(is.na(NumVecinos),0,NumVecinos),
              NumVecinosVirgin=ifelse(is.na(n),0,n),Virgin_status)]
H1=data.table(H)
H1=H1[Z,on=c(ID='ID'),nomatch=0]
```

Ahora realizamos el cálculo considerando una red dirigida. Construimos una red dirigida usando una lista de enlaces que está contenida en la matriz de dos columnas "el". La red se guarda en el grafo g.

```
el=matrix(c(as.character(datos$from),as.character(datos$to)),nc = 2, byrow = F)
g=graph_from_edgelist(el)
E(g)$weight=datos$weight
h=as_ids(V(g))
```

Agregamos a df, anteriormente definido, las columnas correspondientes a la duración y cantidad de llamadas, tanto realizadas como recibidas.

```
df$CallTime_in<- graph.strength(g, mode="in")
df$CallTime_out<- graph.strength(g, mode="out")
df$Call_out=degree(g,v=V(g),mode=c("out"),loops=FALSE,normalized=FALSE)
df$Call_in=degree(g,v=V(g),mode=c("in"),loops=FALSE,normalized=FALSE)
df$TotalCalls=degree(g,v=V(g),mode=c("total"),loops=FALSE,normalized=FALSE)
E(g)$traffictype=as.character(datos$traffictype)
E(g)$destinationzone=as.character(datos$destinationzone)
E(g)$sessioncreationtimestamp=as.character(datos$sessioncreationtimestamp)
```

Considerando las características de red (atributos de los nodos y de los enlaces), es posible crear nuevas variables.

```
g_sub=subgraph.edges(g, E(g)[E(g)$destinationzone=='(on-net)'],
                     delete.vertices = FALSE)
df$CallTimeOutOnNet<-graph.strength(g_sub, mode="out")
df$CallOutOnNet=degree(g_sub,v=V(g_sub),mode=c("out"),loops=FALSE,normalized=FALSE
    )
df$CallTimeIntOnNet=graph.strength(g_sub, mode="in")
df$CallInOnNet=degree(g_sub,v=V(g_sub),mode=c("in"),loops=FALSE,normalized=FALSE)

X=data.table(df)
H=H1[X,on=c(ID='ID'),nomatch=0]

write.csv(H,'atributosParcialesAgosto.csv',row.names=F)
```

Apéndice B

Análisis Exploratorio de los Datos (EDA)

```
H2=read.csv('atributosParcialesAgosto.csv',header=T,dec='.',sep=',')
H2=data.table(H2)

f=function(t){ifelse(is.na(t),0,t)}
Q$NumVecinosVirgin=sapply(Q$NumVecinosVirgin,f)
Q$NumVecinos=sapply(Q$NumVecinos,f)
Q$NumVecinosFugados=sapply(Q$NumVecinosFugados,f)
Q$CallFromChurned=sapply(Q$CallFromChurned,f)
Q$CallTimeFromChurned=sapply(Q$CallTimeFromChurned,f)
Q$CallToChurned=sapply(Q$CallToChurned,f)
Q$CallTimeToChurned=sapply(Q$CallTimeToChurned,f)
Q$tenure=sapply(Q$tenure,f)
Q$TotalCallTime=sapply(Q$TotalCallTime,f)
Q$CallTime_in=sapply(Q$CallTime_in,f)
Q$CallTime_out=sapply(Q$CallTime_out,f)
Q$Call_out=sapply(Q$Call_out,f)
Q$Call_in=sapply(Q$Call_in,f)
Q$TotalCalls=sapply(Q$TotalCalls,f)
Q$CallTimeOutOnNet=sapply(Q$CallTimeOutOnNet,f)
Q$CallOutOnNet=sapply(Q$CallOutOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallInOnNet=sapply(Q$CallInOnNet,f)
```

Gráfico de correlaciones:

```
corr <- round(cor(Q[,-1]), 1)
ggcorrplot(corr, hc.order = TRUE, type = "lower", lab = TRUE, lab_size = 3,
           method="circle",colors = c("tomato2", "white", "springgreen3"),
           title="Correlograma de Q", ggtheme=theme_bw)
```

Compañía De Origen. Datos simulados de acuerdo con la distribución de probabilidades:

```
(m=nrow(H2))
(PClaro=length(which(H2$nombre_cia_origen=='Claro'))/m)
(PEntel=length(which(H2$nombre_cia_origen=='Entel'))/m)
(PFalabella_Movil_Spa=length(which(H2$nombre_cia_origen=='Falabella'))/m)
(PMovistar=length(which(H2$nombre_cia_origen=='Movistar'))/m)
(POtros=length(which(H2$nombre_cia_origen=='Otros'))/m)
(PVTR_Movil=length(which(H2$nombre_cia_origen=='VTR'))/m)
(PWOM=length(which(H2$nombre_cia_origen=='WOM'))/m)
```

Función que permite generar un valor simulado donde haya un NA de acuerdo a las probabilidades calculadas arriba:

```
f_origen_simul=function(i){ifelse(is.na(H2$nombre_cia_origen[i]),sample(c('Claro',
                                'Entel','Falabella','Movistar','Otros','VTR','WOM'),1,
                                replace=TRUE,prob=c(PClaro,PEntel,PFalabella_Movil_Spa,
                                PMovistar,POtros,PVTR_Movil,PWOM)),
                                as.character(y$nombre_cia_origen[i]))}
Q$nombre_cia_origen=sapply((1:m),f_origen_simul)
```

El mismo procedimiento se utiliza para la variable Canal:

```
(m=nrow(H2))
(PKiosco_Portabilidad=length(which(H2$Canal=='Kiosco Portabilidad'))/m)
(PMasivo_y_Mayorista=length(which(H2$Canal=='Mayorista'))/m)
(PRetail=length(which(H2$Canal=='Retail'))/m)
(PConvenios=length(which(H2$Canal=='Convenios'))/m)
(POtros=length(which(H2$Canal=='Otros'))/m)
(PWEB_Virgin=length(which(H2$Canal=='WEB Virgin'))/m)
f_canal_simul=function(i){ifelse(H2$Canal[i]=='',sample(c('Kiosco Portabilidad',
                                'Mayorista','Retail','Convenios','Otros','WEB Virgin'),
                                1,replace=TRUE,prob=c(PKiosco_Portabilidad,
                                PMasivo_y_Mayorista,PRetail,PConvenios,POtros,
                                PWEB_Virgin)), as.character(y$Canal[i]))}
Q$Canal=sapply((1:m),f_canal_simul)
```

Gráficas para la Compañía De Origen:

```
y=Q
orig=ggplot(data = y) +
  geom_bar(mapping = aes(x = nombre_cia_origen, fill = nombre_cia_origen))+
  ggtitle("Distribucion cia de origen")+
  labs(x = "Cia de origen",y = "Cantidad de clientes",fill="Cia")+
  guides(fill=FALSE)
orig

ggplot(data = y) +
  geom_bar(mapping = aes(x = nombre_cia_origen, fill = StatusFugado),
  position = "dodge")+
  ggtitle("Distribucion cia de origen, separado por status")+
  labs(x = "Cia de origen",y = "Cantidad de clientes",fill="Estado")

ggplot(y, mapping = aes(x = activacion ,fill=nombre_cia_origen)) +
  geom_density(alpha=0.5) +
  ggtitle("Distribucion cia de origen, por fecha de activacion")+
  labs(x = "fecha de activacion",y = "Proporcion")+
  scale_x_date(date_breaks = "6 months",date_labels = "%b %y")
```

Gráficas para la Compañía De Destino:

```
dest=ggplot(data = subset(y,!is.na(nombre_cia_destino))) +
  geom_bar(mapping = aes(x = nombre_cia_destino, fill = nombre_cia_destino))+
  ggtitle("Distribucion cia de destino")+
  labs(x = "Cia de destino",y = "Cantidad de clientes",fill="cia")+
  guides(fill=FALSE)
dest
```

Gráficas para el Canal de adquisición:

```
ggplot(data = y) +
  geom_bar(mapping = aes(x = Canal, fill = Canal))+
  ggtitle("Distribucion canal de adquisicion")+
  labs(x = "Canal de adquisicion",y = "Cantidad de clientes",fill="Canal")+
  guides(fill=FALSE)

ggplot(data = y) +
```

```
  geom_bar(mapping = aes(x = Canal, fill = StatusFugado),position = "dodge")+
  ggtitle("Distribucion canal de adquisicion, separado por status")+
  labs(x = "Canal de adquisicion",y = "Cantidad de clientes",fill="Estado")
```

Gráficas para la antigüedad:

```
ggplot(y, aes(x=StatusFugado, y=tenure, fill=StatusFugado))+
  geom_boxplot()+
  ggtitle("Boxplot de tenure separado por status")+
  labs(x = "Estado",y = "Tenure",colour="Estado")+
  guides(fill=FALSE)+ coord_flip()
```

Creamos un Diagrama de Cuerdas, para analiza la proveniencia y el destino del flujo de clientes que pasan por Virgin Mobile.

```
df=subset(y,y$nombre_cia_destino!="")
origen=as.character(df$nombre_cia_origen)
destino=as.character(df$nombre_cia_destino)
df=cbind(origen,destino)
mat=as.matrix(get.adjacency(graph.data.frame(df)))
circos.par(start.degree = 90, clock.wise = FALSE)
grid.col=c("deeppink2","orange","turquoise3","goldenrod4","navyblue",
           "mediumorchid3","green4")
chordDiagram(mat,grid.col = grid.col,order = c("Claro","VTR","WOM","Otros",
             "Entel","Falabella","Movistar"),annotationTrack = "grid",
             preAllocateTracks = list(track.height = uh(5, "mm")),
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -uh(2, "mm"))
for(si in get.all.sector.index()) {
  circos.axis(h = "top", labels.cex = 1.5, sector.index = si, track.index = 2)
}

circos.track(track.index = 1, panel.fun = function(x, y) {
  xlim = get.cell.meta.data("xlim")
  ylim = get.cell.meta.data("ylim")
  sector.name = get.cell.meta.data("sector.index")
  xplot = get.cell.meta.data("xplot")

  circos.lines(xlim, c(mean(ylim), mean(ylim)), lty = 3) # dotted line
  by = ifelse(abs(xplot[2] - xplot[1]) > 30, 0.2, 0.5)
  for(p in seq(by, 1, by = by)) {
    circos.text(p*(xlim[2] - xlim[1]) + xlim[1], mean(ylim) + 0.1,
                paste0(p*100, "%"), cex = 1, adj = c(0.5, 0), niceFacing = TRUE)
  }

  circos.text(mean(xlim), 1, cex=3,sector.name, niceFacing = TRUE,
  adj = c(0.5, 0))}, bg.border = NA)
```

Por último, realizamos un Gráfico de Coordenadas Paralelas, para observar el comportamiento de las variables de red, separadas por Status del cliente.

```
ggparcoord(Q,columns = 2:19,groupColumn="StatusFugado")
```

B.1. Algunas gráficas

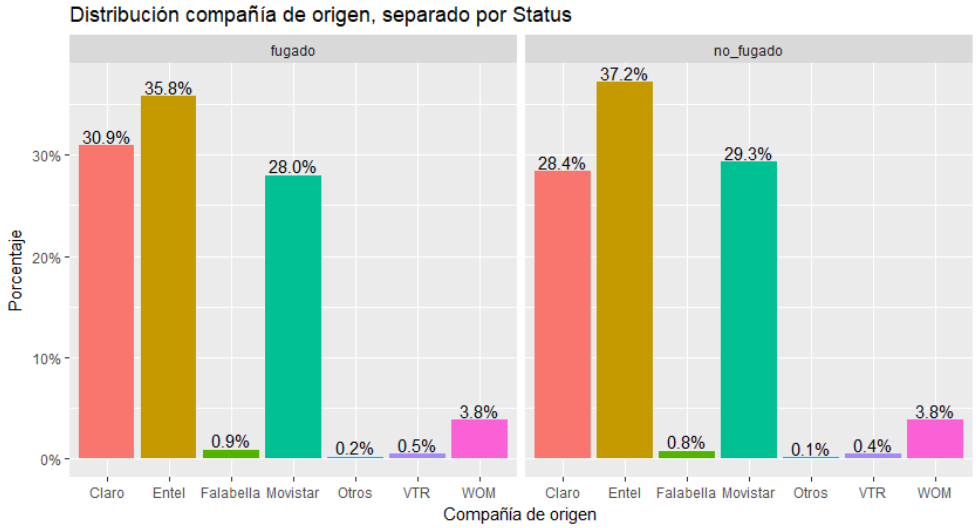


Figura B.1: Compañía de origen de clientes portados a Virgin, separado por Status. En esta gráfica es posible observar la distribución de las compañías de origen, diferenciando si el cliente es fugado o no fugado. La distribución es similar en ambos casos.

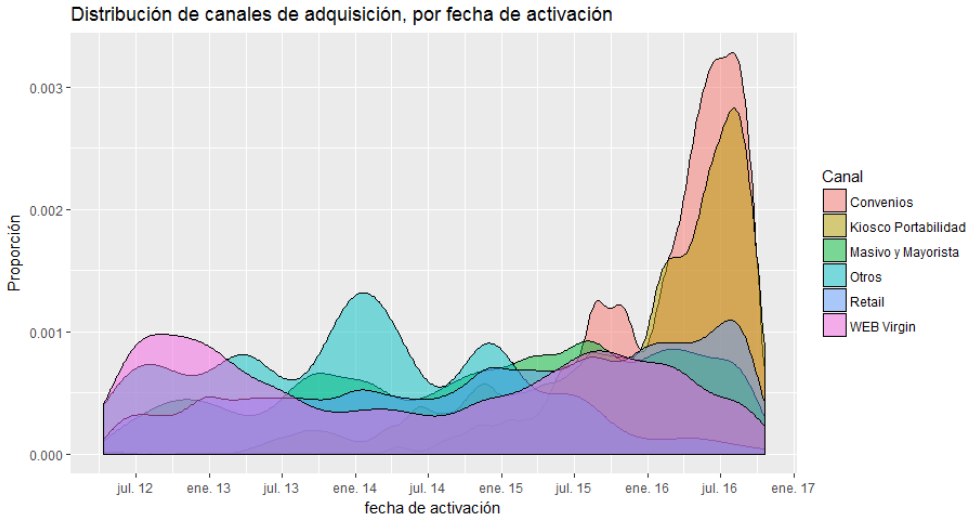


Figura B.2: Distribución de los canales de adquisición respecto de la fecha de activación. La figura muestra las preferencias de los clientes al momento de adquirir el servicio de Virgin a lo largo de los últimos años. En el último año de estudio las principales formas de adquisición son a través de convenios y kioskos de portabilidad.

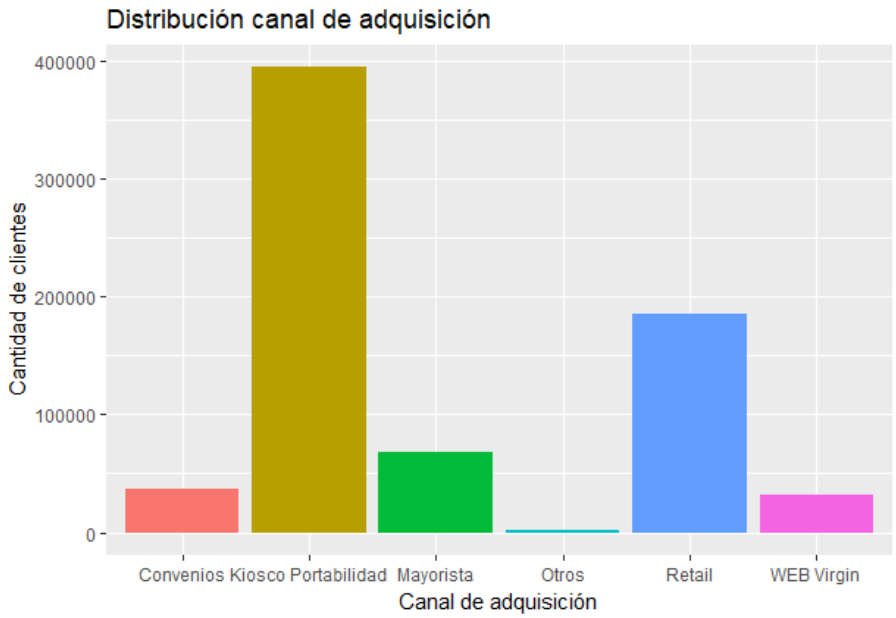


Figura B.3: Distribución canal de adquisición. La figura muestra las preferencias de los clientes al momento de adquirir el servicio de Virgin. Más de 384.088 clientes lo hacen a través de Kioskos de portabilidad, seguido de medios de venta masivos.

Apéndice C

Clasificadores

C.1. Comparando algoritmos: LDA, GLM, GLMNET, CART, KNN, RF

A continuación se muestra el código utilizado para estudiar los algoritmos utilizando todas las variables, sin balance de datos. El resto de los escenarios estudiados son variantes de este código, en el que se aplica SMOTE, y se reduce o aumenta el número de variables utilizadas.

Todos estos códigos fueron ejecutados en el HPC-UDD-Sofía, con el objetivo de obtener resultados de manera más rápida, utilizando la totalidad de los datos disponibles para el estudio.

Cargamos los datos y, al igual que en los algoritmos presentados con anterioridad, evitamos la presencia de NA's.

```
H2=read.csv('atributos_agosto.csv',sep=',',dec='.', header=T)
H2$Canal=as.character(H2$Canal)
H2$Canal[H2$Canal == "Empresas"] <- ""
H2$Canal=as.factor(H2$Canal)
Q=select(H2,NumVecinos,NumVecinosVirgin,NumVecinosFugados,CallFromChurned,
         CallTimeFromChurned,CallToChurned,CallTimeToChurned,tenure,TotalCallTime,
         CallTime_in,CallTime_out,Call_out,Call_in,TotalCalls,CallTimeOutOnNet,
         CallOutOnNet,CallTimeIntOnNet,CallInOnNet)
f=function(t){ifelse(is.na(t),0,t)}
Q$NumVecinosVirgin=sapply(Q$NumVecinosVirgin,f)
Q$NumVecinos=sapply(Q$NumVecinos,f)
Q$NumVecinosFugados=sapply(Q$NumVecinosFugados,f)
Q$CallFromChurned=sapply(Q$CallFromChurned,f)
Q$CallTimeFromChurned=sapply(Q$CallTimeFromChurned,f)
Q$CallToChurned=sapply(Q$CallToChurned,f)
Q$CallTimeToChurned=sapply(Q$CallTimeToChurned,f)
Q$tenure=sapply(Q$tenure,f)
Q$TotalCallTime=sapply(Q$TotalCallTime,f)
Q$CallTime_in=sapply(Q$CallTime_in,f)
Q$CallTime_out=sapply(Q$CallTime_out,f)
Q$Call_out=sapply(Q$Call_out,f)
Q$Call_in=sapply(Q$Call_in,f)
Q$TotalCalls=sapply(Q$TotalCalls,f)
Q$CallTimeOutOnNet=sapply(Q$CallTimeOutOnNet,f)
Q$CallOutOnNet=sapply(Q$CallOutOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallInOnNet=sapply(Q$CallInOnNet,f)
m=nrow(H2)
PClaro=length(which(H2$nombre_cia_origen=='Claro'))/m
PEntel=length(which(H2$nombre_cia_origen=='Entel'))/m
PFalabella_Movil_Spa=length(which(H2$nombre_cia_origen=='Falabella_Movil_Spa'))/m
PMovistar=length(which(H2$nombre_cia_origen=='Movistar'))/m
POtros=length(which(H2$nombre_cia_origen=='Otros'))/m
```

```
PVTR_Movil=length(which(H2$nombre_cia_origen=='VTR_Movil'))/m
PWOM=length(which(H2$nombre_cia_origen=='WOM'))/m
f_origen_simul=function(i){
  ifelse(is.na(H2$nombre_cia_origen[i]),sample(c('Claro','Entel',
         'Falabella_Movil_Spa','Movistar','Otros','VTR_Movil','WOM'),
         1,replace=TRUE,prob=c(PClaro,PEntel,PFalabella_Movil_Spa,PMovistar,
         POtros,PVTR_Movil,PWOM)),as.character(H2$nombre_cia_origen[i]))}
H2$nombre_cia_origen=sapply((1:m),f_origen_simul)
PKiosco_Portabilidad=length(which(H2$Canal=='Kiosco Portabilidad'))/m
PMasivo_y_Mayorista=length(which(H2$Canal=='Masivo y Mayorista'))/m
PRetail=length(which(H2$Canal=='Retail'))/m
PConvenios=length(which(H2$Canal=='Convenios'))/m
POtros=length(which(H2$Canal=='Otros'))/m
PWEB_Virgin=length(which(H2$Canal=='WEB Virgin'))/m
f_canal_simul=function(i){
  ifelse(H2$Canal[i]=='',sample(c('Kiosco Portabilidad',
          'Masivo y Mayorista','Retail','Convenios','Otros','WEB Virgin'),
          1,replace=TRUE,prob=c(PKiosco_Portabilidad,PMasivo_y_Mayorista,
          PRetail,PConvenios,POtros,PWEB_Virgin)),as.character(H2$Canal[i]))}
H2$Canal=sapply((1:m),f_canal_simul)
Q$nombre_cia_origen=H2$nombre_cia_origen
Q$Canal=H2$Canal
Q$Portado=H2$Portado
Q$suscrito_selfcare=H2$suscrito_selfcare
Q$StatusFugado=H2$StatusFugado
```

Algunos de los clasificadores tienen problemas al trabajar con variables de tipo categórico, por lo que es necesario que todas estas variables sean tipo "factor".

```
Q$StatusFugado=as.factor(Q$StatusFugado)
Q$Canal=as.factor(Q$Canal)
Q$Portado=as.factor(Q$Portado)
Q$nombre_cia_origen=as.factor(Q$nombre_cia_origen)
Q$suscrito_selfcare=as.factor(Q$suscrito_selfcare)
```

Una vez realizado los ajustes a la base de datos, separamos el conjunto de observaciones en dos grupos: el 70 % correspondientes a los datos de entrenamiento y el 30 % restante equivalente a los datos de testeo. La separación se hace de manera aleatoria.

```
n=nrow(G)
a=sample(c(1:n), floor(n*0.7),replace=F)
train=G[a,]
test=G[-a,]
```

A continuación es posible realizar pruebas con diferentes clasificadores. Para ello se utiliza el paquete `caret`. Es necesario establecer una métrica para comparar los diferentes algoritmos. Por defecto, esta métrica es "Accuracy", por lo que esta primera etapa se desarrolla bajo este parámetro.

```
trainControl <- trainControl(method="repeatedcv", number=5, repeats=3)

fit.lda <- train(StatusFugado~., data=train, method="lda",
                 preProc=c("center","scale"), trControl=trainControl)
fit.glm <- train(StatusFugado~., data=train, method="glm",
                 preProc=c("center","scale"), trControl=trainControl)
fit.glmnet <- train(StatusFugado~., data=train, method="glmnet",
                 preProc=c("center", "scale"), trControl=trainControl)
fit.cart <- train(StatusFugado~., data=train, method="rpart",
                 preProc=c("center", "scale"), trControl=trainControl)
fit.knn <- train(StatusFugado~., data=train, method="knn",
                 preProc=c("center","scale"), trControl=trainControl)
fit.rf <- train(StatusFugado~., data=train, method="rf",
                 preProc=c("center","scale"), trControl=trainControl)
```

Una vez entrenados los modelos (utilizando el conjunto de datos train), es posible compararlos con la función `resample`.

```
resultsACC <- resamples(list(LDA=fit.lda, GLM=fit.glm,GLMNET=fit.glmnet,
                             CART=fit.cart,KNN=fit.knn, RF=fit.rf))
Summary(resultsACC)
```

También es necesario tener los resultados aplicando los modelos entrenados sobre el conjunto de datos test. Para ello utilizamos la función `predict` en conjunto con `confusionMatrix`.

```
Plda=predict(fit.lda,newdata=test)
confusionMatrix(Plda,test$StatusFugado)

Pglm=predict(fit.glm,newdata=test)
confusionMatrix(Pglm,test$StatusFugado)

Pglmnet=predict(fit.glmnet,newdata=test)
confusionMatrix(Pglmnet,test$StatusFugado)

Pcart=predict(fit.cart,newdata=test)
confusionMatrix(Pcart,test$StatusFugado)

Pknn=predict(fit.knn,newdata=test)
confusionMatrix(Pknn,test$StatusFugado)

Prf=predict(fit.rf,newdata=test)
confusionMatrix(Prf,test$StatusFugado)
```

Una segunda comparación será estableciendo la métrica "ROC". Para ello es necesario que en `trainControl` se establezca: $summaryFunction = twoClassSummary$.

```
trainControl <- trainControl(method="repeatedcv", number=5, repeats=3,
                             classProbs=TRUE,summaryFunction=twoClassSummary)
metric='ROC'
fit.ldaROC <- train(StatusFugado~., data=train, method="lda",
                preProc=c("center","scale"),trControl=trainControl,metric=metric)
fit.glmROC <- train(StatusFugado~., data=train, method="glm",
                preProc=c("center","scale"),trControl=trainControl,metric=metric)
fit.glmnetROC <- train(StatusFugado~., data=train, method="glmnet",
                preProc=c("center", "scale"),trControl=trainControl,metric=metric)
fit.cartROC <- train(StatusFugado~., data=train, method="rpart",
                preProc=c("center", "scale"),trControl=trainControl,metric=metric)
fit.knnROC <- train(StatusFugado~., data=train, method="knn",
                preProc=c("center","scale"),trControl=trainControl,metric=metric)
fit.rfROC <- train(StatusFugado~., data=train, method="rf",
                preProc=c("center","scale"),trControl=trainControl,metric=metric)
```

Al igual que en el caso anterior, comparamos los modelos utilizando tanto el conjunto de datos train como los datos test.

```
resultsROC <- resamples(list(LDA=fit.ldaROC, GLM=fit.glmROC,
                             GLMNET=fit.glmnetROC,CART=fit.cartROC,
                             KNN=fit.knnROC, RF=fit.rfROC))

Plda=predict(fit.ldaROC,newdata=test)
confusionMatrix(Plda,test$StatusFugado)

Pglm=predict(fit.glmROC,newdata=test)
confusionMatrix(Pglm,test$StatusFugado)

Pglmnet=predict(fit.glmnetROC,newdata=test)
confusionMatrix(Pglmnet,test$StatusFugado)

Pcart=predict(fit.cartROC,newdata=test)
```

```
confusionMatrix(Pcart, test$StatusFugado)

Pknn=predict(fit.kmnROC, newdata=test)
confusionMatrix(Pknn, test$StatusFugado)

Prf=predict(fit.rfROC, newdata=test)
confusionMatrix(Prf, test$StatusFugado)
```

En los escenarios en los que se utilizaba SMOTE para balancear los datos, era necesario agregar la siguiente línea luego de separar los datos en conjuntos de entrenamiento y testeo. Este comando debe ser aplicado exclusivamente sobre los datos de entrenamiento.

```
smote_train <- SMOTE(StatusFugado ~ ., data  = train)
```

C.2. Support Vector Machine (SVM)

Se indicará el código principal para establecer el Support Vector Machine. El resto de los escenarios estudiados son variaciones a este modelo, los cuales incluyen balance de datos y reducción/aumento del número de variables utilizadas. Los diferentes modelos SVM estudiados fueron ejecutados en el HPC-UDD-Sofía.

Cargamos los datos y aplicamos los comandos necesarios para limpiar la base de datos.

```
H2=read.csv('variables_agosto.csv',sep=',',dec='.', header=T)

Q=select(H2,ID,NumVecinos,NumVecinosVirgin,NumVecinosFugados,
         CallFromChurned,CallTimeFromChurned,CallToChurned,
         CallTimeToChurned,tenure,TotalCallTime,CallTime_in,
         CallTime_out,Call_out,Call_in,TotalCalls,CallTimeOutOnNet,
         CallOutOnNet,CallTimeIntOnNet,CallInOnNet,Transitivity,
         count_triangles)
f=function(t){ifelse(is.na(t),0,t)}
Q$NumVecinosVirgin=sapply(Q$NumVecinosVirgin,f)
Q$NumVecinos=sapply(Q$NumVecinos,f)
Q$NumVecinosFugados=sapply(Q$NumVecinosFugados,f)
Q$CallFromChurned=sapply(Q$CallFromChurned,f)
Q$CallTimeFromChurned=sapply(Q$CallTimeFromChurned,f)
Q$CallToChurned=sapply(Q$CallToChurned,f)
Q$CallTimeToChurned=sapply(Q$CallTimeToChurned,f)
Q$tenure=sapply(Q$tenure,f)
Q$TotalCallTime=sapply(Q$TotalCallTime,f)
Q$CallTime_in=sapply(Q$CallTime_in,f)
Q$CallTime_out=sapply(Q$CallTime_out,f)
Q$Call_out=sapply(Q$Call_out,f)
Q$Call_in=sapply(Q$Call_in,f)
Q$TotalCalls=sapply(Q$TotalCalls,f)
Q$CallTimeOutOnNet=sapply(Q$CallTimeOutOnNet,f)
Q$CallOutOnNet=sapply(Q$CallOutOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallInOnNet=sapply(Q$CallInOnNet,f)
Q$Transitivity=sapply(Q$Transitivity,f)
Q$count_triangles=sapply(Q$count_triangles,f)
m=nrow(H2)
PClaro=length(which(H2$nombre_cia_origen=='Claro'))/m
PEntel=length(which(H2$nombre_cia_origen=='Entel'))/m
PFalabella_Movil_Spa=length(which(H2$nombre_cia_origen==
                                  'Falabella_Movil_Spa'))/m
PMovistar=length(which(H2$nombre_cia_origen=='Movistar'))/m
POtros=length(which(H2$nombre_cia_origen=='Otros'))/m
PVTR_Movil=length(which(H2$nombre_cia_origen=='VTR_Movil'))/m
```

```
PWOM=length(which(H2$nombre_cia_origen=='WOM'))/m
f_origen_simul=function(i){ ifelse(is.na(H2$nombre_cia_origen[i]),
                        sample(c('Claro','Entel','Falabella_Movil_Spa',
                        'Movistar','Otros','VTR_Movil','WOM'),1,replace=TRUE,
                        prob=c(PClaro,PEntel,PFalabella_Movil_Spa,
                        PMovistar,POtros,PVTR_Movil,PWOM)),
                        as.character(H2$nombre_cia_origen[i]))}
H2$nombre_cia_origen=sapply((1:m),f_origen_simul)
PKiosco_Portabilidad=length(which(H2$Canal=='Kiosco Portabilidad'))/m
PMasivo_y_Mayorista=length(which(H2$Canal=='Masivo y Mayorista'))/m
PRetail=length(which(H2$Canal=='Retail'))/m
PConvenios=length(which(H2$Canal=='Convenios'))/m
POtros=length(which(H2$Canal=='Otros'))/m
PWEB_Virgin=length(which(H2$Canal=='WEB Virgin'))/m
f_canal_simul=function(i){ ifelse(H2$Canal[i]=='',
                         sample(c('Kiosco Portabilidad','Masivo y Mayorista',
                         'Retail','Convenios','Otros','WEB Virgin'),1,
                         replace=TRUE,prob=c(PKiosco_Portabilidad,
                         PMasivo_y_Mayorista,PRetail,PConvenios,POtros,
                         PWEB_Virgin)), as.character(H2$Canal[i]))}
H2$Canal=sapply((1:m),f_canal_simul)
Q$nombre_cia_origen=H2$nombre_cia_origen
Q$Canal=H2$Canal
Q$Portado=H2$Portado
Q$suscrito_selfcare=H2$suscrito_selfcare
Q$StatusFugado=H2$StatusFugado
G=Q[,c(-1)]
G$StatusFugado=as.factor(G$StatusFugado)
G$Canal=as.factor(G$Canal)
G$Portado=as.factor(G$Portado)
G$nombre_cia_origen=as.factor(G$nombre_cia_origen)
G$suscrito_selfcare=as.factor(G$suscrito_selfcare)
```

Separamos los datos en conjuntos de entrenamiento y testeo (70 % y 30 % respectivamente).

```
n=nrow(G)
a=sample(c(1:n), floor(n*0.7),replace=F)
train=G[a,]
test=G[-a,]
```

Definimos el algoritmo para crear un SVM. Se han establecido valores de gamma igual a 0,05 y costo igual a 1. Para crear el modelo es necesaria la utilización del paquete `e1071`.

```
svmfit.opt=svm(StatusFugado~., data=train, kernel="radial",
                    gamma=0.05,cost=1,decision.values=T)
ypred=predict(svmfit.opt,test[,-25])
(tt=table(predichas=ypred, reales=test$StatusFugado))
```

Calculamos diferentes métricas a partir de la información obtenida de la matriz de confusión:

```
accuracy=mean(ypred==test$StatusFugado)
precision=tt[1,1]/(tt[1,1]+tt[1,2])
especificidad=tt[2,2]/(tt[2,2]+tt[1,2])
sensibilidad=tt[1,1]/(tt[1,1]+tt[2,1])
Balanced_Accuracy=(sensibilidad+especificidad)/2
```

C.3. Deep Learning (DL)

Para utilizar Deep Learning en R es necesario cargar el paquete `h2o`[1].Los diferentes escenarios estudiados son variantes del código que aquí se presenta. Debido a la simplicidad del código y la agilidad con la que se ejecuta, en este caso no fue necesaria la utilización del HPC-UDD-Sofía.

Cargamos los datos.

```
H2=read.csv('variables_agosto.csv',sep=',',dec='.', header=T)

Q=select(H2,ID,NumVecinos,NumVecinosVirgin,NumVecinosFugados,CallFromChurned,
        CallTimeFromChurned,CallToChurned,CallTimeToChurned,tenure,TotalCallTime,
        CallTime_in,CallTime_out,Call_out,Call_in,TotalCalls,CallTimeOutOnNet,
        CallOutOnNet,CallTimeIntOnNet,CallInOnNet,Transitivity,count_triangles)
f=function(t){ifelse(is.na(t),0,t)}
Q$NumVecinosVirgin=sapply(Q$NumVecinosVirgin,f)
Q$NumVecinos=sapply(Q$NumVecinos,f)
Q$NumVecinosFugados=sapply(Q$NumVecinosFugados,f)
Q$CallFromChurned=sapply(Q$CallFromChurned,f)
Q$CallTimeFromChurned=sapply(Q$CallTimeFromChurned,f)
Q$CallToChurned=sapply(Q$CallToChurned,f)
Q$CallTimeToChurned=sapply(Q$CallTimeToChurned,f)
Q$tenure=sapply(Q$tenure,f)
Q$TotalCallTime=sapply(Q$TotalCallTime,f)
Q$CallTime_in=sapply(Q$CallTime_in,f)
Q$CallTime_out=sapply(Q$CallTime_out,f)
Q$Call_out=sapply(Q$Call_out,f)
Q$Call_in=sapply(Q$Call_in,f)
Q$TotalCalls=sapply(Q$TotalCalls,f)
Q$CallTimeOutOnNet=sapply(Q$CallTimeOutOnNet,f)
Q$CallOutOnNet=sapply(Q$CallOutOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallTimeIntOnNet=sapply(Q$CallTimeIntOnNet,f)
Q$CallInOnNet=sapply(Q$CallInOnNet,f)
Q$Transitivity=sapply(Q$Transitivity,f)
Q$count_triangles=sapply(Q$count_triangles,f)
m=nrow(H2)
PClaro=length(which(H2$nombre_cia_origen=='Claro'))/m
PEntel=length(which(H2$nombre_cia_origen=='Entel'))/m
PFalabella_Movil_Spa=length(which(H2$nombre_cia_origen=='Falabella_Movil_Spa'))/m
PMovistar=length(which(H2$nombre_cia_origen=='Movistar'))/m
POtros=length(which(H2$nombre_cia_origen=='Otros'))/m
PVTR_Movil=length(which(H2$nombre_cia_origen=='VTR_Movil'))/m
PWOM=length(which(H2$nombre_cia_origen=='WOM'))/m
f_origen_simul=function(i){ ifelse(is.na(H2$nombre_cia_origen[i]),
                    sample(c('Claro','Entel','Falabella_Movil_Spa',
                    'Movistar','Otros','VTR_Movil','WOM'),1,replace=TRUE,
                    prob=c(PClaro,PEntel,PFalabella_Movil_Spa,
                    PMovistar,POtros,PVTR_Movil,PWOM)),
                    as.character(H2$nombre_cia_origen[i]))}
H2$nombre_cia_origen=sapply((1:m),f_origen_simul)
PKiosco_Portabilidad=length(which(H2$Canal=='Kiosco Portabilidad'))/m
PMasivo_y_Mayorista=length(which(H2$Canal=='Masivo y Mayorista'))/m
PRetail=length(which(H2$Canal=='Retail'))/m
PConvenios=length(which(H2$Canal=='Convenios'))/m
POtros=length(which(H2$Canal=='Otros'))/m
PWEB_Virgin=length(which(H2$Canal=='WEB Virgin'))/m
f_canal_simul=function(i){ ifelse(H2$Canal[i]=='',
                     sample(c('Kiosco Portabilidad','Masivo y Mayorista',
                     'Retail','Convenios','Otros','WEB Virgin'),1,
```

[1]R utiliza un API para conectarse con http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html

```
                        replace=TRUE, prob=c(PKiosco_Portabilidad,
                        PMasivo_y_Mayorista, PRetail, PConvenios, POtros,
                        PWEB_Virgin)), as.character(H2$Canal[i]))}
H2$Canal=sapply((1:m),f_canal_simul)
Q$nombre_cia_origen=H2$nombre_cia_origen
Q$Canal=H2$Canal
Q$Portado=H2$Portado
Q$suscrito_selfcare=H2$suscrito_selfcare
Q$StatusFugado=H2$StatusFugado
```

Para crear el modelo Deep Learning, es necesario inicializar `h2o`, contenido en el paquete llamado de la misma manera. Definimos la variable "Class" como el Status Fugado.

```
h2o.init()
GG=Q[,c(-1,-26)]
GG$class=Q$StatusFugado
```

Para aplicar este modelo, es necesario contar con una base de datos formato h2o. Esto lo realizamos utilizando la función `as.h2o`. Además, definimos dos conjuntos de datos: y contiene las etiquetas de los clientes, mientras que x corresponde a las variables predictoras.

```
data<-as.h2o(GG)
y <- "class"
x <- setdiff(names(data), y)
```

Dividimos el set de datos en un conjunto train, equivalente al 70 % de los datos y test, correspondiente al 30 % restante.

```
parts <- h2o.splitFrame(data, ratios = 0.7)
train <- parts[[1]]
test <- parts[[2]]
```

Con la información anterior, definimos el modelo Deep Learning. Para ello es necesario utilizar los conjuntos de datos y, x y *train*. Una vez entrenado el modelo, es posible obtener la matriz de confusión asociada a la predicción del conjunto de datos test.

```
m <- h2o.deeplearning(x, y, train)
p <- h2o.predict(m, test)
S=as.data.frame(p)
h2o.mse(m)
h2o.confusionMatrix(m)
M=h2o.confusionMatrix(m)
tn=M$no_fugado[2]
tp=M$fugado[1]
fp=M$no_fugado[1]
fn=M$fugado[2]
accuracy=(tp+tn)/(tp+tn+fp+fn)
precision=tp/(tp+fp)
sensibilidad=tp/(tp+fn)
especificidad=tn/(tn+fp)
Balanced_Accuracy=(sensibilidad+especificidad)/2
```

Apéndice D

Análisis de Supervivencia

D.1. Modelos no paramétricos

Análisis de supervivencia

```
set.seed(123)
min(ymd(H$activacion))
max(ymd(H$activacion))
```

Usamos tenure como tiempo de falla y estratificamos por Canal.

```
autoplot(survfit(Surv(tenure, Status) ~ Canal, data=H),
type="fill", palette="Pastel2",
fillLineSize=0.4, alpha=0.4,conf.int = T,censor=F,survLineSize = 5)
```

Eliminamos Ótrosén canal.

```
H1=filter(H,Canal!='Otros')
g1=autoplot(survfit(Surv(tenure, Status) ~ Canal, data=H1),
type="fill", palette="Pastel1",
fillLineSize=0.4, alpha=0.7,conf.int = T,censor=F,survLineSize = 5)

p=c("#7bc043","#f37736","#ee4035","#0392cf",'#edc951','#4b3832')

g1 + ggplot2::scale_colour_manual(values=p) +
ggplot2::scale_fill_manual(values=p)+ ggplot2::theme_classic()
```

Estratificación por suscrito_selfcare.

```
g1=autoplot(survfit(Surv(tenure, Status) ~ suscrito_selfcare, data=H),
type="fill", palette="Pastel1",
fillLineSize=0.9, alpha=0.7,conf.int = T,censor=F,survLineSize =5,surv.size=2)
p=c("#7bc043","#f37736")

g1 + ggplot2::scale_colour_manual(values=p) +
ggplot2::scale_fill_manual(values=p)+ ggplot2::theme_classic()
```

Estratificación por Portado.

```
g1=autoplot(survfit(Surv(tenure, Status) ~ Portado, data=H1),
type="fill", palette="Pastel1",
fillLineSize=0.9, alpha=0.7,conf.int = T,censor=F,survLineSize =5,surv.size=2)
p=c("#7bc043","#f37736")

g1 + ggplot2::scale_colour_manual(values=p) +
ggplot2::scale_fill_manual(values=p)+ ggplot2::theme_classic()
```

Estimación KM de la función de supervivencia.

```
g1=autoplot(survfit(Surv(tenure, Status) ~ 1, data=H1),
type="fill", palette="Pastel1",
fillLineSize=0.9, alpha=0.7,conf.int = T,censor=T,survLineSize =5,surv.size=2)
p=c('#d11141')

g1 + ggplot2::scale_colour_manual(values=p) +
ggplot2::scale_fill_manual(values=p)+ ggplot2::theme_classic()
```

Cambiamos el marco de tiempo. Tomamos el tiempo de seguimiento que parte desde el 2016-08-01. Esta información está contenida en la variable tm_obs.

```
H1=as.data.table(H)
f_end= ymd("2016-10-15")
f_ini=ymd("2016-08-01")
a=interval(f_ini,f_end)/ddays(1)
```

No considera la activación después de la fecha final.

```
H1=as.data.table(filter(H1,ymd(activacion)<=f_end))

head(H1)
unique(H1$StatusFugado)
prop.table(table(H1$Status))

H1$fecha_churn_mkt=as.character(H1$fecha_churn_mkt)
H1$fecha_portOut=as.character(H1$fecha_portOut)
```

Definimos la fecha de fuga "ff".

```
H2=H1[,.(ff=ifelse((StatusFugado=='fugado')&(!(fecha_portOut=='')),
fecha_portOut,fecha_churn_mkt))]
H1$ff=H2$ff
head(H1,20)
H2=H1[,.(ff=ifelse(StatusFugado=='no_fugado',NA,ff))]
H1$ff=H2$ff
```

Filtramos dejando solo los que se fugan después del inicio del período de observación.

```
H1=as.data.table(filter(H1,(is.na(ff))|(ymd(ff)>=f_ini)))
```

Definimos en tiempo de observación

```
H2=H1[,.(t1=ifelse((StatusFugado=='no_fugado')|(StatusFugado=='fugado'&(ymd(ff)>=
    f_end)),
(interval(ymd(activacion),f_end)/ddays(1)+a-abs(interval(ymd(activacion),f_end)/
    ddays(1)-a))/2,
(interval(ymd(activacion),ymd(ff))/ddays(1)+interval(f_ini,
ymd(ff))/ddays(1)-abs(interval(ymd(activacion),
ymd(ff))/ddays(1)-interval(f_ini,ymd(ff))/ddays(1)))/2))]
H1$tm_obs=H2$t1

head(H1,20)
```

Reajustamos el Status porque hay observaciones que sabemos que son fugados, pero que al cierre del periodo de observación se mantenian sin fugarse.

```
H3=mutate(H1,Status_new=ifelse(H1$Status==1&H1$tm_obs==a,0,Status))
```

D.2. Modelo de Regresión de Cox

```
datos=read.csv('variables_agosto.csv',header=T,sep=',',dec='.')
datos=data.table(datos)
Q=select(datos,ID,NumVecinos,NumVecinosVirgin,NumVecinosFugados,CallFromChurned,
CallTimeFromChurned,CallToChurned,CallTimeToChurned,tenure,TotalCallTime,
    CallTime_in,
CallTime_out,Call_out,Call_in,TotalCalls,CallTimeOutOnNet,CallOutOnNet,
CallTimeIntOnNet,CallInOnNet,Reciprocidad,NumVecinosFugadosCercanos,
NumVecinosVirginCercanos)

H4 <- Q[H3,on=c(ID='ID'), nomatch=0]

aa=subset(H4,H4$NumVecinosFugados<4)
fit <- coxreg(Surv(tm_obs,Status_new)~ NumVecinosFugados,data = aa)
summary(fit)

fit <- coxreg(Surv(tm_obs,Status_new)~ NumVecinos,data = aa)
summary(fit)

fit <- coxreg(Surv(tm_obs,Status_new)~ NumVecinos+NumVecinosFugados,data = aa)
summary(fit)


fit <- coxreg(Surv(tm_obs,Status_new)~ tenure,data = aa)
summary(fit)

fit <- coxreg(Surv(tm_obs,Status_new)~ tenure+ NumVecinosFugados+NumVecinos,data =
    aa)
summary(fit)


fit <- coxreg(Surv(tm_obs,Status_new)~ TotalCallTime,data = aa)
summary(fit)


fit <- coxreg(Surv(tm_obs,Status_new)~ CallTime_in,data = aa)
summary(fit)


fit <- coxreg(Surv(tm_obs,Status_new)~ TotalCalls,data = aa)
summary(fit)


fit <- coxreg(Surv(tm_obs,Status_new)~ CallFromChurned,data = aa)
summary(fit)

fit <- coxreg(Surv(tm_obs,Status_new)~ CallFromChurned+TotalCalls,data = aa)
summary(fit)

fit <- coxreg(Surv(tm_obs,Status_new)~ CallFromChurned+TotalCalls+tenure+
 NumVecinosFugados,data = aa)
summary(fit)
```

D.3. Modelo de Aalen

```
A=aareg(Surv(tm_obs,Status_new) ~NumVecinosFugados+
NumVecinosVirgin+NumVecinosFugadosCercanos+NumVecinosVirginCercanos, data = H4)
autoplot(A)
summary(A)
```

```
A

A=aareg(Surv(tm_obs,Status_new) ~CallFromChurned+CallTimeFromChurned+
CallToChurned+CallTimeToChurned, data = H4)
autoplot(A)
A
```