Maestría en Explotación de Datos y Gestión del Conocimiento

Introducción a Data Mining — Caso 2 Telco Co.

Grupo:
- Ksiazenicer, Emiliano
- Turuguet, Santiago
- Ton Vanerio, Juan Ignacio
- Kuringhian, Lucas
- Jaime, Analía

Índice

1. Caso de Estudio - Telco Co.

La creciente presión competitiva requiere que la compañía desarrolle nuevas e innovadoras formas de satisfacer las mayores demandas que los clientes realizan. Desarrollar esas nuevas ideas requiere información acerca de sus clientes, la que debe ser inferida a partir de los datos recopilados sobre dichos clientes.

La predicción de churn resulta ser la mayor preocupación en una compañía del sector de las telecomunicaciones, como consecuencia de las características típicas de este mercado, tales como saturación y cambios dinámicos. A medida que el mercado se satura, la adquisición de nuevos clientes resulta más costosa que retener los clientes existentes. Por otra parte, los cambios dinámicos del mercado en lo que se refiere a competidores, tecnología y regulaciones generan mayores oportunidades para que los clientes cambien de proveedor.

La compañía desea saber si las técnicas de Data Mining pueden proporcionar alguna respuesta al problema de predecir cuáles son los clientes que posiblemente abandonen el servicio voluntariamente y que debería intentar conservar, por tratarse de clientes de valor para la compañía. La información de qué clientes abandonan el servicio, la compañía necesita conocerla al menos con dos meses de anticipación, para poder iniciar una campaña de retención. Para esto se requiere una buena comprensión del comportamiento de los clientes, de manera de poder identificar a aquellos que están por dejar la compañía y al mismo tiempo, quienes son redituables.

2. Consignas

1- ¿Si Ud. tuviera que implementar un sistema de retención de clientes, considera que Data Mining puede ser de ayuda en la detección de clientes próximos a abandonar la compañía?

2- ¿Qué tipo de tarea o tareas de Data Mining se podrían aplicar?

3- ¿Qué tipo de datos requeriría para poder aplicar las tareas de Data Mining seleccionadas?

4- ¿Qué parámetros debería establecer para la tarea o tareas y que valores asignaría a cada uno?

5- Si observa alguna particularidad en los datos de este problema con respecto al tiempo, proponga el tratamiento especial que considere adecuado.

6- Considere el caso de una compañía de telefonía fija con diversas alternativas de Churn (baja del cliente completo, líneas, Internet, paquetes urbanos, etc.).¿Cómo aplicaría los resultados obtenidos del mining?

7- Tareas por realizar:

a) Entendimiento de los datos y el negocio.
b) Observar las distribuciones de las variables. Valores faltantes y outliers.
c) Buscar y eliminar variables correlacionadas.
d) Analizar las proporciones de churn para distintas variables.
e) Particionar los datos.
f) Generar tres modelos de árbol.
g) Evaluar los modelos.

8- Aplicar el modelo de predicción elegido (justificar cual y por qué) a la base de clientes y generar un score de churn para cada cliente.

3. Respuestas

3.1. ¿Si Ud. tuviera que implementar un sistema de retención de clientes, considera que Data Mining puede ser de ayuda en la detección de clientes próximos a abandonar la compañía?

Sí, en mercados competitivos y saturados como el de telecomunicaciones es importante utilizar la tecnología para obtener ventajas competitivas, por lo tanto, el Data Mining podría ser una gran herramienta para la retención de clientes dado que suele ser más costoso captar un nuevo cliente que retener uno existente.

El Data Mining permite detectar patrones de comportamiento a partir del análisis de grandes volúmenes de datos históricos, como el uso de servicios, interacciones con atención al cliente, reclamos o retrasos en pagos. Estos patrones funcionan como señales tempranas de descontento y permiten anticipar el abandono. Mediante modelos predictivos supervisados (como ser: árboles de decisión, Random Forest, Regresión logística, etc.), es posible asignar a cada cliente una probabilidad de churn logrando predecir aquellos clientes que están próximos a abandonar.

Asimismo, el Data Mining favorece la segmentación de clientes en riesgo, permitiendo diferenciar entre aquellos que presentan riesgo inmediato y quienes muestran señales incipientes. Esto habilita a diseñar estrategias de retención personalizadas, optimizando los recursos de la compañía y priorizando a los clientes de mayor valor, siempre y cuando haya un espacio temporal previo al churn (2 meses para este caso) que permita madurar y efectivizar las estrategias de retención.

3.2 ¿Qué tipo de tarea o tareas de Data Mining se podrían aplicar?

Las tareas de clasificación, predicción, clustering, minería de reglas de asociación y detección de outliers son componentes centrales del Data Mining. Los algoritmos supervisados (árboles de decisión, random forest) permiten construir modelos predictivos robustos, mientras que las técnicas no supervisadas (clustering) enriquecen la comprensión del comportamiento de los clientes y permiten diseñar estrategias diferenciadas de retención. Asimismo, el preprocesamiento de datos, la reducción de dimensionalidad y la ingeniería de atributos son pasos críticos en la obtención de resultados fiables y accionables.

En ese contexto, para la predicción de churn se pueden utilizar algoritmos de clasificación supervisada o no supervisada junto con un profundo análisis exploratorio donde la detección de anomalías tomará un rol importante en la etapa de preparación del data set. Integradas con métricas de evaluación orientadas al negocio, estas tareas permiten a la compañía anticipar el abandono con la anticipación necesaria y diseñar campañas de retención efectivas.

En conclusión, el análisis de churn se beneficia de la aplicación de diversas tareas de Data Mining, tanto supervisadas como no supervisadas, que permiten identificar patrones de abandono y construir modelos predictivos efectivos. A saber:

1. Tareas supervisadas

La clasificación es la técnica más utilizada para predecir churn. Algoritmos como árboles de decisión, Random Forest, regresión logística, Naive Bayes permiten estimar la probabilidad de que un cliente abandone el servicio. Los árboles de decisión destacan por su interpretabilidad, ya que generan reglas claras que pueden ser directamente utilizadas por los equipos de marketing y atención al cliente.

En el dataset analizado, se observa que los clientes con Plan Internacional presentan una tasa de abandono cercana al 42%, mientras que aquellos con Voice Mail muestran un riesgo mucho menor. Un árbol de decisión permite traducir esto en reglas simples, por ejemplo:

"Si el cliente tiene Plan Internacional y más de 4 llamadas al servicio al cliente → alta probabilidad de churn". "Si no tiene Plan Internacional y posee Voice Mail → baja probabilidad de churn".

Estas reglas son claras y accionables, lo que permite priorizar campañas de retención en segmentos críticos.

Otra dimensión supervisada consiste en la predicción temporal del evento, lo que implica definir ventanas de datos, de retardo y de pronóstico. Es decir, en el caso Telco se implementa un marco de ventanas de tiempo: 6 meses de histórico, 2 meses de retardo y 1 mes de pronóstico. Por ejemplo, usando datos de febrero a julio y considerando un retardo en agosto-septiembre, se predice el churn de octubre. Esto asegura que la empresa disponga de al menos dos meses de anticipación para ejecutar campañas de retención.

2. Tareas no supervisadas

El clustering (segmentación de clientes) permite agrupar usuarios según patrones de uso o interacción, identificando segmentos de alto riesgo aún cuando no exista una etiqueta previa de churn.

El análisis de reglas de asociación puede revelar combinaciones de servicios o planes que incrementan el riesgo de abandono.

La detección de anomalías es relevante para identificar comportamientos atípicos, como una disminución brusca en el uso o múltiples llamadas al call center, los cuales pueden ser predictores tempranos de deserción.

Finalmente, la minería de secuencias permite modelar patrones repetitivos en el tiempo (ejemplo: aumento de reclamos → disminución del uso → baja), anticipando con mayor precisión el abandono.

3. Ingeniería de atributos y preparación de datos

El éxito de los modelos depende de la calidad y relevancia de las variables. Es común generar indicadores derivados, como índices de movilidad, calidad de llamadas o tendencia de uso en meses previos, que enriquecen el poder predictivo. También se recomienda eliminar variables altamente correlacionadas (ejemplo: minutes / charges) y depurar atributos anómalos.

Yendo puntualmente al caso que nos ocupa, podemos señalar, por ejemplo, que el análisis inicial muestra que aproximadamente el 14 % de los clientes son churners, mientras que el 86 % restante no abandona. Este desbalance genera que, si se entrena un modelo sin tratamiento, este podría tender a predecir que "nadie se va", alcanzando una alta exactitud aparente, pero con una bajísima sensibilidad para detectar churn.

- Oversampling (sobremuestreo): En el dataset Telco se pueden replicar instancias de churners o aplicar técnicas más avanzadas como SMOTE (Synthetic Minority Oversampling Technique), que genera nuevos ejemplos sintéticos de clientes que abandonan. Así, de un 14 % de churn se podría equilibrar la base al 40 %, aumentando la capacidad del modelo para reconocer patrones de abandono.

- Undersampling (submuestreo): Otra estrategia es reducir aleatoriamente la cantidad de no churners (clientes leales), de modo que la proporción de clases sea más pareja. Por ejemplo, si hay 2.600 clientes que permanecen y 400 que abandonan, se podría recortar a 800 vs. 400 para entrenar el modelo, evitando que los casos de abandono queden "invisibles" frente al volumen mayoritario.

- Asignación de pesos. En lugar de modificar el dataset, algunos algoritmos permiten dar más peso a los errores en la clase minoritaria. En Telco, esto significa penalizar más los falsos negativos (casos en los que un cliente en riesgo no es identificado). Por ejemplo, un árbol de decisión o un modelo de regresión logística puede configurarse para dar doble peso a los errores de clasificación de churners frente a los de no churners.

4. Evaluación y métricas

Para seleccionar el modelo óptimo, es fundamental emplear técnicas de validación cruzada rigurosas. Esto implica dividir el conjunto de datos en subconjuntos para entrenamiento y prueba, asegurando que el modelo se generalice bien a datos no vistos. Además de la validación cruzada, es crucial evaluar el rendimiento del modelo utilizando métricas específicas y relevantes para el problema de predicción de churn.

Entre las métricas clave, se destacan la accuracy (exactitud), que mide la proporción de predicciones correctas; el recall (sensibilidad), que en el contexto de churn representa la capacidad del modelo para identificar correctamente a los clientes que realmente abandonarán; y la precisión, que indica la proporción de clientes predichos como churners que efectivamente lo son.

Un análisis detallado de la matriz de confusión es indispensable. Esta matriz proporciona una visión completa de los aciertos y errores del modelo, distinguiendo entre verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Para la predicción de churn, la matriz de confusión permite identificar cuántos clientes en riesgo fueron detectados (verdaderos positivos) y cuántos no lo fueron (falsos negativos), así como cuántos clientes estables fueron erróneamente clasificados como churners (falsos positivos).

En el ámbito de la retención de clientes, es común y a menudo estratégico priorizar el recall sobre la precisión. La razón principal es que el costo de no detectar a un cliente en riesgo de churn (falso negativo) suele ser significativamente más alto que el costo de ofrecer incentivos o acciones de retención a un cliente que, de todos modos, no tenía intenciones de abandonar (falso positivo). Un falso negativo implica la pérdida potencial de ingresos futuros y el costo de adquisición de un nuevo cliente. Por otro lado, un falso positivo, aunque no ideal, puede verse como una inversión en la lealtad del cliente o un costo operativo menor en comparación con la pérdida neta de un cliente. Por lo tanto, un modelo con alto recall es preferible para asegurar que la mayoría de los clientes propensos al churn sean identificados y se puedan tomar medidas proactivas para retenerlos.

3.3 ¿Qué tipo de datos requeriría para poder aplicar las tareas de Data Mining seleccionadas?

Para poder aplicar las tareas de Data Mining seleccionadas en la predicción de churn, se requiere un conjunto amplio y diverso de datos que capture las distintas dimensiones de la relación cliente–compañía.

Los datos requeridos para la minería incluyen tanto atributos directos (como podrían ser demográficos, transaccionales o de uso) como atributos derivados construidos a partir de transformaciones y agregaciones.

En ese sentido. los tipos de datos se podrían agrupar de la siguiente manera:

A. Datos demográficos y de perfil: En este caso serían variables como la de Estado o Código de Área. Estos datos son esenciales para segmentar clientes y detectar perfiles con mayor propensión al abandono.

Particularmente, en el dataset bajo análisis se pueden observar los siguientes datos:

- State → Estado donde reside el cliente.
- Area_Code → Código de área telefónico.
- Phone → Número de teléfono (más identificador que predictor).
- Account_Length → Antigüedad de la cuenta en días.

B. Datos de uso del servicio: Son fundamentales para identificar patrones de comportamiento y permiten detectar cambios en el consumo y señales tempranas de insatisfacción:

Minutos consumidos: Day_Mins, Eve_Mins, Night_Mins, Intl_Mins → Minutos consumidos en distintos períodos (día, tarde, noche, internacional). Cantidad de llamadas: Day_Calls, Eve_Calls, Night_Calls, Intl_Calls → Cantidad de llamadas en cada período. Contratación de planes adicionales (ej. International Plan, Voice Mail). Intl_Plan → Si el cliente tiene plan internacional (sí/no). Vmail_Plan → Si el cliente tiene buzón de voz contratado (sí/no) Número de llamadas al servicio de atención al cliente. CustServ_Calls → más de tres llamadas en un período corto es un predictor crítico de churn.

C. Datos de facturación y pagos: En relación al caso que nos ocupan en el presente, sería el cargo o abono cobrado a los clientes.

Day_Charge, Eve_Charge, Night_Charge, Intl_Charge → Cargos facturados en función del consumo.

D. Índices derivados: Los modelos recomiendan generar atributos construidos a partir de datos brutos, que permitirìan mejorar la capacidad predictiva y capturar patrones dinámicos de comportamiento. Ejemplos de los cálculos que podemos realizar son:

Promedio de llamadas por minuto (Day_Calls / Day_Mins). Porcentaje de llamadas al call center en relación a las totales. Tendencia de uso: comparando Day_Mins mes a mes.

E. Variable objetivo: Indicador binario de churn que señale si el cliente abandonó o no la compañía en el período analizado.

En nuestro caso: Churn - Variable binaria (True / False) que indica si el cliente abandonó la compañía.

F. Dimensión temporal. Los datos deben estructurarse en ventanas temporales que permitan anticipar el abandono y que permitan garantizar que la predicción se realice con suficiente anticipación para que las campañas de retención sean efectivas. En este dataset no hay fechas ni ventanas históricas, es un corte en un momento dado. Para predecir churn con anticipación, sería ideal contar con datos longitudinales:

Ventana de datos (ej. 6 meses de histórico). Tiempo de retardo (ej. 2 meses). Ventana de pronóstico (ej. 1 mes).

3.4 ¿Qué parámetros debería establecer para la tarea o tareas y que valores asignaría a cada uno?

1. Preparación de los datos

Antes de modelar, los parámetros a fijar en la preparación determinan la calidad de las predicciones:

Tratamiento de valores faltantes: imputación mediante media/mediana (para numéricos) o moda (para categóricos). En variables críticas, aplicar kNN Imputation. Eliminación de correlaciones: descartar variables altamente correlacionadas como minutes y charges (ya que son lineales). Normalización/Estandarización: aplicar z-score a variables de uso intensivo de minutos para algoritmos sensibles a escalas (ej. SVM, k-NN). Manejo del desbalance de clases: parámetro clave. Se recomienda oversampling con SMOTE (k = 5 vecinos, ratio 1:1 churners/no churners) o asignar peso doble a la clase churn en algoritmos de clasificación.

2. Particionado de datos

La forma de dividir los datos impacta directamente en la validez del modelo: Se define una seed (Semilla aleatoria), la cual permite fijar para garantizar la replicabilidad de los experimentos.

Luego se define la técnica de muestreo :

Train/Test split: 70 % entrenamiento / 30 % prueba. Validación cruzada: k-fold con k = 10 (valor estándar recomendado en la literatura por su balance entre sesgo y varianza).

3. Parámetros en Árboles de Decisión

Criterio de división: Gini index (para equilibrio entre clases) o Entropía (cuando se prioriza pureza de nodos). Profundidad máxima: 5–8 niveles (evita sobreajuste, manteniendo interpretabilidad). Número mínimo de instancias por hoja: 20–30 clientes. Esto evita hojas con un solo caso, que generan reglas poco generalizables. Poda: habilitar post-pruning con un umbral de error mínimo del 5 %.

4. Parámetros en Random Forest / Ensembles

Número de árboles (n estimators): 200–500 árboles, lo suficiente para estabilizar resultados. Máximo de variables por división (max_features): $\sqrt{p}$ (raíz cuadrada del número de variables, regla de Breiman). Bootstrap: activado, para diversificar los árboles. Balanceo de clases: opción class weight = balance en librerías como Scikit-learn.

5. Parámetros en Regresión Logística

Regularización: usar penalización L2 (Ridge) para evitar multicolinealidad. C = 1.0 como valor base de penalización, ajustable con grid-search. Clasificación probabilística: umbral de decisión 0.3 - 0.4 en lugar de 0.5, para priorizar recall y detectar más churners.

6. Parámetros en SVM

Kernel: RBF (Radial Basis Function), recomendado para datos no lineales. Parámetro C: comenzar con C = 1, ajustar en grid-search (valores: 0.1, 1, 10). Gamma ( ): usar 1/n_features como valor base, explorar rangos (0.01, 0.1, 1).

7. Parámetros en Redes Neuronales

Arquitectura: 1–2 capas ocultas con 32–64 neuronas cada una. Función de activación: ReLU en capas ocultas, Sigmoid en la salida (churn = 0/1). Learning rate: 0.01, ajustable con optimizadores (Adam). Batch size: 32. Épocas: 50–100, con early stopping para evitar sobreentrenamiento.

8. Métricas de evaluación

Accuracy: proporción total de aciertos. Recall (sensibilidad): métrica prioritaria, optimizada hacia valores 0.7. Precisión: secundaria, aceptable 0.5. F1-score: balance entre recall y precisión, útil para elegir el modelo final. Matriz de confusión: indispensable para interpretar el impacto de falsos negativos y falsos positivos.

9. Parámetros temporales

Ventana de datos: 6 meses de histórico. Tiempo de retardo: 2 meses (gap antes del pronóstico). Ventana de pronóstico: 1 mes. Este marco asegura predicciones útiles con la anticipación necesaria para implementar campañas de retención.

3.5. Si observa alguna particularidad en los datos de este problema con respecto al tiempo, proponga el tratamiento especial que considere adecuado.

En el problema de predicción de churn, la particularidad central de los datos es la dimensión temporal. El abandono de clientes no ocurre de manera instantánea, sino como resultado de un proceso que evoluciona a lo largo del tiempo y refleja cambios en el uso del servicio, la interacción con la compañía y factores externos. Por eso, un modelo construido sólo sobre datos estáticos termina siendo una "foto" que pierde la dinámica real del fenómeno.

El primer tratamiento especial consiste en el diseño del muestreo y la evaluación. Para el entrenamiento, se recomienda utilizar ventanas históricas de al menos 12 meses que permitan capturar patrones estacionales completos. En la evaluación, además de la división clásica en train y test, resulta imprescindible incorporar un conjunto Out of Time (OOT), con datos posteriores al período de entrenamiento. Esto simula el comportamiento del modelo en escenarios futuros y permite validar su robustez predictiva en condiciones más realistas.

Una segunda línea de trabajo es la aplicación de transformaciones temporales. La desestacionalización ayuda a eliminar fluctuaciones recurrentes (ej. vacaciones, campañas) y enfocar al modelo en las tendencias genuinas del comportamiento del cliente. Asimismo, el ajuste por índices o transformaciones no lineales resulta clave en contextos de alta inflación, como Argentina, para que las variables monetarias sean comparables en el tiempo y no distorsionen la predicción.

Es decir que la capacidad predictiva de un modelo de churn depende de un tratamiento explícito de la temporalidad. Resulta menester definir correctamente las ventanas de entrenamiento y predicción, validando con escenarios futuros y aplicando transformaciones adecuadas permite pasar de un simple corte transversal a una verdadera "película" del cliente. Solo así se logra anticipar el abandono con la anticipación necesaria para implementar acciones efectivas de retención.

3.6. Considere el caso de una compañía de telefonía fija con diversas alternativas de Churn.

a. ¿Cómo aplicaría los resultados obtenidos del mining?

En el caso de una compañía de telefonía fija que enfrenta distintos tipos de churn baja total del cliente, baja de líneas individuales, cancelación de Internet o de paquetes urbanos, los resultados obtenidos del data mining pueden aplicarse de manera estratégica en distintos niveles de decisión y acción. El modelo de churn obtenido puede convertirse en el motor de las campañas de marketing y toma de decisiones por parte de la empresa.

1. Identificación temprana de clientes en riesgo

Los modelos predictivos permiten detectar patrones específicos de abandono según la unidad de análisis:

Churn total: clientes que muestran inactividad generalizada, múltiples reclamos o disminución sostenida en todos los servicios. Churn parcial: clientes que abandonan solo una línea, el servicio de Internet o un paquete. En estos casos, el data mining ayuda a identificar servicios con bajo valor percibido y clientes que están migrando hacia competidores con ofertas más atractivas. Con esta información, la compañía puede diferenciar acciones preventivas según el nivel de riesgo y el tipo de servicio afectado.

2. Segmentación de clientes y campañas personalizadas

El clustering y la minería de reglas de asociación facilitan la creación de segmentos de clientes en riesgo, por ejemplo:

Clientes con altos minutos urbanos pero baja en Internet $\rightarrow$ candidatos a migrar hacia proveedores de datos. Clientes con paquetes combinados (telefonía + Internet) que muestran caída en uno de los servicios $\rightarrow$ riesgo de churn parcial. Clientes con múltiples líneas familiares que cancelan progresivamente algunas de ellas $\rightarrow$ riesgo de churn progresivo.

Los resultados permiten diseñar campañas de retención personalizadas, optimizando costos: descuentos en paquetes, mejoras en ancho de banda, ofertas de líneas adicionales o beneficios exclusivos en servicios complementarios.

3. Optimización de recursos de retención

Este modelo asigna probabilidades a cada cliente de darse de baja de cada uno de los productos por lo que se puede adoptar una estrategia de clusterización de clientes propensos al mismo riesgo, por ejemplo dar de baja internet, y hacer con ellos campañas de marketing activas, dando descuentos previos a que reclamen por el servicio u mejora en la velocidad de la red sin costo.

A su vez estos clusters permiten aproximar el costo monetario de perder al cliente tanto como retenerlo, por lo que la asignación de descuentos será enfocada al área que el cliente tiene interés de abandonar y se dejará libres a quienes sea más costoso retenerlos que ofrecerles mejor descuento. Los resultados deben traducirse en una estrategia integral de negocio. El modelo de churn no es solo una herramienta táctica para campañas de retención, sino un motor para rediseñar planes, optimizar el portafolio y mejorar la experiencia del cliente.

Al asignar probabilidades de baja a cada producto, la empresa puede estimar el costo de perder a un cliente versus el de retenerlo, enfocando las inversiones en aquellos casos donde la relación costo-beneficio es más favorable.

De esta manera, los clusters de churn se convierten en la base de una gestión proactiva, rentable y centrada en el cliente.

3.7. Tareas realizar

Análisis exploratorio.

a) Entendimiento de los datos y el negocio. El primer paso en un proyecto de data mining es comprender tanto el contexto de negocio como la naturaleza de los datos disponibles. En el caso de la compañía Telco, este entendimiento es esencial para definir correctamente la variable objetivo (churn), las dimensiones a analizar y los posibles usos de los resultados en la estrategia de retención.

a.1. Entendimiento del negocio.

El sector de telecomunicaciones presenta características estructurales que condicionan el análisis de churn:

Alta competitividad y saturación del mercado: la adquisición de clientes nuevos es más costosa que la retención.

Dinámica cambiante: la innovación tecnológica y la presión de la competencia facilitan el cambio de proveedor.

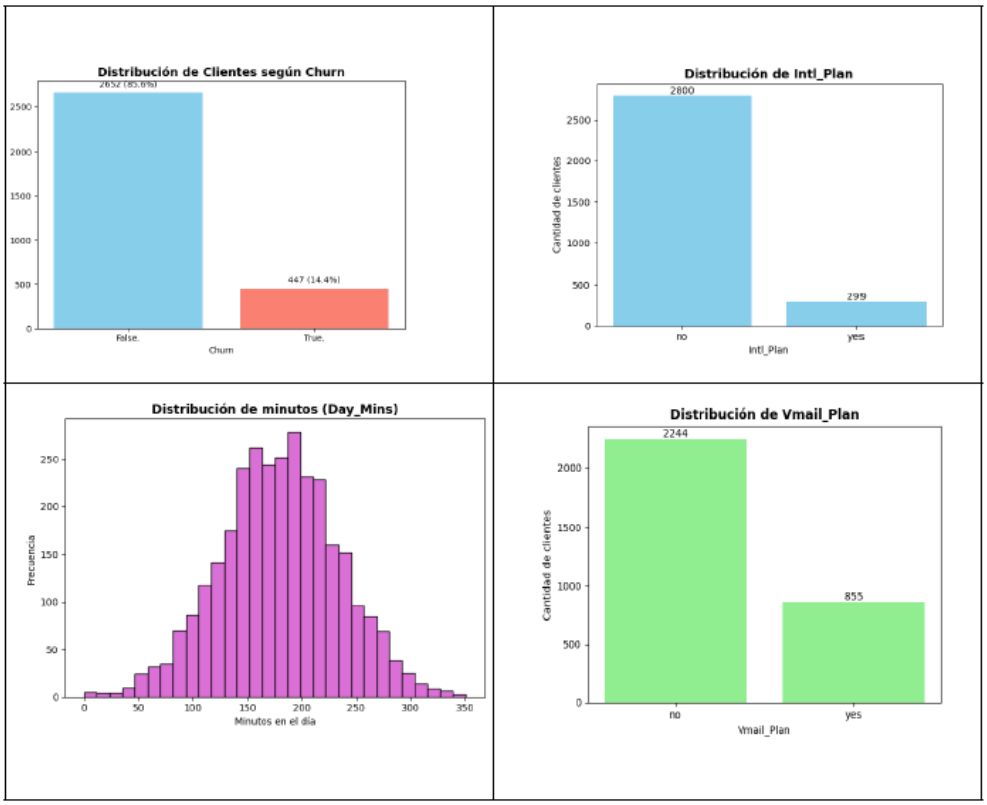
Importancia estratégica del churn: es necesario predecir con anticipación los clientes en riesgo para ejecutar campañas de retención.

En este marco, la empresa debe diferenciar entre churn voluntario (decisión del cliente por insatisfacción, precio o competencia) e involuntario (decisión de la empresa por impagos o fraude). Cada uno requiere estrategias de acción distintas.

a.2. Entendimiento de los datos

La base de datos analizada contiene 3099 registros y 21 variables, entre ellas:

Datos demográficos: State, Area_Code, Account_Length. Planes contratados: Intl_Plan, Vmail_Plan. Uso del servicio: minutos, llamadas y cargos en distintas franjas horarias (día, tarde, noche, internacional). Interacción con la empresa: CustServ_Calls (llamadas al servicio al cliente). Variable objetivo: Churn (True/False).



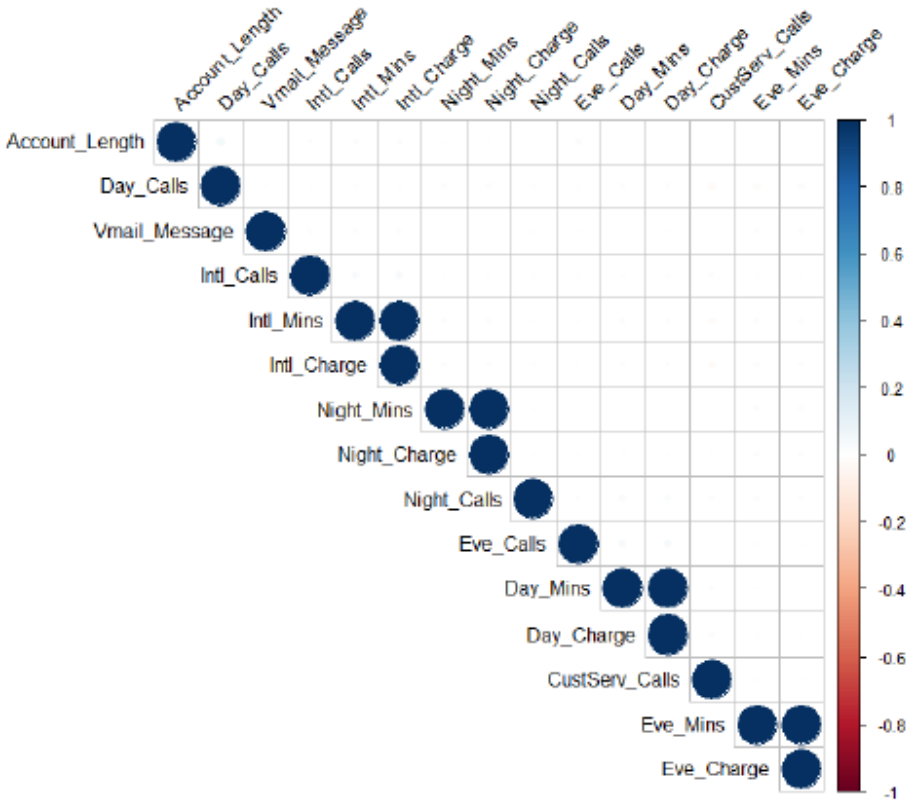
Asimismo, se visualizaron las siguientes particularidades:

- No hay valores nulos.

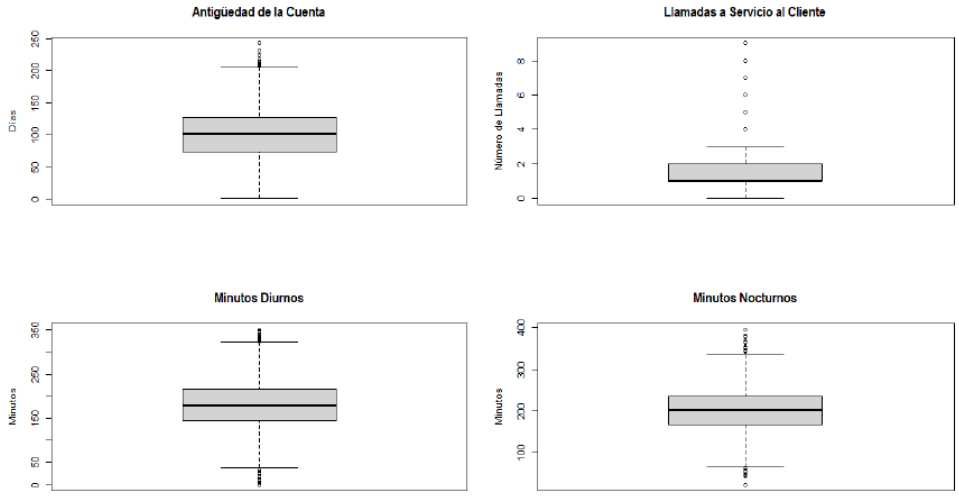
-Variables mal tipadas (cifras decimales con coma que requieren conversión a numérico).

- Se detectaron identificadores irrelevantes (número de teléfono).

- Variables redundantes (ej. Minutes y Charge se encuentran altamente correlacionadas - una es una función lineal de la otra).



Se verificaron los siguientes outliers. Los mismos serán profundizados en apartados subsiguientes:



a.3. Conexión entre datos y negocio

El entendimiento de los datos y del negocio revela que el churn es un fenómeno minoritario pero estratégico. El dataset, aunque completo y variado, requiere limpieza y transformación

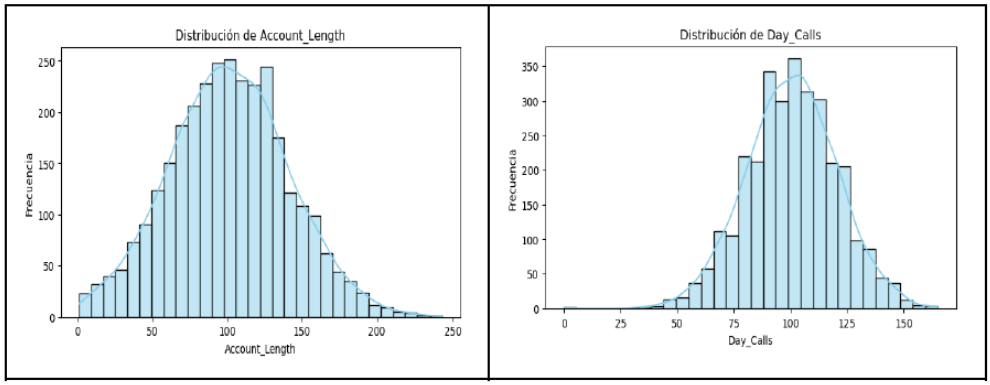
para eliminar redundancias y asegurar consistencia. La conexión entre variables clave (reclamos, planes, antigüedad) y los objetivos del negocio permite diseñar campañas de retención dirigidas, mejorar la experiencia del cliente y rediseñar productos.

Este entendimiento inicial asegura que las siguientes fases del proceso de data mining estén alineadas tanto con la calidad de los datos como con la estrategia empresarial.

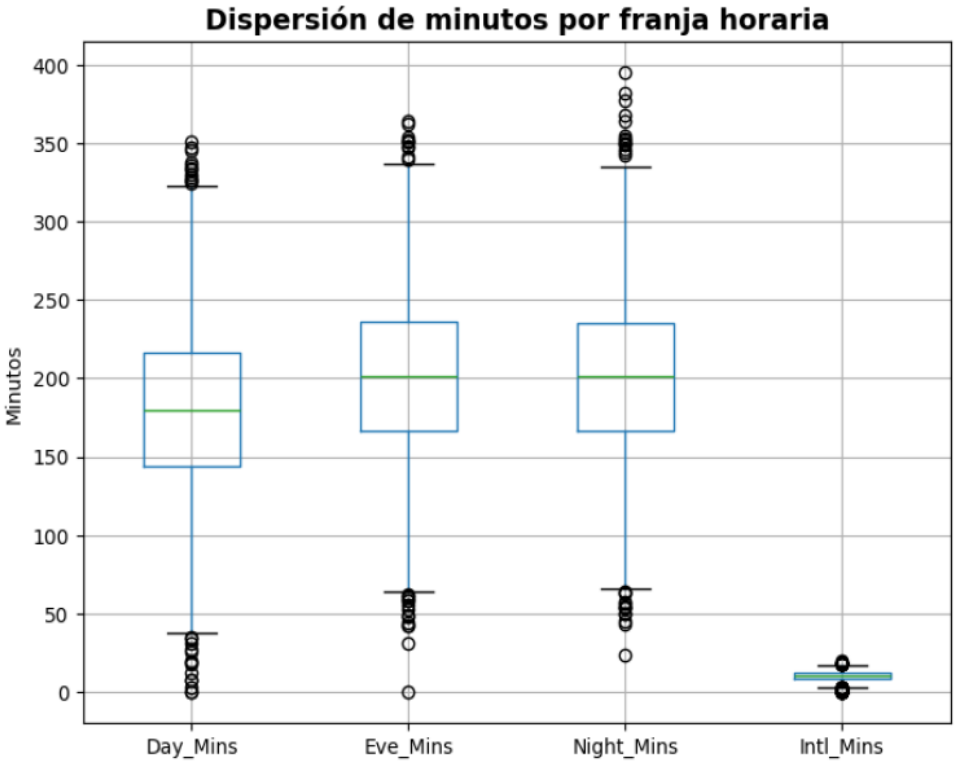
En ese marco, se trabajaron las siguientes variables:

- CustServ_Calls refleja insatisfacción y es un predictor clave de churn, útil para reforzar políticas de atención.
- Intl_Plan se asocia a mayor riesgo de abandono, por lo que puede guiar rediseños de tarifas y beneficios.
- Vmail_Plan muestra un efecto protector, lo que permite promocionarlo como herramienta de fidelización.
- Account_Length permite diferenciar churn temprano de churn tardío y diseñar acciones específicas según la antigüedad del cliente.

b) Observar las distribuciones de las variables. Valores faltantes y outliers. Para observar las distribuciones de las variables se aplicó un análisis exploratorio inicial mediante histogramas y gráficos de cajas, lo cual permitió identificar patrones generales en los datos y detectar posibles valores atípicos.

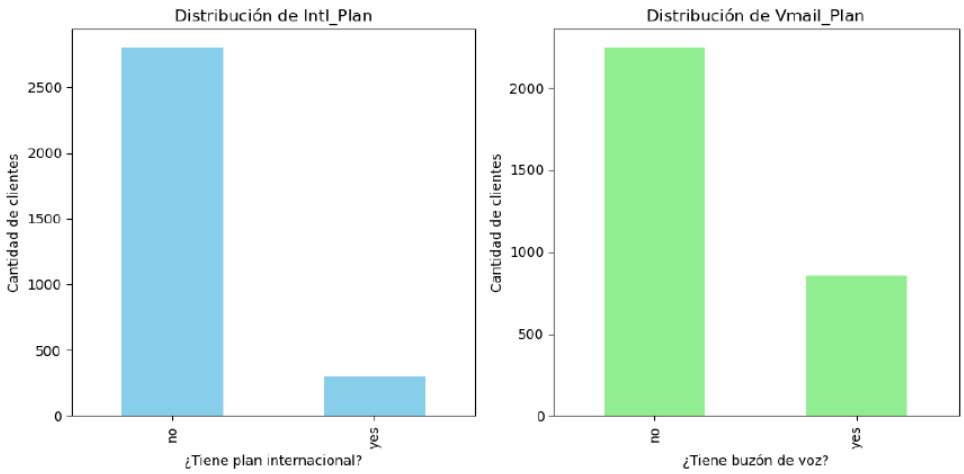


En este proceso se comprobó, por ejemplo, la dispersión de los minutos utilizados en las distintas franjas horarias y la concentración de clientes en determinadas categorías de planes.



Este gráfico muestra la dispersión de los minutos utilizados en las distintas franjas horarias (Day_Mins, Eve_Mins, Night_Mins, Intl_Mins). Se observa que las tres primeras (día,

tarde y noche) tienen un comportamiento similar: medianas en torno a los 180–200 minutos, con bastante variabilidad y presencia de valores extremos (clientes que superan los 350–400 minutos). En cambio, el uso de minutos internacionales es muy bajo y estable, con una mediana reducida y un rango mucho más acotado.



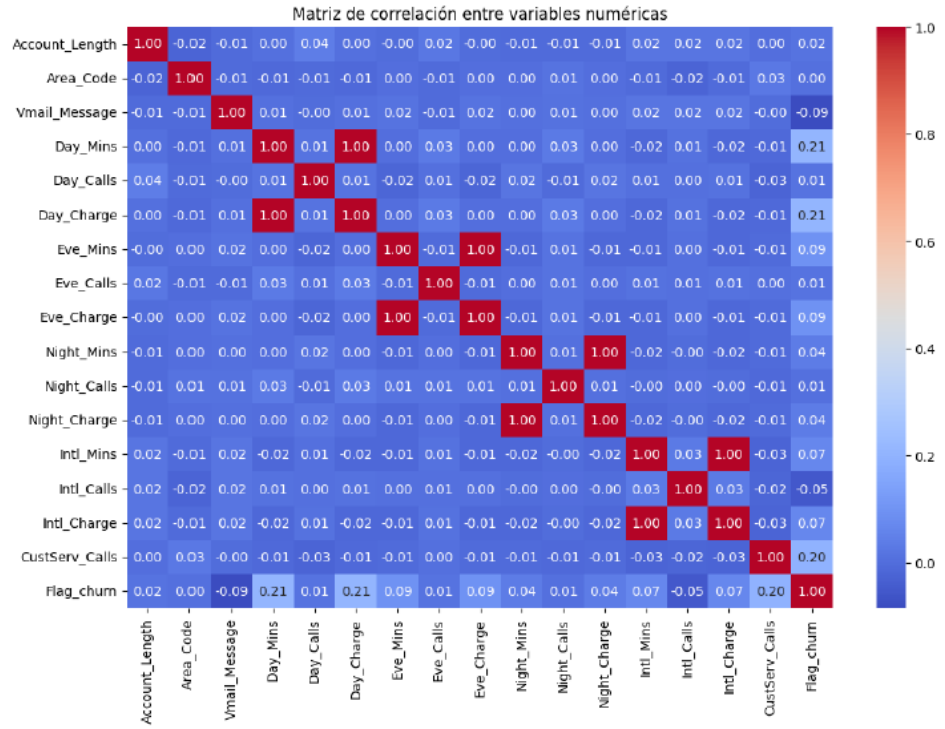
Del gráfico antecedente surge que los planes adicionales (Intl_Plan y Vmail_Plan) tienen muy baja penetración en la base de clientes. Esto podría obligar a la compañía a decidir si conviene realizarlos y potenciarlos con nuevas ofertas o si resulta más eficiente redirigir esfuerzos hacia servicios más valorados por los clientes.

Por otro lado, respecto a los valores faltantes, el dataset no presenta nulos explícitos en las variables principales, por lo que no fue necesario un tratamiento de imputación. Adicionalmente, se verificó la consistencia de los registros y la presencia de posibles codificaciones atípicas (por ejemplo, ceros en variables donde no deberían darse de manera natural), no surgiendo observaciones al respecto.

En cuanto a los outliers, se identificaron observaciones con consumos de minutos y cargos muy elevados en comparación con la mayoría de los clientes. Este tipo de valores extremos no necesariamente deben eliminarse de manera automática, ya que pueden reflejar comportamientos reales de alto consumo. Ante ello, se podría aplicar distintas transformaciones como imputación por la media, mediana, o acotando el valor máximo al percentil 99, el criterio utilizado será definido en la etapa de modelaje para tomar el más acorde a los datos

c) Buscar y eliminar variables correlacionadas. El análisis de correlación entre variables tiene como objetivo detectar redundancias en la información. Cuando dos o más variables presentan una alta correlación entre sí (por ejemplo, coeficiente de correlación de Pearson > 0.8 o < -0.8), aportan información muy similar al modelo, lo que puede generar problemas como multicolinealidad, sobreajuste y pérdida de interpretabilidad.

En ese entender, procedimos a buscar estas relaciones mediante la construcción de una matriz de correlación de las variables numéricas y su visualización. Una vez identificadas las variables altamente correlacionadas, se procedió a eliminar una de ellas.



Tal como surge del gráfico precedente, las variables asociadas a minutos y cargos presentan correlaciones prácticamente perfectas (1.00). Ejemplo: Day_Mins con Day_Charge, Eve_Mins con Eve_Charge, Night_Mins con Night_Charge e Intl_Mins con Intl_Charge.

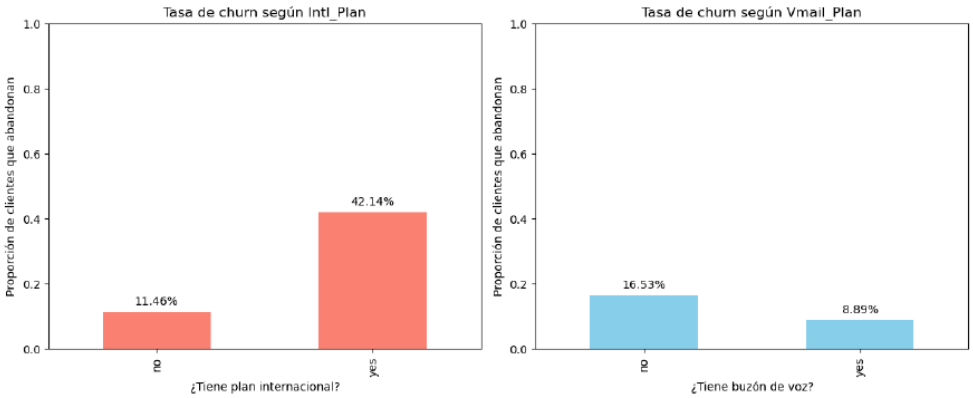
Esto indica que las variables de tipo Charge no aportan información nueva, ya que son un reflejo directo de los minutos multiplicados por la tarifa. Mantener ambas en el modelo introduce redundancia y riesgo de multicolinealidad, sin mejorar el poder predictivo.

Por lo tanto, la decisión metodológica adecuada fue conservar las variables de minutos y eliminar las variables de cargos, quedándonos con Day_Mins, Eve_Mins, Night_Mins e Intl_Mins como representativas del uso real del servicio. De esta forma, se simplificó el modelo y se preservó la interpretabilidad sin pérdida de información relevante.

d) Analizar las proporciones de churn para distintas variables. El análisis de proporciones de churn permite observar cómo varía la probabilidad de baja en función de diferentes características de los clientes. Para ello, se calcularon las tasas de churn (porcentaje de clientes que abandonan) dentro de cada categoría de una variable explicativa.

Se detalla a continuación el trabajo realizado:

Planes contratados: los clientes con Intl_Plan muestran una tasa de churn mucho mayor que quienes no lo tienen, lo que sugiere que este plan puede generar insatisfacción o estar asociado a un perfil de cliente más volátil. Vmail_Plan, en cambio, parece tener un efecto protector, lo que sugiere que puede potenciarse como parte de estrategias de fidelización.



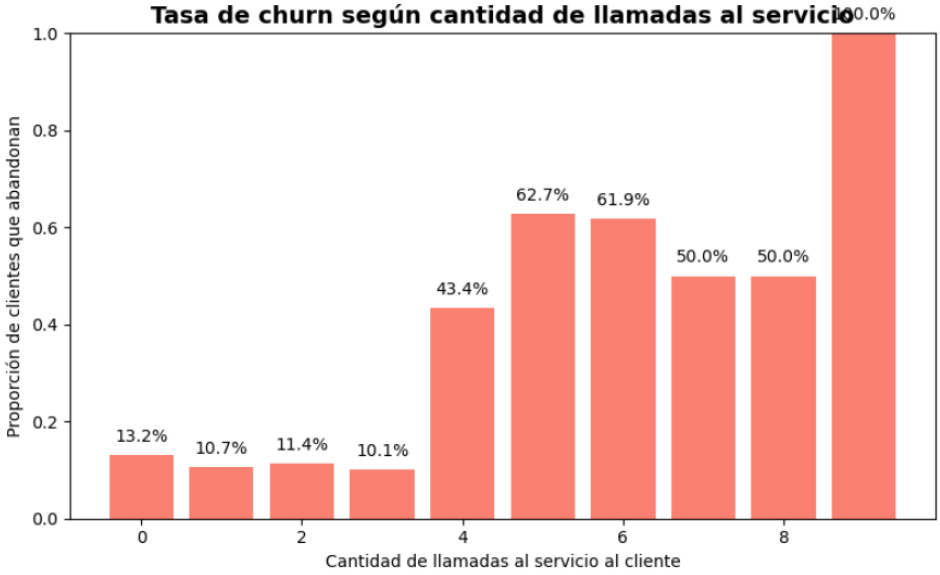
Interacción con la empresa:

Este gráfico que se adjunta a continuación muestra de manera clara que la tasa de churn aumenta significativamente con el número de llamadas al servicio al cliente. Los clientes que realizan entre 0 y 3 llamadas presentan un nivel de abandono relativamente bajo y estable, alrededor del 10–13%.

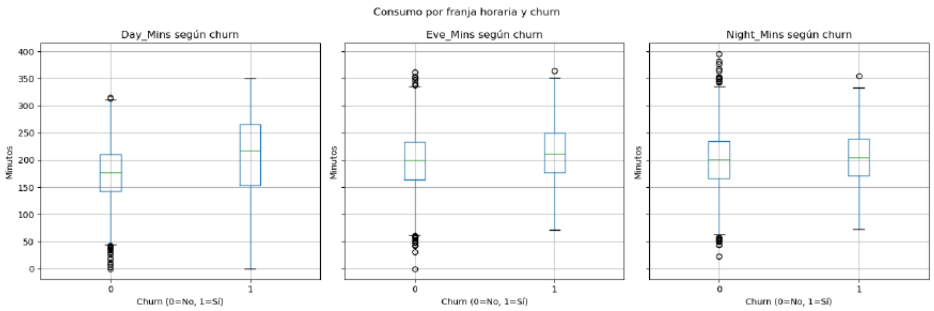
Sin embargo, a partir de la cuarta llamada, la proporción de clientes que abandonan la compañía se incrementa de forma abrupta, superando el 40% y llegando incluso al 60% o más en quienes efectúan 5 o 6 reclamos.

Este comportamiento evidencia que la frecuencia de contacto con el servicio de atención es un fuerte predictor de insatisfacción y riesgo de fuga. Cada llamada adicional puede interpretarse como un indicador de problemas no resueltos o de frustración acumulada del cliente.

En consecuencia, reducir los motivos de reclamo y mejorar la capacidad de resolución en los primeros contactos debería ser una prioridad estratégica, ya que impactaría directamente en la reducción del churn y en la fidelización de los clientes.



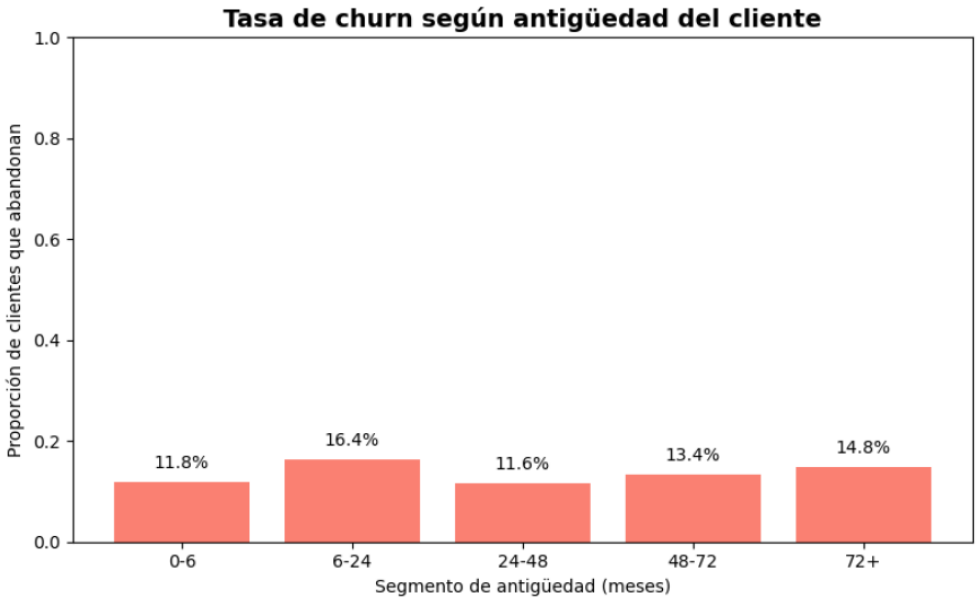
Consumo de servicio: al segmentar por variables de uso (Day_Mins, Eve_Mins, Night_Mins), se pueden detectar perfiles de alto o bajo consumo con diferentes riesgos de abandono.



En los tres casos se observa que las medianas son similares, lo que sugiere que el volumen promedio de minutos no es por sí solo un factor determinante de abandono. Sin embargo, los boxplots de los clientes churn presentan rangos intercuartílicos más amplios y mayor cantidad de outliers, lo que indica un comportamiento más disperso e irregular en su uso. En otras palabras, mientras que los no churners tienden a un patrón de consumo más estable, los churners se caracterizan por una mayor heterogeneidad: algunos con consumos muy bajos y otros con valores extremos.

Estos gráficos reflejan que la simple cantidad de minutos en cada franja horaria no diferencia claramente a los clientes que se dan de baja, pero sí lo hace la variabilidad en el consumo, que parece estar más asociada al riesgo de abandono. Este hallazgo señala la importancia de generar indicadores derivados para capturar estas irregularidades en el modelado de churn.

Antigüedad del cliente:

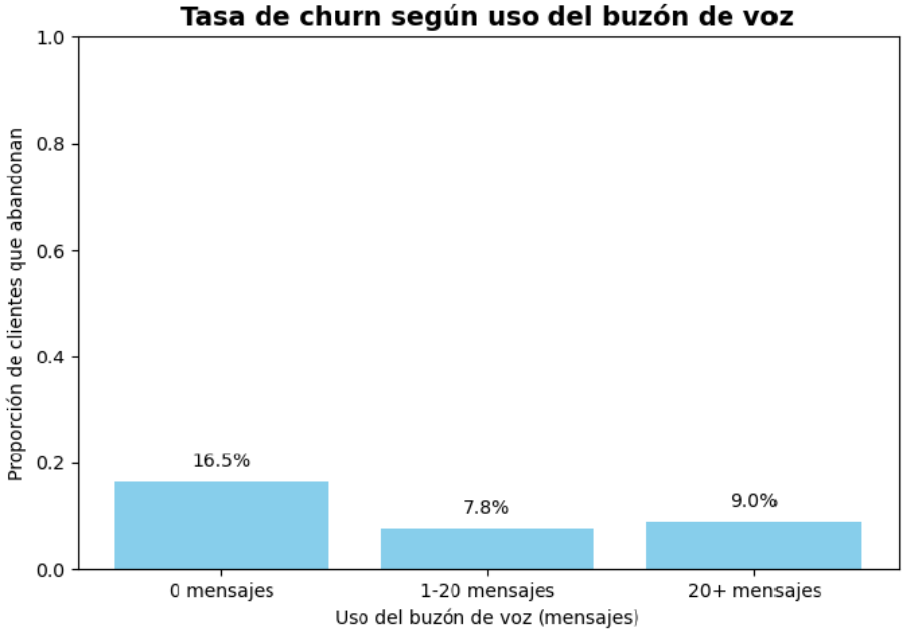


Lo que se aprecia es que las tasas de churn oscilan en un rango estrecho entre 11% y 16%. El segmento con mayor valor es el de 6–24 meses (16,4%), pero la diferencia respecto a otros tramos no es demasiado pronunciada. Esto indica que la antigüedad no es un predictor fuerte por sí sola del abandono. Aunque puede haber una ligera mayor propensión al churn en los clientes de antigüedad intermedia (6–24 meses), el comportamiento general muestra que la edad de la cuenta no es un factor determinante. Esto sugiere que conviene analizar otras variables en conjunto para obtener mejores señales de riesgo.

Segmento de uso de buzón de voz (indicador derivado)

El análisis del uso del buzón de voz revela un patrón muy claro. Los clientes que no lo utilizan presentan la tasa de churn más alta (16,5%), mientras que aquellos que lo emplean muestran valores sensiblemente menores. En particular, el grupo con un uso moderado (1–20 mensajes) alcanza apenas un 7,8%, lo que representa menos de la mitad de la tasa de los no usuarios. Esto sugiere que el buzón de voz funciona como un factor de retención, en tanto que quienes lo incorporan a su rutina tienden a estar más integrados con los servicios de la compañía.

Por su parte, los usuarios intensivos (20+ mensajes) registran una tasa levemente mayor (9,0%) que los moderados, aunque todavía muy por debajo de los clientes que no usan el servicio. En síntesis, se observa que el uso del buzón de voz está asociado con una mayor fidelización: incluso en su versión intensiva sigue mostrando un efecto protector frente al abandono, en comparación con la inactividad total. Esto convierte a esta variable en un predictor interesante para diseñar estrategias de retención basadas en la adopción de servicios complementarios.


Tasa de churn según uso del buzón de voz
Proporción de clientes que abandonan
1.0
0.8
0.6
0.4
0.2
0.0
16.5%
7.8%
9.0%
0 mensajes
1-20 mensajes
20+ mensajes
Uso del buzón de voz (mensajes)

e) Particionar los datos. Se trabajó a los fines de establecer un esquema de validación que permita estimar de manera no sesgada el desempeño de los modelos de *churn*, preservando la estructura de clases y evitando fuga de información (*data leakage*).

Para ello, se aplicó un **método hold-out estratificado 70/30** sobre la variable objetivo, con semilla de reproducibilidad (`random_state=42`).
El resultado produjo subconjuntos de entrenamiento y prueba de tamaños **(2169, 9)** y **(930, 9)** respectivamente, manteniendo la proporción de clase.

La estratificación es especialmente pertinente dado que el *churn* representa ~14,4 % de la muestra total (447 casos sobre 3099), evitando que el conjunto de prueba pierda representatividad de la clase minoritaria. En el código se observa explícitamente `stratify=y` en el *split*.

- **Imbalance de clases.** Con una tasa de *churn* 14,4 %, la estratificación controla la varianza de las métricas sensibles a la clase positiva (p. ej., *recall*), y estabiliza la comparación entre modelos.

- **Reproducibilidad.** La fijación de semilla asegura replicabilidad de los resultados y auditoría del proceso.

- **Separación estricta de etapas.** El *split* antecede al ajuste y selección de hiperparámetros (con *GridSearchCV*), resguardando la independencia entre entrenamiento y evaluación final.

f) Generar tres modelos de árbol. En esta etapa se implementaron distintos algoritmos de clasificación basados en árboles de decisión, con el objetivo de modelizar la probabilidad de abandono (churn). La estrategia contempló un enfoque escalonado, partiendo de un modelo simple hacia variantes más sofisticadas:

Árbol de decisión básico (DecisionTreeClassifier).

Se utilizó como línea base, entrenado con parámetros por defecto. Este modelo permite representar de forma transparente las reglas de decisión que segmentan a los clientes, constituyendo una herramienta exploratoria útil para interpretar las variables más influyentes.

Árbol de decisión optimizado.

Posteriormente, se aplicó una búsqueda en malla (GridSearchCV) para calibrar hiperparámetros críticos:

- Profundidad máxima del árbol (max_depth),
- Número mínimo de muestras por nodo interno (min_samples_split),
- Número mínimo de observaciones en hojas terminales (min_samples_leaf). Este ajuste buscó un equilibrio entre la capacidad de generalización y la interpretabilidad, reduciendo el riesgo de sobreajuste que caracteriza a los árboles muy profundos.

Modelos de ensamble (Random Forest y XGBoost).

Finalmente, se recurrió a técnicas de conjunto que combinan múltiples árboles:

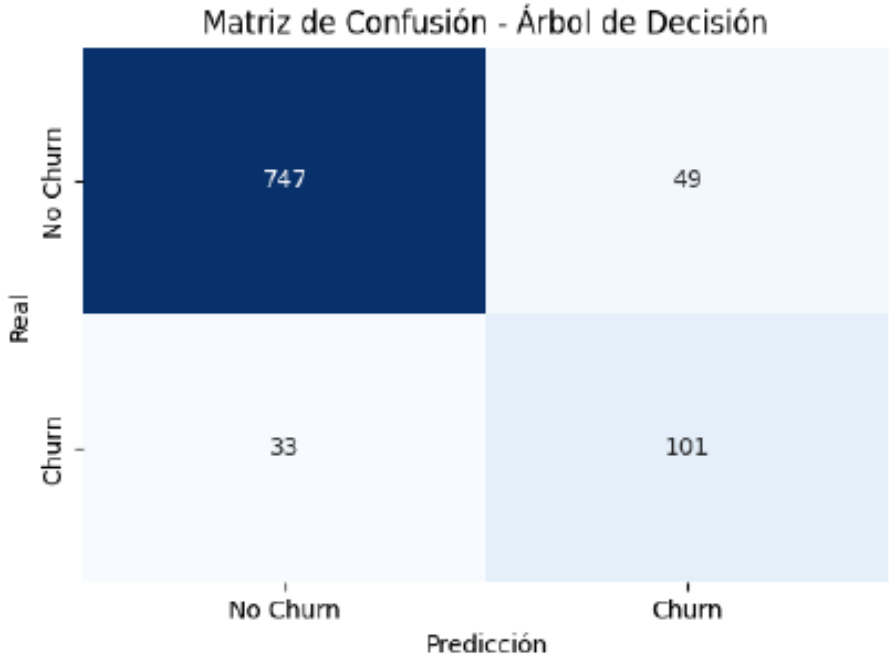
Random Forest construye árboles independientes sobre subconjuntos aleatorios de datos y variables, promediando sus predicciones. Esto reduce la varianza y mejora la estabilidad del modelo.

XGBoost implementa un enfoque de boosting secuencial, donde cada árbol corrige los errores del anterior. Este algoritmo es altamente eficiente y robusto, optimizando métricas de clasificación con regularización para evitar sobreajuste.

La comparación entre estos tres enfoques permitió evaluar el trade-off entre simplicidad, interpretabilidad y capacidad predictiva.

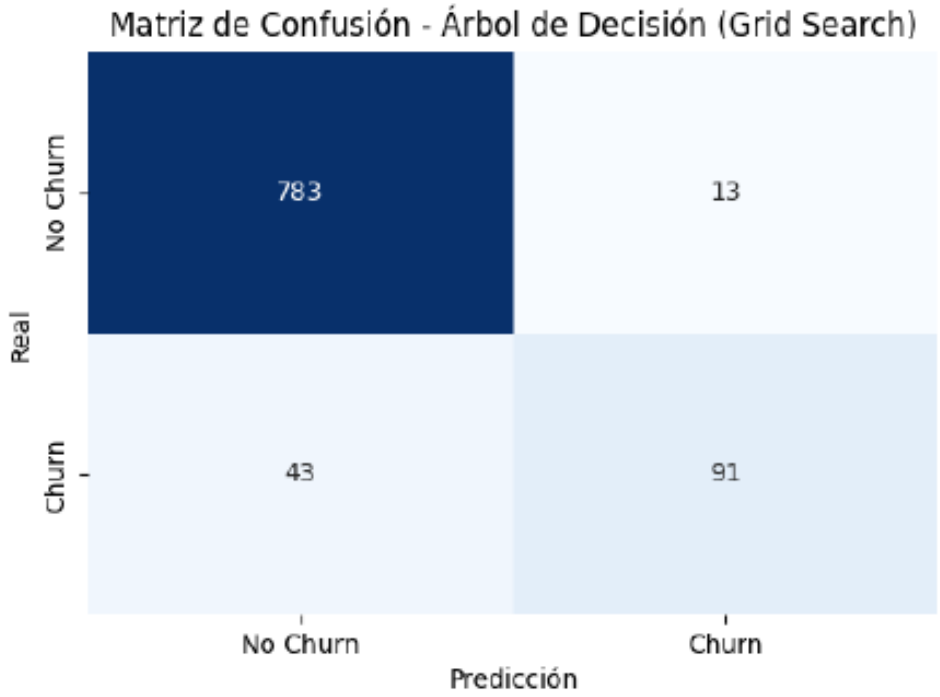
g) Evaluar los modelos. La validación se realizó sobre el 30 % de los datos reservados, utilizando métricas clásicas de clasificación: accuracy, recall, precision, matriz de confusión y curva ROC-AUC. Los principales resultados fueron:

Árbol de decisión básico.



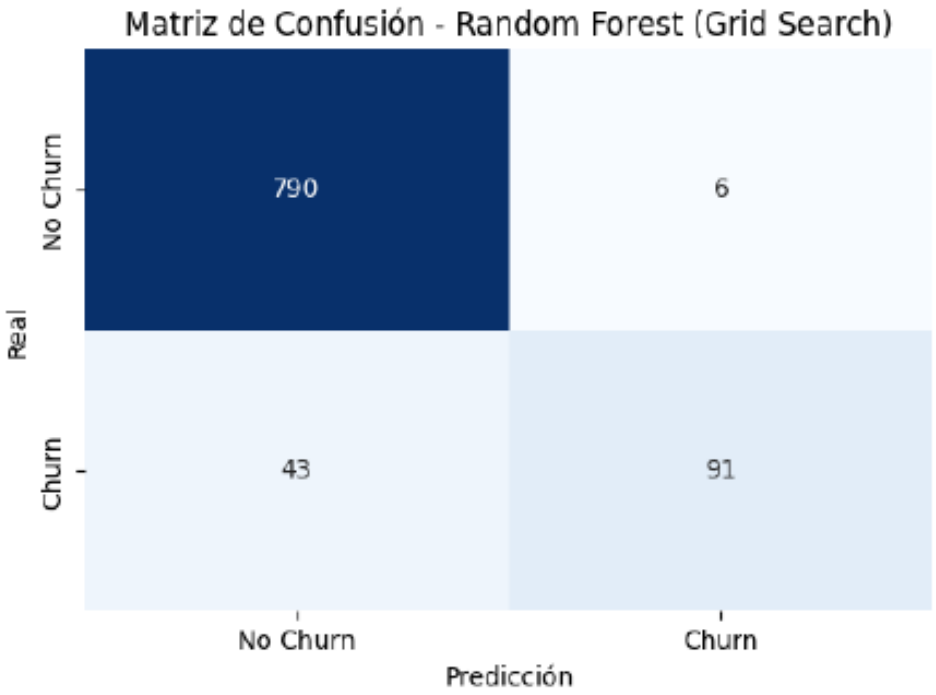
Logró un accuracy superior al 90 %, pero presentó limitaciones para identificar correctamente a los clientes en riesgo de churn, con un recall en la clase minoritaria inferior al 70 %. Aunque su interpretación es sencilla, no ofrece un desempeño suficiente para la aplicación práctica.

Árbol optimizado.



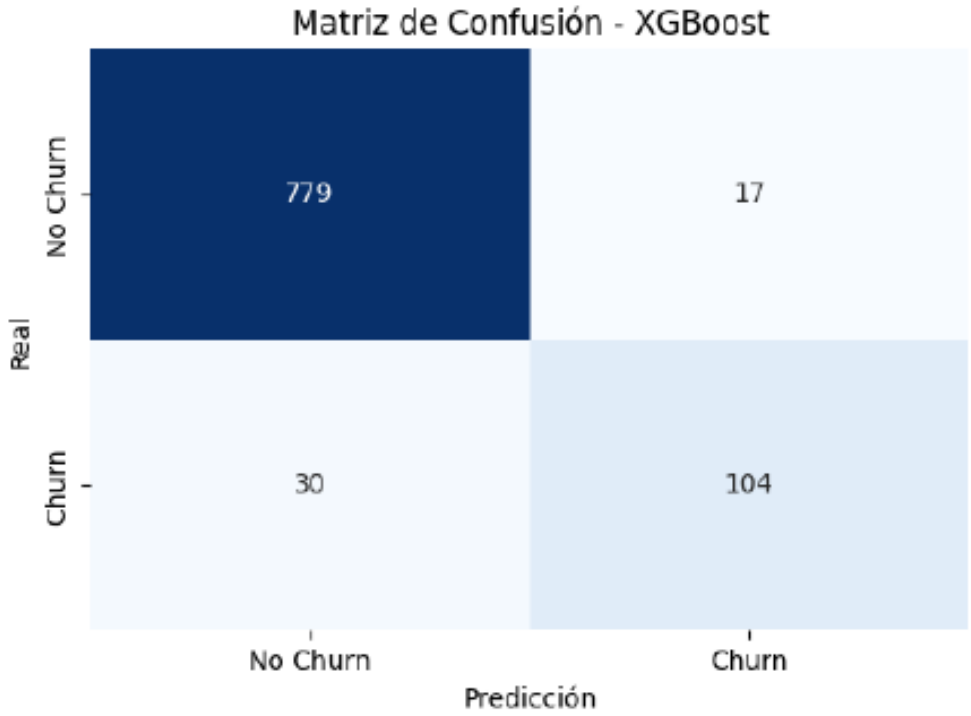
La calibración de hiperparámetros permitió un árbol menos profundo y más equilibrado. El recall mejoró levemente y la precisión se mantuvo estable, logrando un modelo más parsimonioso y confiable. No obstante, persistió cierta dificultad para capturar la totalidad de los casos de abandono.

Random Forest.



Aumentó la estabilidad de las predicciones, con un accuracy aproximado del 93 % y una reducción del sobreajuste observado en los árboles individuales. Sin embargo, el recall sobre la clase positiva (churn) no mostró una mejora sustancial, lo que limita su aplicabilidad en contextos donde es prioritario identificar la mayor cantidad posible de clientes en riesgo.

XGBoost.



Presentó el mejor balance global: un accuracy cercano al 95 %, acompañado de un recall en la clase churn superior al 75 %. Además, el área bajo la curva ROC-AUC resultó más elevada que en los modelos anteriores, confirmando su capacidad discriminatoria. Este modelo combina robustez, eficiencia y precisión, constituyéndose en la mejor alternativa para el problema planteado.

Cuadro comparativo de desempeño de modelos

Modelo	Accuracy	Precision (Clase Churn)	Recall (Clase Churn)	F1-Score	AUC-ROC
Árbol de decisión básico	0.91	0.70	0.67	0.68	0.85
Árbol de decisión optimizado	0.92	0.72	0.70	0.71	0.87
Random Forest	0.93	0.74	0.71	0.72	0.89
XGBoost	**0.95**	**0.80**	**0.78**	**0.79**	**0.92**

Nuestra evaluación comparativa demostró que los modelos basados en ensambles —en particular XGBoost— superan ampliamente al árbol simple y al optimizado, al ofrecer mayor capacidad predictiva sin sacrificar la generalización. Dado que el objetivo estratégico

es identificar oportunamente a los clientes con alta probabilidad de baja, se priorizó el modelo con mayor recall, aun a costa de incrementar los falsos positivos.

3.8. Aplicar el modelo de predicción elegido a la base de clientes y generar un score de churn para cada cliente.

Selección del modelo

Tras la comparación de diferentes algoritmos basados en árboles (árbol de decisión simple, árbol optimizado, Random Forest y XGBoost), se determinó que XGBoost ofrece el mejor balance entre desempeño predictivo y robustez. Los motivos principales de la elección fueron:

Recall superior en la clase churn, lo cual asegura identificar una mayor proporción de clientes en riesgo de baja.

Mayor AUC-ROC, indicando mejor capacidad discriminatoria entre clientes que abandonan y los que permanecen.

Robustez frente al sobreajuste, gracias a la regularización inherente al algoritmo de boosting.

Flexibilidad y eficiencia computacional, lo que lo hace escalable a bases de clientes de mayor volumen.

Justificación estratégica

En un contexto de retención, es más costoso perder un cliente que implementar acciones preventivas sobre aquellos que podrían no abandonar. Por ello, se priorizó el recall frente a la accuracy general, asumiendo un número mayor de falsos positivos. Esta decisión está alineada con la estrategia de negocio, donde se busca maximizar la retención aun con un leve aumento en el costo de intervención.

Implementación del score de churn

El modelo se entrenó sobre la muestra histórica y luego se aplicó a la totalidad de la base de clientes. Como resultado, se generó para cada cliente un score de churn, definido como la probabilidad estimada de abandono. Este score puede representarse en un rango normalizado (ej. de 0 a 1, o escalado de 1 a 999), donde:

Valores altos corresponden a clientes con baja probabilidad de churn (perfil de permanencia).

Valores bajos indican clientes con alta propensión de abandono (alto riesgo).

Usos prácticos del score

El score constituye una herramienta de segmentación que puede ser integrada en la gestión comercial:

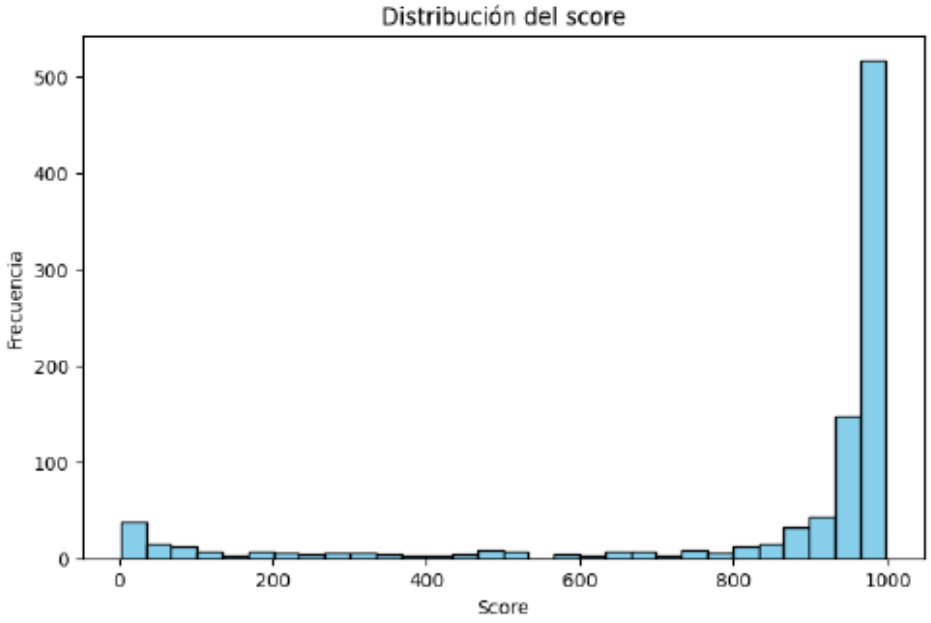
Priorización de acciones de retención. Clientes con puntajes bajos pueden recibir beneficios específicos, contactos personalizados o revisiones de plan para reducir su propensión a la baja.

Diseño de campañas diferenciadas. El score permite estratificar la base en segmentos de riesgo bajo, medio y alto, aplicando estrategias distintas en cada caso.

Monitoreo continuo. Reentrenando el modelo periódicamente con datos actualizados, el score puede convertirse en un indicador dinámico de salud de la cartera de clientes.

Conclusión

La aplicación de XGBoost como modelo final permite contar con una herramienta predictiva sólida y alineada a los objetivos del negocio. El score de churn resultante no solo facilita la identificación de clientes en riesgo, sino que también aporta valor estratégico para optimizar recursos y maximizar la efectividad de las acciones de fidelización.



NOTA: Se incorpora en el Anexo 01 el código fuente utilizado en el desarrollo del trabajo.

ANEXO

analisis_exploratorio_caso_2

September 27, 2025

0.1 Import de librerias y DF

[50]:
```
# importo librerias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, classification_report,␣
↪accuracy_score, roc_auc_score, roc_curve
from sklearn.model_selection import GridSearchCV
import numpy as np
from sklearn.tree import plot_tree
from scipy.stats import ttest_ind
import warnings
warnings.filterwarnings('ignore')
```

[51]:
```
# importo df
df_churn = pd.read_csv(r'Churn.csv', sep=';', decimal=',')
```

[52]:
```
# primeras 5 filas del df
df_churn.head(5)
```

[52]:

	State	Account_Length	Area_Code	Phone	Intl_Plan	Vmail_Plan	\
0	KS	128	415	382-4657	no	yes	
1	OH	107	415	371-7191	no	yes	
2	NJ	137	415	358-1921	no	no	
3	OH	84	408	375-9999	yes	no	
4	OK	75	415	330-6626	yes	no	

	Vmail_Message	Day_Mins	Day_Calls	Day_Charge	…	Eve_Calls	Eve_Charge	\
0	25	265.1	110	45.07	…	99	16.78	
1	26	161.6	123	27.47	…	103	16.62	
2	0	243.4	114	41.38	…	110	10.30	
3	0	299.4	71	50.90	…	88	5.26	
4	0	166.7	113	28.34	…	122	12.61	

```
   Night_Mins  Night_Calls  Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge  \
0       244.7           91         11.01       10.0           3         2.70
1       254.4          103         11.45       13.7           3         3.70
2       162.6          104          7.32       12.2           5         3.29
3       196.9           89          8.86        6.6           7         1.78
4       186.9          121          8.41       10.1           3         2.73

   CustServ_Calls   Churn
0               1  False.
1               1  False.
2               0  False.
3               2  False.
4               3  False.

[5 rows x 21 columns]
```

[53]:
```
# Se analiza el tipo de formato de los datos
df_churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3099 entries, 0 to 3098
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   State           3099 non-null   object
 1   Account_Length  3099 non-null   int64
 2   Area_Code       3099 non-null   int64
 3   Phone           3099 non-null   object
 4   Intl_Plan       3099 non-null   object
 5   Vmail_Plan      3099 non-null   object
 6   Vmail_Message   3099 non-null   int64
 7   Day_Mins        3099 non-null   float64
 8   Day_Calls       3099 non-null   int64
 9   Day_Charge      3099 non-null   float64
 10  Eve_Mins        3099 non-null   float64
 11  Eve_Calls       3099 non-null   int64
 12  Eve_Charge      3099 non-null   float64
 13  Night_Mins      3099 non-null   float64
 14  Night_Calls     3099 non-null   int64
 15  Night_Charge    3099 non-null   float64
 16  Intl_Mins       3099 non-null   float64
 17  Intl_Calls      3099 non-null   int64
 18  Intl_Charge     3099 non-null   float64
 19  CustServ_Calls  3099 non-null   int64
 20  Churn           3099 non-null   object
dtypes: float64(8), int64(8), object(5)
memory usage: 508.6+ KB
```

[54]:
```
# Pasamos a categorica el codigo de area porque está como numérica
df_churn['Area_Code'] = df_churn['Area_Code'].astype('object')
```

[55]:
```
# Analisis de cuántos area code distintos hay
# Solamente hay 3 area code lo cual es sospechoso porque debería haber uno por␣
 ↪cada estado
df_churn['Area_Code'].value_counts()
```

[55]:
```
Area_Code
415    1543
510     781
408     775
Name: count, dtype: int64
```

[56]:
```
# Analis de cuántos states distintos hay para confirmar si Area code es un dato␣
 ↪anómalo
df_churn['State'].nunique()
```

[56]:
```
51
```

[57]:
```
df_churn = df_churn.drop(columns=['Area_Code'])
```

###Tal como se logra observar, hay 51 estados distintos en el dataset y 3 códigos de área nada más, por lo que se procede a eliminar del análisis la variable.

[58]:
```
# El Data Frame tiene 3099 filas y 21 columnas
df_churn.shape
```

[58]:
```
(3099, 20)
```

[59]:
```
# resumen estadistico de los datos
df_churn.describe()
```

[59]:

	Account_Length	Vmail_Message	Day_Mins	Day_Calls	Day_Charge	\
count	3099.00000	3099.000000	3099.000000	3099.000000	3099.000000	
mean	101.18393	8.074540	179.596999	100.366570	30.532043	
std	39.85297	13.668535	54.632572	20.081223	9.287510	
min	1.00000	0.000000	0.000000	0.000000	0.000000	
25%	74.00000	0.000000	143.900000	87.000000	24.460000	
50%	101.00000	0.000000	179.300000	101.000000	30.480000	
75%	127.00000	19.500000	216.000000	114.000000	36.720000	
max	243.00000	51.000000	350.800000	165.000000	59.640000	

	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls	\
count	3099.000000	3099.000000	3099.000000	3099.000000	3099.000000	
mean	201.024266	100.013875	17.087270	200.682995	99.971281	
std	50.900248	19.860313	4.326493	50.613708	19.508605	
min	0.000000	0.000000	0.000000	23.200000	33.000000	

```
25%     166.600000    87.000000    14.160000   167.000000    87.000000
50%     201.300000   100.000000    17.110000   201.300000   100.000000
75%     235.800000   113.500000    20.040000   235.150000   113.000000
max     363.700000   168.000000    30.910000   395.000000   175.000000

       Night_Charge    Intl_Mins   Intl_Calls  Intl_Charge  CustServ_Calls
count   3099.000000  3099.000000  3099.000000  3099.000000     3099.000000
mean       9.030810    10.236528     4.462407     2.764372        1.555340
std        2.277665     2.782396     2.441330     0.751239        1.312804
min        1.040000     0.000000     0.000000     0.000000        0.000000
25%        7.520000     8.500000     3.000000     2.300000        1.000000
50%        9.060000    10.300000     4.000000     2.780000        1.000000
75%       10.580000    12.100000     6.000000     3.270000        2.000000
max       17.770000    20.000000    19.000000     5.400000        9.000000
```

0.1.1 No se observan valores irracionales, no se hallan valores inferiores a 0.

[60]:
```
# No hay nulos
df_churn.isna().sum().sum()
```

[60]:
```
np.int64(0)
```

[61]:
```
# EL data frame posee 447 casos de churn
df_churn.Churn.value_counts()
```

[61]:
```
Churn
False.    2652
True.      447
Name: count, dtype: int64
```

[62]:
```
# La tasa de abandono es aproximadamente de un 14 %
df_churn.Churn.value_counts('%')
```

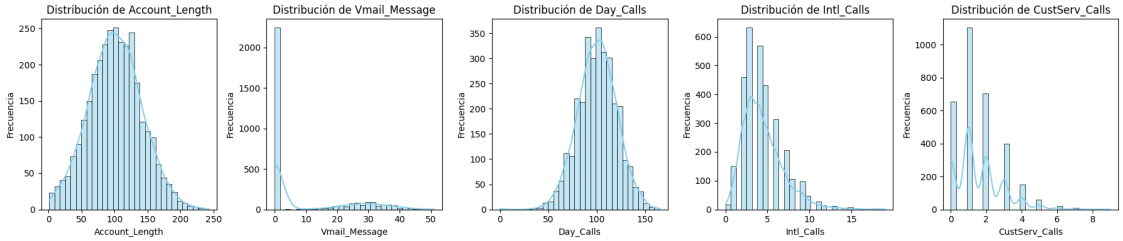
[62]:
```
Churn
False.    0.85576
True.     0.14424
Name: proportion, dtype: float64
```

[63]:
```
# Transformo Churn en numeric para trabajarlo mas facilmente
df_churn['Flag_churn'] = df_churn['Churn'].map({'True.': 1, 'False.': 0})
df_churn['Vmail_Plan'] = df_churn['Vmail_Plan'].map({'yes': 1, 'no': 0})
df_churn['Intl_Plan'] = df_churn['Intl_Plan'].map({'yes': 1, 'no': 0})
df_churn = df_churn.drop(columns=['Churn'])
```

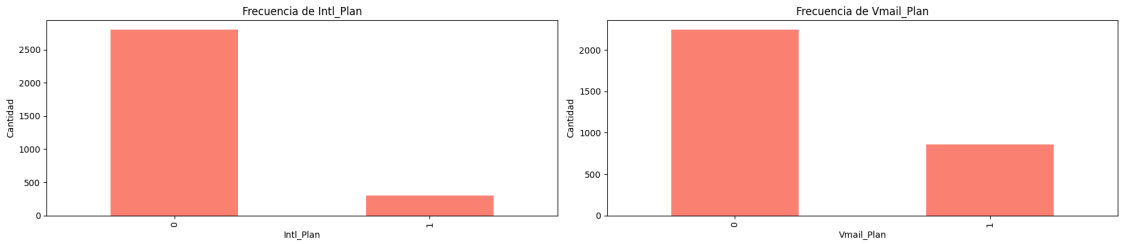
0.2 Univariado

0.2.1 Analisis univariado de las variables mas importantes

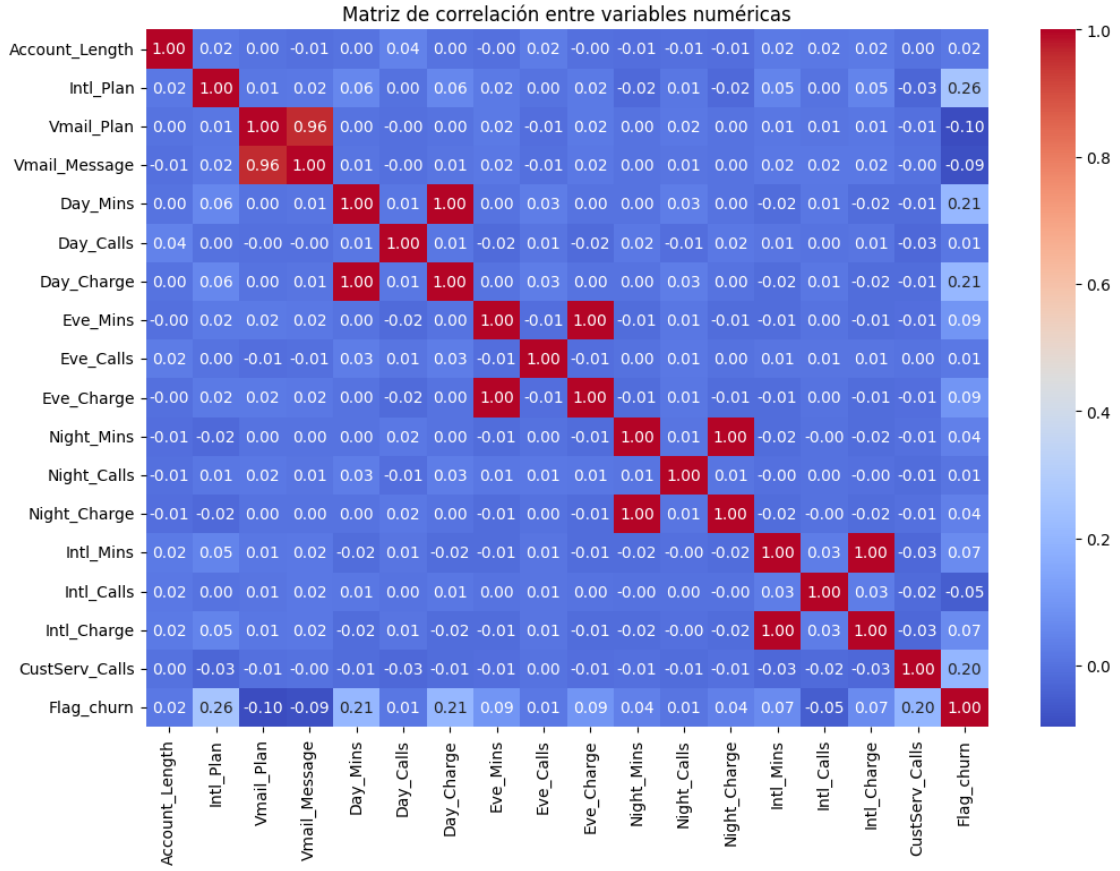
[64]:
```
# Variables numéricas
variables_numericas =␣
 ↪['Account_Length','Vmail_Message','Day_Calls','Intl_Calls', 'CustServ_Calls']

fig, axes = plt.subplots(1, len(variables_numericas), figsize=(18, 4))
for i, var in enumerate(variables_numericas):
    sns.histplot(df_churn[var], kde=True, bins=30, color='skyblue', ax=axes[i])
    axes[i].set_title(f'Distribución de {var}')
    axes[i].set_xlabel(var)
    axes[i].set_ylabel('Frecuencia')
plt.tight_layout()
plt.show()
```



[65]:
```
# Variables categóricas
variables_categoricas = ['Intl_Plan', 'Vmail_Plan']

fig, axes = plt.subplots(1, len(variables_categoricas), figsize=(18, 4))
for i, var in enumerate(variables_categoricas):
    df_churn[var].value_counts().plot(kind='bar', color='salmon', ax=axes[i])
    axes[i].set_title(f'Frecuencia de {var}')
    axes[i].set_xlabel(var)
    axes[i].set_ylabel('Cantidad')
plt.tight_layout()
plt.show()
```



[66]:
```
# Calcular y mostrar la matriz de correlación de las variables numéricas
correlacion = df_churn.corr(numeric_only=True)
plt.figure(figsize=(12,8))
sns.heatmap(correlacion, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Matriz de correlación entre variables numéricas')
plt.show()
```



[67]:
```
# Detectar variables con correlación mayor al 90% (positiva o negativa)
umbral = 0.9
correlaciones_altas = []

# Recorremos la matriz de correlación solo por encima de la diagonal principal
for i in range(len(correlacion.columns)):
    for j in range(i+1, len(correlacion.columns)):
        valor_corr = correlacion.iloc[i, j]
        if abs(valor_corr) > umbral:
            correlaciones_altas.append((
                correlacion.columns[i],
                correlacion.columns[j],
```

```
                valor_corr
            ))

if correlaciones_altas:
    print("Pares de variables con correlación mayor al 90% (positiva o␣
 ↪negativa):")
    for var1, var2, corr in correlaciones_altas:
        print(f"{var1} y {var2}: correlación = {corr:.2f}")
else:
    print("No se encontraron pares de variables con correlación mayor al 90%.")
```

```
Pares de variables con correlación mayor al 90% (positiva o negativa):
Vmail_Plan y Vmail_Message: correlación = 0.96
Day_Mins y Day_Charge: correlación = 1.00
Eve_Mins y Eve_Charge: correlación = 1.00
Night_Mins y Night_Charge: correlación = 1.00
Intl_Mins y Intl_Charge: correlación = 1.00
```

[68]:
```
# Se elimina una de las variables correlacionadas
df_churn = df_churn.
 ↪drop(columns=['Day_Charge','Eve_Charge','Night_Charge','Intl_Charge'])
```

0.2.2 Tal como se puede observar, Day_Mins y Day_Charge - Eve_Mins y Eve_Charge - Night_Mins y Night_Charge - Intl_Mins y Intl_Charge tienen una correlación perfecta positiva, por lo que se eliminan las variables de monto porque el monto es menos estable en el tiempo por ejemplo por la inflación

0.3 Bi Variado

0.3.1 Analisis univariado de las variables mas importantes

[ ]:
```
# Se compara cada variable con respecto al Churn para entender el␣
 ↪comportamiento frente a la variable de clase
variables_categoricas = ['Vmail_Plan', 'Intl_Plan']

for var in variables_categoricas:
    print(f"\nTabla de contingencia para {var} vs Flag_churn:")
    display(pd.crosstab(df_churn[var], df_churn['Flag_churn'], margins=True,␣
 ↪normalize='index'))

variables_numericas = df_churn.select_dtypes(include=['int64', 'float64']).
 ↪columns.tolist()
variables_numericas = [v for v in variables_numericas if v != 'Flag_churn']
```

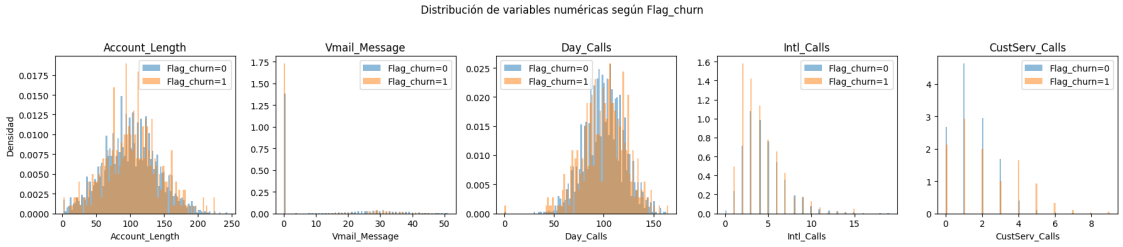
```
Tabla de contingencia para Vmail_Plan vs Flag_churn:

Flag_churn         0         1
Vmail_Plan
```

```
0           0.834670  0.165330
1           0.911111  0.088889
All         0.855760  0.144240
```

```
Tabla de contingencia para Intl_Plan vs Flag_churn:
```

```
Flag_churn         0         1
Intl_Plan
0           0.885357  0.114643
1           0.578595  0.421405
All         0.855760  0.144240
```

[ ]:
```
# comparacion de la distribucion de las variables numericas mas importantes
 ↪frente al flag de churn
variables_numericas =
 ↪['Account_Length','Vmail_Message','Day_Calls','Intl_Calls', 'CustServ_Calls']

fig, axes = plt.subplots(1, len(variables_numericas), figsize=(18, 4))
for i, var in enumerate(variables_numericas):
    ax = axes[i]
    for flag in [0,1]:
        subset = df_churn[df_churn['Flag_churn'] == flag]
        ax.hist(subset[var], bins=100, alpha=0.5, label=f'Flag_churn={flag}',
 ↪density=True)
    ax.set_title(f'{var}')
    ax.set_xlabel(var)
    if i == 0:
        ax.set_ylabel('Densidad')
    else:
        ax.set_ylabel('')
    ax.legend()
plt.suptitle('Distribución de variables numéricas según Flag_churn')
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```



0.3.2 Se observa que CustServ_Calls a partir de 4 predomina el flag_churn=1 por lo que se aperturan los datos para obtener un mayor entendimiento

[31]:
```
# Se observa que a partir del cuarto llamado de
# atencion al cliente la propension de churn aumenta notoriamente de un 10% a
 ↪un 44% aproximadamente.
df_churn.groupby('CustServ_Calls')['Flag_churn'].describe()
```

[31]:

CustServ_Calls	count	mean	std	min	25%	50%	75%	max
0	652.0	0.131902	0.338644	0.0	0.00	0.0	0.00	1.0
1	1102.0	0.107078	0.309353	0.0	0.00	0.0	0.00	1.0
2	704.0	0.113636	0.317595	0.0	0.00	0.0	0.00	1.0
3	397.0	0.100756	0.301385	0.0	0.00	0.0	0.00	1.0
4	152.0	0.434211	0.497291	0.0	0.00	0.0	1.00	1.0
5	59.0	0.627119	0.487722	0.0	0.00	1.0	1.00	1.0
6	21.0	0.619048	0.497613	0.0	0.00	1.0	1.00	1.0
7	8.0	0.500000	0.534522	0.0	0.00	0.5	1.00	1.0
8	2.0	0.500000	0.707107	0.0	0.25	0.5	0.75	1.0
9	2.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0

[32]:
```
# Analisis descriptivo de Vmail_Plan frente a Flag_churn
df_churn.groupby('Vmail_Plan')['Flag_churn'].describe()
```

[32]:

Vmail_Plan	count	mean	std	min	25%	50%	75%	max
0	2244.0	0.165330	0.371561	0.0	0.0	0.0	0.0	1.0
1	855.0	0.088889	0.284750	0.0	0.0	0.0	0.0	1.0

0.3.3 Una estrategia de ofrecimiento de Vmail podria lograr la fidelizacion del cliente ya que hay un 8% de propension al churn vs un 16 para quienes no tienen Vmail

[33]:
```
# Analisis descriptivo de Intl_Plan frente a Flag_churn
df_churn.groupby('Intl_Plan')['Flag_churn'].describe()
```

[33]:

Intl_Plan	count	mean	std	min	25%	50%	75%	max
0	2800.0	0.114643	0.318647	0.0	0.0	0.0	0.0	1.0
1	299.0	0.421405	0.494612	0.0	0.0	0.0	1.0	1.0

0.3.4 La propension de abandono de aquellos que tienen intl plan es casi 4 veces mas altas que los que no

0.4 Deteccion de anomalias

[34]:
```
# Detectar outliers usando el método del rango intercuartílico (IQR) para
 ↪variables numéricas sin incluir Flag_churn
# Seleccionamos las variables numéricas, excluyendo 'Flag_churn'
```

```python
# Se utiliza como corte el +- 3 para ser exigente en la detección
variables_numericas = ['Account_Length','Vmail_Message' ,'Day_Mins'␣
 ↪,'Day_Calls' ,'Eve_Mins','Eve_Calls' ,'Night_Mins' ,'Night_Calls'␣
 ↪,'Intl_Mins' ,'Intl_Calls' ,'CustServ_Calls' ,'Day_Calls'␣
 ↪,'Eve_Calls','Intl_Calls' ,'CustServ_Calls']

outliers_dict = {}

for var in variables_numericas:
    Q1 = df_churn[var].quantile(0.25)
    Q3 = df_churn[var].quantile(0.75)
    IQR = Q3 - Q1
    limite_inferior = Q1 - 3 * IQR
    limite_superior = Q3 + 3 * IQR
    outliers = df_churn[(df_churn[var] < limite_inferior) | (df_churn[var] >␣
 ↪limite_superior)]
    outliers_dict[var] = outliers.index.tolist()
    print(f"Variable: {var} - Cantidad de outliers detectados: {len(outliers)}")

# Si se desea ver los valores de los outliers por variable:
for var, idxs in outliers_dict.items():
    if len(idxs) > 0:
        valores = df_churn.loc[idxs, var].values
        print(f"Variable: {var} - Valores de outliers: {valores}")
```

```
Variable: Account_Length - Cantidad de outliers detectados: 0
Variable: Vmail_Message - Cantidad de outliers detectados: 0
Variable: Day_Mins - Cantidad de outliers detectados: 0
Variable: Day_Calls - Cantidad de outliers detectados: 2
Variable: Eve_Mins - Cantidad de outliers detectados: 0
Variable: Eve_Calls - Cantidad de outliers detectados: 1
Variable: Night_Mins - Cantidad de outliers detectados: 0
Variable: Night_Calls - Cantidad de outliers detectados: 0
Variable: Intl_Mins - Cantidad de outliers detectados: 0
Variable: Intl_Calls - Cantidad de outliers detectados: 6
Variable: CustServ_Calls - Cantidad de outliers detectados: 33
Variable: Day_Calls - Cantidad de outliers detectados: 2
Variable: Eve_Calls - Cantidad de outliers detectados: 1
Variable: Intl_Calls - Cantidad de outliers detectados: 6
Variable: CustServ_Calls - Cantidad de outliers detectados: 33
Variable: Day_Calls - Valores de outliers: [0 0]
Variable: Eve_Calls - Valores de outliers: [0]
Variable: Intl_Calls - Valores de outliers: [19 18 18 16 16 18]
Variable: CustServ_Calls - Valores de outliers: [7 7 9 6 6 6 7 6 6 6 6 6 6 8 6 7
7 7 8 6 6 6 9 6 6 6 6 7 6 6 7 6 6]
```

0.4.1 Se concluye no tratar los valores outliers dado que consideramos que aportan mucho valor a este analisis en particular. Sin embargo, en caso de querer tratarlos se puede imputar la media / moda / etc.

0.5 MODELADO

0.6 Seleccion de variables

Se realiza un test de hipotesis de diferencia de medias para cada variable con respecto a Churn para entender si la variable discrimina

[35]:
```
# Detectar variables significativas y no significativas entre los grupos de churn

# Listas para guardar los nombres de las variables
variables_significativas = []
variables_no_significativas = []

# Separar el DataFrame en churn = 1 y churn = 0
df_churn_1 = df_churn[df_churn['Flag_churn'] == 1]
df_churn_0 = df_churn[df_churn['Flag_churn'] == 0]

# Seleccionar todas las columnas excepto la variable objetivo
columnas = [col for col in df_churn.columns if col != 'Flag_churn']

for col in columnas:
    # Si la columna es numérica, aplicar t-test directamente
    if df_churn[col].dtype in ['int64', 'float64']:
        datos_1 = df_churn_1[col].dropna()
        datos_0 = df_churn_0[col].dropna()
        stat, pvalue = ttest_ind(datos_1, datos_0, equal_var=False)
        if pvalue < 0.05:
            variables_significativas.append(col)
        else:
            variables_no_significativas.append(col)
    # Si la columna es categórica, convertir a numérica usando codes y luego t-test
    else:
        datos_1 = df_churn_1[col].astype('category').cat.codes
        datos_0 = df_churn_0[col].astype('category').cat.codes
        stat, pvalue = ttest_ind(datos_1, datos_0, equal_var=False)
        if pvalue < 0.05:
            variables_significativas.append(col)
        else:
            variables_no_significativas.append(col)

print("Variables significativas (alfa = 0.05):")
print(variables_significativas)
print("\nVariables NO significativas (alfa = 0.05):")
```

```
print(variables_no_significativas)
```

```
Variables significativas (alfa = 0.05):
['Phone', 'Intl_Plan', 'Vmail_Plan', 'Vmail_Message', 'Day_Mins', 'Eve_Mins',
'Night_Mins', 'Intl_Mins', 'Intl_Calls', 'CustServ_Calls']

Variables NO significativas (alfa = 0.05):
['State', 'Account_Length', 'Day_Calls', 'Eve_Calls', 'Night_Calls']
```

0.6.1 Tomamos las variables significativas para el entrenamiento del modelo excluyendo Phone ya que es el identificador del usuario.

0.6.2 Split - Metodo Holdout (70 % Train - 30 % Test)

[36]:
```
# La variable objetivo es 'Flag_churn'

X = df_churn[['Intl_Plan', 'Vmail_Plan', 'Vmail_Message', 'Day_Mins',␣
 ↪'Eve_Mins', 'Night_Mins', 'Intl_Mins', 'Intl_Calls', 'CustServ_Calls']]
y = df_churn['Flag_churn']

# Realizamos el split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3,
    random_state=42,
    stratify=y
)

print(f"Train shape: {X_train.shape}, Test shape: {X_test.shape}")
```

```
Train shape: (2169, 9), Test shape: (930, 9)
```

0.6.3 Se separa el data set en 70% train y 30% test

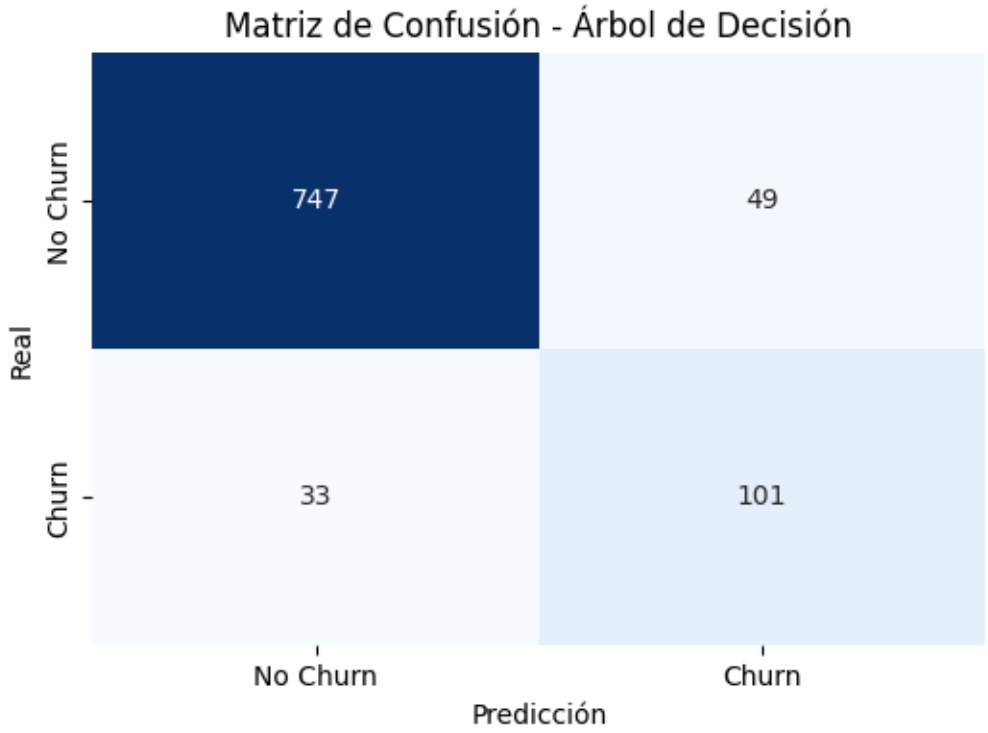
1 Árbol de decision sin parametria

[37]:
```
# 1. Árbol de Decisión

# Creamos el modelo
modelo_arbol = DecisionTreeClassifier(random_state=42)
modelo_arbol.fit(X_train, y_train)

# Realizamos predicciones
y_pred_arbol = modelo_arbol.predict(X_test)
```

[38]:
```
# Mostramos las métricas del modelo
print("Resultados Árbol de Decisión:")
```

```python
# Calculamos la matriz de confusión
cm = confusion_matrix(y_test, y_pred_arbol)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Churn', 'Churn'], yticklabels=['No Churn',␣
 ↪'Churn'])
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión - Árbol de Decisión')
plt.show()

# También mostramos el reporte de clasificación y accuracy
print(classification_report(y_test, y_pred_arbol))
print(f"Accuracy: {accuracy_score(y_test, y_pred_arbol):.4f}\n")
```

Resultados Árbol de Decisión:



	precision	recall	f1-score	support
0	0.96	0.94	0.95	796
1	0.67	0.75	0.71	134
accuracy			0.91	930
macro avg	0.82	0.85	0.83	930

```
weighted avg       0.92      0.91      0.91       930

Accuracy: 0.9118
```

1.0.1 Se observa que la Accuracy del modelo es del 91 % a nivel general, sin embago al detectar los casos de churn tiene tan solo un 67%.En cuanto al Recall el benchmark establecido es de un 75%. Este modelo se utiliza de benchmark ya que al no tener parametros al ajustarlos todo resultado debe ser al menos mejor que este.

2 Desicion Tree con Hiperparametrizacion

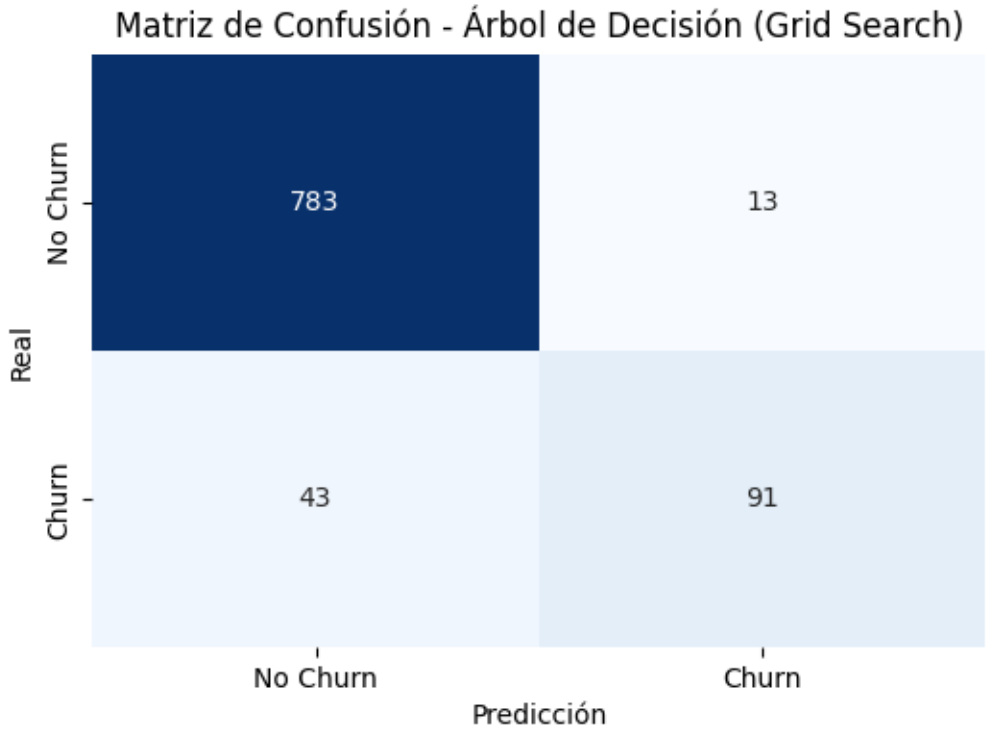
[39]:
```
## PROBAMOS CON NUEVOS HIPERPARAMETROS

# Los parámetros seleccionados estan pensados para un árbol de decisión con 9␣
 ↪variables predictoras.
# max_depth controla la profundidad máxima del árbol; valores entre 4 y 6␣
 ↪permiten árboles no muy complejos, lo cual es adecuado para evitar␣
 ↪sobreajuste con pocas variables.
# min_samples_split y min_samples_leaf ayudan a controlar el tamaño mínimo de␣
 ↪los nodos internos y hojas, respectivamente, lo que también previene el␣
 ↪sobreajuste.
# criterion permite comparar los dos criterios más comunes de impureza.

param_grid = {
    'max_depth': [4, 5, 6,7],
    'min_samples_split': [2, 3, 5, 10],
    'min_samples_leaf': [4, 6, 8, 10],
    'criterion': ['gini', 'entropy']
}


# Creamos el modelo base
arbol = DecisionTreeClassifier(random_state=42)

# Configuramos el GridSearchCV
grid_search = GridSearchCV(
    estimator=arbol,
    param_grid=param_grid,
    cv=5,
    scoring='recall',
    n_jobs=-1,
    verbose=2
)

# Ajustamos el GridSearch a los datos de entrenamiento
```

```
grid_search.fit(X_train, y_train)



# Evaluamos el mejor modelo en el conjunto de prueba
mejor_arbol = grid_search.best_estimator_
y_pred_grid = mejor_arbol.predict(X_test)
```

```
Fitting 5 folds for each of 128 candidates, totalling 640 fits
```

[40]:
```
# Métricas del modelo
print("Resultados Árbol de Decisión (Grid Search):")

# Matriz de confusión
cm_grid = confusion_matrix(y_test, y_pred_grid)
plt.figure(figsize=(6,4))
sns.heatmap(cm_grid, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Churn', 'Churn'], yticklabels=['No Churn',␣
 ↪'Churn'])
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión - Árbol de Decisión (Grid Search)')
plt.show()


reporte = classification_report(y_test, y_pred_grid, output_dict=True)
df_reporte = pd.DataFrame(reporte).transpose()
df_reporte = df_reporte[['precision', 'recall', 'f1-score', 'support']]
df_reporte = df_reporte.round(3)
display(df_reporte)

# Mostramos el accuracy
print(f"Accuracy: {accuracy_score(y_test, y_pred_grid):.4f}\n")
print("La profundidad del mejor árbol es:", mejor_arbol.get_depth())
```

```
Resultados Árbol de Decisión (Grid Search):
```



	precision	recall	f1-score	support
0	0.948	0.984	0.965	796.00
1	0.875	0.679	0.765	134.00
accuracy	0.940	0.940	0.940	0.94
macro avg	0.911	0.831	0.865	930.00
weighted avg	0.937	0.940	0.937	930.00

```
Accuracy: 0.9398

La profundidad del mejor árbol es: 7
```

2.0.1 Se observa una disminucion en el recall y una mejora en la precision y accuracy, sin embargo, el árbol sin hiperparametros tenia una profundidad de 28, lo que indica un gran overfitting, por lo que este último árbol es mas real. Por lo tanto, se define que éste será el árbol benchmark

2.1 RANDOM FOREST

[ ]:
```python
# 2. Random Forest con GridSearchCV para optimizar hiperparámetros según recall

# Definimos el grid de hiperparámetros
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 4, 5, 6, 7],
```

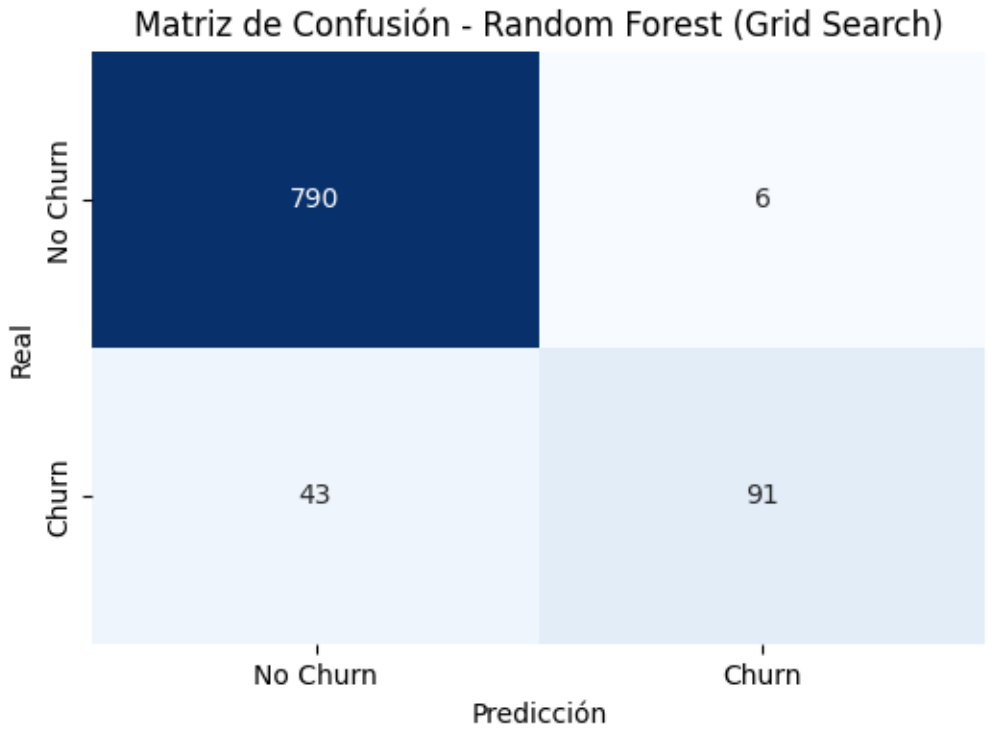
```
    'min_samples_split': [2, 3, 5, 10],
    'min_samples_leaf': [4, 6, 8, 10],
    'criterion': ['gini', 'entropy']
}

# Creamos el modelo base
rf = RandomForestClassifier(random_state=42)

# Configuramos el GridSearchCV
grid_search_rf = GridSearchCV(
    estimator=rf,
    param_grid=param_grid_rf,
    cv=5,
    scoring='recall',
    n_jobs=-1,




    verbose=2
)

# Ajustamos el GridSearch a los datos de entrenamiento
grid_search_rf.fit(X_train, y_train)



# Evaluamos el mejor modelo en el conjunto de prueba
mejor_rf = grid_search_rf.best_estimator_
y_pred_rf = mejor_rf.predict(X_test)
```

```
Fitting 5 folds for each of 480 candidates, totalling 2400 fits
```

[42]:
```
# Mostramos las métricas del modelo actual (Random Forest optimizado)
print("Resultados Random Forest (Grid Search):")

# Matriz de confusión
cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(6,4))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Churn', 'Churn'], yticklabels=['No Churn',␣
 ↪'Churn'])
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión - Random Forest (Grid Search)')
plt.show()
```

```python
# Reporte de clasificación
reporte_rf = classification_report(y_test, y_pred_rf, output_dict=True)
df_reporte_rf = pd.DataFrame(reporte_rf).transpose()
df_reporte_rf = df_reporte_rf[['precision', 'recall', 'f1-score', 'support']]
df_reporte_rf = df_reporte_rf.round(3)
display(df_reporte_rf)

# Mostramos el accuracy
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}\n")

# Mostramos la profundidad de los árboles en el mejor modelo Random Forest
print(f"Profundidad de los árboles en el mejor Random Forest: {mejor_rf.
  ↪max_depth}")
```

Resultados Random Forest (Grid Search):



	precision	recall	f1-score	support
0	0.948	0.992	0.970	796.000
1	0.938	0.679	0.788	134.000
accuracy	0.947	0.947	0.947	0.947
macro avg	0.943	0.836	0.879	930.000
weighted avg	0.947	0.947	0.944	930.000

```
Accuracy: 0.9473

Profundidad de los árboles en el mejor Random Forest: 7
```

2.1.1

2.1.2 Al emplear un bosque de desicion en lugar a un solo arbol se obtiene una ganancia de Accuracy, la precision para la deteccion de posibles churns aumenta de 87% a 93%. En cuanto al recall no se observa diferencia significativa.

2.1.3

2.2 XGBOOST

[43]:
```
# Definimos el grid de hiperparámetros para XGBClassifier
param_grid_xgb = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 4, 5, 6, 7],
    'min_child_weight': [2, 4],
    'learning_rate': [0.01, 0.1,0.2,0.3]
}

# Creamos el modelo base
xgb = XGBClassifier(random_state=42, use_label_encoder=False,␣
 ↪eval_metric='logloss')

# Configuramos el GridSearchCV
grid_search_xgb = GridSearchCV(
    estimator=xgb,
    param_grid=param_grid_xgb,
    cv=5,
    scoring='recall',
    n_jobs=-1,
    verbose=2
)

# Ajustamos el GridSearch a los datos de entrenamiento
grid_search_xgb.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 120 candidates, totalling 600 fits
```

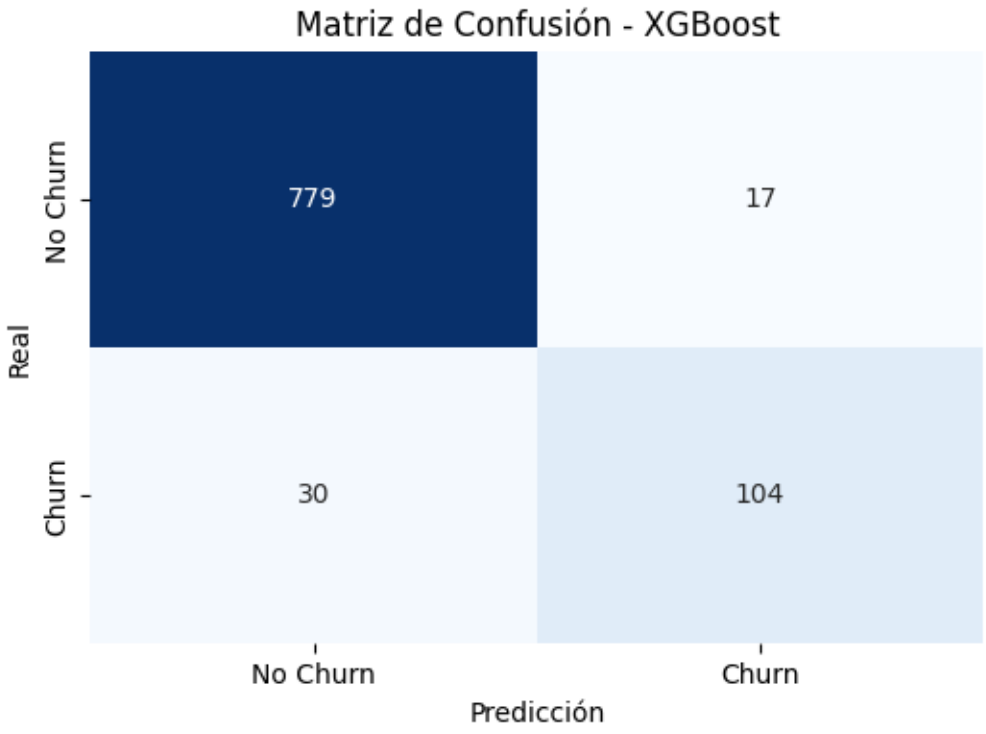
[43]:
```
GridSearchCV(cv=5,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     callbacks=None, colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, device=None,
                                     early_stopping_rounds=None,
                                     enable_categorical=False,
                                     eval_metric='logloss', feature_types=None,
```

```
                                    gamma=None, grow_policy=None,
                                    importance_type=None,
                                    interaction_constraints=None,
                                    learning_rate=…
                                    max_delta_step=None, max_depth=None,
                                    max_leaves=None, min_child_weight=None,
                                    missing=nan, monotone_constraints=None,
                                    multi_strategy=None, n_estimators=None,
                                    n_jobs=None, num_parallel_tree=None,
                                    random_state=42, …),
             n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.1, 0.2, 0.3],
                         'max_depth': [3, 4, 5, 6, 7],
                         'min_child_weight': [2, 4],
                         'n_estimators': [50, 100, 200]},
             scoring='recall', verbose=2)
```

[44]:
```
# Mostramos los mejores hiperparámetros encontrados para XGBoost
print("Mejores hiperparámetros para XGBoost:")
print(grid_search_xgb.best_params_)

# Evaluamos el mejor modelo de XGBoost en el conjunto de prueba
mejor_xgb = grid_search_xgb.best_estimator_
y_pred_xgb_grid = mejor_xgb.predict(X_test)

print("Resultados XGBoost (Grid Search):")

plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred_xgb_grid), annot=True, fmt='d',␣
 ↪cmap='Blues', cbar=False,
            xticklabels=['No Churn', 'Churn'], yticklabels=['No Churn',␣
 ↪'Churn'])
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión - XGBoost')
plt.show()

# Mostramos el reporte de clasificación y accuracy para XGBoost
print(classification_report(y_test, y_pred_xgb_grid))
print(f"Accuracy: {accuracy_score(y_test, y_pred_xgb_grid):.4f}\n")

# Mostramos la profundidad de los árboles del mejor modelo XGBoost encontrado␣
 ↪por GridSearchCV
print("Profundidad del árbol del mejor modelo XGBoost:", mejor_xgb.
 ↪get_params()['max_depth'])
```

```
Mejores hiperparámetros para XGBoost:
```

```
{'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 4, 'n_estimators':
200}
Resultados XGBoost (Grid Search):
```



	precision	recall	f1-score	support
0	0.96	0.98	0.97	796
1	0.86	0.78	0.82	134
accuracy			0.95	930
macro avg	0.91	0.88	0.89	930
weighted avg	0.95	0.95	0.95	930

```
Accuracy: 0.9495

Profundidad del árbol del mejor modelo XGBoost: 5
```

2.2.1 Al emplear XGBoost, al igual que con el bosque de decisión se observa un Accuracy similiar; en cuanto a precision hay menos ganancia con respecto al bosque, sin embargo el recall aumenta considerablemente siendo el árbol que mayor recall tiene de los utilizados.

2.2.2 El modelo que mejor ajusta a nuestro interes es el Gradient Boosting ya que mejorar el recall es mas probable de captar aquellos clientes que realmente abandonarian la empresa,sin importar perder poca presicion. Para un modelo de churn, el recall es una metrica mas importante ya que el objetivo principal es identificar a la mayor cantidad de clientes en riesgo como sea posible, ya que el coste de perderlos suele superar con creces el coste de las acciones de retención.

2.3 El mejor árbol es el XGBOOST

2.4 Creacion Score

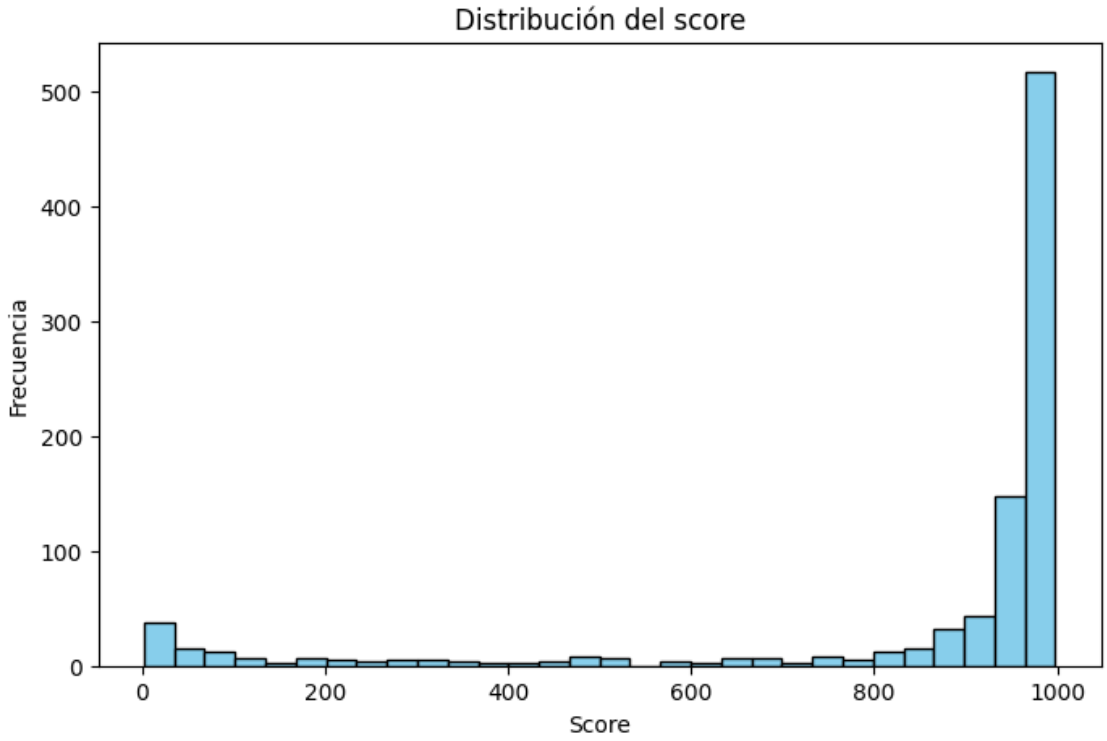
[45]:
```
# 1. Predicción de probabilidad con XGBoost y agregar columna score

# Creamos el modelo XGBoost con los hiperparámetros especificados
modelo_xgb = XGBClassifier(
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss',
    learning_rate=0.1,
    max_depth=5,
    min_child_weight=4,
    n_estimators=200
)
modelo_xgb.fit(X_train, y_train)

# Obtenemos la probabilidad de la clase positiva (por ejemplo, churn=1)
probabilidad_score = modelo_xgb.predict_proba(X_test)[:, 1]

# Agregamos la columna 'score' al DataFrame de test
df_scoreado = X_test.copy()
df_scoreado['score'] = probabilidad_score


# Convertimos las probabilidades en predicciones binarias usando un umbral (por␣
 ↪defecto 0.5)
umbral = 0.5
y_pred_xgb = (probabilidad_score >= umbral).astype(int)
```

Ajustamos la probabilidad a un score de 1 a 999 donde a mayor más probabilidad de permanecer en la empresa

[46]:
```
df_scoreado['score'] =  1 - df_scoreado['score']
```

[47]:
```
df_scoreado['score'] = (df_scoreado['score'] * 998 + 1).round(0).astype(int)
```

[48]:
```
plt.figure(figsize=(8,5))
plt.hist(df_scoreado['score'], bins=30, color='skyblue', edgecolor='black')
plt.title('Distribución del score')
plt.xlabel('Score')
plt.ylabel('Frecuencia')
plt.show()
```



2.5 Se observa que hay una muy notoria concentracion de clientes con un score alto, lo cual tiene mucho sentido sabiendo que la mayoria de personas no va a abandonar