Master in Management + Analytics

Modelo de Predicción de Churn de Repartidores de Apps

Autoría: *Ilkow, Damián*
Año académico: *2023*

UNIVERSIDAD TORCUATO DI TELLA

Escuela de Negocios

MASTER IN MANAGEMENT + ANALYTICS

Modelo de Predicción de Churn de Repartidores de Apps

Ilkow, Damian

Mayo 2023

Director: Agustín Gravano

"Creía en infinitas series de tiempos, en una red creciente y vertiginosa de tiempos divergentes, convergentes y paralelos. Esa trama de tiempos que se aproximan, se bifurcan, se cortan o que secularmente se ignoran, abarca todas las posibilidades. No existimos en la mayoría de esos tiempos; en algunos existe usted y no yo; en otros, yo, no usted; en otros, los dos."

El Jardín de los Senderos que se bifurcan,

Jorge Luis Borges

Resumen

El advenimiento de la tecnología y la aparición de nuevas empresas, impulsó la aparición de apps de delivery. Una de las problemáticas que deben enfrentar este tipo de organizaciones, es el mantenimiento de la flota de repartidores haciendo frente al *churn*, es decir al abandono de los mismos. El objetivo de este trabajo, es, por un lado, crear un modelo predictivo que permita predecir aquellos repartidores que realizarán churn, con el fin de poder tomar algún tipo de acción al respecto; por otro lado, se propone explorar un nuevo modelo, llamado *modelo de supervivencia*, que nos permitirá conocer desde el inicio, el tiempo que un repartidor se encontrará activo en la plataforma. De esta manera, se propondrá una nueva perspectiva explorando y comparando los resultados con el modelo original, más convencional.

Abstract

The emergence of technology and the appearance of new companies have led to the creation of delivery apps. One of the problems faced by this type of organizations is the maintenance of the delivery fleet, facing the churn, i.e. the desertion of the delivery riders. The objective of this work is, on one hand, to create a predictive model that allows knowing in advanced who of said riders will churn, in order to be able to take some kind of action in this regard; on the other hand, it is proposed to explore a new model, called survival model, which will allow us to know from the beginning how long a rider will be active on the platform. In this way, a new perspective will be proposed by exploring and comparing the results with the original, more conventional one.

Índice

Objetivo

El objetivo de la presente tesis es desarrollar y proponer un modelo predictivo que permita estimar la probabilidad de que un repartidor de una app de delivery realice churn en las próximas semanas (W + 1, W + 2). De este modo se le podría enviar una determinada oferta a tiempo antes de que, efectivamente, deje de repartir. El objetivo que persigue el modelo es incrementar la retención en la flota y reducir los costos asociados al churn. Dicho esto, podemos afirmar que el problema descrito se trata de un problema de clasificación donde para cada observación buscaremos predecir si su condición será *churn* o *no churn*.

Además, los resultados de la tesis servirán como disparadores para nuevos desafíos relacionados con el desarrollo de diferentes propuestas que se podrían enviar a los repartidores, siempre persiguiendo el objetivo de que continúen prestando servicios y no realicen churn.

El desafío de este modelo proviene de que la mayoría de los modelos predictivos de churn analizados y realizados, se centran en evitar la fuga de clientes de las compañías. Pero, en este caso en particular, no nos encontraremos modelando la fuga de clientes, sino de repartidores cuyo objetivo no es adquirir productos sino obtener ganancias a partir del uso de la aplicación que proveen las organizaciones de *q-commerce*.

El enfoque propuesto para llevar adelante este trabajo consta de tres partes, a saber:

- Análisis Descriptivo, donde se llevará adelante una investigación exhaustiva sobre los datos recabados que nos permitirá arriesgar hipótesis, hallar relaciones entre las diferentes variables y comprender su relevancia frente a la variable dependiente.
- Análisis Predictivo, donde se propondrán diferentes modelos predictivos basados en técnicas estadísticas y de machine learning (algoritmos de clasificación) , que tomará como punto de partida el primer análisis (descriptivo) y nos permitirá calcular la probabilidad de churn para cada repartidor.
- Análisis Prescriptivo, donde a partir del desarrollo del modelo predictivo nos encontraremos en posición de proponer diferentes alternativas que permitan alcanzar el objetivo del modelo (disminuir el churn) y tomar la mejor decisión consecuentemente.

Introducción

Modelo de Negocio de Plataformas

Hacia principios de los 2000, impulsadas por la incipiente explosión de internet, comienzan a ganar popularidad las páginas webs. Luego de este primer acercamiento de los individuos a la internet, apoyadas en este desarrollo mencionado, surge el modelo de negocio de plataformas, denominadas por Andrei Hagiu (2015) como las "Multi Sided Platforms" por su característica de unir a dos o más grupos interdependientes de clientes con necesidades complementarias. Según el autor, el negocio de las plataformas se basa en dos puntos fundamentales: i) por un lado logra reducir los costos relacionados con la búsqueda de productos y ii) por otro lado logra reducir, también, los costos de la transacción compartida. En lo sucesivo, nos encargaremos de profundizar en estos conceptos.

Si bien el negocio de plataformas se trata de un fenómeno que lleva tiempo, su crecimiento ha cobrado impulso en los últimos años. Desde la explosión de la burbuja de las punto com, hacia esta parte, internet ha jugado un rol preponderante en el desarrollo de los nuevos negocios. Las plataformas, aprovechando estos movimientos y las nuevas tecnologías pujantes, han apalancado su negocio para crecer y ganar protagonismo.

Según Andrei Hagiu (2015), Una Plataforma Multi Lado (Multi Sided Platform) provee un soporte que facilita la interacción (o transacción) entre dos o más partes. Mientras más miembros de una de las partes haya, es más probable que más miembros de la otra parte se unan a la plataforma. Por ejemplo, si tomamos el caso de un Marketplace, podríamos suponer que mientras más oferentes (uno de los grupos de usuarios de las plataformas) haya y la variedad y cantidad de productos sea mayor, encontraremos también una mayor cantidad de personas que se inscribirán (segundo grupo de usuarios) para comprar los productos que ofrece el primer grupo de usuarios. De este modo, podríamos pensar a una plataforma como un facilitador entre nexos, entre grupos de usuarios con necesidades complementarias.

Los Modelos de Negocio de Plataformas poseen tres fuentes principales de generación de valor para sus grupos de usuarios:

1. Efectos de Red Directos (With-in the Groups Network Effects): esta fuente de valor, se basa en que los usuarios de las plataformas generan valor interactuando entre ellos, con otros usuarios de su mismo grupo. Los usuarios crean una red que los conecta con otros de su mismo tipo. Un ejemplo de esto podría ser un foro de *reviews* de películas, donde los consumidores de películas dejan una crítica sobre las que vieron y cualquier usuario que desee ver una película puede aprovechar esa review para decidir qué ver o dejar de ver. En estos casos, tenemos un único grupo de usuarios.
2. Efectos de Red Indirectos (Cross-Groups Network Effects): en este caso, y a diferencia de la definición anterior, el agregado de valor se produce a partir de la interacción de usuarios de diferentes grupos. En estos casos, un usuario adicional en uno de los grupos, incita a más usuarios a pertenecer al otro grupo. Un ejemplo de esto serían los marketplaces, como los mencionados anteriormente.
3. Funcionalidades (Stand-Alone Utility): esta última definición de valor acapara a todas aquellas plataformas que poseen un valor en sí mismas, independientemente de los grupos de usuarios que interactúen. Como por ejemplo, aquellas que tienen un editor de texto, como la que está siendo utilizada en este momento para redactar estas líneas.

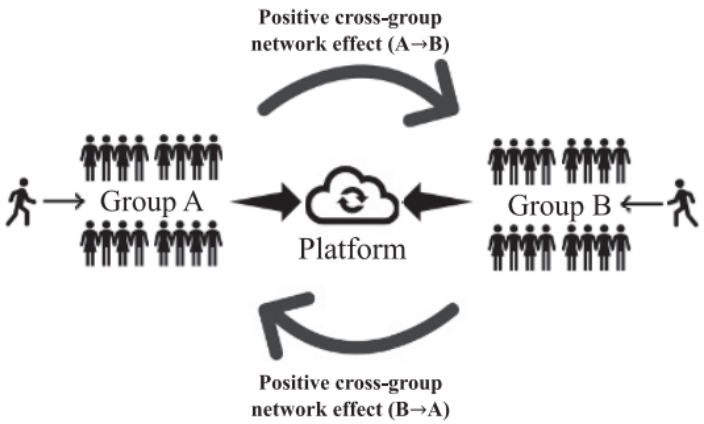


Gráfico Efectos de red - Andrei Hagiu (2015)

Los efectos de red tratan de un efecto propio de las empresas de plataforma. Estos podrían ser considerados de un modo similar a las economías de escala, pero del lado de la demanda. Esto significa que, en la medida en que crecen los grupos de clientes,

disminuyen costos para las empresas de plataformas. La imagen *efectos de red* muestra cómo se relacionan los grupos de usuarios para un modelo de negocio de plataforma cuya principal propuesta valor pasa por efectos de red indirectos (dos grupos de usuarios con necesidades complementarias).

Desafíos de las Plataformas de dos lados

La perspectiva ofrecida por Hagiu en la sección Modelo de Negocio de Plataformas, nos ofrece un marco para interpretar y comprender cómo funciona el Modelo de negocios de las plataformas y, en particular, las plataformas de dos lados (*Multi Sided Platforms*). Sin embargo, podría resultar interesante abordar el tipo de desafíos que estas plataformas de dos lados utilizan para crear valor y retener a sus usuarios. Podemos encontrar términos análogos al de Hagiu, como el de Plataformas de dos lados o Two sided platforms, del inglés original. En el texto referido "Strategies for Two Sided Markets" (Eisenmann et al., 2006), se sostiene que en los modelos de negocio de dos lados, las plataformas actúan como intermediarios entre dos grupos de usuarios interdependientes, como proveedores y consumidores. Estas plataformas crean valor al facilitar la interacción y el intercambio de bienes, servicios o información entre los dos lados. Del mismo modo, coincide con el texto citado con anterioridad - Andrei Hagiu 2015 - en que el valor que ofrece la plataforma dependerá ampliamente de los efectos de red que puedan generarse a ambos lados, es decir en los diferentes grupos de usuarios. De este modo, el valor particular de cada usuario crecerá en la medida que la demanda se vea satisfecha por el otro grupo de usuarios. Por ejemplo, mientras más usuarios utilicen el buscador de Google, más atractivo será para quienes busquen publicitar en su sitio web.

Los autores mencionan una serie de desafíos a considerar para los modelos de negocio de plataformas:

1. Fijación de precios en la plataforma: A diferencia de los modelos convencionales de fijación de precios en industrias competitivas, donde el costo marginal tiende a ser relevante al momento de fijar precios, en los modelos de negocio de plataformas funciona de un modo particular. Los proveedores de plataformas deben elegir un precio para cada lado, considerando el impacto que esto pueda

tener en el crecimiento y la disposición a pagar del otro lado. Por lo general, las redes de dos lados tienen un "lado subsidiado", es decir, un grupo de usuarios que, cuando se atraen en volumen, son altamente valorados por el otro grupo de usuarios. Debido a esto, el lado monetario de la plataforma debe pagar un valor mayor de lo que se pagaría en un mercado independiente. Por ello, el objetivo de los modelos de negocio de plataformas es generar efectos de red cruzados, si el proveedor de la plataforma puede atraer suficientes usuarios subsidiados, los usuarios del lado monetario pagarán generosamente para acceder a ellos.

2. Dinámicas de ganador se lleva todo: los rendimientos a escala que pueden generarse a partir de los mercados de plataformas, podrían impulsar a llegar a una dinámica donde la plataforma que se impone se “lleva todo”, situaciones similares a monopolios en mercados convencionales. Es por ello que se vuelve relevante definir si el mercado objetivo puede ser atendido por una o más plataformas. Según los autores, un mercado puede será atendido por una única plataforma, cuando se cumplan tres condiciones, a saber:
    a. Los costos de cambio entre plataformas son altos para al menos un lado de los usuarios. Los costos de cambio incluyen todos los gastos en los que incurren los usuarios de la red, como la adopción, la operación y el costo de oportunidad del tiempo, para establecer y mantener la afiliación a la plataforma.
    b. Los efectos de red son positivos y fuertes, al menos para los usuarios del lado de la red con altos costos de cambio. Cuando los efectos de red cruzados son positivos y fuertes, esos usuarios de la red tienden a converger en una sola plataforma.
    c. Ninguno de los lados de los usuarios tiene una preferencia fuerte por características especiales. Si ciertos usuarios tienen necesidades únicas, entonces las plataformas más pequeñas y diferenciadas pueden enfocarse en esas necesidades y abrirse paso en la sombra de un competidor más grande.
3. Amenaza de absorción: En último lugar, los autores mencionan que existe la amenaza de que la plataforma sea absorbida o "envuelta" por un proveedor de plataforma adyacente que ingresa a su mercado porque las plataformas

habitualmente tienen bases de usuarios superpuestas. Aprovechar estas relaciones compartidas puede hacer que sea fácil y atractivo para un proveedor de plataforma absorber la red de otro.

Como conclusión, podemos afirmar que en el contexto empresarial actual, el modelo de negocio de plataformas ha surgido como un modelo de negocio innovador y disruptivo. Estas plataformas actúan como intermediarios entre proveedores y usuarios, creando un entorno donde se facilita la interacción y el intercambio de servicios. Sin embargo, para garantizar el éxito y la sostenibilidad de estas plataformas, es fundamental considerar los desafíos a los que se enfrentan, en un mercado en constante y pujante crecimiento.

Apps de Delivery

En el decenio transcurrido entre el 2010 y el 2020, impulsado por el *boom* de internet que empieza ya a consolidarse y la aparición de nuevas tecnologías, entre ellas los teléfonos inteligentes, surgen las primeras aplicaciones cuya finalidad, siendo escuetos en su definición, es unir restaurantes con comensales. Con el paso del tiempo, ese negocio se fue perfeccionando, iterando sobre cada una de sus verticales, incluyendo otros servicios y, además, la logística propia para unir estos dos actores del sistema, incorporando un tercer actor fundamental en su construcción, la figura del repartidor.

En Argentina los dos principales competidores en este sector, son PedidosYa y Rappi. La primera es una empresa fundada en 2009 en Uruguay, mientras que la segunda fue fundada en Colombia, en el 2015. Ambas empresas extienden sus negocios a lo largo de América Látina.

Las empresas de q-commerce (Apps de delivery), son una particularidad del negocio de plataformas, definidos por los autores mencionados anteriormente. En este caso, el modo en que estas plataformas agregan valor es a través de efectos de red indirectos, donde dos o más grupos de usuarios se encuentran, plataforma mediante, para cubrir necesidades complementarias.

El éxito de las compañías (apps) de q-commerce, entendiendo por éxito al alcance de los objetivos finales de cada una de ellas, depende de tres soportes fundamentales, a

saber: clientes finales, restaurantes o comercios (a los que llamaremos Partners) y repartidores. De la conjunción e interacción de estos tres actores fundamentales, surge el funcionamiento de estas organizaciones (sin omitir todos aquellos factores internos que hacen a su cultura, empleados y tecnología) y su consecuente alcance de los objetivos. Es por ello que este tipo de plataformas se vuelve un caso particular de las "Multi sided platforms" en las que tenemos tres tipos de usuarios, definidos por los tres actores fundamentales ya mencionados.

Problemática

Impulsada por el advenimiento de la tecnología, la irrupción de la pandemia y la incorporación de nuevos hábitos de consumo, las organizaciones (apps) de q-commerce, han visto cómo se incrementó la cantidad de órdenes repartidas en los últimos años. Según la Cámara argentina de comercio, en el primer semestre del año 2022 las ventas electrónicas crecieron un 73% respecto al mismo período del año anterior. Asimismo, de todas estas ventas casi el 60% se realizaron a través de un celular. Además de ello, la principal categoría de producto vendida fue Alimentos y Bebidas [1].

Sin embargo, ese crecimiento exponencial vivenciado en los últimos tres años, no se encuentra exento de problemáticas. Las altas tasas de rotación y los cortos períodos de tiempo en que se mantienen activos los repartidores representan un desafío para este tipo de organizaciones. Los costos de publicidad, costos de incorporación, costos de entrenamiento y costos de oportunidad relacionados a la curva de aprendizaje de los repartidores, reflejan la cara visible de esta problemática. Esta decisión de los repartidores de dejar de prestar servicios (no encontrarse más activos), es lo que definiremos como churn en lo sucesivo. Con el objetivo de reducir la fuga de repartidores a partir de acciones específicas, es que será propuesto el modelo de predicción.

Sin embargo, una pregunta razonable que podría surgir, a partir del desarrollo realizado hasta aquí, es ¿cuál es el impacto del churn, de cualquiera de las partes, en los Modelos de Negocio de Plataformas?. Como se menciona en la sección *escalando efectos de red*, del libro *Platform revolution* (Parker et al., 2016), los efectos de red dependen del tamaño

de la red. Por lo tanto, las plataformas consideradas efectivas son aquellas capaces de expandirse y crecer rápida y fácilmente, aumentando así el valor que se deriva de los efectos de red. Sin embargo, escalar una red requiere que ambos lados de la plataforma aumenten su tamaño de forma proporcional, si esto no sucediera podría generarse una inequidad entre la demanda y la oferta. Por ejemplo: si aumentan desproporcionadamente los usuarios que utilizan Uber para viajar, pero no crecen del mismo modo los conductores, podría generarse una fuga de los clientes hacia otras plataformas debido a la insatisfacción de la demanda, así como también sucedería lo mismo si se redujera la cantidad de usuarios que viajan, los conductores se fugarían hacia otras aplicaciones donde existiera la demanda a satisfacer. Así mismo, si esto sucediera, los efectos de red mencionados por Hagiu perderían fuerza, generando vínculos más débiles entre los diferentes lados de la plataforma. Es por ello que la fuga de usuarios de cualquiera de los tres lados de las plataformas conformadas por las apps de delivery, generarían un desbalance entre la oferta y la demanda, o el servicio en el caso de los repartidores. Consecuentemente a esta caída en la oferta, demanda o servicio, se verían afectadas otras partes de la plataforma, generando así un perjuicio en la plataforma.

Marco de Trabajo

Definiciones

A continuación, listamos una serie de definiciones que se consideran relevantes para el desarrollo del trabajo:

Rider: Término utilizado para referirse a los repartidores y repartidoras de las apps de delivery.

Churn: Se define como *churn* a la acción, en este caso de los riders, de abandonar la plataforma. Su definición proviene del término anglosajón homónimo, donde según el diccionario de Cambridge su significado es que un cliente abandone un servicio. Según definiciones de negocio, consideraremos que un rider realizó churn cuando dejó de conectarse a repartir por la plataforma, durante tres semanas consecutivas.

Churn Rate: Cociente entre la cantidad de riders que realizaron churn, en un determinado período de tiempo, y el total de riders.

Hiperparámetros: Según James, G. et al. (2022), se denominan hiperparámetros a aquellos parámetros de un algoritmo de aprendizaje automático, que se definen con anterioridad al entrenamiento del modelo e influyen en cómo el modelo aprende. Los valores de los hiperparámetros pueden contribuir a aumentar o controlar el overfitting de los modelos.

En el capítulo dedicado al análisis de los datos, profundizaremos en los nombres específicos de determinadas variables cuyo significado no es del todo transparente con su nombre, o bien pueden presentar definiciones propias del negocio.

Medidas de éxito de un modelo predictivo

Debido a la incorporación del segundo modelo y por su condición de novedad, debemos incorporar este apartado sobre cómo medimos el éxito de un modelo predictivo.

Empezaremos por los modelos convencionales, para finalizar con el modelo de Survival y comparar ambos casos.

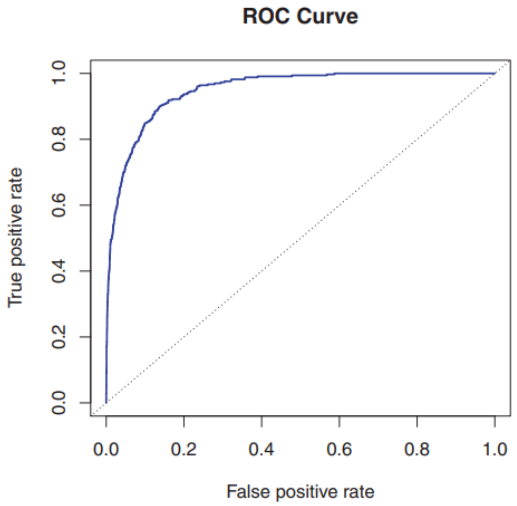
Según James, G. et al. (2022), la performance de un modelo predictivo habitual, como pueden ser los creados a través de algoritmos como Random Forest o XGBoost, también mencionados y desarrollados en el mismo libro, los mismos que utilizaremos a lo largo del presente trabajo, se mide con métricas como el Accuracy, el AUC o el F1. De este modo, las métricas que podemos hallar en el desarrollo del mismo documento, son las siguientes:

Accuracy: Ante la pregunta de cuán bueno resulta un modelo predictivo, definiremos el accuracy como la métrica que define la proporción de observaciones que se encuentran correctamente clasificadas (sobre el total de observaciones). El accuracy es una métrica que se utiliza para problemas de clasificación, como el problema tratado en este caso. El accuracy de un modelo, es definido del siguiente modo:

$$1 - \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \widehat{y_i})$$

De este modo, el valor del accuracy se encontrará delimitado entre 0 y 1. Mientras mayor sea el valor del accuracy mejor será un modelo, ya que la proporción de observaciones correctamente clasificadas aumentará frente a otros.

AUC: Área bajo la curva ROC (Area Under Curve). Este indicador devuelve un resumen de las combinaciones para cada umbral posible del modelo sobre ambos tipos de error, error de tipo uno y error de tipo dos. Del mismo modo que el accuracy, el valor del AUC se encontrará entre los valores 0 y 1. A continuación se muestra un ejemplo de cómo se compone la métrica AUC y los diferentes tipos de errores.



Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

Gráfico Curva ROC James, G. et al. (2022)

El área debajo de la curva azul es el área bajo la curva ROC, la mencionada AUC. Mientras que la imagen de la derecha nos enseña las diferentes combinaciones de ratios de acierto o error que podemos encontrar en las predicciones de un modelo.

Validación de datos

Con el objetivo de encontrar los mejores hiperparámetros para el modelo, es necesario realizar la validación de los resultados que se obtienen para cada combinación de hiper parámetros, técnica mencionada, relacionada y recomendada en James, G. et al. (2022). Como solución a esta necesidad surgen diferentes técnicas que permiten validar, utilizando datos de entrenamiento, los valores obtenidos de alguna métrica de éxito para una determinada combinación de hiperparámetros. La técnica más utilizada es *cross validation* (validación cruzada), dentro de la que encontramos k-fold cross validations y LOOCV (Live one out cross validation). Para los fines del desarrollo de nuestra problemática nos quedaremos con la primera de las técnicas mencionadas.
K-fold cross validation requiere dividir el set de datos de entrenamiento en K grupos (o folds) aproximadamente del mismo tamaño. Secuencialmente, cada uno de los k grupos se utiliza como set de test, mientras que con los demás k-1 grupos se entrena el modelo. De esta forma, se repite este proceso k veces hasta recorrer todos los k grupos.

En cada una de las k iteraciones del proceso, se calcula la métrica de éxito que se haya definido anteriormente. De este modo, una vez finalizado el proceso, obtendremos k valores para la métrica definida y el valor final será la media de todos esos valores.
De esta forma, si para cada combinación de hiperparámetros se realiza un k-fold validation, computando el valor de la métrica de éxito como la media que se definió anteriormente, aquel juego de hiperparámetros que maximice, o minimice, dependiendo el caso, la métrica de éxito, será el valor finalmente elegido.

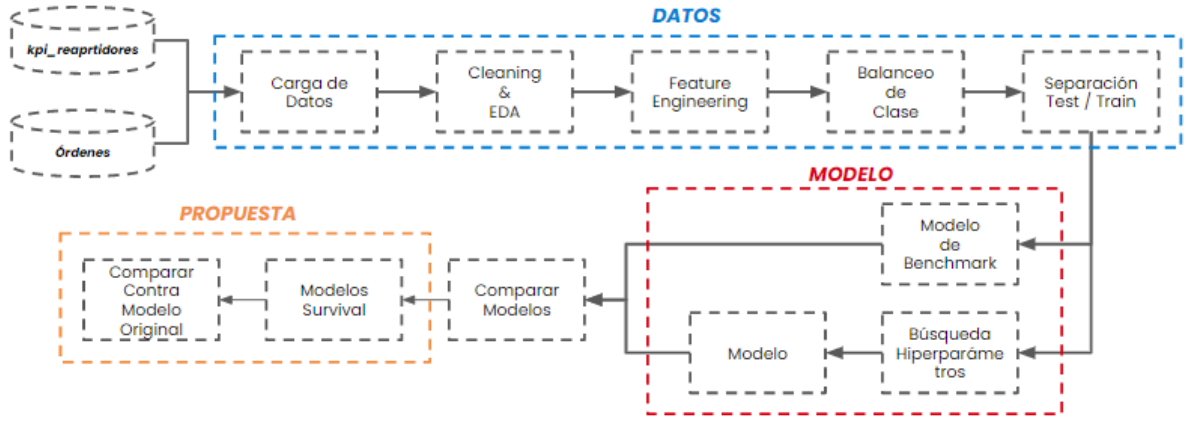
Escalado de Datos

Habitualmente, los modelos de aprendizaje automático tienden a performar mejor cuando los datos que se utilizan como input se encuentran escalados. Escalar los datos significa transformarlos a partir de algún tipo de escala. Existen varios tipos de escalado de datos dependiendo la escala que se utilice y cómo se realice su transformación.
En este caso, se utilizará el escalado estándar. Este tipo de escalado transforma los datos al estadístico Z Normal estándar. Para ello, a cada valor de la observación se le resta la media y se lo divide sobre el desvío estándar de la variable que se está escalando. La nueva variable escalada tendrá una media de 0 y un desvío estándar de 1.

Análisis, Limpieza de datos y Creación de variables

Origen de los datos

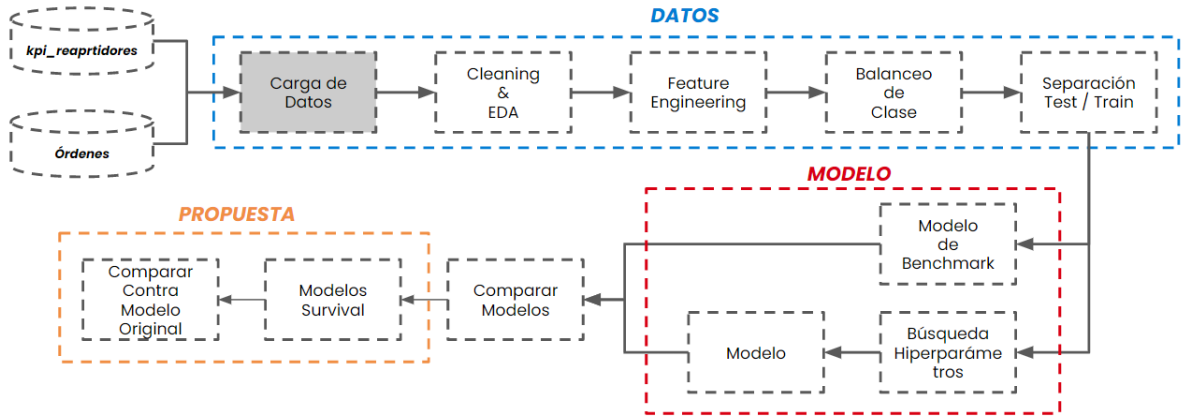
El trabajo fue realizado con datos obtenidos de una empresa de delivery por app, ubicada en Argentina. Dichos datos, corresponden a los repartidores de la app y a las órdenes generales realizadas. El período de tiempo abarcado por los datos comienza en el período de la semana uno del año 2020, finalizando en la semana 27 del año 2022.
Por cuestiones de confidencialidad, dichas bases no serán visibles en el trabajo. Dicho esto, para garantizar el anonimato, todos los datos que pudieran considerarse información sensible fueron debidamente esfumados con el fin de ser ocultados.
El pipeline del modelo, posee la siguiente forma:



El flujo de los datos que sigue el modelo consta de cinco grandes estadíos, a su vez repartidos en tareas más pequeñas. En primer lugar, se debe acceder a las bases de datos propias de la compañía y descargar los datos. Luego, comienza el estadío de los datos respectivamente, que contiene cinco tareas, a saber: a) cargar los datos descargados de las bases originales, b) limpiar los datos y realizar un primer análisis descriptivo de los mismos, c) crear nuevas variables que puedan ser útiles para el algoritmo, d) balancear las clases -categorías existentes en la variables dependiente- debido al problema de desbalanceo, e) separar las observaciones entre aquellas que se utilizarán para entrenar el modelo y aquellas que se utilizarán para testearlo. Luego de este primer estadío, llega el momento de crear el modelo propiamente dicho, constando, a su vez, de las siguientes tareas: a) crear un modelo de benchmark que nos sirva como

punto de partida y siente un objetivo a mejorar, b) buscar los hiperparámetros más adecuados para entrenar el algoritmo final, c) entrenar el algoritmo con los hiperparámetros que maximicen las métricas de éxito elegidas. Luego se compararán los modelos de Benchmark y el modelo final, con el objetivo de describir cuál fue el desempeño alcanzado. Finalmente, el último estadío consiste en la propuesta de un nuevo modelo alternativo, proponiendo un enfoque diferente al llevado adelante en los primeros desarrollos. En este último estadío, en primer lugar desarrollaremos el modelo y, en segundo lugar, lo compararemos contra el modelo original.

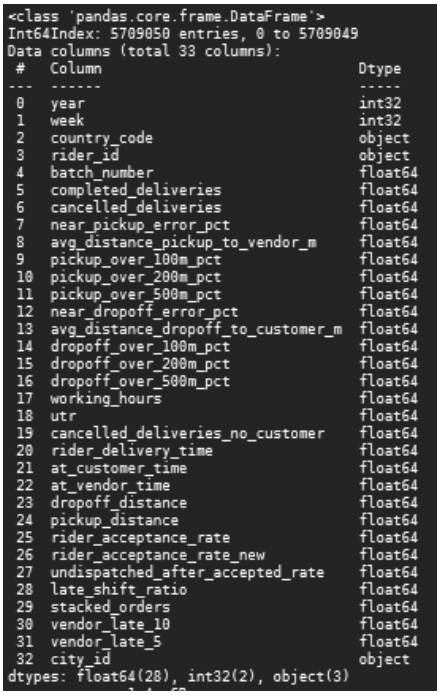
Carga de datos



Las fuentes de datos de donde obtuvimos el dataset original, fueron dos tablas de la empresa, una de ellas contiene datos sobre los repartidores y la otra contiene datos sobre las órdenes realizadas por los clientes, ya sea por medio de la app o la página web. Debido a la gran cantidad de datos contenidos en el dataset original, fue necesario dividirlo en tres partes para proceder con su descarga y posterior carga. Cada una de estas partes generadas se corresponde con cada uno de los años analizados, conteniendo 2020, 2021 y 2022 por separado. El dataset contiene datos tanto para la Argentina y otros países de la región (LATAM).

El dataset original contiene 5.709.050 observaciones y 33 columnas correspondientes a las 33 variables. Cada observación consta de un año, una semana y un repartidor particular. Es decir que la unidad de análisis es semana-repartidor. Por otro lado, en el dataset encontramos poco más de 200 mil repartidores únicos.

El dataset analizado posee la particularidad de tratarse de una serie de datos de panel, donde podemos hallar años, semanas (dimensiones temporales) y repartidores (dimensión transversal). Esta particularidad debe ser considerada a la hora de tomar cualquier tipo de decisión sobre el conjunto de datos, como por ejemplo eliminar observaciones, o bien separar entre test y train.



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5709050 entries, 0 to 5709049
Data columns (total 33 columns):
 #   Column                                Dtype
---  ------                                -----
 0   year                                  int32
 1   week                                  int32
 2   country_code                          object
 3   rider_id                              object
 4   batch_number                          float64
 5   completed_deliveries                  float64
 6   cancelled_deliveries                  float64
 7   near_pickup_error_pct                 float64
 8   avg_distance_pickup_to_vendor_m       float64
 9   pickup_over_100m_pct                  float64
 10  pickup_over_200m_pct                  float64
 11  pickup_over_500m_pct                  float64
 12  near_dropoff_error_pct                float64
 13  avg_distance_dropoff_to_customer_m    float64
 14  dropoff_over_100m_pct                 float64
 15  dropoff_over_200m_pct                 float64
 16  dropoff_over_500m_pct                 float64
 17  working_hours                         float64
 18  utr                                   float64
 19  cancelled_deliveries_no_customer      float64
 20  rider_delivery_time                   float64
 21  at_customer_time                      float64
 22  at_vendor_time                        float64
 23  dropoff_distance                      float64
 24  pickup_distance                       float64
 25  rider_acceptance_rate                 float64
 26  rider_acceptance_rate_new             float64
 27  undispatched_after_accepted_rate      float64
 28  late_shift_ratio                      float64
 29  stacked_orders                        float64
 30  vendor_late_10                        float64
 31  vendor_late_5                         float64
 32  city_id                               object
dtypes: float64(28), int32(2), object(3)
```

Fuente de datos - Imagen 1: Variables contenidas en el Dataset original

El dataset contiene observaciones para cada repartidor, para cada semana y año en que se encuentra activo. Además de ello, contiene variables como la cantidad de horas que se encontró repartiendo esa semana, la cantidad de órdenes que repartió, la ciudad y el país en el que se encuentra, las órdenes canceladas y las distancias recorridas, entre otras.

A continuación detallaremos algunas de las principales variables presentes en el dataset:

rider_id: identificador único para cada rider (lo utilizamos oculto en todo momento, con el fin de guardar la información sensible).

batch_number: valor numérico dentro del intervalo del 1 al 6, que indica la categoría a la que pertenece cada rider según indicadores definidos por la empresa. Esta categoría puede cambiar semana a semana. Se encuentra definida dentro de una escala ordinal donde los valores más bajos resultan mejores.

completed_deliveries: cantidad de órdenes completadas por un rider para cada semana.

cancelled_deliveries: cantidad de órdenes canceladas por un rider para cada semana.

working_hours: cantidad total de horas que un rider se encontró prestando servicio durante una semana determinada independientemente de la cantidad de días en que se hayan realizado esas horas.

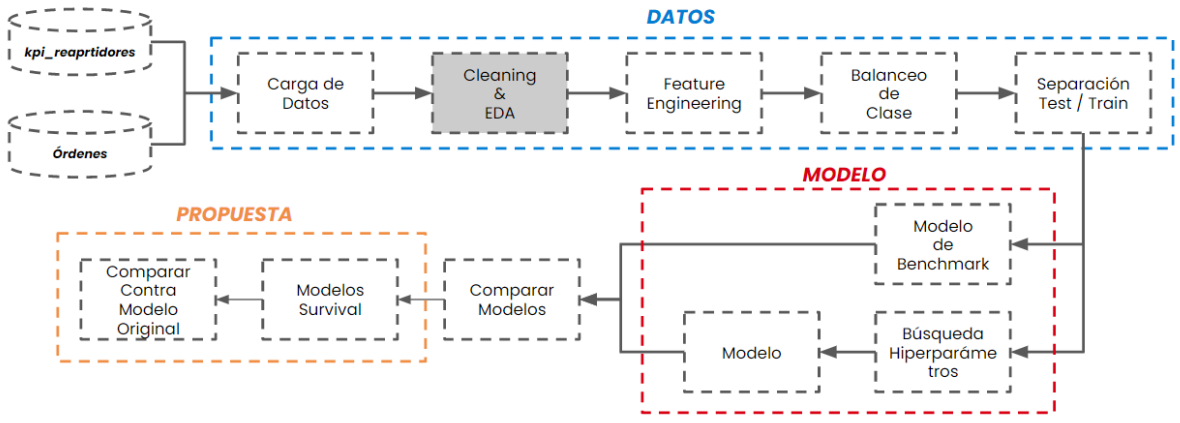
utr: ratio entre la cantidad de órdenes y la cantidad de horas. Indica la cantidad de órdenes por hora que reparte un rider.

stacked_orders: indica la cantidad de órdenes stackeadas que llevó un rider en una semana determinada. Una orden se encuentra stackeada cuando comparte viaje con otra orden, es decir que un rider lleva más de una orden a más de un punto diferente.

late_shift_ratio: proporción de días en los que un rider comenzó tarde su turno de horas.

rider_acceptance_rate: proporción de órdenes que un rider aceptó en una semana determinada.

Carga y limpieza de datos



El primer desafío que se nos presentó con el análisis de los datos, fue tomar decisiones respecto a su completitud, precisión y posible distorsión. Estas acciones se dividieron en tres: a) decidir qué hacer con los valores nulos; b) decidir qué hacer con los valores

extremos (outliers); c) entender el impacto que podría haber generado la pandemia y decidir qué hacer con ello:

a. *Valores faltantes (NaN values)*: decidimos eliminar las observaciones que poseían valores faltantes. Sin embargo, como cada repartidor se podía encontrar más de una vez a lo largo del dataset (porque podría tener valores para cada semana que se encuentra activo y encontrarse activo para más de una semana), al momento de eliminar observaciones por valores faltantes fue necesario hacerlo a nivel repartidor. De forma que si una observación relacionada a un repartidor, o una repartidora, contenía valores faltantes era eliminada en todas las semanas en las que pudiera aparecer. Del mismo modo, decidimos eliminar tres variables completas (rider_acceptance_rate_new, vendor_late_5, vendor_late_10) debido a que ellas poseían más del 80% de sus valores nulos.
b. *Valores extremos*: si bien el dataset originalmente no contenía una gran cantidad de datos con valores extremos (outliers), igual que como fue hecho con los valores faltantes, decidimos eliminarlos a nivel repartidor. El ejemplo más claro de estos valores extremos lo representan las órdenes, de este modo se eliminaron todos los repartidores que, al menos en una de las semanas observadas, poseían una cantidad de órdenes que se encontraba a una distancia mayor que 1.5 veces la distancia intercuartil. Este método es utilizado es el mismo que se utiliza en los boxplots para determinar qué es un valor extremo y qué no. Siguiendo esta metodología, eliminamos a cualquier repartidor que contenga algún valor extremo, en cualquiera de las observaciones analizadas, en algún período de tiempo. Asimismo, la última semana observada del año 2022 (27) fue eliminada por poseer todos valores extremos. Si bien desconocemos el motivo preciso sobre porqué esta semana presenta este comportamiento atípico en sus valores, podemos afirmar tanto desde la perspectiva del análisis de datos, como de la perspectiva de negocio, que los datos observados no presentan coherencia alguna.
c. *Impacto de la pandemia*: por último, fue necesario considerar el impacto de los datos relacionados a la aparición de la pandemia. La base contenía datos desde inicios de 2020, de modo que los primeros datos contenidos, aproximadamente las primeras 10 semanas, se trataban de datos anteriores a la pandemia mundial

y al inicio de la cuarentena en la mayoría de los países de la región. Las principales métricas contenidas en el dataset presentaban grandes distorsiones en las primeras dos semanas del comienzo de la pandemia (gran impacto en el churn rate calculado a partir de la variable dependiente del modelo).

Debido a la condición de serie de datos de panel que presenta el dataset fue necesario eliminar las observaciones al nivel id de repartidor, tal como fue mencionado para el caso de los valores extremos y faltantes. De esta forma, conseguimos eliminar toda su historia y no sólo fragmentos, que pudieran generar ciertas distorsiones en su propia evolución.

Una vez eliminadas las observaciones que presentaban las condiciones descritas anteriormente, el nuevo dataset contiene 5.007.851, es decir que se eliminaron cerca de 700.000 observaciones y tres variables, obteniendo finalmente 30 variables y 185.927 repartidores diferentes para el análisis.

Luego de realizar esta primera limpieza, fue necesario crear una variable dependiente, la variable a predecir. En este caso, por la naturaleza del problema analizado, la variable a predecir es una variable binaria - categórica que puede tomar los valores: churn y *no_churn*. Para ello, como mencionamos anteriormente, se definirá como churn a aquellos riders que no hayan completado al menos una orden durante tres semanas consecutivas, esa definición responde a una concepción propia de la organización que provee los datos, sustentada en un análisis no facilitado, aunque mencionado, junto con los datos en el que afirman que la probabilidad de retorno de un repartidor que dejó de repartir durante tres semanas consecutivas corresponde al 0,06. Para definir si un repartidor realizó churn, para cada observación, comparamos el rider_id, el año y la semana entre una entrada y la consecutiva posterior. De esta forma, un repartidor podría encontrarse en distintas ocasiones, a lo largo de su ciclo de vida en la organización, en situación de churn o *no_churn*, dependiendo la distancia temporal en sus períodos de actividad consecutivos.

	Churn	No Churn	*Total*
Valores Absolutos	206.750	4.662.122	4.868.872
Proporción	4,25%	95.75%	100%

Fuente de datos - Tabla 1 : Cantidad de observaciones según variable dependiente

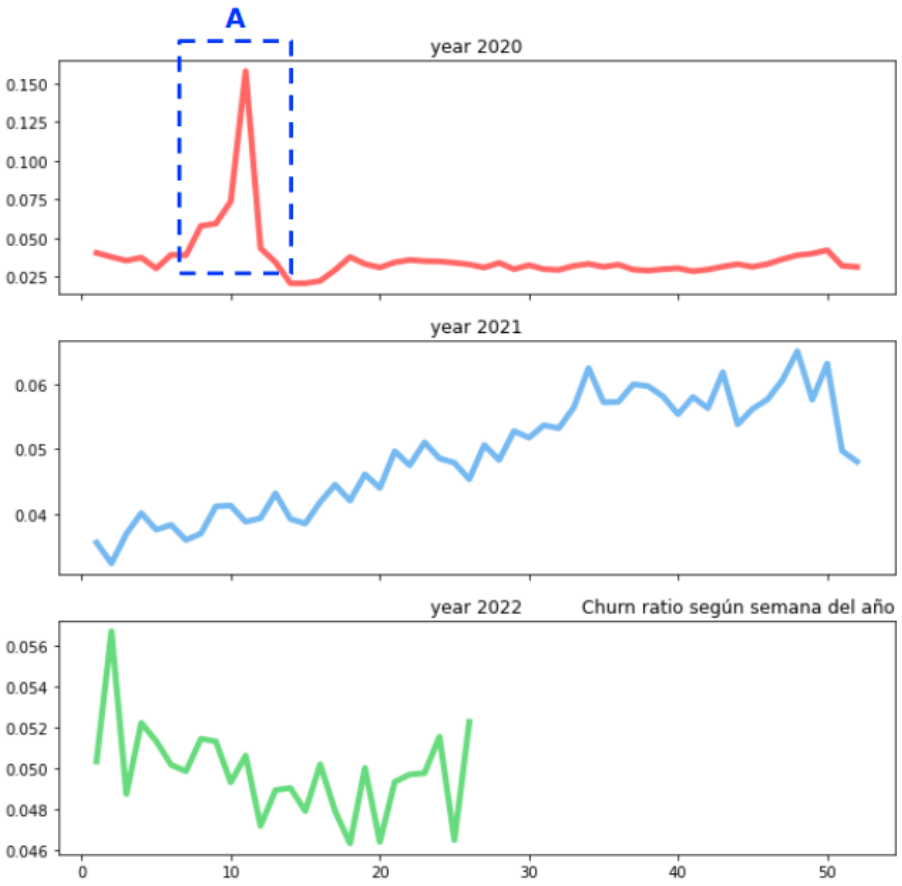
Análisis exploratorio de los datos

En este apartado analizaremos el dataset original, cada variable que contiene y cuáles de ellas podrían resultar relevantes para el análisis futuro.

La primera observación, la más relevante, es que el dataset se encuentra altamente desbalanceado. Una de las clases, en este caso *no_churn*, ocupa el 96% de las observaciones, mientras que la otra ocupa el 4% restante. Esta relación se puede apreciar en la *tabla 1*. Con el objetivo de mitigar el impacto de este desbalance generado entre clases, se utilizaron diferentes estrategias que desarrollaremos en el momento oportuno.

En primer lugar, antes de introducir el desarrollo de *feature engineering*, consideramos necesario crear una variable que nos permita entender el ratio de churn para cada período de tiempo. Esta variable la llamamos *churn rate* y se calcula como la proporción de repartidores que hicieron churn sobre el total de repartidores para cada período de tiempo.

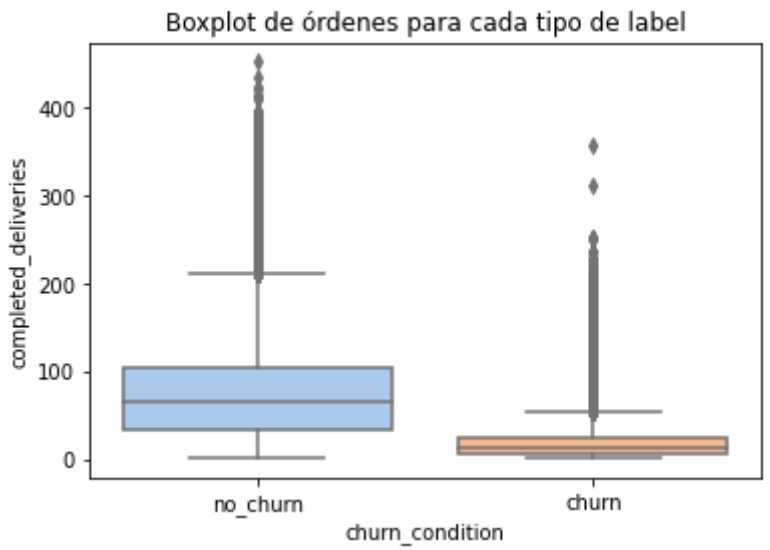
$$Churn\ Rate\ = \frac{Repartidores\ en\ Churn}{Total\ de\ Repartidores}$$



Fuente de datos - Imagen 2: Evolución semanal del Churn Rate

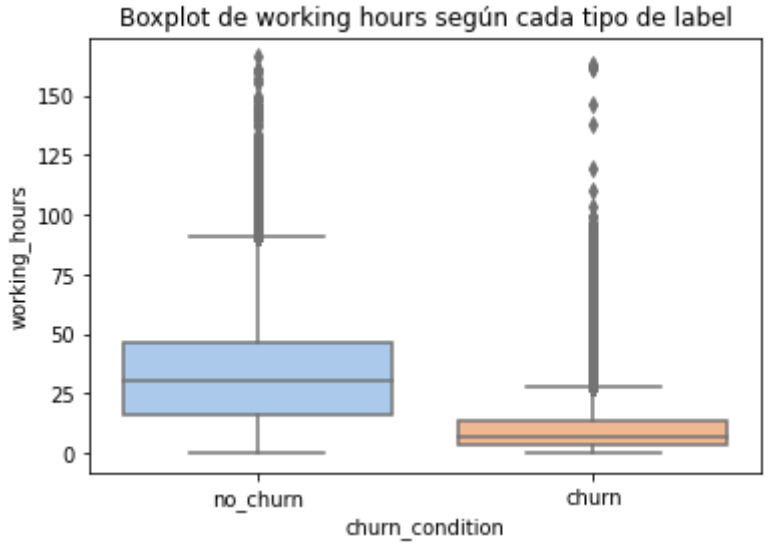
En la imagen 3 - Evolución semanal de Churn rate, podemos apreciar la evolución del churn rate a lo largo del tiempo desde una perspectiva semanal. Tal como se menciona en el apartado anterior, se puede ver la distorsión generada por la irrupción de la pandemia (Marzo 2020) y la decisión de los diferentes gobiernos locales de comenzar cuarentenas estrictas. Es por ello que todas las observaciones anteriores a la pandemia fueron desechadas. Por otro lado, también se puede apreciar una tendencia creciente, desde el comienzo hacia el fin, del churn rate para el segundo año de análisis (2021), único año que posee observaciones para todas sus semanas.

Según el conocimiento de negocio previo, el primer análisis fue enfocado en dos variables que, por su naturaleza, resultan relevantes. Estas fueron: Working Hours cuyo significado son las horas semanales que un repartidor se encontró repartiendo, y *completed_deliveries* que representa la cantidad total de órdenes que un repartidor entregó para cada semana.



Fuente de datos - Imagen 3: Boxplot para órdenes totales según churn o no_churn

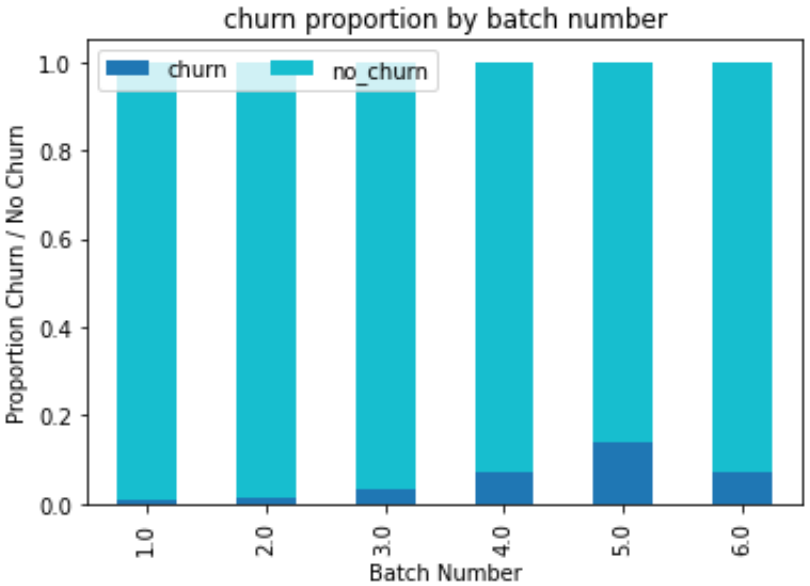
Como podemos apreciar en la imagen 4, la distribución de órdenes para *churn* o *no churn*, es sumamente inequitativa. Incluso el boxplot de la derecha tiene su tercer cuartil por debajo del primer cuartil del diagrama de la izquierda. Esto significa que el 75% repartidores que hacen churn, la semana antes de hacer churn, reparten menos órdenes que el 25% de los repartidores que no hacen churn. De esta forma, se puede concluir que la variable *completed_deliveries* (órdenes totales) podría resultar de gran valor predictivo.



Fuente de datos - Imagen 4: Boxplot para horas totales según churn o no_churn

En boxplot de la imagen 5, la variable que analizamos según *churn* o *no_churn*, es las horas totales que un repartidor se encontró repartiendo durante la semana. Del mismo modo que en la gráfica de órdenes totales, se puede observar fácilmente que para el caso de repartidores que terminan haciendo churn, la distribución es menor que para los que no. Asimismo, esta variable working hours podría resultar de gran valor predictivo para el modelo.

Por último, nos parece importante destacar los valores obtenidos para otra variable relevante para el negocio. Esta variable se denomina batch, y representa una suerte de categoría en la que se encuentra el repartidor según el valor obtenido en determinadas métricas, a saber: horas planificadas sobre horas de conexión, turnos tomados en los que efectivamente se presentó, días en los que se conectó a tiempo al turno tomado. Se trata de una variable discreta y con una escala ordinal, cuyos valores van del 1 al 6, siendo 1 el mejor valor posible y 5 el peor valor posible, el valor 6 se encuentra reservado para los nuevos repartidores que no llegan a tener historia suficiente para ser comparados e ingresar al sistema.

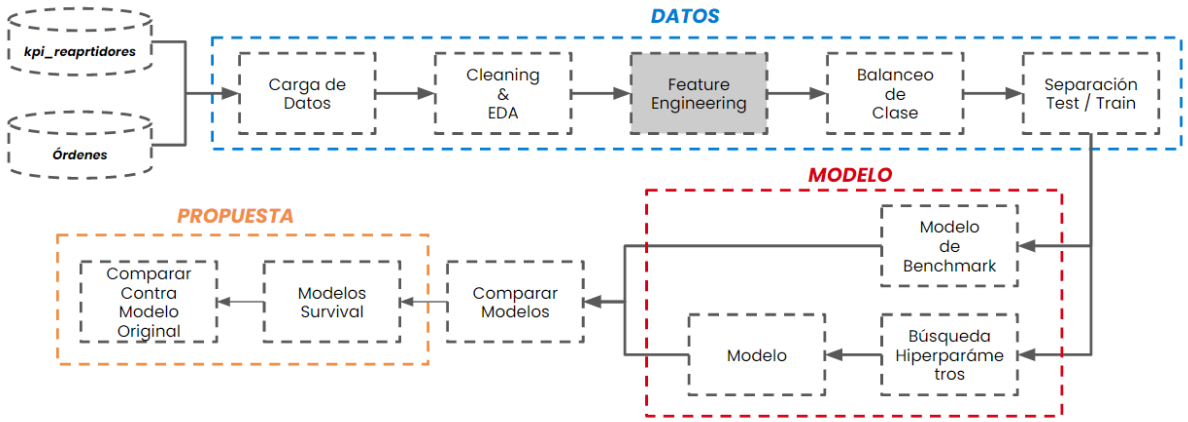


Fuente de datos - Imagen 5: Distribución de churn o no_churn según número de batch.

En este caso, en la imagen también puede apreciarse cómo aumenta la proporción de repartidores que hacen churn a medida que aumenta el valor de la variable batch. Esto significa que mientras peor es el batch de un repartidor, mayor es la proporción de churn. Lo cual hace perfectamente sentido con la regla de negocio que mencionamos

anteriormente, la conformación del sistema de ranking, ya que los repartidores más comprometidos son aquellos que se encuentran en los batchs superiores y se muestran menos propensos a abandonar la plataforma.

Generación de nuevas variables



El siguiente estadío del proyecto, consiste en generar nuevas features que puedan resultar de ayuda para el modelo a la hora de predecir la variable dependiente. Como se menciona en Feature Engineering (Zheng et al. 2018), la creación de nuevas variables es una tarea importante en el proceso de crear modelos predictivos, ya que se podrían llegar a captar relaciones entre las variables que, a priori, los algoritmos podrían no captar. Dentro de las variables más relevantes que fueron creadas en este estadío, se pueden mencionar las siguientes:

Seniority: esta variable representa la antigüedad de cada repartidor en semanas. Es la diferencia entre la fecha de la semana en la que se encuentra la observación y la primera orden repartida por este repartidor.

Suma cuadrática de órdenes y Working Hours: en este caso, buscamos emular la distancia de un punto al origen en el plano. Para ello, calculamos la raíz cuadrada de la suma al cuadrado de órdenes totales y working hours. La creación de esta variable, resultó ser una gran acierto ya que terminará teniendo un alto valor predictivo.

Tres componentes PCA: aplicamos un algoritmo de componentes principales sobre las variables numéricas del dataset original, convirtiendo algunas de las

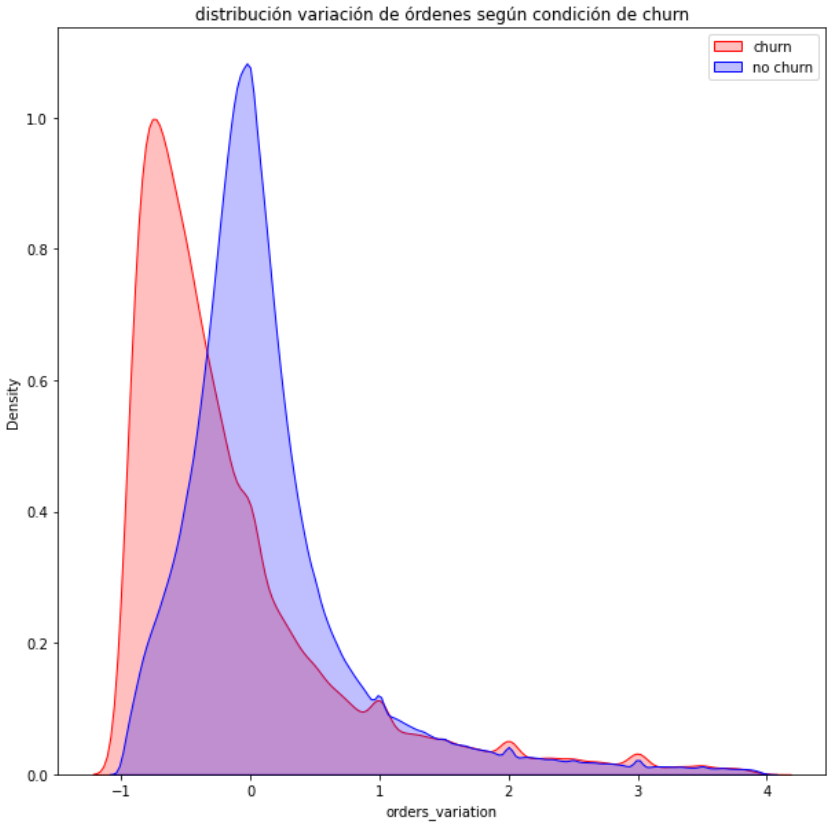
categóricas en numéricas, como por ejemplo país, a través de *one hot enconding*. El resultado de componentes principales, no fue demasiado alentador, obteniendo apenas una variabilidad explicada de 0.43 con tres componentes.

Variación de horas / órdenes / UTR: debido a que el dataset es una serie de datos de panel, decidimos crear una variable que sea capaz de captar la variación de ciertas métricas a través del tiempo y su evolución respecto del período inmediatamente anterior (t-1). Las variables para las que se calculó esta variación fueron working hours, órdenes totales y UTR (las órdenes promedio por hora). La variación fue calculada como la variación porcentual del valor posterior sobre el valor precedente.

Decil de órdenes según ciudad: La última variable de relevancia creada, fue el decil que cada repartidor ocupa para la cantidad de órdenes repartidas en esa semana para cada ciudad en la que se encuentra. Esto se debe a que la demanda no se presenta de un modo uniforme a lo largo de todas las ciudades, mostrando una propensión a ser mayor en las ciudades donde la población también es mayor.

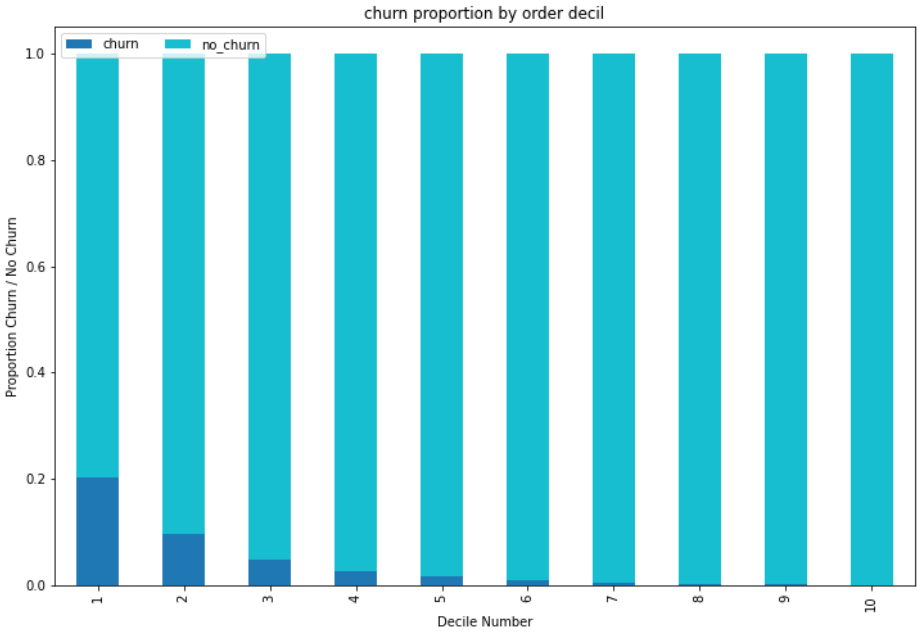
Transformaciones varias: por último, llevamos adelante algunas transformaciones mínimas necesarias para poder utilizar el dataset como input del modelo. Dentro de ellas, podemos encontrar la conversión de variables categóricas binarias a numéricas reemplazando una categoría por 1 y la otra por 0. Esta técnica habitualmente se conoce como que se conoce como generación de variables dummy o variables indicadoras.

Luego de haber creado estas variables, volvemos a analizar su impacto a través de algunas visualizaciones.



Fuente de datos - Imagen 6: Distribución de churn / no_churn según variación de órdenes

La variable *orders_variation* fue creada para captar la variación en cada período de tiempo. En el gráfico, esta variable se encuentra diferenciada según la variable dependiente, mostrando una distribución para cada caso. De esta forma, se puede apreciar una diferencia entre ambas categorías a predecir. La variable churn concentra valores entre el -1 y el 0, es decir que en la medida en que un repartidor disminuye su ritmo de órdenes entregadas respecto a la semana w-1 parecería ser que presenta una mayor tendencia a realizar churn. En cambio, la media para *no_churn* se mantiene positiva, cercana al cero, es decir que la hipótesis podría ser que un repartidor que no realiza *churn* mantiene, o bien aumenta, en una baja proporción la cantidad de órdenes repartidas.

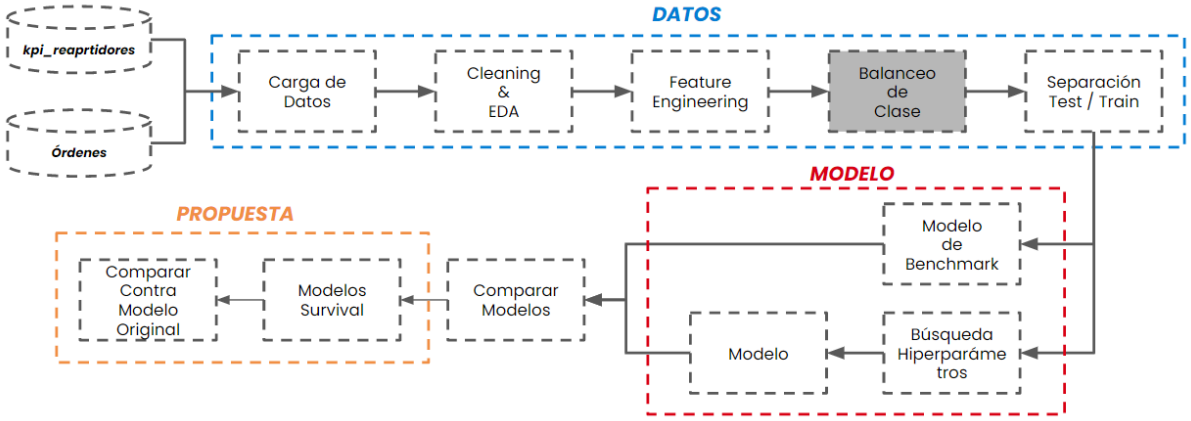


Fuente de datos - Imagen 7: Distribución de churn / no_churn según decil de órdenes

Por último, se analizó la proporción de churn y no_churn para los deciles de órdenes que ocupa cada repartidor según la ciudad en la que se encuentra, buscando de este modo mitigar el impacto de la demanda en el análisis. En el gráfico de la Imagen 7: Distribución de churn / no_churn según decil de órdenes, se puede observar cómo aquellos repartidores que se encuentran en los deciles más bajos muestran una mayor proporción de churn que aquellos que se encuentran en los deciles superiores. Es decir que la hipótesis aquí podría ser que mientras mayor cantidad de órdenes completa un repartidor menor es su intención de churn.

Análisis Predictivo I

Balanceo de Clases



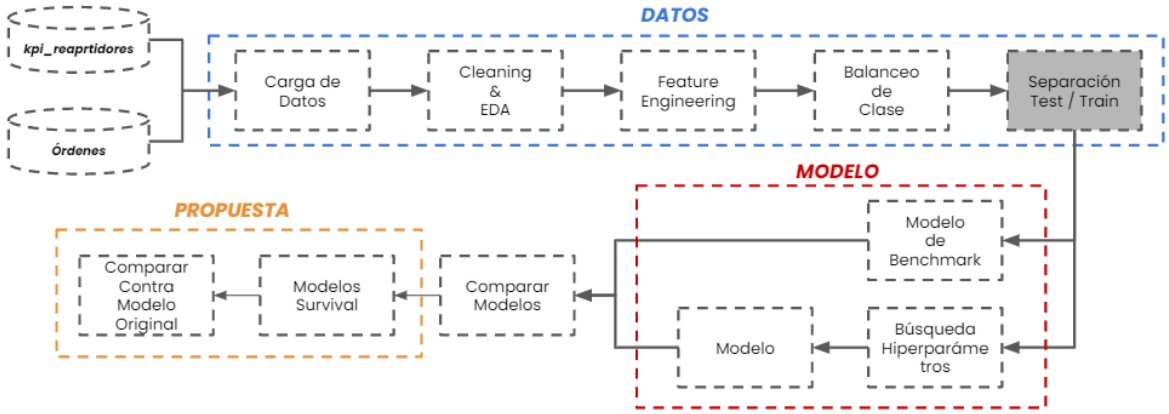
Como hemos observado en el capítulo anterior, una de las principales problemáticas que presenta el dataset utilizado es que posee un gran desbalance en sus clases. Podemos observar un 5% de observaciones definidas como *churn* y un 95% de observaciones definidas como *no_churn*. Debido a que esta desproporción entre observaciones podría generar un rendimiento no deseado en el modelo, es necesario tomar medidas para mitigar su impacto en el desarrollo del modelo. La primera medida tomada para reducir el efecto del desbalance de clases fue eliminar observaciones de la clase sobrerrepresentada. La eliminación de las observaciones se llevó adelante con las siguientes consideraciones:

1. Se eliminaron los user_id, es decir los repartidores, que nunca hicieron churn en ningún momento de la serie de datos de panel.
2. Como la primera medida no fue suficiente porque continuaba la desproporción, decidimos eliminar el 20% (percentil 20) de los repartidores que poseían el ratio churn / no churn más bajo.

Es decir que se eliminaron los repartidores que se encuentran en gran cantidad de semanas como no churn respecto a las semanas churn o bien en ningún período de tiempo realizan churn. Luego de esta eliminación, la proporción de observaciones clasificadas como churn son del 12,5% mientras que el 87,5% restante se encuentran clasificadas como no churn. A pesar de haber tomado esta medida, eliminando

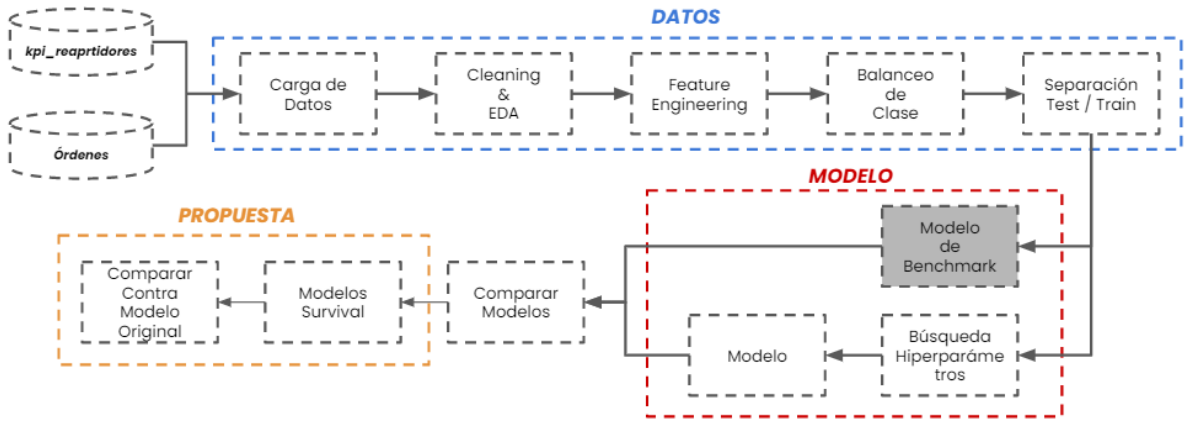
observaciones que sobre representaban no churn, el dataset continuaba presentándose sumamente desbalanceado. Es por ello que, en lo sucesivo, controlaremos el desbalance entre clases redefiniendo el umbral a partir de la probabilidad predicha por los modelos para cada clase.

Separación en test y train



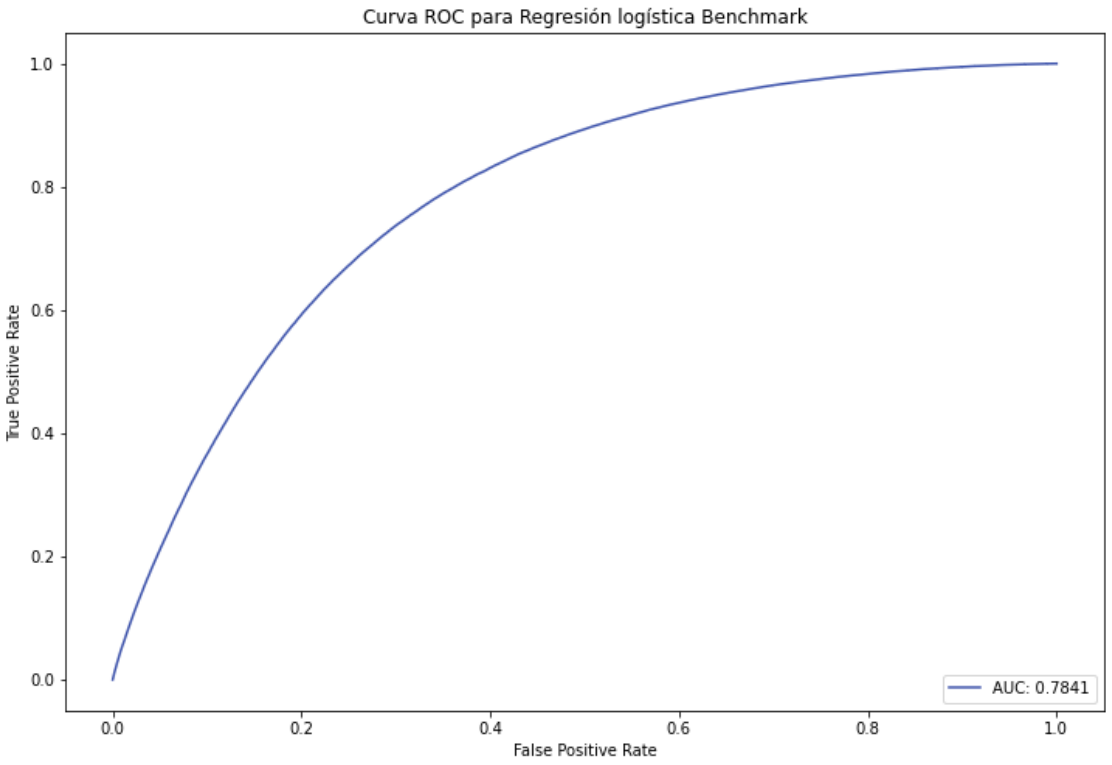
Como ya hemos mencionado en varias oportunidades, a lo largo del desarrollo del presente trabajo, el dataset analizado tiene la particularidad de ser una serie de datos de panel. Esta condición es relevante a la hora de separar en *test* y *train* porque un mismo repartidor no puede pertenecer a ambos grupos a la vez para diferentes períodos de tiempo. Es por ello que la decisión de separar en *test* y *train*, fue tomada a nivel id de repartidor, garantizando que cada observación se mantenga todos los períodos de tiempo correspondientes en un mismo grupo, test o train. Gracias a la gran cantidad de observaciones que posee el dataset, se ha podido destinar una gran proporción de observaciones a train, un 95%, mientras que para test se destinó el 5% restante. En cada uno de estos grupos se mantienen aproximadamente las proporciones originales de 12,5% y 87,5% para churn y no churn, respectivamente.

Modelo de Benchmark

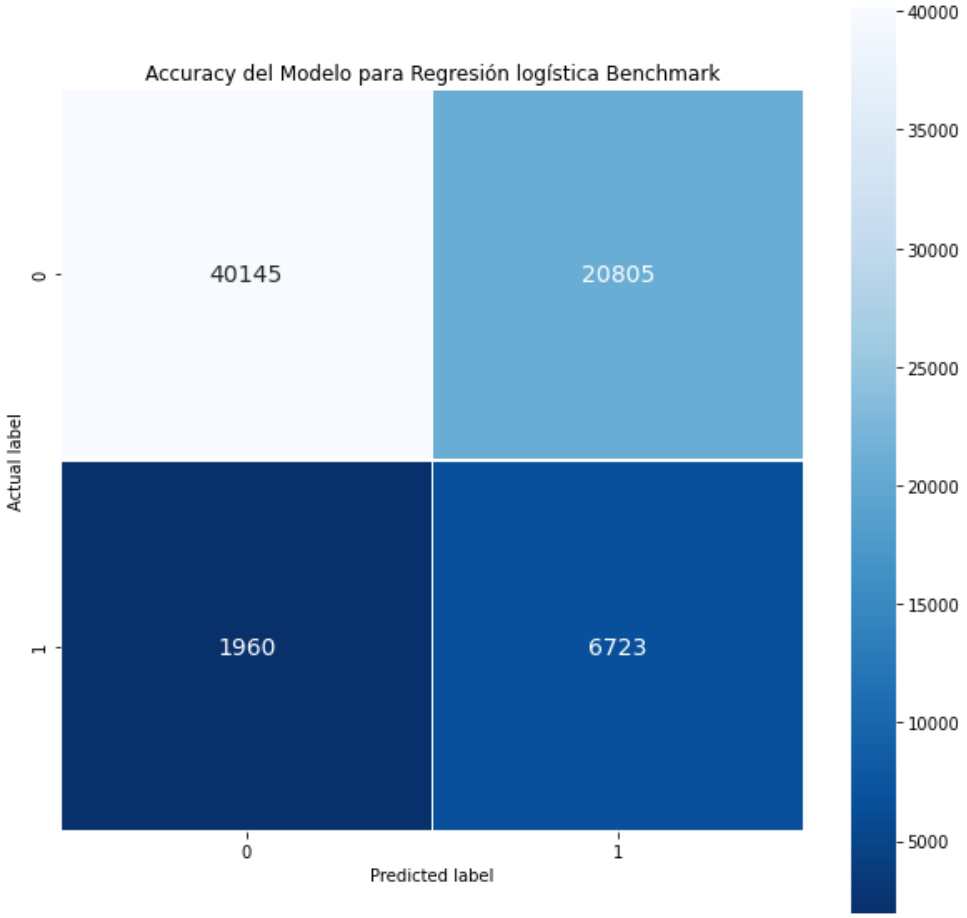


Antes de comenzar a desarrollar el modelo final, creamos dos modelos más simples que utilizaremos a modo de benchmark con el objetivo de compararlos con el modelo final. Se crearon una regresión logística, por un lado, y por otro un modelo de Random Forest.

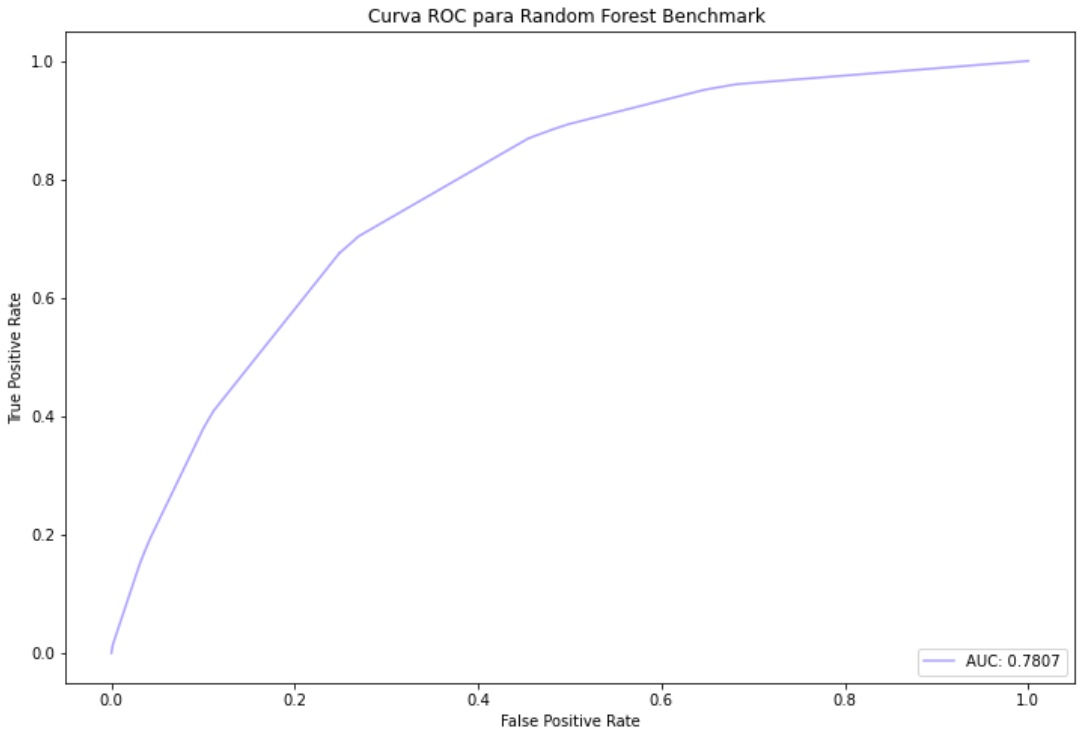
Antes de entrenar los modelos de benchmark se escalaron los datos del dataset según una distribución normal estándar. Luego, entrenamos ambos modelos con los valores de hiperparámetros que tienen por defecto. Sin embargo, una vez entrenado el modelo, llegado el momento de predecir, nos encontramos con que las predicciones no se ajustaban a los valores reales (recordamos del capítulo anterior que, con el objetivo de atacar el problema del desbalance, decidimos buscar una umbral para mejor asignación de clases según probabilidad). Dado esto, siguiendo las prácticas mencionadas en Brownlee, J. (2021), optamos por variar los umbrales prediciendo la probabilidad de pertenencia a la clase en lugar de la categoría. Una vez obtenida la predicción de probabilidad, buscamos estos umbrales utilizando sólamente los datos de entrenamiento para evitar *data leakage*. De esta forma, nos quedamos con el umbral que maximice la métrica de éxito del modelo, en este caso elegimos el coeficiente de Youden como métrica de éxito (descripto más abajo). Una vez definido este umbral con los valores de train, se predice siguiendo el mismo procedimiento pero esta vez sobre los valores de test, prediciendo probabilidad primero y luego, a partir del umbral hallado en train, prediciendo clase.



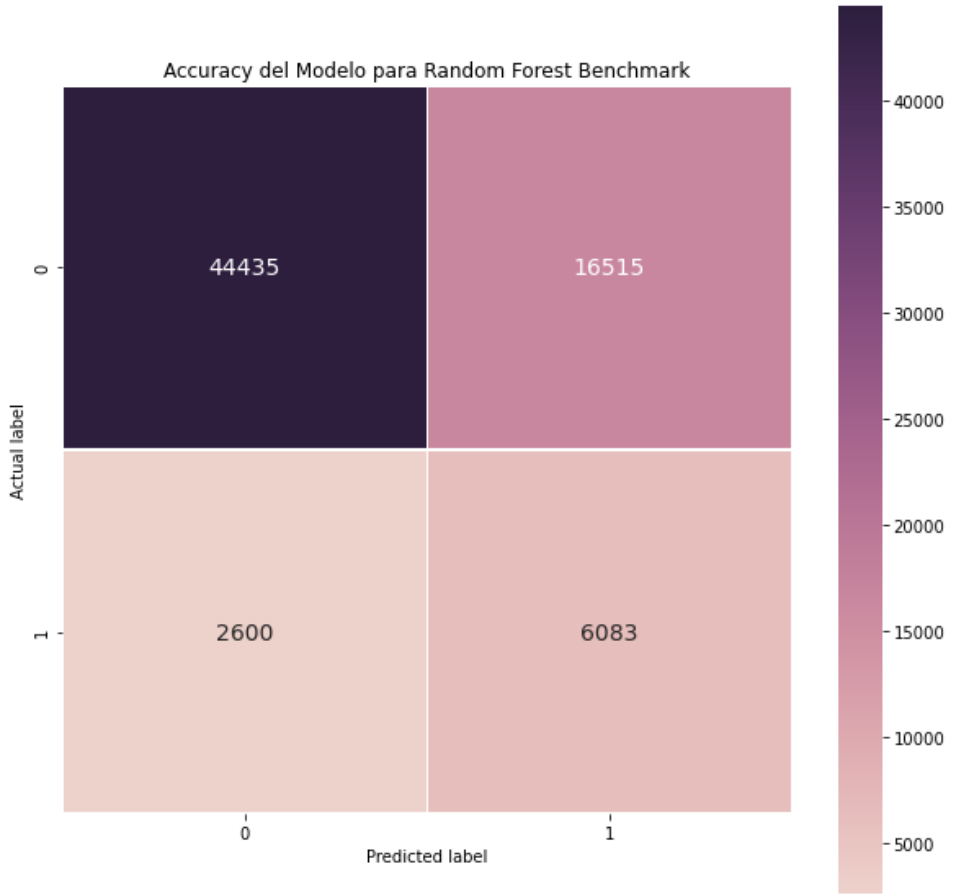
Análisis Predictivo I - Imagen 8: Curva ROC Regresión Logística



Análisis Predictivo I - Imagen 9: Matriz Confusión Regresión Logística



Análisis Predictivo I - Imagen 10: Curva ROC Regresión Logística

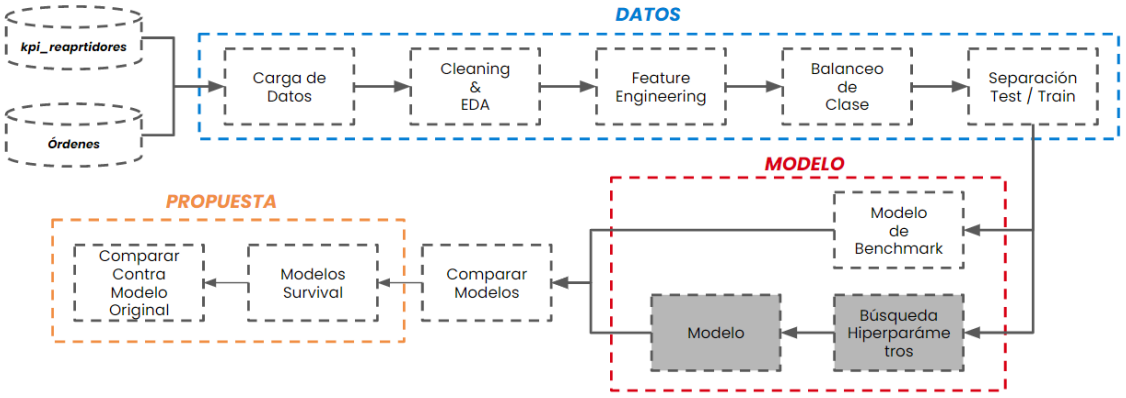


Análisis Predictivo I - Imagen 11: Matriz Confusión Regresión Logística

Luego de entrenar los modelos de benchmark, encontrar el valor más adecuado de umbrales y predecir en test, se obtuvieron los resultados que se observan en las imágenes precedentes 8, 9, 10 y 11. Se entrenaron los dos modelos correspondientes, obteniendo los siguientes valores en test:

- Regresión Logística AUC: 0.7772
- Random Forest AUC: 0.7727

Entrenamiento del Modelo - Búsqueda de Hiperparámetros



Luego de entrenar y testear los modelos de benchmark, creamos el modelo final. Optamos por un modelo de Boosting (Hastie, T. et al. 2017), por su ventaja como modelo de ensamble al combinar más de un modelo débil para lograr un modelo que posea un mejor desempeño. El modelo es aplicado a través de XGBoost. Sin embargo, antes de dar lugar a su desarrollo fue necesario buscar aquellos hiperparámetros óptimos para el mismo. Para ello, creamos una grilla, como se menciona en Gupta, A. (2021), de hiperparámetros para las variables: *max_depth, eta, colsample by tree, min child weight, max delta step, gamma, subsample*. Según la propia documentación de la librería a utilizar (sickit learn), el significado de cada hiperaparámetro, es el siguiente:

max_depth: Es la máxima profundidad que puede poseer cada árbol del modelo.

eta: Su valor se encuentra entre 0 y 1. Es el ratio de aprendizaje del algoritmo.

colsample by tree: Define la proporción de columnas que toma el modelo en cada paso de entrenamiento.

min child weight: Define la cantidad mínima de la suma de los pesos que debe tener una hoja en cada árbol.

max delta step: Determina el peso de cada árbol en el modelo. Este parámetro puede ayudar a mejorar la clasificación en casos de clases desbalanceadas.

gamma: Variable que define la mínima reducción de pérdida requerida para realizar una división en un nodo del árbol.

subsample: Define la proporción de observaciones que serán aleatoriamente tomadas para cada entrenamiento.

Luego de crear la grilla de hiperparámetros, se validaron los valores del mismo buscando aquellos que maximicen el coeficiente de Youden, a través de un random grid search. La validación de la grilla de hiperparámetros, se realizó a partir de un *k-fold cross validation* con un valor de k igual a tres. El tiempo total que demoró la búsqueda de hiper parámetros fue 914 minutos, aproximadamente 15hs.

Finalmente, los valores obtenidos para cada uno de los hiperparámetros buscados fueron:

best_params = {'colsample_bytree': 0.710513222270019,
'eta': 0.08035985346443755,
'gamma': 1.7749542059016372,
'max_delta_step': 6,
'max_depth': 10,
'min_child_weight': 28,
'subsample': 0.9827203626796498}

Luego de definir los valores para cada uno de los hiperparámetros, se entrenó el modelo de boosting. Este modelo fue entrenado del mismo modo que el modelo de benchmark, es decir que se predijo para cada observación la probabilidad de pertenencia a la clase 1 y se buscó el umbral que maximice el índice de Youden utilizando valores de train.

El estadístico de Youden (1950), es una medida estadística utilizada para evaluar el rendimiento de una prueba o modelo de clasificación binaria. Cuantifica la capacidad de una prueba para identificar correctamente los casos positivos y evitar los falsos positivos. Se define como la diferencia entre la sensitividad (tasa de verdaderos

positivos) y el complemento de la especificidad (tasa de verdaderos negativos) de un clasificador binario. Se calcula de la siguiente manera:

$$Índice\ Youden\ (J)\ =\ Sensitividad\ +\ Especificidad\ -\ 1$$

Sabiendo lo siguiente:

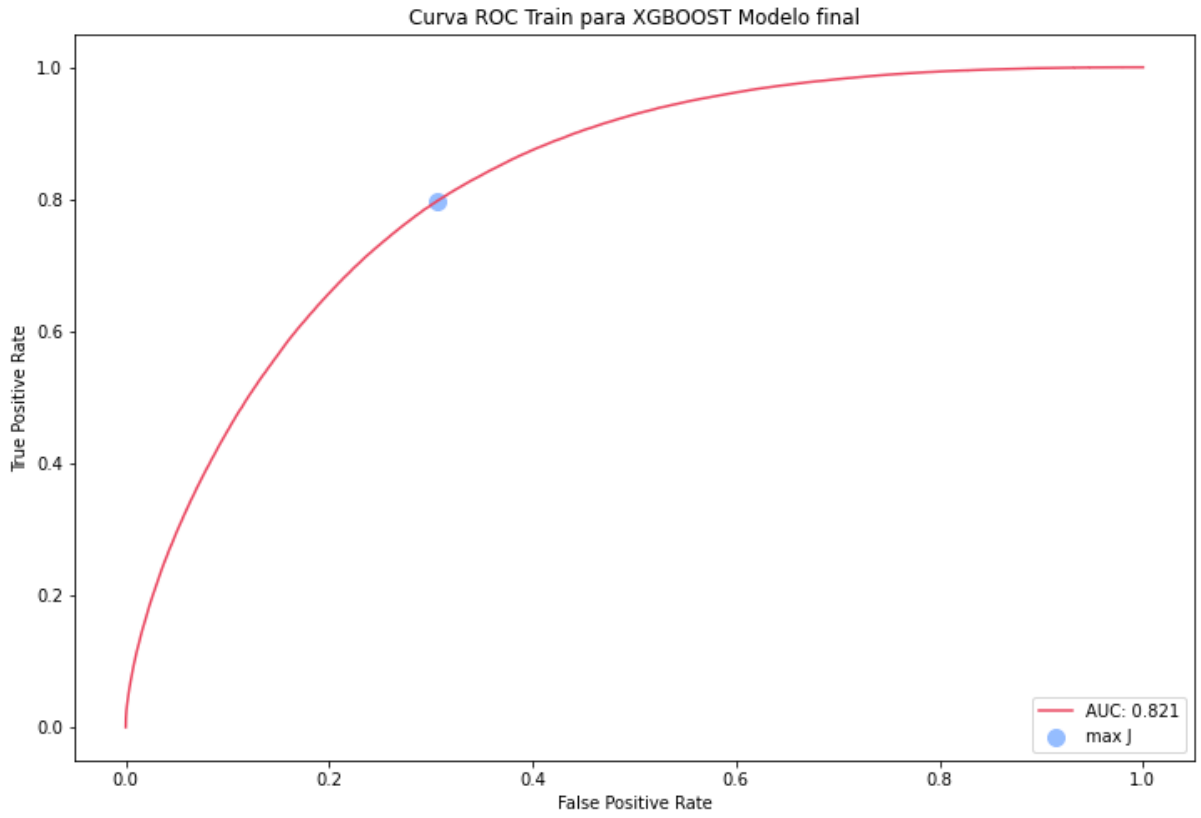
$$Sensitividad\ =\ \frac{Verdaderos\ positivos}{(verdaderos\ positivos + falsos\ negativos)}$$

$$Especificidad\ =\ \frac{Verdaderos\ negativos}{(verdaderos\ negativos + falsos\ positivos)}$$

A partir de esto, podemos reexpresar el estadístico de Youden del siguiente modo:

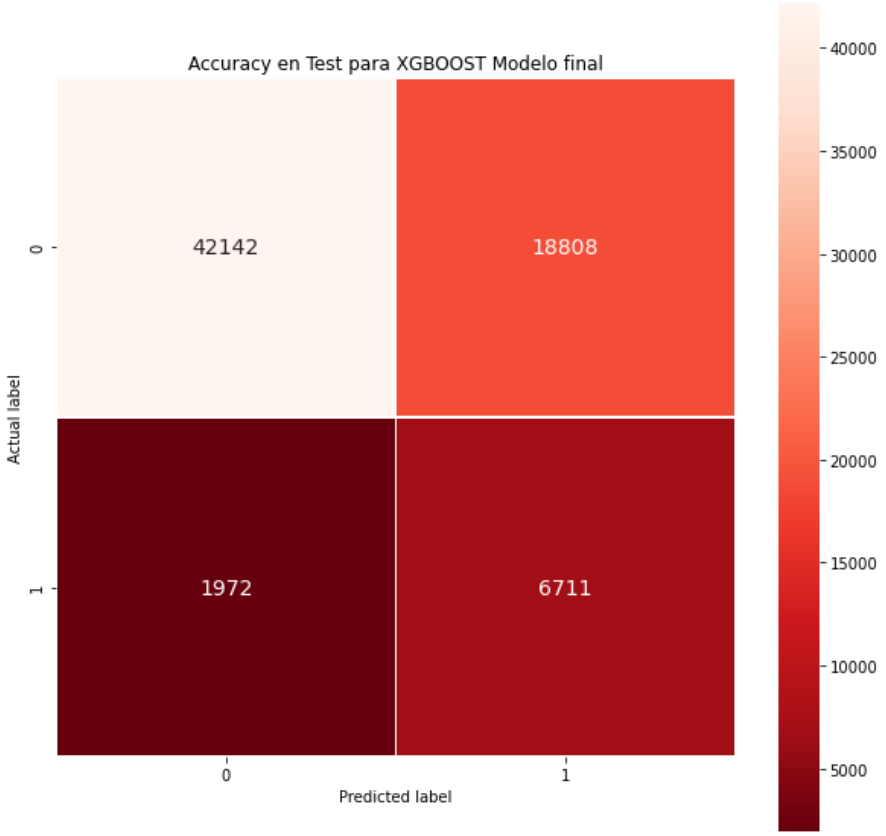
$$Índice\ Youden\ (J)\ =\ Proporción\ de\ Verdaderos\ positivos\ -\ Proporción\ de\ verdaderos\ negativos$$

Al maximizar el Índice de Youden, es posible determinar el umbral óptimo para un clasificador binario que logre el mejor equilibrio entre sensitividad y especificidad.



Análisis Predictivo I - Imagen 12: Búsqueda umbral y curva ROC

La imagen 12: búsqueda del umbral y curva ROC, muestra cómo evoluciona la combinación de los ratios de verdaderos positivos y verdaderos negativos en cada punto de la curva, en el modelo con datos de test para cada valor de umbral. El umbral elegido fue el que maximiza el valor del estadístico de Youden.



Análisis Predictivo I - Imagen 13: Matriz de Confusión

Una vez definido el valor de umbral, se testeó el modelo con valores de test y se obtuvo la matriz de confusión que se puede apreciar en la imagen 13. El AUC para test fue de 0.806

De este modo, el valor de AUC para el nuevo modelo es +0.025 mejor que el obtenido para el mejor modelo de benchmark. Del mismo modo como se mencionó en el marco teórico, debido al desbalance que presenta la clase y la mayor importancia en clasificar correctamente las observaciones que realizan churn y debido a la importancia para el negocio, se decidió utilizar el AUC como métrica de éxito en lugar del accuracy u otras métricas. Es decir, para el negocio tiene mayor importancia predecir correctamente las observaciones que son churn, que las que no son no churn. Dado que puede accionar sobre las primeras enviándole ofertas de retención y enviar una oferta de retención a un repartidor que no abandona la compañía, es menos grave que perder un repartidor.

Debajo podemos apreciar los valores de las métricas de éxito obtenidos para cada uno de los modelos entrenados.

	Regresión Logística	*Random Forest*	*XGBoost*
Train	0.787	0.782	0.822
Test	0.784	0.781	0.806

Como mencionamos anteriormente, la medida de éxito elegida fue el AUC. Esto se debe a que el accuracy no alcanza a captar la complejidad del negocio. Y, encontrándonos con un dataset altamente desbalanceado, un modelo ingenuo que prediga siempre la clase mayoritaria, acertaría 9 de cada 10 veces. Sin embargo, por la mencionada complejidad de negocio resulta más importante la clasificación de los verdaderos positivos, es decir clasificar correctamente a un repartidor que hará churn frente a clasificar incorrectamente a un repartidor que no hará churn.

Análisis predictivo II: Survival model

Modelo - Survival Analysis

Hacia el final del presente trabajo, proponemos un modelo alternativo para el problema de *churn* donde la cuestión no pasa por predecir si un rider dejará de repartir, sino por predecir desde el inicio, de su período como rider, el tiempo que se encontrará activo en la aplicación. Este nuevo enfoque, permite la posibilidad de tomar diferentes decisiones de negocio desde el inicio del vínculo rider-organización.

Este nuevo modelo, ajeno a todos los modelos antes utilizados, se denomina *Survival Analysis*. Según Emmert-Streib, F. et al. (2019), *Survival Analysis* es un método utilizado para, a partir de una colección longitudinal de datos, obtener como salida del modelo una variable que se encuentra definida en una escala de tiempo (días, semanas, meses, años). Habitualmente, se utiliza en diferentes campos como la medicina y el marketing. Este modelo genera a partir de diferentes atributos (variables), una función de probabilidad particular para cada observación (rider). De esta forma, se define una curva de probabilidad *S(t)* en función del tiempo. S(t) se denomina la función de supervivencia (*Survival Function*). Por lo tanto, S(t) podemos definirlo como la probabilidad de que una variable aleatoria T sea mayor a un determinado tiempo t.

$$S(t) = P(T > t)$$

Este modelo de Survival se encuentra definido a partir de un estimador no paramétrico llamado el estimador de *Kaplan-Meier*. Este estimador puede ser calculado para toda una población e individualizado para cada observación a partir de los valores de las variables independientes. De este modo, se puede calcular el estimador de forma recursiva con la siguiente ecuación:

$$S_{KM}(t_k) = \frac{n_{k-1} - d_{k-1}}{n_{k-1}} S_{KM}(t_{k-2})$$

Donde n representa el número de individuos en un período determinado de tiempo y d representa el número de individuos que experimentaron el evento de interés en el mismo período de tiempo, mientras que k representa el instante de tiempo analizado.

Este estimador representa el desafío de que no todas las observaciones habrán alcanzado el momento en el tiempo en que dejarán de "sobrevivir". Para ello, deberemos definir las observaciones que se encuentran *censuradas*. Entendiendo por censuradas a aquellas observaciones que aún no hayan finalizado su período, en el caso de los riders serían aquellos riders que no hayan hecho *churn*.

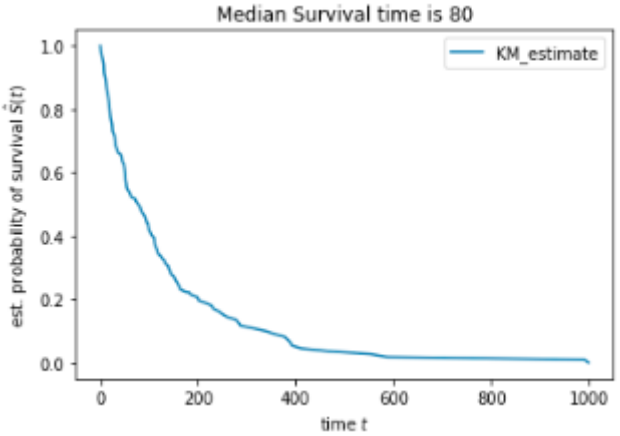


Imagen 14 - Evolución probabilidad de survival

En la Imagen 14 - Evolución probabilidad de survival, tomada a modo de ejemplo para graficar el objetivo del *survival analysis*, podemos apreciar cómo evoluciona la probabilidad de supervivencia para para un conjunto de observaciones. La mediana de supervivencia de las observaciones que componen el universo para crear dicha función, es de 80 períodos de tiempo, pudiendo estos ser días, semanas, meses, años.

Librería Scikit Survival

Scikit Survival es una librería de Python que permite aplicar modelo de *survival analysis*. Dicha librería se encuentra basada en la popular librería Scikit Learn. Según la descripción de la propia librería, su objetivo es: "El objetivo del análisis de supervivencia (también conocido como tiempo hasta el evento o análisis de confiabilidad) es establecer una conexión entre las covariables y el tiempo de un evento". Lo que hace que el análisis de supervivencia difiera del aprendizaje automático tradicional es el hecho de que partes de los datos de entrenamiento solo se pueden observar parcialmente: están censurados".

Esta librería, ofrece todas las funcionalidades de Scikit learn con el agregado de aquellas específicas para los modelos de *survival analysis*. Dentro de estas funcionalidades

específicas podemos encontrar una función para medir el concordance-index, otra para crear el modelo de survival propiamente dicho.

Surgimiento del modelo alternativo

El modelo originalmente propuesto tiene como objetivo responder a la pregunta de si un repartidor que se encontrara activo en una semana W realizaría churn de allí en adelante. De este modo, esta solución permitiría a la organización idear algún plan de acción. Veremos algunos ejemplos de aplicación en el último capítulo, con el objetivo de retener a esos repartidores. Sin embargo, existe otro enfoque posible relacionado al tiempo de permanencia que se espera que un rider se encuentre activo en la plataforma. Así, si por ejemplo pudiéramos conocer con certeza que un rider se encontrará activo durante n cantidad de semanas, se podrían tomar decisiones en consecuencia.

Esta segunda alternativa mencionada, y esta segunda problemática, es la que busca responder este nuevo enfoque. La ventaja que posee sobre el primero es que agrega información, ya que no se trata de si un repartidor hará, o no, churn; sino del momento en que hará churn. Si este momento del tiempo fuese demasiado extenso, podríamos concluir que nunca hará churn. Este nuevo enfoque permite tomar decisiones vinculadas con el recupero de los costos asociados al inicio de los repartidores en la app, la curva de aprendizaje y demás.

Tal como se mencionó en el apartado anterior, el modelo originalmente propuesto predice para cada período de tiempo W si un repartidor hará churn en el período posterior. La principal ventaja de ese modelo, frente al nuevo propuesto, es que otorga información inmediata sobre los períodos de tiempo inmediatamente posteriores. Sin embargo, la desventaja que presenta es que para el momento W_0 no es posible conocer, efectivamente, la cantidad de intervalos de tiempo que un rider se encontrará activo.

Por su lado, la principal ventaja que posee el modelado alternativo -*survival*- es que desde el inicio del ciclo del rider podemos conocer el período promedio de tiempo de actividad, a partir de una función de probabilidad individual para el repartidor. Es decir que no sólo podríamos conocer el momento en que realizaría churn, sino además dentro de cuánto tiempo.

Una diferencia notable entre ambos modelos es que, el primer modelo se trataba de un modelo de clasificación donde la variable dependiente era una variable categórica (*churn / no churn*), en cambio este segundo modelo posee una variable dependiente continua, el tiempo de actividad de los repartidores en la app.

Preprocesamiento de los datos

El dataset original, como se puede apreciar en el apartado dedicado exclusivamente a su análisis, consta de una serie de tiempo particular para cada rider, es decir una serie de datos de panel. Sin embargo, el nuevo modelo propuesto toma los datos (inputs) de un modo diferente. El nuevo modelo, se detiene en el momento cero (t_0) y a partir de ciertas características de cada una de las observaciones, define su curva de "supervivencia", calculando a partir de una función de probabilidad única, la probabilidad de que la observación continúe activa para cada período (t_n) de tiempo futuro. Además de esta primera limitación del modelo, se presenta el inconveniente de que el modelo no posee un buen manejo de datasets con alta dimensionalidad, es decir con dataset que posean una gran cantidad de variables.
De este modo, debido a las dos limitaciones mencionadas anteriormente, fue necesario tomar dos medidas:

1. Por un lado fue necesario agregar los valores a las dos primeras semanas de cada repartidor. Es decir que se creó un nuevo dataset donde se posee una única línea por observación, esta línea contiene datos referidos a un único repartidor para las dos primeras semanas de observación. Esto se debe a que el funcionamiento del algoritmo predice hacia adelante a partir de un momento inicial. En este caso y por cuestiones relacionadas a la operación del negocio particular de la empresa que brinda los datos, se decidió tomar ese período de tiempo inicial de dos semanas.
2. En segundo lugar, debido a que el modelo no funciona de un modo óptimo en contextos de alta dimensionalidad, fue necesario definir algún modo adecuado para seleccionar las features más relevantes. Para ello, se aprovechó el modelo de Random Forest utilizado como benchmark anteriormente y, luego se consultar

por las variables más relevantes, estas fueron las elegidas como input para el modelo.

De esta forma, el nuevo dataset contiene 108.099 observaciones que se corresponden con 108.099 riders id únicos. Y la variables contenidas en el mismo son:

- *completed_deliveries*
- *cancelled_deliveries*
- *avg_distance_pickup_to_vendor_m*
- *working_hours*
- *stacked_orders*
- *rider_delivery_time*
- *late_shift_ratio*
- *duration*
- *event* (define si un evento se encuentra censurado, o no, según las definiciones del modelo)
- *sqr_orders_working_hours*
- *log_sqr_orders_working_hours*

Según el output de la importancia se variables para el modelo de random forest, se obtuvo el siguiente resultado:

	importance
sqr_orders_working_hours	0.499037
log_sqr_orders_working_hours	0.462874
near_dropoff_error_pct	0.038089

Dado esto, dos de las variables que resultan más relevantes para el modelo son variables creadas en la sección de feature engineering: *sqr_orders_working_hours* y *log_sqr_orders_working_hours*, la raíz cuadrática de la suma cuadrática de las órdenes y working hour y el logaritmos de dicha raíz, respectivamente. Variables que fueron creadas como parte del desarrollo de *feature engineering*. Esto podría sugerir que existe algún tipo de relación no lineal entre las variables que el algoritmo llega a captar en su forma agregada.

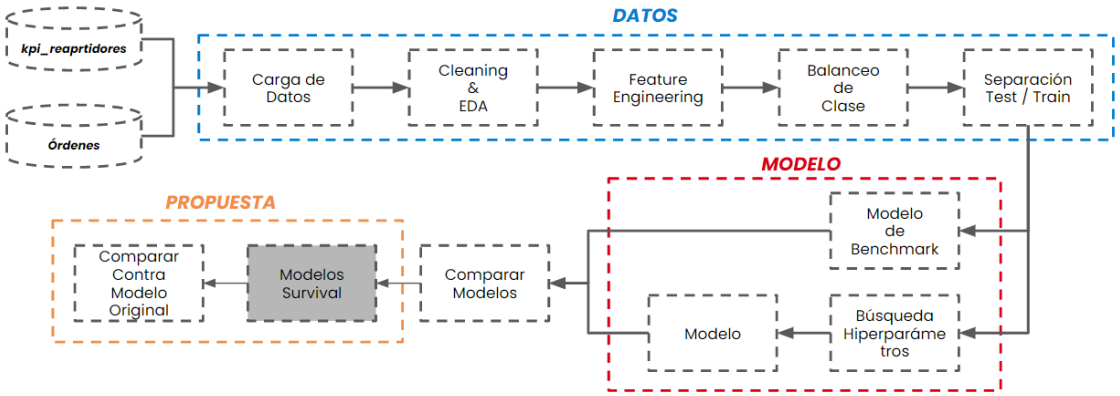
Métrica de Éxito Modelo Survival (C-Index)

Todo este apartado sobre métricas de éxito de los modelos predictivos cobra sentido cuando incorporamos el modelo de *Survival Analysis*. Como se menciona en Raykar et. al (2007), Debido a las características propias del modelo de survival, este posee su propia métrica de éxito asociada, el concordance index. Sin embargo, existe un modo para comparar ambos modelos.

Concordance Index: la primera métrica que analizaremos será la llamada *concordance index* (c-index). El índice de concordancia se podría interpretar como la fracción de todos los pares de observaciones cuya predicción de tiempo de supervivencia se encuentra correctamente ordenada. Dicho de otro modo, es la probabilidad de concordancia que existe entre las predicciones y las observaciones reales. Este índice puede tomar valores entre 0 y 1, siendo su interpretación similar al AUC, en tanto y cuanto un c-index con un valor igual a 0.5 equivale a una predicción aleatoria.

AUC Dinámico: así como se utiliza el AUC para los modelos de clasificación habituales, si consideramos cada período de tiempo como una instancia única y un umbral definido, en ese momento una observación puede haber realizado churn o no. Es por ello que se podría calcular el AUC para el período de tiempo entre t_0 y t_n. A este AUC para cada período de tiempo, es lo que llamaremos AUC dinámico, su condición de dinámico se debe principalmente a la característica de posicionarse en cada período de tiempo. De este modo, podríamos calcular el AUC promedio para todos los períodos de tiempo y compararlo con AUC del primer modelo con el objetivo de entender si un modelo resulta mejor que otro.

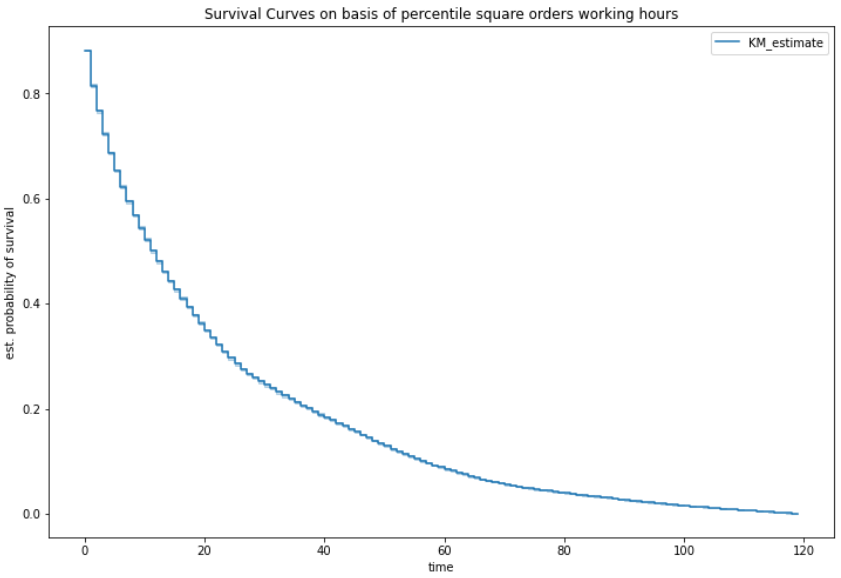
Modelo survival base - benchmark



Del mismo modo a como se operó para el primer modelo, en este caso también se creó un modelo base (*benchmark*), sin modificar hiperparámetros ni incorporar ninguna otra variable de relevancia.

Debido a que este nuevo dataset posee una observación única por línea, la serie de datos de panel ya no existe y la variable dependiente pasa a ser una variable continua, de este modo se puede realizar una separación entre test y train de forma convencional, sin la preocupación de que un mismo id se encuentre en más de una clase. Se decidió conservar un 30% de las observaciones para test, mientras que el 70% se destinaron a entrenar el modelo.

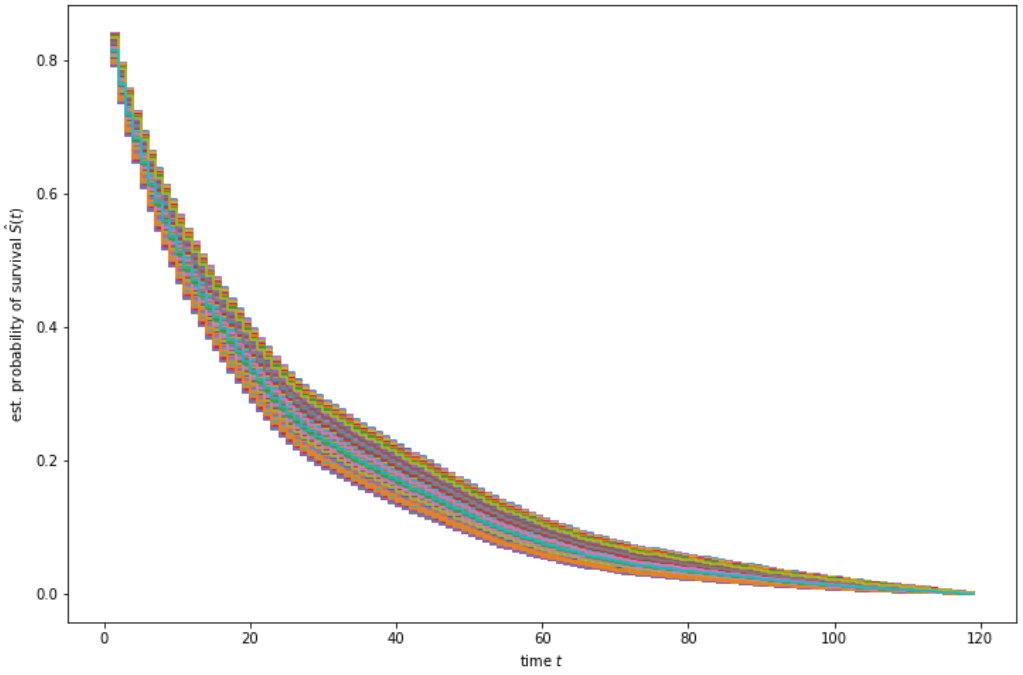
Si bien la sintaxis de la librería propia para aplicar este tipo de modelo, posee su particularidad, esencialmente el *pipeline* es el mismo que para cualquier otro modelo.



Análisis Predictivo II - Imagen 15: Curva de probabilidad -Kaplan Meier-

La curva de probabilidad para toda la población, conserva la forma que puede apreciarse en Análisis Predictivo II - Imagen 15: Curva de probabilidad -Kaplan Meier-.

Donde se puede ver cómo, en la medida en que transcurre el tiempo, la probabilidad de encontrar activa una observación disminuye, para tender asintóticamente a cero hacia las 120 semanas.



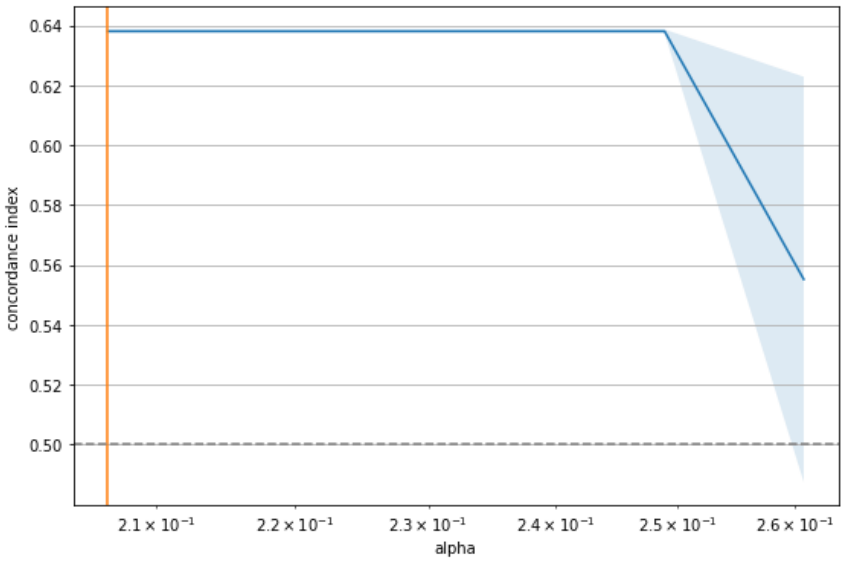
Análisis Predictivo II - Imagen 16: Curva de probabilidad -Kaplan Meier- para cinco observaciones

Asimismo, se puede graficar la curva de "supervivencia", a partir de la función de probabilidad particular para cada observación. En la imagen 16 se puede observar cómo evolucionan, de una forma muy similar las curvas de probabilidad para un conjunto aleatorio de observaciones.

Para este primer modelo propuesto, el indicador C-index, que luego será comparado con la versión mejorada del modelo, arrojó el valor de 0.61 para datos de test.

Modelo survival mejorado

Del mismo modo como con los modelos habituales, como por ejemplo el desarrollado a través de XGBoost en este mismo trabajo, se buscan hiperparámetros. Para este modelo de survival también existe esta búsqueda de hiperparámetros. Como se menciona en Harrell, F.E. (2015) el modelo survival, posee un hiperarámetro llamado α (alfa), cuya función es definir la velocidad con la que aprende el modelo penalizándolo.



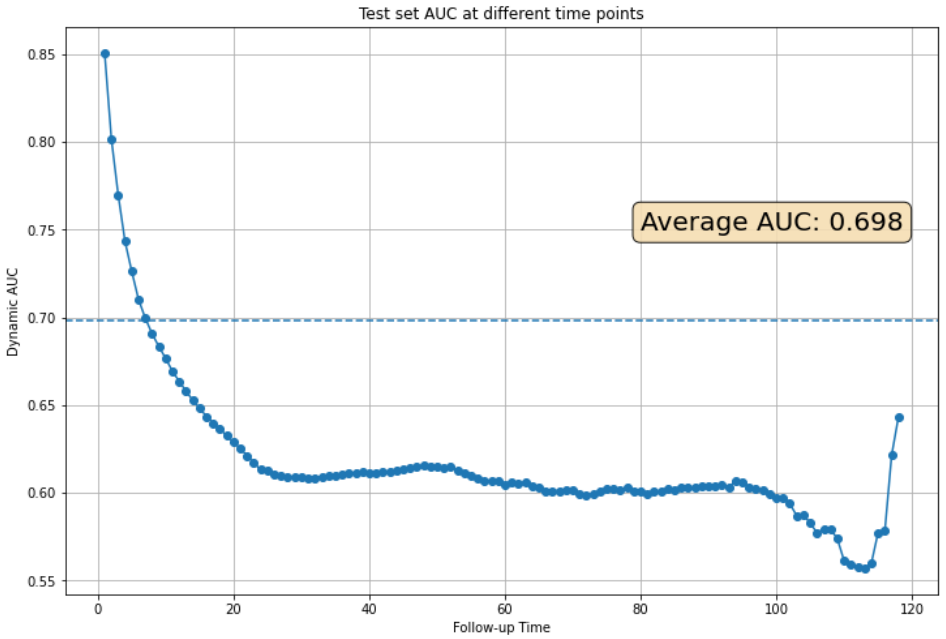
Análisis Predictivo II - Imagen 17: Búsqueda de hiperparámetro alpha

Esta búsqueda de hiperparámetro, se realizó a través de un K-Fold validation, con un valor de K igual a 5. La métrica de éxito contra la que se fue comparando esta búsqueda del hiperparámetro, fue el concordance index. Luego de la búsqueda de hiperparámetros, el mejor valor de concordance index hallado fue de 0.63, en un valor de alpha igual a 0.21. Tal como puede apreciarse en la imagen 17.

Finalmente, luego de encontrar el mejor valor para el hiperparámetro alfa, se entrenó y testeó el modelo de survival. El valor de c-index para el modelo final de survival fue de 0.69, generando una mejora de 0.08 respecto al modelo de survival que se utilizó como benchmark.

Comparación entre modelos

Luego de desarrollar el nuevo modelo de survival. Surge la cuestión sobre cómo se comporta este nuevo modelo respecto al original y, de ello, la necesidad de compararlos. Debido a que ambos modelos utilizan métricas de éxito diferentes, fue necesario converger la métrica de los modelos hacia una homogénea, que sea comparable entre ambos. Para este fin, se utilizó el AUC dinámico que se presentó en la sección de métrica de performance.



Análisis Predictivo II - Imagen 18: AUC dinámico para modelo de survival fine tuning

En la imagen 18 se puede apreciar cómo se comporta el AUC para cada período de tiempo W_k. De esta manera, se puede obtener un AUC promedio para todos los períodos de tiempo analizados. Luego de obtener el AUC promedio para todos los períodos de tiempo, se puede comparar ese AUC contra el obtenido en el modelo original.

Para el modelo categórico original, creado con el algoritmo XGBoost, se obtuvo un valor de AUC igual a 0.802, es decir que el modelo categórico creado a través de boosting es mejor en 0.104 puntos respecto al modelo de survival. Sin embargo, existe un trade off entre la efectividad del modelo y el momento en que se posee la información. Es decir, el modelo original brinda información de forma más inmediata, en cuanto a que nos dice qué es lo que sucederá en la próxima semana. En cambio, el segundo modelo nos dice qué sucederá en todos los períodos de tiempo. En el apartado siguiente profundizaremos en la comparación de los modelos y las ventajas y desventajas de tomar decisiones en cada período de tiempo.

Análisis prescriptivo

Comparación de modelos en la toma de decisiones

Las decisiones de negocio, en su mayoría, se encuentran circunscritas al ámbito de la incertidumbre. Los acontecimiento que influyen en los negocios son, generalmente, inciertos. Esta incertidumbre dominante dificulta la toma de decisiones, ya que desconocer hacia dónde se moverán ciertos parámetros, o qué sucederá, implica en muchas ocasiones no terminar poder ser del todo eficaces en las acciones que se implementan.

Una de las funciones de los algoritmos de aprendizaje automático aplicados a decisiones de negocio, es reducir la incertidumbre proporcionando información oportuna para la toma de decisiones. Sin embargo, es importante considerar las implicancias de los errores de predicción (tipo I y tipo II) en relación con la estrategia de negocio que será ejecutada posteriormente.

En el caso de este estudio, se evaluaron dos modelos: el modelo de boosting de clasificación y el modelo survival. Si bien el modelo de boosting de clasificación demostró un mejor rendimiento en términos de precisión de la predicción, es necesario tener en cuenta las diferencias claves entre ambos modelos.

Como se mencionó en el apartado de comparación de modelos al final del capítulo de survival, la principal diferencia que ofrecen ambos modelos es el momento en que puede tomarse una decisión asociada al churn. En el caso del modelo original, este brinda información en la semana anterior a que un rider realice, o no, churn, mientras que el modelo de survival nos brinda la información desde el momento inicial W_0. Esta diferencia de tiempo presente entre ambos modelos puede resultar crucial para tomar decisiones oportunas, más allá de la diferencia de rendimiento.

Dicho esto, como mencionamos al inicio de este apartado, resulta relevante considerar las implicancias de los errores de predicción en la estrategia de negocio. Por ejemplo, los errores de tipo I, es decir, marcar un churn que no se materializa, podrían llevar a la implementación de acciones innecesarias, como ofrecer incentivos de retención a repartidores que no tienen intención de abandonar la plataforma. Esto podría generar costos adicionales y recursos desperdiciados.

Por otro lado, los errores de tipo II, es decir, no detectar a tiempo un churn que sí ocurre, podrían resultar en la pérdida de repartidores valiosos para el desempeño del negocio y así ofrecer al cliente final una experiencia por debajo de la esperada, dañando la reputación de la plataforma. Además, la falta de detección temprana del churn puede dificultar la implementación de estrategias de retención efectivas y pertinentes.

Es fundamental considerar estos costos asociados al tomar decisiones basadas en los resultados del modelo de predicción de churn. La elección del modelo adecuado y la comprensión de sus diferencias temporales pueden ayudar a minimizar los errores y tomar decisiones más informadas y oportunas en el contexto de las organizaciones circunscritas al modelo de negocio de plataformas.

Decisiones en el modelo original

El modelo planteado originalmente tenía como objetivo tomar decisiones inmediatas para reducir el churn de riders. Sin embargo, es importante profundizar en la aplicación práctica de este modelo en el negocio y considerar estrategias más detalladas para abordar el churn. La gestión efectiva del churn requiere acciones específicas por parte de las organizaciones para retener a los clientes que el modelo identifica como propensos a abandonar.

Por ejemplo, una estrategia recomendada podría ser enviar ofertas especiales a los riders identificados como propensos a churn en la semana previa al abandono. Estas ofertas podrían incluir incentivos financieros adicionales, como pagos adicionales por orden, bonos especiales por cumplir determinada cantidad de órdenes o beneficios exclusivos para fomentar la lealtad del rider, apalancado en un programa llamado *rewards* donde los repartidores suman puntos para canjear por diferentes premios (por ejemplo otorgar puntos especiales o bonos por puntos). Otras opciones podrían ser brindar preferencias en la asignación de turnos semanales, como la posibilidad de elegir horarios preferidos, o proporcionar incentivos especiales durante ciertas horas de mayor demanda. Algunas de estas estrategias, aunque aplicadas al campo de los clientes finales y no de repartidores que ocupan otro rol en la plataforma, han probado resultar para disminuir el churn, aumentando el valor de los clientes (Kumar et al., 2018).

Es importante considerar que la implementación de estas estrategias conllevan costos y recursos asociados para brindar servicios personalizados. El costo de un abandono en general puede ser significativo para las empresas, ya que podría implicar la pérdida de ingresos futuros, afectar negativamente la reputación de la plataforma y, en este caso en particular, la experiencia del usuario final ofreciendo un menor nivel de servicio. Por lo tanto, es fundamental buscar el equilibrio entre los costos de retención y los beneficios potenciales derivados de la reducción del churn.

Finalmente, una vez implementada la política de retención para aquellos repartidos predichos como churn, podrían realizarse distintos tipos de test (por ejemplo un AB test) con el objetivo de comprobar si la aplicación de una oferta lanzada proactivamente logra reducir el abandono frente a aquellos que no la reciben. Así mismo, podrían realizarse diferentes tests sobre los montos a enviar buscando maximizar la completitud de la flota de repartidores, a un costo que resulte óptimo.

Decisiones en el modelo alternativo

Como se mencionó con anterioridad, el modelo alternativo difiere del original en el espacio de tiempo en que pueden tomarse las decisiones de negocio. Al adelantarse el espacio de tiempo en el que se puede tomar una decisión, la cantidad de acciones que pueden tomarse se amplía considerablemente. Por ejemplo, si se pudiera conocer desde el inicio del ciclo de los repartidores el tiempo que se encontrarán activos, podrían calcularse valores de recupero relacionados a la inversión inicial que tiene la organización en costos relacionados con equipamiento, entrenamiento y reclutamiento de repartidores.

A partir del conocimiento de la probabilidad de encontrarse activo para un período de tiempo determinado, se podría calcular la esperanza de ingresos como la multiplicación del ingreso para cada período de tiempo por la probabilidad de mantenerse activo para cada uno de los mismos y, en consecuencia, podría calcularse el retorno de dichas inversiones. A partir de allí, se podrían establecer parámetros mínimos de actividad a partir de los cuáles se realiza el recupero de la inversión.

Conclusiones

Objetivos propuestos

Los objetivos propuestos inicialmente para este trabajo eran, por un lado, crear un modelo predictivo que permita predecir si un repartidor hará *churn*; y, por otro, explorar y proponer un modelo alternativo que permita tomar otro tipo de decisiones, en un período de tiempo diferente al primero. Dicho esto, podemos concluir que hemos alcanzado ambos objetivos. Por un lado, hemos propuesto un modelo que predice el churn de los repartidores que, si bien es cierto que la mejora que presenta frente a los modelos de benchmark no son lo suficientemente amplias, predice con un alto grado de éxito el churn de los repartidores. Por otro lado, hemos propuesto un novedoso y alternativo modelo que predice la probabilidad de supervivencia para cada período *t* de tiempo, para cada repartidor particular, con las ventajas que esto ofrece de cara al negocio.

Oportunidades de mejora

Las principales oportunidades de mejora detectadas a lo largo del trabajo son dos puntuales. Por un lado, iterar y mejorar el modelo final de XGBoost, de modo de alcanzar un modelo que sea mucho mejor que los propuestos como benchmark, esto podría conseguirse iterando sobre los hiper parámetros, creando nuevas variables, o buscando nuevas alternativas. Algunas de las nuevas variables a incorporar podrían ser el área donde se encontró el repartidor a partir de algún tipo de georreferenciación o centroide, alguna dimensión temporal extra, como períodos de vacaciones o semanas de alta demanda. Se podría analizar la existencia de algún tipo de comportamiento no lineal entre las variables debido a que las variables de mayor importancia en los modelos fueron aquellas que obtuvimos como resultado de combinaciones no lineales. Por otro lado, también se podría entender la posibilidad de haber introducido algún tipo de sesgo a partir de la decisión de eliminar todos los datos relacionados a la pandemia y, en caso de haber sido así, modificarlo.

La segunda oportunidad de mejora, viene del modelo de *Survival*, que terminó por tener valores de éxito mucho menores a los obtenidos por el modelo originalmente propuesto. En este caso, podrían incluirse algunas de las variables que se dejaron fuera del modelo buscar nuevas formas de buscar qué variables incorporar o iterar sobre algunos de los hiperparámetros, ya que solamente iteramos sobre dos de ellos. Así mismo, una de las grandes posibilidades para ambos modelos, sería la incorporación de nuevos datos que permitan continuar aprendiendo agregando información a los ya incorporados. Estos nuevos datos podrían ser datos relativos a los repartidores, datos que actualmente el modelo de datos de la compañía no se encuentra almacenando y/o midiendo, como por ejemplo: el momento en que se toman los turnos y cuántos se dejan y/o cambian. Por otro lado, podría resultar beneficioso agregar datos cualitativos que puedan provenir de encuestas, entrevistas y/o focus groups de los repartidores. Esto último, podría ayudar a realizar algún tipo de segmentación que podría incorporarse al dataset con el que se llevó adelante el análisis y la creación del modelo. Así mismo, en la medida en que vaya transcurriendo el tiempo, podría moverse la ventana temporal de los datos y terminar de quitar, por completo, los datos que se vean afectados, de algún modo, por la pandemia.

Agregado de valor

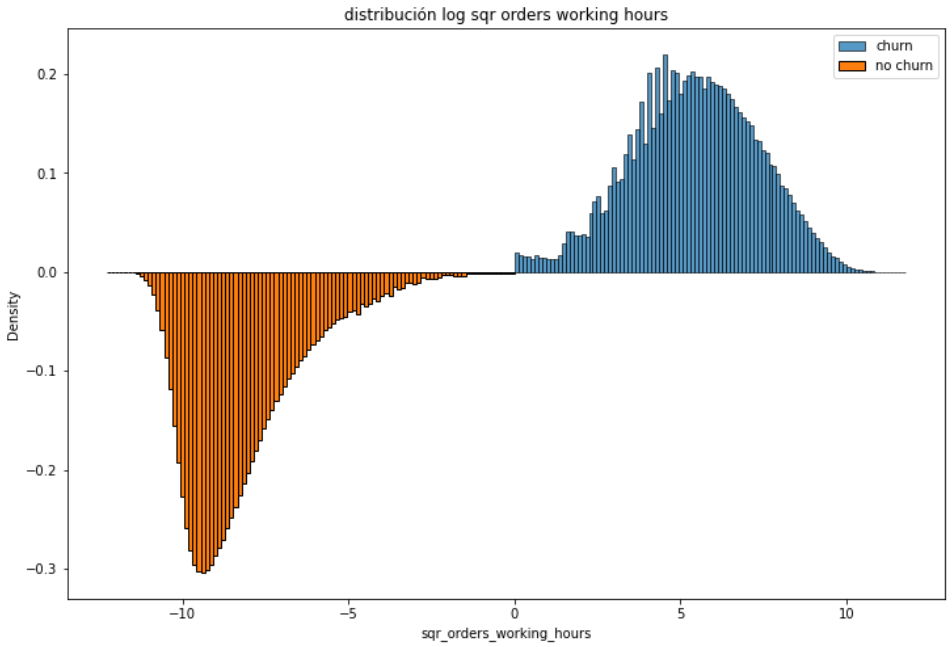
El principal agregado de valor del trabajo consiste en la incorporación de un análisis de churn, habitualmente aplicado a los clientes de empresas, a los repartidores, una suerte de modelo para clientes internos, cambiando el foco de los mismos. Asimismo, se produce otro agregado de valor que vale la pena mencionar al crear un modelo de *survival*, un modelo que, si bien no posee las características propias de los modelos de aprendizaje automáticos actuales, es novedoso en este dominio y entrega un agregado de valor al ofrecer un tipo de información diferente a los modelos convencionales.

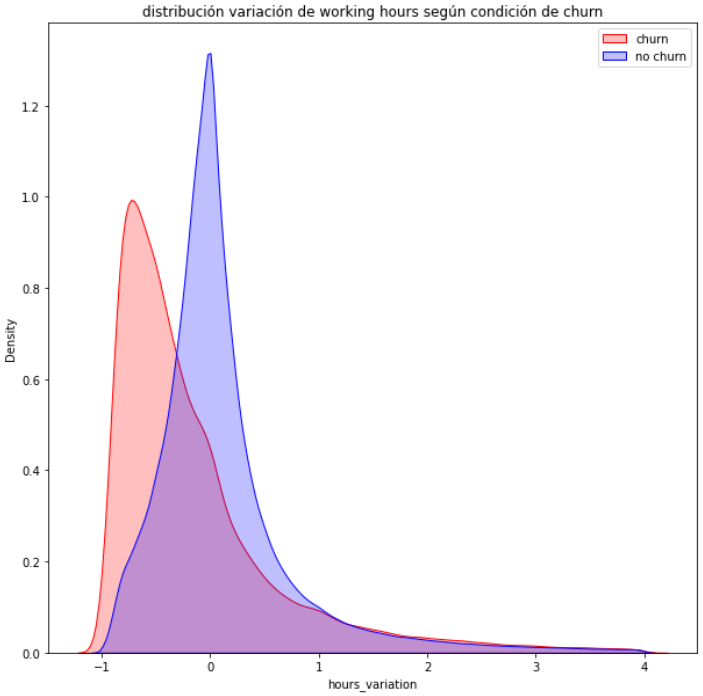
Anexos

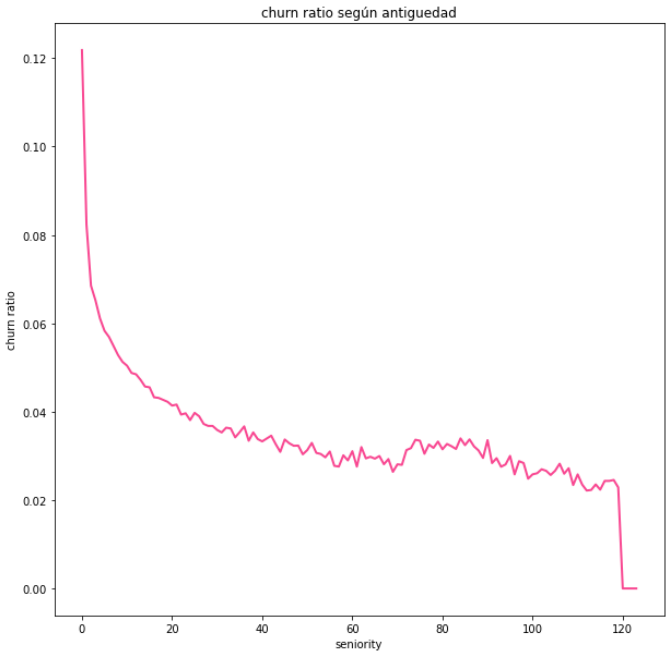
Anexo – Variables modelo survival

```
Data columns (total 11 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   completed_deliveries            108099 non-null  float64
 1   cancelled_deliveries            108099 non-null  float64
 2   avg_distance_pickup_to_vendor_m 108099 non-null  float64
 3   working_hours                   108099 non-null  float64
 4   stacked_orders                  108099 non-null  float64
 5   rider_delivery_time             108099 non-null  float64
 6   late_shift_ratio                108099 non-null  float64
 7   duration                        108099 non-null  int64
 8   event                           108099 non-null  bool
 9   sqr_orders_working_hours        108099 non-null  float64
 10  log_sqr_orders_working_hours    108099 non-null  float64
dtypes: bool(1), float64(9), int64(1)
memory usage: 13.2+ MB
```

Anexo – Gráficos no incorporados al trabajo







Anexo – Librerías Utilizadas

- xgboost
- scikit-survival
- scikit-learn
- lifelines

Anexo – Código Python

Link de acceso al código de Python

Bibliografía

- Hagiu, A. (2007) "Multi-sided platforms: From microfoundations to design and expansion strategies," SSRN Electronic Journal [Preprint]. Available at: https://doi.org/10.2139/ssrn.955584.
- Raykar, V.C., Steck, H. and Krishnapuram, B. (2007) "On Ranking in Survival Analysis: Bounds on the Concordance Index."
- Harrell, F.E. (2015) Regression modeling strategies: With applications to linear models, logistic regression, and survival analysis. Cham: Springer.
- Albanese, N.C. (2022) "How to Evaluate Survival Analysis Models," Towards Data Science, 28 May. Available at: https://towardsdatascience.com/how-to-evaluate-survival-analysis-models-dd67bc10caae.
- Emmert-Streib, F. and Dehmer, M. (2019) "Introduction to survival analysis in practice," Machine Learning and Knowledge Extraction, 1(3), pp. 1013–1038. Available at: https://doi.org/10.3390/make1030058.
- James, G. et al. (2022) An introduction to statistical learning: With applications in R. Boston: Springer.
- Hastie, T., Friedman, J. and Tisxbshirani, R. (2017) The elements of Statistical Learning: Data Mining, Inference, and prediction. New York: Springer.
- Diccionario Cambridge inglés: Significados Y Definiciones (no date) Diccionario Cambridge inglés: Significados y Definiciones. Available at: https://dictionary.cambridge.org/es/diccionario/ingles/ (Accessed: April 6, 2023).
- Gupta, A. (2021) XGBoost hyperparameters-explained, Medium. Medium. Available at: https://amangupta16.medium.com/xgboost-hyperparameters-explained-bb6ce580501d (Accessed: April 6, 2023).
- Zheng, A. and Casari, A. (2018) Feature Engineering for Machine Learning: Principles and techniques for Data scientists. Beijing: O'Reilly.
- Youden, W.J. (1950). Index for rating diagnostic tests. Cancer, 3(1), 32-35.
- Brownlee, J. (2021) *A gentle introduction to threshold-moving for Imbalanced Classification*, *MachineLearningMastery.com*. Available at:

https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/ (Accessed: 13 May 2023).

- Eisenmann, T., Parker, G., & Van Alstyne, M. (2006). Strategies for two-sided markets. Harvard Business Review, 84(10), 92-101.
- Parker, Geoffrey G., Van Alstyne, Marshall W., Choudary, Sangeet Paul (2006). Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You. W. W. Norton & Company, 22-24 .
- Wang, X., Yu, C., & Wei, Y. (2015). "Churn prediction and customer segmentation using clickstream data in the online gaming industry". Journal of Interactive Marketing, 31, 27-45.
- Kumar, V., & Reinartz, W. (2018). Customer relationship management. Springer-Verlag GmbH 2012, 2018.