Master in Management + Analytics

Predicción de baja de usuarios en el sistema público de bicicletas de la ciudad de Buenos Aires

Autoría: Merlini Serasio, Juan Manuel
Año académico: *2023*

UNIVERSIDAD TORCUATO DI TELLA

MASTER IN MANAGEMENT + ANALYTICS

PREDICCIÓN DE BAJA DE USUARIOS EN EL SISTEMA PÚBLICO DE BICICLETAS DE LA CIUDAD DE BUENOS AIRES

TESIS

Juan Manuel Merlini Serasio

04 2023

Tutor: Ignacio Martín Vilieri

Resumen

Este trabajo de fin de posgrado se centra en la realización de un caso de estudio que consiste en la predicción sobre la probabilidad de baja de usuarios en el servicio público de bicicletas de la Ciudad Autónoma de Buenos Aires, mejor conocido como “Ecobicis”.

Para el desarrollo del mismo fueron necesarios conocimientos sobre aprendizaje automático de tipo supervisado y feature engineering para la determinación de dos grupos de usuarios (*churn* y *no churn*). En particular, el estudio basado en clasificación mediante boosting (haciendo foco en modelos basados en XGBoost y CatBoost), la búsqueda de los mejores parámetros basado en técnicas de optimización de los mismos (GridSearch) para su implementación y mejor rendimiento así como también el diseño de variables que nos ayuden a mejorar la predictibilidad y poder obtener mejores insights de los datos.

Abstract

This postgraduate dissertation focuses on the development of a case study that consists of the prediction of the probability of users leaving the Ciudad Autónoma de Buenos Aires's public bicycle service, better known as "Ecobicis".

The development of this case study required knowledge of supervised machine learning and feature engineering for the determination of two groups of users (*churn* and *non-churn*). In particular, the study is based on boosting classification (focusing on models based on XGBoost and CatBoost), the search for the best parameters based on optimization techniques (GridSearch) for their implementation and better performance as well as the design of variables that help us to improve predictability and to obtain better insights from the data.

Índice

Índice de Tablas

Índice de Ilustraciones

1. Introducción

Los servicios públicos de bicicletas ponen a disposición de la población un servicio de transporte de bajo costo (dependiendo del tiempo de uso) y en algunos casos (o momentos) de manera gratuita, en dónde se diferencia de los tradicionales, dando algunas ventajas extra, como pueden ser el hacer ejercicio mientras la persona se moviliza hacia el sitio deseado, evitar el tráfico, reducir los gases de efecto invernadero y es una excelente opción para la mejora del humor y el bienestar en general (basado en el ejercicio).

Vemos que en todo el mundo, este movimiento está cobrando cada vez más relevancia junto con todas las políticas medioambientales que van surgiendo y la inclinación de la población mundial hacia un mundo más libre de emisiones de CO2.

En algunos casos, contamos con algunas ventajas excelentes que nos ahorran tiempo, como puede ser, la disponibilidad de un puerto USB para poder cargar el celular mientras nos desplazamos, generando energía desde el mismo movimiento de la bicicleta, por ejemplo.

Por estas razones detalladas y más, este trabajo apunta a poder retener la mayor cantidad posible de usuarios en el servicio público de bicicletas y poder extenderlo a más personas.

1.1. Contexto

El programa de Ecobicis nació con la visión de transformar la Ciudad de Buenos Aires en una ciudad más verde, inclusiva, creativa e innovadora. Fue el motor principal de acciones estratégicas como el fomento del uso de la bicicleta como medio de transporte.

Antes de lanzar este programa, se llevó adelante un estudio de opinión que dio como resultado que en la Ciudad de Buenos Aires había 1.000.000 de bicicletas en los hogares y que, para la mayoría de los porteños, era un recuerdo de la infancia compartido con seres queridos. Se buscó recuperar ese momento y proyectar una ciudad donde la bicicleta, esencialmente saludable y amigable con el medio ambiente, sea una verdadera alternativa de transporte. (Gobierno de la Ciudad de Buenos Aires, s.f.)

En Argentina, existen varias políticas públicas que han sido implementadas para fomentar la utilización de bicicletas como medio de transporte. Algunas de ellas son:

- Plan Nacional de Transporte Sostenible: Este plan busca desarrollar una movilidad sostenible, inteligente y resiliente para las generaciones futuras, promoviendo el

desarrollo federal. Disminuir las emisiones de Gases de Efecto de Invernadero (GEI) y su efecto local. Mejorar la salud y la calidad de vida por la reducción de la contaminación acústica en áreas urbanas y de internaciones asociadas a enfermedades respiratorias (reducción del gasto sanitario). (Gobierno de la República Argentina, s.f.)

- Ley de Promoción del Uso de la Bicicleta: Esta ley, aprobada en la Ciudad de Buenos Aires en 2010, establece medidas para promover el uso de la bicicleta como medio de transporte en la ciudad. Entre otras cosas, la ley establece la creación de un sistema de bicicletas públicas, la construcción de ciclovías y la promoción de campañas de concientización sobre el uso de la bicicleta. (Resolución 173, Subsecretaría de Transporte, 2010)
- Plan de Movilidad en Bicicleta de la Ciudad de Rosario: La ciudad de Rosario, en la provincia de Santa Fe, ha implementado un plan de movilidad en bicicleta que busca fomentar el uso de este medio de transporte en la ciudad. El plan incluye la construcción de ciclovías, la promoción del uso de bicicletas públicas y la educación sobre el uso seguro de la bicicleta en el tráfico urbano. (Ente de la Movilidad de Rosario, 2018)
- Plan de Movilidad en Bicicleta de la Ciudad de Córdoba: La ciudad de Córdoba, en la provincia de Córdoba, ha implementado un plan de movilidad en bicicleta que incluye la construcción de más de 100 kilómetros de ciclovías y la promoción del uso de bicicletas públicas. También se ha establecido un sistema de incentivos para las empresas que promuevan el uso de la bicicleta entre sus empleados. (Graciela Español, 2014)

Estas son algunas de las políticas públicas que se han implementado en Argentina para fomentar el uso de bicicletas como medio de transporte. Se espera que estas políticas continúen y se expandan en el futuro para mejorar la movilidad urbana y reducir la contaminación ambiental. (Dirección de Investigación Accidentológica & Dirección Nacional de Observatorio Vial, 2021)

1.2. Caso a abordar

En continuación del punto anterior, para apoyar esta iniciativa por parte de la Ciudad de Buenos Aires, nos enfocamos en las nuevas técnicas y tecnologías que tenemos disponibles, como el machine learning y los modelos estadísticos de predicción, para ayudar a lograr una mayor utilización del servicio y también lograr reducir la cantidad de

personas que dejan de hacer uso de este o que no van a hacerlo por un largo período de tiempo.

Es por esto que consideramos de alta relevancia analizar y aportar en la identificación de personas que van a dejar de utilizarlo, como se describió anteriormente, a esto en machine learning lo llamamos ***churn***. El *churn* es el concepto que engloba la baja o terminalidad en el uso de un servicio por parte de un usuario (Kumar, V., & Reinartz, W., 2012). En nuestro caso, identificamos esto como aquellos usuarios que no van a viajar por al menos 60 días consecutivos en el siguiente trimestre al momento de la predicción, esto lo basamos en los tiempos que en promedio toma a una persona la generación de un hábito, que en nuestro caso es el que queremos evitar, el del *sedentarismo* y la costumbre de no usar este medio de transporte.

1.3. Objetivo

El objetivo de esta tesis es el de crear un modelo de machine learning que, mediante el uso de datos públicos del uso del servicio público de bicicletas "Ecobicis", puestos a disposición por el gobierno de la Ciudad de Buenos Aires y datos del tiempo y condiciones meteorológicas, sea capaz de predecir, con un rendimiento aceptable, la baja o la intermitencia por largos períodos de tiempo de los usuarios que utilizan el servicio de transporte de bicicletas públicas.

Habiendo comentado lo anterior, el problema será abordado desde un proceso de clasificación, tabularizando los datos y generando las métricas derivadas necesarias para poder obtener el mejor rendimiento posible con los datos disponibles.

Con esto se busca no sólo dejar disponible el modelo para predicciones, sino también poder aportar en el entendimiento de las variables que más impacto tienen en estos casos y cómo poder generar mejores iniciativas usándolo en nuestro favor, evitando el efecto de caja negra, visto en muchos modelos de machine learning y así facilitar la adopción por parte del negocio.

2. Datos

2.1. Origen de los datos

La obtención de los datos se realizó mediante el portal de datos abiertos de la Ciudad de Buenos Aires, ya que están disponibles para su utilización sin ningún costo y/o licencia por el Gobierno de la Ciudad de Buenos Aires.

Dentro de los datasets disponibles, para este proyecto se decidió utilizar los años 2019 y 2020 de los recorridos, ya que cuentan con información más actual y son los años anteriores al inicio del servicio de tipo pago.

El servicio comenzó a ser arancelado a partir del 13 de marzo de 2021 en días de fin de semana y feriados, continuando su gratuidad en días hábiles, aunque los turistas siempre deberán pagar por la utilización de este servicio. Se excluyen los años posteriores a 2020 ya que al ser pago el servicio es necesario realizar un análisis por separado debido a que el impacto del arancel puede generar un comportamiento diferente a cuando este no estaba instaurado.

Para la base de usuarios sí se utilizó la historia completa disponible, es decir, a partir del año 2015 hasta 2020.

Yendo a los datasets utilizados para agregar información, contamos con información meteorológica, es decir, datos de temperaturas y precipitaciones de la Ciudad de Buenos Aires, acotada a los años de los recorridos (2019 y 2020).

Los links de los datasets comentados anteriormente se detallan a continuación:

- Recorridos bicicletas públicas (Ecobicis) de la Ciudad Autónoma de Buenos Aires – Años 2019 y 2020 (BA Data - Ecobicis, s.f.)
- Base de usuarios de Ecobicis de la Ciudad Autónoma de Buenos Aires – Años 2015, 2016, 2017, 2018, 2019 y 2020 (BA Data - Ecobicis, s.f.)
- Registro de precipitaciones (1991-Actualidad) de la Ciudad Autónoma de Buenos Aires – Años 2019 y 2020 (BA Data - Precipitaciones, s.f.)
- Registro de temperaturas (1991-Actualidad) de la Ciudad Autónoma de Buenos Aires – Años 2019 y 2020 (BA Data - Temperaturas, s.f.)

2.2. Estructura inicial de los datos

Siguiendo lo dicho en el punto anterior, a continuación veremos un pantallazo sobre la estructura inicial de las tablas con las que comenzamos y comentaremos las modificaciones que hemos ido haciendo a estos datos en siguientes puntos.

Se anexan los siguientes archivos a la presente tesis para dar un ejemplo de los datos con los que estaremos trabajando, también se adjunta el set de datos final con el que entrenaremos nuestros modelos, a continuación los nombres y una breve descripción de qué incluyen:

- recorridos_total_head.xlsx (primeras 5 filas del dataset final de recorridos)
- base_usuarios_completa_head.xlsx (primeras 5 filas del dataset final de usuarios)
- Precipitaciones CABA 2019-2020.xlsx (dataset de precipitaciones 2019-2020 CABA)
- Temperaturas CABA 2019-2020.xlsx (dataset de temperaturas 2019-2020 CABA)
- ecobicis_final_dataset_head.xlsx (primeras 5 filas del dataset final con todas las métricas incluidas)

Base de recorridos de Ecobicis, esta tabla cuenta con un total de 15 columnas y 7.223.898 registros, los cuales cada uno representa un recorrido de un usuario. El período incluye registros de los años 2019 y 2020. El detalle de los campos de la misma es el siguiente:

Columna	Tipo de dato	Descripción
periodo	int64	Año del recorrido
usuario_id	int64	Identificador de usuario
duracion_recorrido	int64	Duración del recorrido
fecha_origen_recorrido	datetime64[ns]	Fecha y hora de inicio del recorrido
id_estacion_origen	int64	Identificador de la estación de origen del recorrido
nombre_estacion_origen	string	Nombre de la estación de inicio del recorrido
long_estacion_origen	float64	Longitud de la estación de inicio del recorrido
lat_estacion_origen	float64	Latitud de la estación de inicio del recorrido
direccion_estacion_origen	string	Dirección de la estación de inicio del recorrido
fecha_destino_recorrido	datetime64[ns]	Fecha y hora de fin del recorrido

id_estacion_destino	int64	Identificador de la estación de fin del recorrido
nombre_estacion_destino	string	Nombre de la estación de fin del recorrido
long_estacion_destino	float64	Longitud de la estación de fin del recorrido
lat_estacion_destino	float64	Latitud de la estación de fin del recorrido
direccion_estacion_destino	string	Dirección de la estación de fin del recorrido

Tabla 1. *Estructura inicial de la base de recorridos de Ecobicis*

Base de usuarios de Ecobicis, esta tabla cuenta con un total de 6 columnas y 280.557 registros, los cuales cada uno representa a un usuario, el período incluye registros desde 2015 hasta 2020 inclusive. El detalle de los campos de la misma es el siguiente:

Columna	Tipo de dato	Descripción
usuario_id	int64	Identificador de usuario
usuario_genero	string	Sexo del usuario
usuario_edad	int64	Edad del usuario
fecha_alta	string	Fecha alta usuario
hora_alta	string	Hora alta usuario
fecha_hora_alta	datetime64[ns]	Fecha y hora alta usuario

Tabla 2*. Estructura inicial de la base de usuarios de Ecobicis*

Base de precipitaciones, esta tabla cuenta con un total de 5 columnas y 24 registros, los cuales cada uno representa las mediciones promedio de temperatura de cada mes desde 2019 hasta 2020. El detalle de los campos de la misma es el siguiente:

Columna	Tipo de dato	Descripción
periodo	int64	Año de la medición
mes	int64	Mes de la medición
periodo_mes	string	Año y mes de la medición
mm_lluvia_mes	float64	Cantidad de milímetros de lluvia de la medición
dias_lluvia_mes	int64	Cantidad de días de lluvia registrados en el *periodo_mes*

Tabla 3. *Estructura inicial de la base de precipitaciones de la Ciudad Autónoma de Buenos Aires*

Base de temperaturas, esta tabla cuenta con un total de 6 columnas y 24 registros, los cuales cada uno representa las mediciones promedio de precipitaciones mensuales y días

de lluvia totales de cada mes desde 2019 hasta 2020. El detalle de los campos de la misma es el siguiente:

Columna	Tipo de dato	Descripción
periodo	int64	Año de la medición
mes	int64	Mes de la medición
periodo_mes	string	Año y mes de la medición
temp_media	float64	Temperatura media registrada en el *periodo_mes*
temp_max_media	float64	Temperatura máxima registrada en el *periodo_mes*
temp_min_media	float64	Temperatura mínima registrada en el *periodo_mes*

Tabla 4. *Estructura inicial de la base de temperaturas de la Ciudad Autónoma de Buenos Aires*

2.2.1. Tratamiento de los datos

Habiendo hecho la introducción de la estructura de datos inicial de la que partimos, procedemos a detallar el tratamiento que se les ha realizado a los mismos para poder llegar a la estructura final que utilizamos para el modelo.

Dentro de las modificaciones, adaptaciones y correcciones que se les hicieron a los datos, tenemos las siguientes:

Dentro de la **base de recorridos de las Ecobicis** se procedió al renombramiento de campos para llevarlos a un estándar y tener los mismos encabezados. Sumado a esto, se quitaron las filas con campos nulos y aquellas duplicadas, de haberlas. En caso de los campos nulos, se optó por su remoción porque no representaban un número considerable de casos. También se recalcularon los tiempos de viaje con los datos de partida y llegada, ya que se encontraron datos erróneos o cero en ese campo. Finalmente se procedió a unificar todos los datasets en uno único.

Dentro de la **base de usuarios de Ecobicis**, al igual que en la base de recorridos, se realizó el renombramiento de los campos para estandarizar los encabezados, se quitaron nulos y duplicados. También se realizaron transformaciones al campo de *usuario_genero* para estandarizar los valores de todos los datasets. Siendo que la edad legal para poder obtener una cuenta dentro de Ecobicis es a partir de los 16 años, se quitaron aquellas cuentas en la que los usuarios contaban con una fecha de nacimiento que daba una edad menor a esta y se quitaron los usuarios mayores a 80 años, ya que

en muchos casos puede ser debido a errores o malas cargas de datos, aunque no representaban nada de peso dentro de la cantidad de recorridos totales. Finalmente se procedió a unificar todo en un único dataset.

A diferencia de estos dos set de datos, a las bases de precipitaciones y temperaturas no se le realizaron cambios.

A continuación, seguiremos con el proceso de integración de los datos.

2.3. Integración de los datos en el dataset final

Teniendo lo anterior listo, se procedió al ensamble del dataset final, en donde se cruzaron las bases de recorridos con la base de usuarios, mediante *usuario_id*, luego se agregaron los datos de precipitaciones y temperaturas de la Ciudad Autónoma de Buenos Aires, mediante el campo *periodo_mes*. Los anteriores tipos de cruces se hicieron mediante *inner join* para evitar la generación de registros que contuvieran valores nulos.

Consiguiendo la siguiente estructura final del dataset:

Columna	Tipo de dato	Descripción
periodo	int64	Año del recorrido
usuario_id	int64	Identificador de usuario
duracion_recorrido	int64	Duración del recorrido
fecha_origen_recorrido	datetime64[ns]	Fecha y hora de inicio del recorrido
id_estacion_origen	int64	Identificador de la estación de origen del recorrido
nombre_estacion_origen	string	Nombre de la estación de inicio del recorrido
long_estacion_origen	float64	Longitud de la estación de inicio del recorrido
lat_estacion_origen	float64	Latitud de la estación de inicio del recorrido
direccion_estacion_origen	string	Dirección de la estación de inicio del recorrido
fecha_destino_recorrido	datetime64[ns]	Fecha y hora de fin del recorrido
id_estacion_destino	int64	Identificador de la estación de fin del recorrido
nombre_estacion_destino	string	Nombre de la estación de fin del recorrido
long_estacion_destino	float64	Longitud de la estación de fin del recorrido
lat_estacion_destino	float64	Latitud de la estación de fin del recorrido

direccion_estacion_destino	string	Dirección de la estación de fin del recorrido
usuario_id	int64	Identificador de usuario
usuario_genero	string	Sexo del usuario
usuario_edad	int64	Edad del usuario
fecha_alta	string	Fecha alta usuario
hora_alta	string	Hora alta usuario
fecha_hora_alta	datetime64[ns]	Fecha y hora alta usuario
mm_lluvia_mes	float64	Cantidad de milímetros de lluvia de la medición
dias_lluvia_mes	int64	Cantidad de días de lluvia registrados en el *periodo_mes*
temp_media	float64	Temperatura media registrada en el *periodo_mes*
temp_max_media	float64	Temperatura máxima registrada en el *periodo_mes*
temp_min_media	float64	Temperatura mínima registrada en el *periodo_mes*

Tabla 5. *Estructura del dataset final*

2.4. Feature Engineering

El objetivo de estos modelos es poder generar predicciones precisas sobre nueva información, es decir, que no haya sido vista en los datos de entrenamiento.

Esto se vuelve un poco difícil, pues, no podremos predecir algo de lo que no tenemos información. Es por eso que es muy importante el cómo se genera la división entre los sets de entrenamiento, validación y prueba.

Luego de lo comentado y para evitar la generación de un modelo que dé lugar a falsas predicciones o a predicciones basadas en fuga de información (data leakage), que se da cuando, por ejemplo, la distribución de una variable se calcula sobre el dataset total y después se procede a la división datos de entrenamiento y prueba, es decir, la información que va a cada uno de estos contiene información sobre data que aún no se ha visto, por lo que es esencial primero generar la separación y luego hacer los cálculos sobre cada set por separado.

Para poder dividirlos y entrenar los modelos de una manera óptima, inicialmente se procedió a partir el dataset final en períodos trimestrales para realizar los cálculos de las variables derivadas sin generar data leakage, que los modelos resultantes pudieran ser válidos y así también evitar falsos sobreajustes (overfittings) que serían muy probablemente vistos en casos de leakage. Para hacer un testeo más intenso del modelo y llevándolo un poco más allá para ver si la generalización del mismo es buena, para los sets de entrenamiento (*t*), validación (*t+1*) y prueba (*t+2*) se tomarán los trimestres 1, 2 y 3, respectivamente, es decir, entrenaremos con el trimestre 1 (*t*), luego se hará la validación del entrenamiento en el trimestre 2 (*t+1*) y finalizaremos haciendo una evaluación de la performance y predictibilidad más a largo plazo con el trimestre 3 (*t+2*).

Hecho esto, se continuó con la creación de las variables derivadas para enriquecer la data ya disponible y dar mejores predicciones.

2.4.1. Creación inicial de campos

En esta sección hablaremos sobre los campos que fueron creados y agregados al dataset inicial para sumar a los datos que teníamos y así poder dar lugar a un modelo más preciso.

Para comenzar, se segmentaron los días en feriado, fin de semana y día de semana haciendo uso de la librería de Python *holidays* para identificarlos más fácilmente. Con esta misma información, se generó el campo de día laborable (si un día no es fin de semana y no es feriado, entonces es laborable). [1]

- Campos creados:
    - ***es_dia_semana*** *(1 si es True, 0 si es False)*
    - ***es_fin_semana*** *(1 si es True, 0 si es False)*
    - ***es_dia_laborable*** *(1 si es True, 0 si es False)*

Luego normalizamos los campos *mes_recorrido*, *dia_recorrido*, *periodo_mes_recorrido* agregando un cero por delante, en caso de ser necesario, para mantener el formato MM (ej: 09), DD (ej: 05) y YYYYMM (ej: 201906), respectivamente.

1 https://pypi.org/project/holidays/

Creamos el campo ***mes_dia_recorrido*** como la concatenación de *mes_recorrido* y *dia_recorrido* (ambos ya normalizados en el paso anterior), para poder contar con toda la información en un único lugar.

Generamos el campo ***duracion_media_recorrido_mes*** como la media mensual de los recorridos totales, con el objetivo de contar con información asociada a la totalidad de usuarios, referente a las duraciones de los recorridos que realizaron.

Seguimos con el campo ***duracion_recorrido_sobre_media***, siendo este el ratio de la duración individual de cada recorrido sobre la media de la duración de los recorridos totales del trimestre, para ganar visibilidad de la relación entre las duraciones de los recorridos individuales por sobre los generales:

$$duracion\_recorrido\_sobre\_media = \frac{\text{Duración del recorrido}}{\text{Media de los viajes totales del trimestre}} \quad (1)$$

El campo ***duracion_media_recorrido_mes*** fue calculado como la duración media total de los recorridos de cada mes del trimestre, así logramos contar con parámetros de comparación de los viajes individuales por sobre la media general de duración de los recorridos del mes.

Luego, creamos el campo ***duracion_recorrido_sobre_media_mensual***, calculado como el ratio de la duración individual de cada recorrido sobre la media de la duración de los recorridos totales del mes del recorrido. Usando el indicador comentado en el párrafo anterior, conseguimos obtener la relación entre los viajes individuales y la media general de la duración de los recorridos del mes. La forma de cálculo es la siguiente:

$$duracion\_recorrido\_sobre\_media\_mensual = \frac{\text{Duración del recorrido}}{\text{Media de los viajes totales del mes}} \quad (2)$$

En la siguiente tabla veremos la creación de múltiples métricas basados en cada usuario en particular, es decir, el nivel de detalle de cada campo está calculado con una granularidad a nivel de usuario, los campos son los siguientes:

Nombre Campo	Descripción / Forma de cálculo
total_viajes_mes_usuario	Total de recorridos de un usuario en el mes
total_viajes_feriados_mes_usuario	Total de recorridos en días feriados de un usuario en el mes
total_viajes_fin_semana_mes_usuario	Total de recorridos en días de fin de semana de un usuario en el mes
total_viajes_dia_semana_mes_usuario	Total de recorridos en días de semana de un usuario en el mes
total_viajes_dia_laborable_mes_usuario	Total de recorridos en días laborables de un usuario en el mes
media_total_viajes_usuario	Media total de recorridos de un usuario en el trimestre
viajes_mes_n_trimestre_usuario	Total de recorridos del usuario en el mes n del trimestre. n = 1, 2, 3
viajes_feriados_mes_n_trimestre_usuario	Total de recorridos en días feriados de un usuario en el mes n del trimestre. n = 1, 2, 3
viajes_dia_semana_mes_n_trimestre_usuario	Total de recorridos en días de semana de un usuario en el mes n del trimestre. n = 1, 2, 3
viajes_fin_semana_mes_n_trimestre_usuario	Total de recorridos en días de fin de semana de un usuario en el mes n del trimestre. n = 1, 2, 3
viajes_dia_laborable_mes_n_trimestre_usuario	Total de recorridos en días laborables de un usuario en el mes n del trimestre. n = 1, 2, 3
dif_viajes_mes_1_vs_mes_2	$\text{viajes\_mes\_2\_trimestre\_usuario} - \text{viajes\_mes\_1\_trimestre\_usuario}$
dif_viajes_mes_2_vs_mes_3	$\text{viajes\_mes\_3\_trimestre\_usuario} - \text{viajes\_mes\_2\_trimestre\_usuario}$
dif_viajes_mes_1_vs_mes_3	$\text{viajes\_mes\_3\_trimestre\_usuario} - \text{viajes\_mes\_1\_trimestre\_usuario}$
ratio_viajes_feriados_mes_n	$\frac{\text{viajes\_feriados\_mes\_n\_trimestre\_usuario}}{\text{viajes\_mes\_n\_trimestre\_usuario}}$ n = 1, 2, 3
ratio_viajes_fin_semana_mes_n	$\frac{\text{viajes\_fin\_semana\_mes\_n\_trimestre\_usuario}}{\text{viajes\_mes\_n\_trimestre\_usuario}}$ n = 1, 2, 3
ratio_viajes_dia_semana_mes_n	$\frac{\text{viajes\_dia\_semana\_mes\_n\_trimestre\_usuario}}{\text{viajes\_mes\_n\_trimestre\_usuario}}$

	n = 1, 2, 3
ratio_viajes_dia_laborable_mes_n	$\frac{\text{viajes\_dia\_laborable\_mes\_n\_trimestre\_usuario}}{\text{viajes\_mes\_n\_trimestre\_usuario}}$ n = 1, 2, 3
cantidad_recorridos_sobre_media_mensual_total	$\frac{\text{total\_viajes\_mes\_usuario}}{\text{duracion\_media\_recorrido\_mes}}$
cantidad_recorridos_sobre_media_total_usuario	$\frac{\text{total\_viajes\_mes\_usuario}}{\text{media\_total\_viajes\_usuario}}$

Tabla 6. *Nuevas métricas derivadas a nivel de usuario creadas*

Para intentar inferir la distancia media aproximada del recorrido elegimos contemplar la distancia entre el punto geográfico de partida y el de llegada haciendo uso de las coordenadas de latitud y longitud de las estaciones de bicicletas. Aquí nos resultó interesante y necesario considerar, más allá de la distancia Euclidea entre los puntos (distancia medida mediante una recta entre el punto A y B), la curvatura terrestre. Para tal fin, hicimos uso de la fórmula de cálculo de Haversine, que hace consideración de estas premisas para obtener la distancia entre los puntos de inicio y fin del recorrido. Para dicho cálculo, hicimos uso de la librería de Python *haversine*, que ya implementa las funciones necesarias para tal fin. Así creamos el campo ***distancia_km_aprox_recorrido***, que nos aporta la información asociada a la distancia de los recorridos que realiza el usuario.[2] Consideramos esto importante porque creemos que aquellos usuarios que hacen un uso más intensivo de las bicicletas en cada recorrido, es decir, recorren distancias más largas con respecto a otros usuarios, tengan una probabilidad menor de dejar de usar el servicio, ya que podrían estar optando por utilizar el transporte público, como colectivos o subte, pero a pesar de dichas opciones optan por realizar ejercicio haciendo uso de las Ecobicis de manera deliberada y no por comodidad.

Siguiendo con el estudio de los datos, nos encontramos con que muchos usuarios iniciaban y concluían el viaje en la misma estación, dando como resultado una distancia de cero. Es por esto que decidimos incluir el campo ***inicio_fin_misma_estacion*** (1 para True, 0 para False) para así poder determinar si el usuario inició y finalizó su viaje en la misma estación, basados en la métrica de distancia comentada anteriormente y para no tener como resultado una velocidad de cero en lo que se describe a continuación.

2 https://pypi.org/project/haversine/

Haciendo uso de la métrica de distancia creamos el campo ***velocidad_kmh_media_viaje*** para obtener una velocidad promedio aproximada de desplazamiento desde el inicio hasta el fin del recorrido, excepto que haya sido la misma estación, en donde pierde sentido este campo, ya que la distancia aproximada resultante es de cero (inicio y fin coinciden), resultando en una velocidad de 0 km/h. Con lo anterior ya hecho, hicimos análisis de la distribución de las velocidades, enfocándonos en los extremos y la media, nos encontramos con que teníamos algunos registros que marcaban velocidades superiores a las que un usuario promedio podría alcanzar, considerando también los tipos de bicicletas disponibles, es por esto que decidimos eliminar aquellos registros con velocidades mayores a 32 km/h del dataset, siendo una cantidad ínfima de filas, probablemente debidas a errores de la data o las mediciones que ya se encontraban dentro del dataset al bajarlo.

Por último, se creó ***dias_hasta_prox_viaje*** que determina la cantidad de días entre el viaje actual y el siguiente dentro de los registros. En caso de no existir un registro posterior, este campo se calcula con una fecha fin igual al fin del trimestre seleccionado. Este campo es quizá el más importante de todos los creados, ya que es el que se utilizará para crear el *target*, que es lo que queremos predecir.

2.5. Confección del dataset final

Una vez generadas todas las métricas que se consideraron de importancia para el modelo, nombradas en el punto anterior, se procedió a agregar las mismas reduciendo el dataset a una granularidad de nivel de usuario, es decir, el agrupamiento se realizó por ***usuario_id*** en todos los campos.

Para intentar perder la menor información posible brindada por la totalidad de registros contenidos en el dataset inicial, se incluyeron campos de cada métrica que contemplan características de su distribución.

A continuación, se detallan las métricas que fueron creadas para denotar sus comportamientos.

Referencia:

- *count*: contar
- *sum*: sumatoria
- *mean*: media
- *median*: mediana
- *var*: varianza
- *std*: desvío estándar
- *min*: mínimo
- *max*: máximo
- *first*: primer registro

Columna	Agregaciones realizadas
es_dia_semana	sum, mean, var, std, min, max, median
es_fin_semana	sum, mean, var, std, min, max, median
es_feriado	sum, mean, var, std, min, max, median
es_dia_laborable	sum, mean, var, std, min, max, median
duracion_recorrido	sum, mean, var, std, min, max, median
mm_lluvia_mes	sum, mean, var, std, min, max, median
dias_lluvia_mes	sum, mean, var, std, min, max, median
temp_media_mes	mean, var, std, min, max, median
temp_max_media_mes	mean, var, std, min, max, median
temp_min_media_mes	mean, var, std, min, max, median
usuario_edad	first
usuario_genero	first
duracion_recorrido_sobre_media	sum, mean, var, std, min, max, median
duracion_media_recorrido_mes	sum, mean, var, std, min, max, median
duracion_recorrido_sobre_media_mensual	sum, mean, var, std, min, max, median
total_viajes_mes_usuario	count, mean, var, std, min, max, median
media_viajes_mes	sum, mean, var, std, min, max, median
media_total_viajes_usuario	sum, mean, var, std, min, max, median
cantidad_recorridos_sobre_media_mensual_total	sum, mean, var, std, min, max, median
cantidad_recorridos_sobre_media_total_usuario	sum, mean, var, std, min, max, median
distancia_km_aprox_recorrido	sum, mean, var, std, min, max, median

velocidad_kmh_media_viaje	mean, var, std, min, max, median
viajes_mes_n_trimestre_usuario	first {n = 1, 2, 3}
ratio_viajes_feriados_mes_n	first {n = 1, 2, 3}
ratio_viajes_fin_semana_mes_n	first {n = 1, 2, 3}
ratio_viajes_dia_semana_mes_n	first {n = 1, 2, 3}
ratio_viajes_dia_laborable_mes_n	first {n = 1, 2, 3}
dif_viajes_mes_1_vs_mes_2	first
dif_viajes_mes_2_vs_mes_3	first
dif_viajes_mes_1_vs_mes_3	first
dias_hasta_prox_viaje	mean, var, std, max, median

Tabla 7. *Agregaciones realizadas a ciertas columnas del dataset inicial*

Los campos fueron generados con la siguiente lógica de nombres:

[nombre agregación]_[nombre columna]

Por ejemplo:

Nombre campo: *dias_hasta_prox_viaje*

Agregación a aplicar: *mean*

Nombre campo final: *mean_dias_hasta_prox_viaje*

Después, se procedió a crear las siguientes derivaciones de ***usuario_genero***:

Nombre campo final	Lógica aplicada
first_usuario_genero_hombre	1 SI EL VALOR ES IGUAL A “M”, SINO 0
first_usuario_genero_mujer	1 SI EL VALOR ES IGUAL A “F”, SINO 0
first_usuario_genero_otro	1 SI EL VALOR ES IGUAL A “O”, SINO 0

Tabla 8. *Campos creados basados en los valores de usuario_genero*

Para finalizar, creamos el campo a predecir, el ***target.***

Lógica de creación de ***target*** es, tomando la última fecha de viaje de un usuario en un trimestre dado: 1 SI *max_dias_hasta_prox_viaje* >= 60, SINO 0

Definición de ***target***: todo usuario que dentro del trimestre no haya viajado en al menos 60 días previos al fin del trimestre, es decir, los últimos dos tercios del mismo, se considerará como uno (1), de lo contrario cero (0).

El campo ***target*** identifica si un usuario fue o no considerado como "***churn***" o baja, en el trimestre correspondiente. Luego, esta variable del trimestre actual (*t*) se asignará al trimestre anterior (*t-1*) para poder generar la predicción con información que tenemos (*t-1*) para el trimestre que queremos predecir (*t*).

2.6. Definición general de churn

En este proyecto, vamos a utilizar el enfoque de clasificación en dos clases, es decir, determinar si, en base a cierta información dada, el usuario pertenece al grupo etiquetado con 1 (uno) o al grupo etiquetado con 0 (cero), que en este caso representa si el usuario es una potencial baja o no lo es, respectivamente.

Para estos casos, surge el concepto de *churn*, que determina aquellos enfoques de aplicación de algoritmos de machine learning en donde se intenta determinar si un usuario tiene mayor probabilidad de darse de baja o de seguir utilizando el servicio.

Por lo que, más formalmente, diríamos que ***churn*** se define como ***el porcentaje de clientes o suscriptores que dejan de utilizar los servicios que ofrece una empresa durante un período de tiempo determinado.*** (Kumar, V., & Reinartz, W., 2012)

¿Por qué nos interesa conocer la probabilidad de *churn* de un usuario en particular?

Ilustración 1. *Imagen ilustrativa de churn*

Dentro de los objetivos podemos encontrar:

- Retención de clientes o usuarios
- Identificar de mejor manera a los clientes o usuarios

- Mejorar las ofertas y los productos o servicios que se ofrecen
- Fidelización de clientes o usuarios

Algunos ejemplos de aplicación:

- What Influences Bike-Sharing Churn? A Case Study of Citi Bike: estudio sobre los factores que influyen en la tasa de churn en el sistema de bicicletas públicas de Citi Bike en Nueva York. (Yu, Q., Liu, Y., & Gong, Y., 2019)
- Assessing the Value of Social Networks to Predict Bike Sharing Churn: en esta investigación se evalúan el valor de las redes sociales para predecir la tasa de churn en el sistema de bicicletas públicas. (González, D., Llorca, J., & Coque, J. J., 2018)
- Bike Share Usage Patterns and User Demographics in a Mid-Sized U.S. City: este paper analiza los patrones de uso y las características demográficas de los usuarios del sistema de bicicletas públicas en la ciudad de Eugene, Oregon, en los Estados Unidos. (Bigazzi, A. S., 2017)
- Handling Churn in a DHT: en este paper se analiza el *churn* en sistemas de archivos distribuidos de tipo Peer to Peer (P2P), basados en DHT (Distributed Hash Tables) y se realizan experimentos en una red emulada de este tipo. (Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz, 2004)
- Minimizing Churn in Distributed Systems: en este paper, se analiza el *churn* en sistemas distribuidos, basándose en el análisis de nodos. Se realizan experimentos de implementación de estrategias para lograr mitigar la baja en los nodos. (P. Brighten Godfrey, Scott Shenker, and Ion Stoica)
- Predicting Churn Rate Of The Massively Multiplayer Online Role-Playing Game (MMORPG) Users By Analyzing Playing Behavior: en este paper se realiza un análisis de *churn* de jugadores de juegos masivos multiplayer en linea. Se hace énfasis a la capacidad de los usuarios de interactuar socialmente dentro del juego, lo cual logró ser un factor clave en la identificación de aquellos usuarios que iban a dejar de jugar. (Han-Soon Sin, Woojin Paik, 2019)

2.7. Definición de churn para este proyecto

Habiendo definido el concepto general y considerando el contexto del problema que se aborda en el presente documento, determinamos el *churn* de usuarios de bicicletas públicas "Ecobicis" de la Ciudad de Buenos Aires como todos aquellos usuarios que, en un período

de 3 meses tengan un lapso entre su último recorrido y el fin del período mencionado de 60 días o más.

Ejemplos, consideramos el lapso entre 01-01-2019 y 31-03-2019:

Último viaje usuario	Fecha fin trimestre	Días desde último viaje	Es churn
01-01-2019	31-03-2019	89	1 (Sí)
30-03-2019	31-03-2019	1	0 (No)

Tabla 9. *Ejemplos de churn para este proyecto*

En la tabla anterior se muestran dos casos en los que se determina si la variable target será un 1 o un 0. Como dijimos anteriormente, el target define como 1 a todo aquel usuario que no haya viajado en los últimos 60 días del trimestre (es el caso de la primera fila de la tabla, como 89>60, entonces se considera churn), los demás serán considerados como 0. Vale aclarar que la confección de estos dos grupos (churn y no churn) se construyen sobre el trimestre actual (*t*) para luego ser quitada del mismo y añadidos como target en el trimestre anterior (*t-1*), para poder predecir en base a información actual un comportamiento futuro.

Estos 60 días entre fechas para la determinación del threshold (*churn* o *no churn*) fueron tomados de acuerdo a la representatividad de este período dentro del trimestre y el tiempo promedio para generar un nuevo hábito, que según el artículo de investigación publicado por el University College of London sobre cómo se forman los hábitos, llevado a cabo en Inglaterra, el cual se enmarca dentro de un marco de estudio de economía del comportamiento. Este estudio se realiza por un período de 12 semanas (84 días) con un total de 39 participantes (luego de algunas iteraciones del estudio), en donde los individuos debieron reportar sus hábitos a lo largo de este tiempo, entre estos se engloban aquellos referidos a la ingesta de líquidos, de alimentos y realización de actividad física. La data generada se usó para entrenar una regresión no lineal, la cual muestra que la mediana de tiempo que se necesita para alcanzar el 95% de la asíntota (valor al cual se aproxima para determinar que un hábito fue formado), es de 66 días. (Phillippa Lally, Cornelia H. M. Van Jaarsveld, 2009)

A continuación un ejemplo de lo anteriormente citado, para ilustrar lo que se comenta:

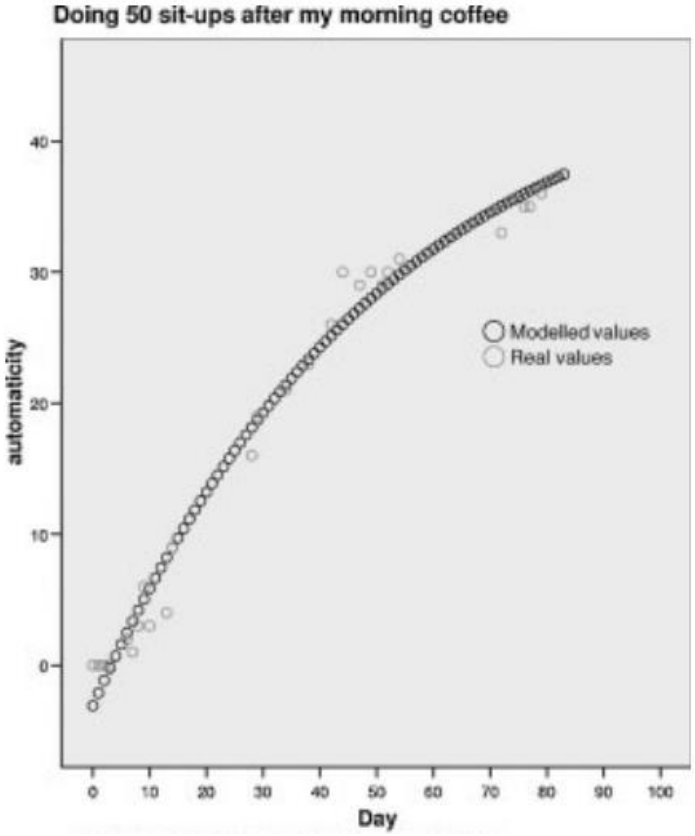


Ilustración 2. Representación de modelo no lineal para alcanzar hábito de ejercicio

De esta manera, este rango temporal nos ayuda a discernir de mejor manera a aquellos usuarios que podrían llegar a dejar de utilizar el servicio o no hacerlo por un largo período de tiempo.

2.8. Análisis exploratorio

En esta sección vamos a centrarnos en el estudio de ciertas variables y supuestos que nos interesa estudiar para probarlos o refutarlos, los cuales pueden ayudarnos luego para determinar nuevas variables que aporten en mejorar las predicciones del modelo acerca de la identificación de aquellas personas con mayor probabilidad de *churn*.

Para comenzar, vamos a ver cómo se comporta la variable target respecto del tiempo y en proporción de *churns* vs *no churns*.

En la *Ilustración* 3 graficamos el comportamiento de la variable que va a ser nuestra variable target, es decir, cuantos usuarios hacen *churn* en el mismo trimestre que estamos

evaluando, el cual luego vamos a desplazar un trimestre hacia atrás, para poder entrenar el modelo para predecir el trimestre siguiente en base al comportamiento del trimestre actual:

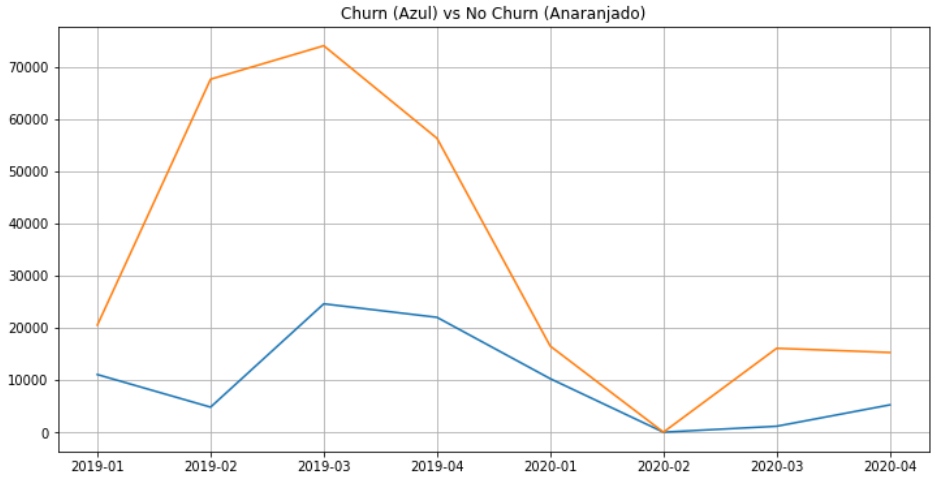


Ilustración 3. *Comportamiento target trimestre actual por usuario*

Para poder ver con mayor claridad la relación entre estas dos curvas graficamos el porcentaje de *churn* en relación a la totalidad de usuarios del trimestre (representados en un rango del eje Y de 0 a 1), es decir, cuántos usuarios hicieron *churn* ese mismo trimestre:

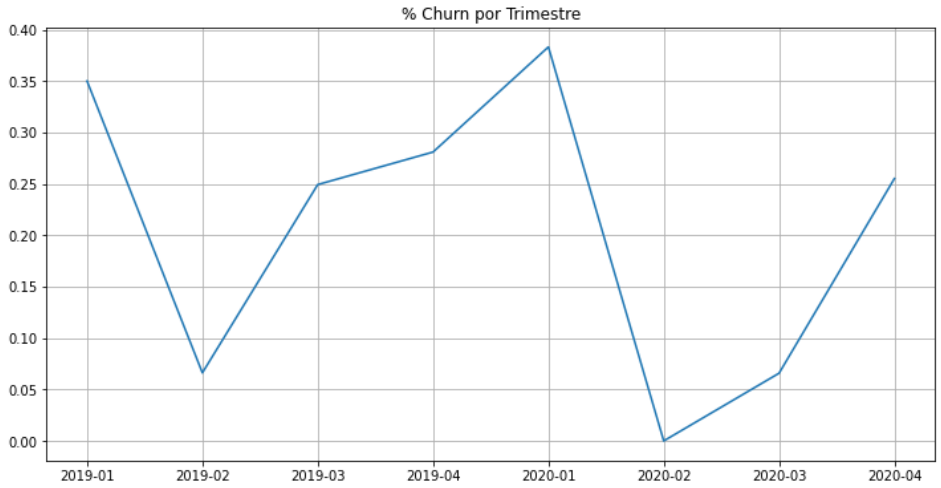


Ilustración 4. *Porcentaje de usuarios que hicieron churn trimestre actual*

Estas imágenes sólo son para poder identificar algún tipo de comportamiento cíclico o estacional sobre nuestra futura variable target, el cual parece evidente al ver la gráfica anterior, pero tenemos que contemplar los períodos en los que por hechos de COVID el servicio fue suspendido, que fue entre marzo 2020 y aproximadamente julio 2020 (segundo trimestre de 2020 y parte del tercero), lo cual puede verse reflejado en las gráficas, ya que el uso del servicio fue cero, por lo que se decidió quitar dicho período (no se utilizará el

período 2020-01 para predecir el 2020-02 ni éste anterior para predecir el 2020-03) del análisis en cuestión, ya que representa un valor atípico para el estudio.

Continuando en esta misma dirección, los gráficos que siguen, muestran estas mismas métricas, sólo que atrasadas un trimestre (para *t-1*), ya listo para que el algoritmo pueda utilizar la data del período actual para predecir el *churn* del siguiente, es decir, por ejemplo, al dataset del primer trimestre de 2019 (2019-01) se le agrega como *target* (0 o 1) a aquellos usuarios que en el trimestre siguiente (2019-02) hayan tenido un período de 60 o más días entre su último viaje y el final de dicho trimestre como 1 y aquellos que no cumplan esa condición como cero. También se contemplarán a aquellos usuarios que tengan actividad en el período actual pero no en el siguiente como *churn*, es decir 1. Gracias a éste acople de la variable a predecir que corresponde al próximo período, podremos hacer uso de la variable ***dias_hasta_prox_viaje*** sin incurrir en un data leakage (sección 2.4).

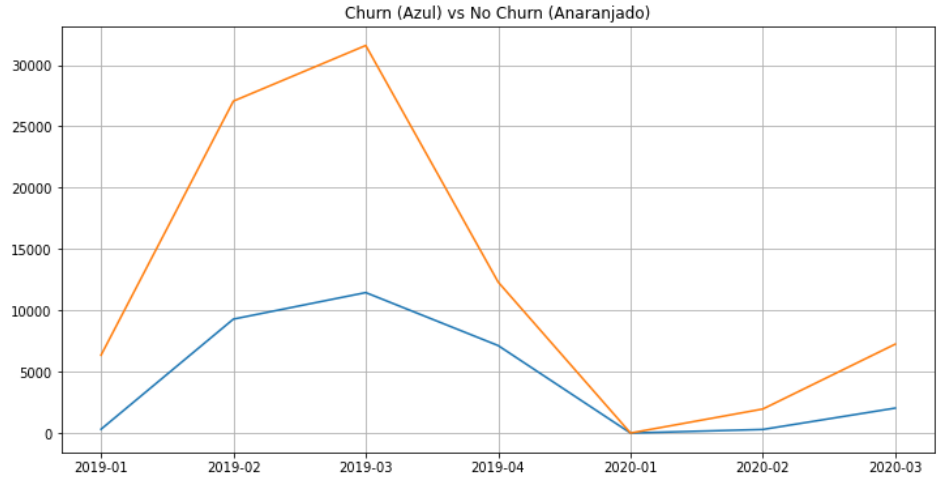


Ilustración 5. *Comportamiento target trimestre siguiente por usuario*

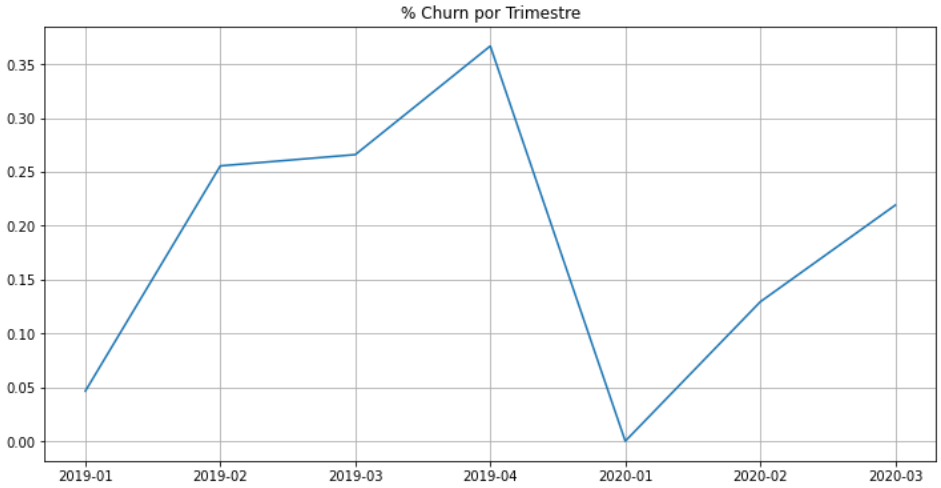


Ilustración 6. *Porcentaje de usuarios que hicieron churn trimestre siguiente*

Dentro de la sección 2.4.1 hicimos la observación, dentro de la descripción de la variable ***distancia_km_aprox_recorrido***, que creíamos que aquellas personas que hacían uso más intensivo de las bicicletas tenían menor probabilidad de *churn* que los demás:

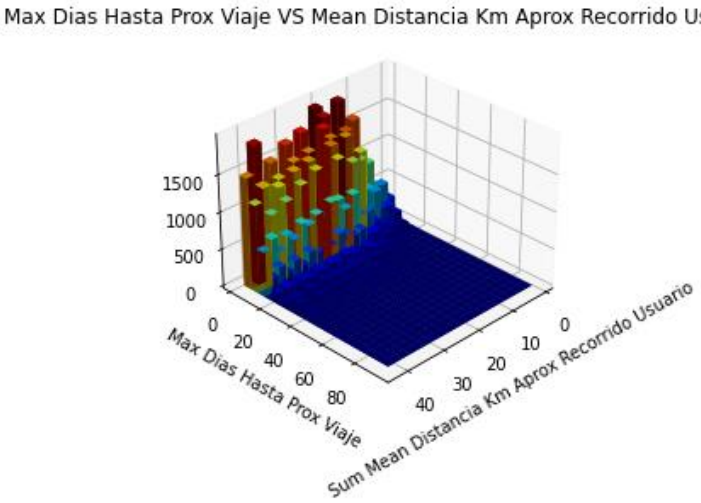


Ilustración 7. *Target VS Distancia Media Total (km)*

Tal cual podemos ver en la *Ilustración 7*, no notamos signos de que esto sea un determinante para ayudar a identificar a aquellos usuarios que van a hacer *churn*.

Otra idea que podemos pensar que puede tener impacto y vale la pena investigar, es ver cuánto incide la edad del usuario en relación de la distribución de máximo de días sin utilizar el servicio, que como veremos a continuación, su distribución es bastante uniforme respecto a la edad y los días máximos hasta su próximo viaje:

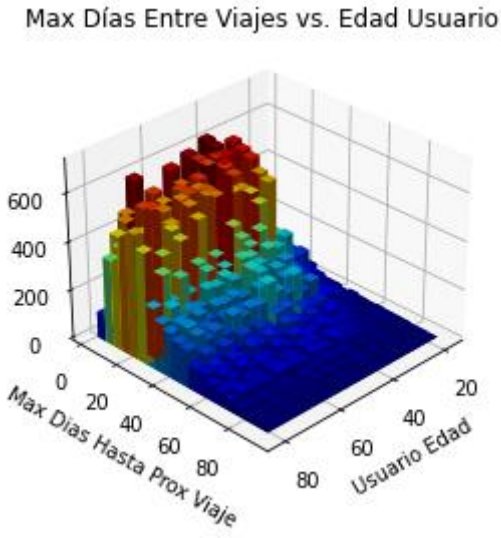


Ilustración 8. *Max Días Hasta Prox Viaje VS Usuario Edad (vista 1)*

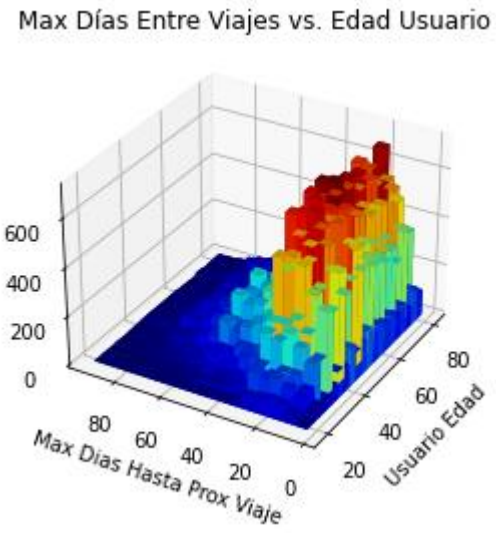


Ilustración 9. *Max Días Hasta Prox Viaje VS Usuario Edad (vista 2)*

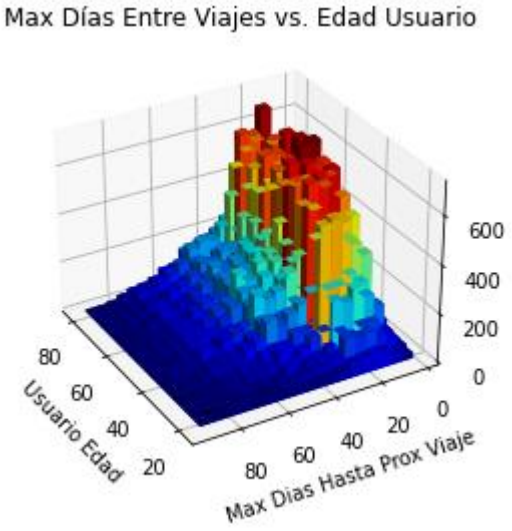


Ilustración 10. *Max Días Hasta Prox Viaje VS Usuario Edad (vista 3)*

Pero, vemos en la *Ilustración 11* que la media de días hasta el próximo uso tiene una distribución proporcional a la edad del usuario (colores modificados en términos de mejorar la visibilidad de los valores bajos):

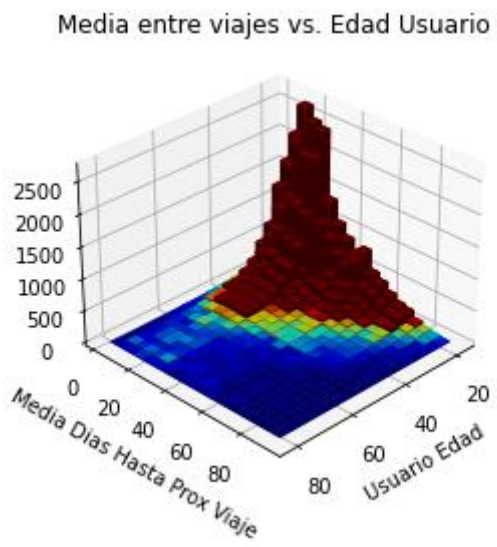


Ilustración 11. *Media entre viajes VS Edad usuario*

Corriéndonos un poco de la variable target, como insights de la búsqueda y estudio realizados, encontramos ciertas variables interesantes y que vale mencionar, acerca de cómo hace uso la gente del servicio de Ecobicis. Para esto, hicimos foco en los diferentes días y calculamos las métricas de uso, en donde discriminamos los días por días de semana, días de fin de semana y feriados, y vimos como se distribuye respecto de la edad de los usuarios:

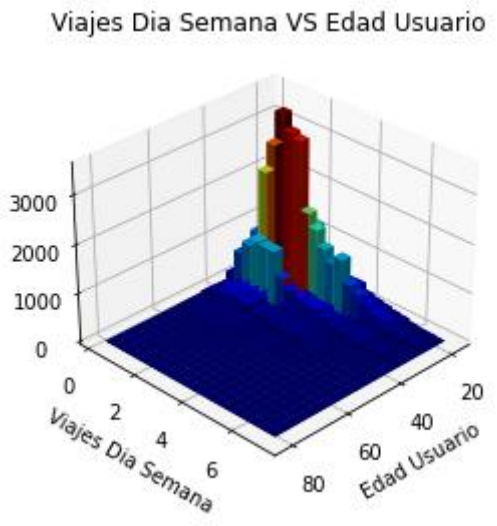


Ilustración 12. *Media viajes día semana VS Edad Usuario*

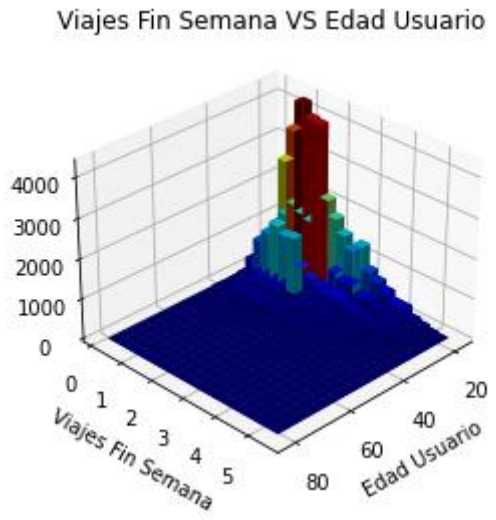


Ilustración 13. *Media viajes fin semana VS Edad usuario*

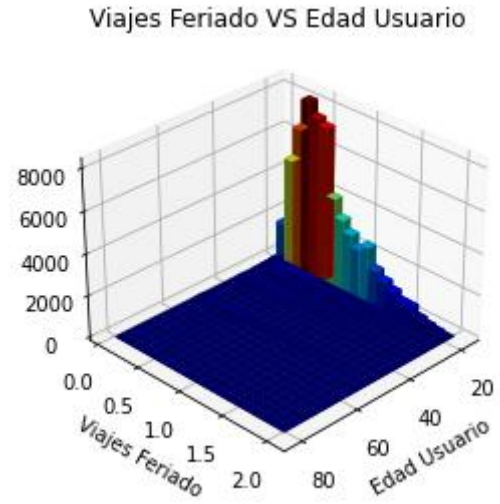


Ilustración 14. *Media viajes feriado VS Edad usuario*

Como vimos, en promedio, los días de fin de semana y feriados es cuando más viajes se realizan, mientras que en los días de semana es cuando menos recorridos se hacen. Respecto del segmento etario correspondiente a cada uno, vemos que se mueve en el rango de entre 18 y 40 años, aproximadamente, cayendo de manera abrupta en edades posteriores. También podemos observar que, en días de fin de semana, la distribución etaria es más amplia que los demás días.

Para continuar con el estudio de algunas variables, parece interesante también poder evaluar cuáles son las estaciones que tienen más viajes entre sí, por lo que se realizó el siguiente mapa de CABA con los puntos del top 20 de estaciones con más recorridos. Primero evaluaremos un ranking de estaciones sin filtro alguno, sólo agrupamientos por

estación de origen y destino y contar los recorridos. Luego, ya con los resultados, vamos a ver si consideramos filtrar de alguna manera la información obtenida.

En la *Ilustración 15* tenemos una captura obtenida mediante la carga de la información de recorridos a la aplicación de Google My Maps: [3]

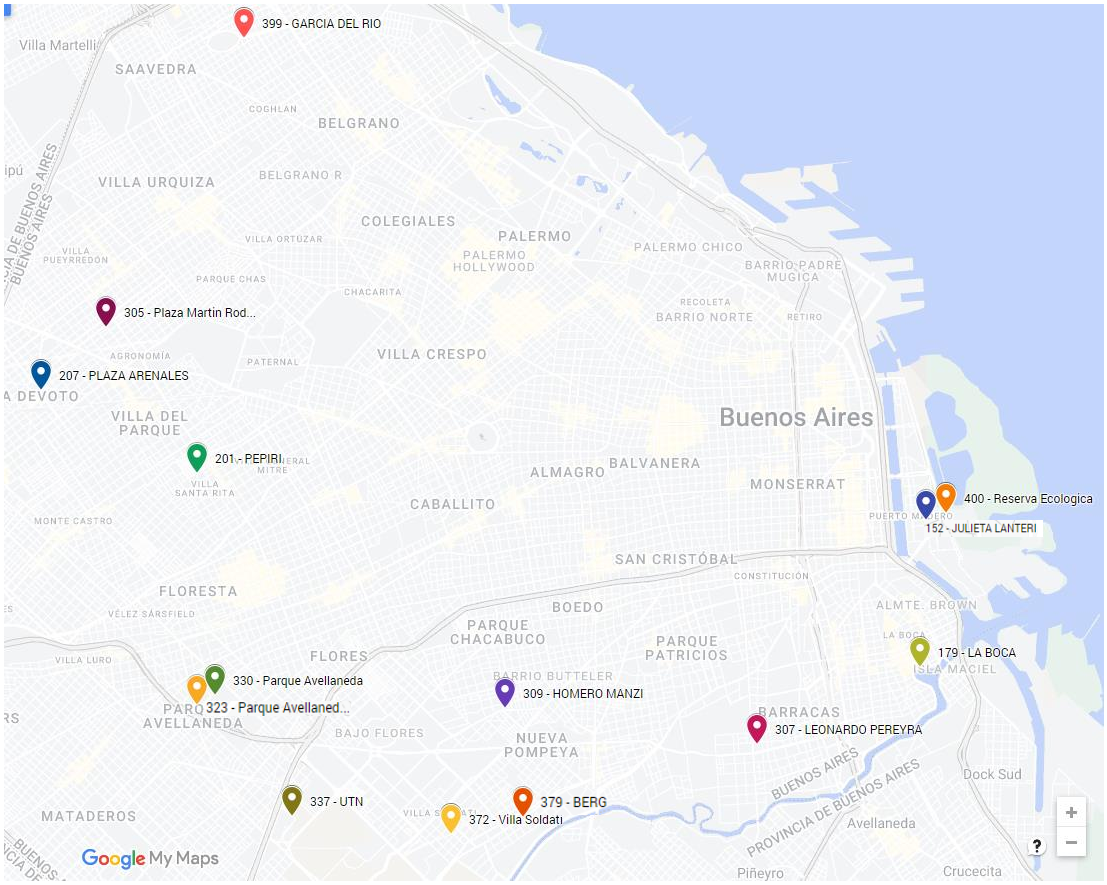


Ilustración 15. *TOP 20 recorridos, agrupados por origen-destino*

Para un poco más de detalle entre estación de origen y destino de los recorridos, se adjunta la *Tabla 10*, conteniendo nombre de estación de origen, nombre de estación de destino y cantidad de viajes totales realizados entre ambas:

3 https://www.google.com/maps/d/?hl=es

	nombre_estacion_origen	nombre_estacion_destino	viajes_totales	es_dia_semana	es_fin_semana	es_dia_laborable	es_feriado
0	337 - UTN	337 - UTN	1625	1	0	1	False
1	372 - Villa Soldati	372 - Villa Soldati	1550	1	0	1	False
2	207 - PLAZA ARENALES	207 - PLAZA ARENALES	1260	1	0	1	False
3	307 - LEONARDO PEREYRA	307 - LEONARDO PEREYRA	1115	1	0	1	False
4	207 - PLAZA ARENALES	207 - PLAZA ARENALES	1091	0	1	0	False
5	379 - BERG	379 - BERG	1052	1	0	1	False
6	399 - GARCIA DEL RIO	399 - GARCIA DEL RIO	941	1	0	1	False
7	337 - UTN	337 - UTN	857	0	1	0	False
8	GARCIA DEL RIO	GARCIA DEL RIO	819	1	0	1	False
9	330 - Parque Avellaneda	330 - Parque Avellaneda	798	1	0	1	False
10	400 - Reserva Ecologica	400 - Reserva Ecologica	749	0	1	0	False
11	179 - LA BOCA	179 - LA BOCA	729	1	0	1	False
12	400 - Reserva Ecologica	400 - Reserva Ecologica	692	1	0	1	False
13	372 - Villa Soldati	372 - Villa Soldati	678	0	1	0	False
14	152 - JULIETA LANTERI	152 - JULIETA LANTERI	652	0	1	0	False
15	305 - Plaza Martin Rodriguez	305 - Plaza Martin Rodriguez	638	1	0	1	False
16	309 - HOMERO MANZI	309 - HOMERO MANZI	636	1	0	1	False
17	201 - PEPIRI	201 - PEPIRI	629	1	0	1	False
18	323 - Parque Avellaneda II	323 - Parque Avellaneda II	615	1	0	1	False
19	Parque Avellaneda	Parque Avellaneda	614	1	0	1	False

***Tabla 10**. TOP 20 recorridos, agrupados por origen-destino*

Como podrán notar, se evalúan demasiados recorridos hechos de una estación hacia la misma, lo cual resulta curioso, por lo que optamos por realizar algunas pruebas filtrando todos aquellos posibles recorridos que hayan sido de una duración demasiado corta, es decir, que el usuario se haya arrepentido de utilizar la bici y la haya devuelto a la estación. Ahora evaluamos tiempos hasta 10 minutos de viaje y no obtuvimos mayores cambios, por lo que podemos considerar dichos datos como verdaderos y no a causa de algún error.

Por lo anterior, hicimos un filtrado para generar un ranking, también del top 20 de estaciones con más cantidad de recorridos, pero la condición fue que no se permitía que la estación de origen y destino fueran la misma. Debajo encontrarán el mapa (*Ilustración 16*) con lo citado y luego en la *Tabla 11* los datos para su evaluación:

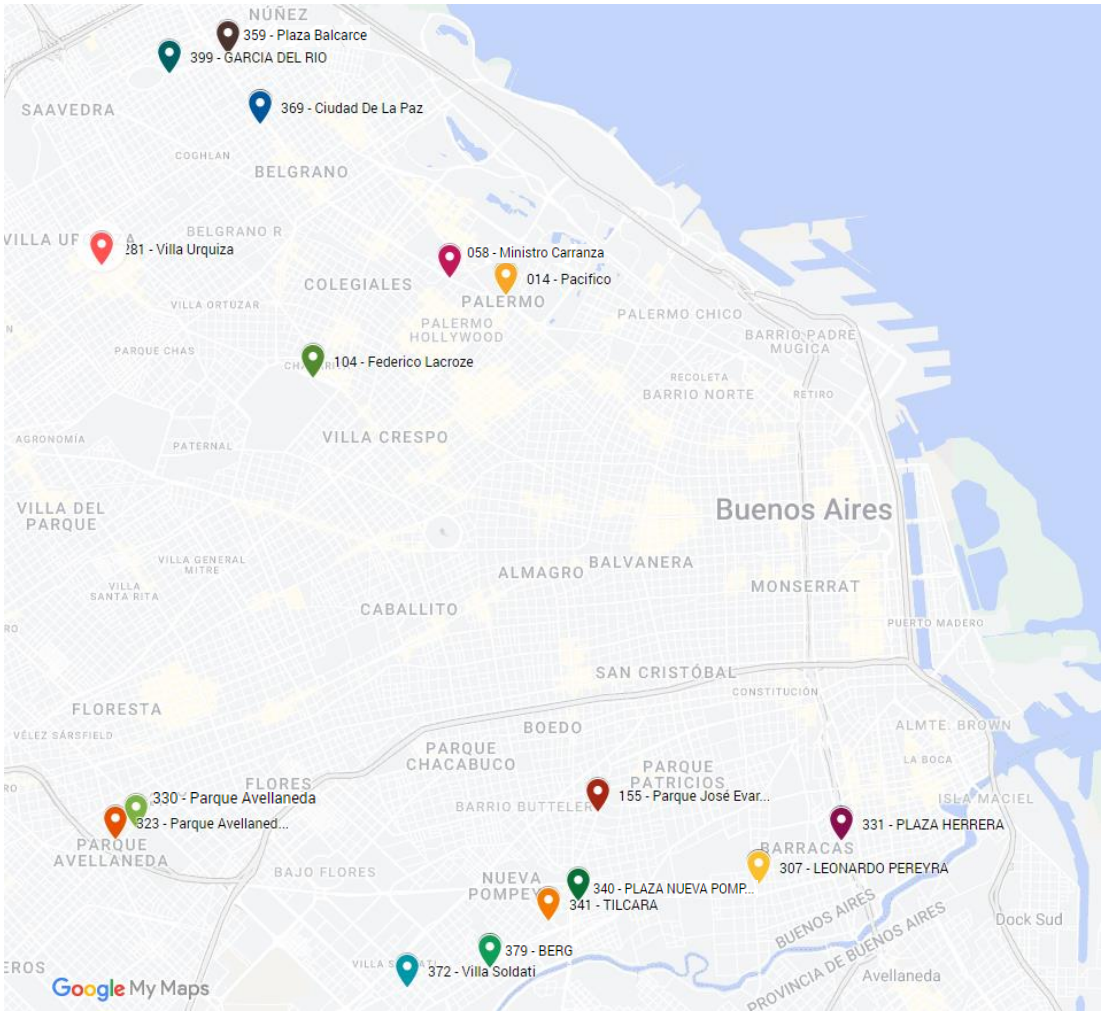


Ilustración 16. *TOP 20 recorridos, agrupados por origen-destino, entre estaciones distintas*

	nombre_estacion_origen	nombre_estacion_destino	viajes_totales	es_dia_semana	es_fin_semana	es_dia_laborable	es_feriado
0	372 - Villa Soldati	379 - BERG	484	1	0	1	False
1	379 - BERG	372 - Villa Soldati	430	1	0	1	False
2	331 - PLAZA HERRERA	307 - LEONARDO PEREYRA	386	1	0	1	False
3	379 - BERG	341 - TILCARA	324	1	0	1	False
4	281 - Villa Urquiza	234 - Valdenegro	317	1	0	1	False
5	307 - LEONARDO PEREYRA	331 - PLAZA HERRERA	309	1	0	1	False
6	323 - Parque Avellaneda II	330 - Parque Avellaneda	306	1	0	1	False
7	369 - Ciudad De La Paz	399 - GARCIA DEL RIO	301	1	0	1	False
8	341 - TILCARA	379 - BERG	271	1	0	1	False
9	340 - PLAZA NUEVA POMPEYA	155 - Parque José Evaristo Uriburu	266	1	0	1	False
10	399 - GARCIA DEL RIO	359 - Plaza Balcarce	259	1	0	1	False
11	359 - Plaza Balcarce	399 - GARCIA DEL RIO	249	1	0	1	False
12	331 - PLAZA HERRERA	366 - Plaza Diaz Velez	249	1	0	1	False
13	155 - Parque José Evaristo Uriburu	340 - PLAZA NUEVA POMPEYA	249	1	0	1	False
14	330 - Parque Avellaneda	323 - Parque Avellaneda II	242	1	0	1	False
15	014 - Pacifico	381 - Matienzo Y Arce	235	1	0	1	False
16	307 - LEONARDO PEREYRA	366 - Plaza Diaz Velez	232	1	0	1	False
17	Ciudad De La Paz	GARCIA DEL RIO	228	1	0	1	False
18	058 - Ministro Carranza	381 - Matienzo Y Arce	224	1	0	1	False
19	104 - Federico Lacroze	208 - Segui	217	1	0	1	False

Tabla 11. *TOP 20 recorridos, agrupados por origen-destino, entre estaciones distintas*

Como vemos, las estaciones con más cantidad de viajes son aquellas que están comprendidas más a las afueras de la capital. Esto podemos asociarlo a la disminución en las opciones de transporte mientras más nos alejamos del centro, es decir, ya empezamos a no contar con subtes, aunque sí con colectivos, pero probablemente es la falta de subtes o altos tiempos de espera de los colectivos lo que probablemente dispara el uso de bicicletas en estas zonas más que en la parte céntrica.

No podíamos dejar fuera de un análisis a las correlaciones entre las variables y el target, por lo que realizamos un screening de las 10 variables que más correlacionan positivamente y aquellas 10 que correlacionan más negativamente, a continuación presentamos un par de imágenes al respecto:

```
dif_viajes_mes_1_vs_mes_2                0.111
dif_viajes_mes_1_vs_mes_3                0.106
inicio_fin_misma_estacion                0.086
duracion_recorrido_sobre_media           0.074
es_fin_semana                            0.070
duracion_recorrido                       0.070
duracion_recorrido_sobre_media_mensual   0.069
dias_lluvia_mes                          0.069
temp_min_media_mes                       0.066
lat_estacion_origen                      0.058
Name: dias_hasta_prox_viaje, dtype: float64
```

Ilustración 17. *10 variables con mayor correlación positiva contra target*

```
viajes_dia_laborable_mes_1_trimestre_usuario    -0.291
viajes_dia_semana_mes_1_trimestre_usuario       -0.292
ratio_viajes_dia_semana_mes_2                   -0.297
ratio_viajes_dia_laborable_mes_2                -0.299
viajes_mes_1_trimestre_usuario                  -0.300
total_viajes_dia_semana_mes_usuario             -0.333
total_viajes_dia_laborable_mes_usuario          -0.333
media_total_viajes_usuario                      -0.337
total_viajes_mes_usuario                        -0.340
cantidad_recorridos_sobre_media_mensual_total   -0.346
Name: dias_hasta_prox_viaje, dtype: float64
```

Ilustración 18. *10 variables con mayor correlación negativa contra target*

En estas últimas dos imágenes, podemos ver que no tenemos variables que tengan un impacto grande en término de correlaciones positivas sobre nuestro *target*, pero esto cambia cuando nos vamos hacia el otro extremo, lo que, cuando vemos cuáles son las que correlacionan más negativamente contra el *target*, nos hace sentido, ya que son aquellas que representan la frecuencia de viajes de los usuarios y, lógicamente, mientras más recorridos tenga realizado un usuario, podemos asumir que la cantidad de días entre viajes será menor y la probabilidad de *churn* disminuirá, ya que contemplamos como *churn* (o baja) a todo usuario que no viaje en al menos 60 días previos al fin del trimestre.

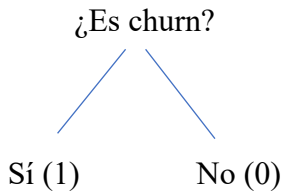
Habiendo realizado este estudio por sobre las variables consideradas como relevantes a criterio, nos volcamos hacia la aplicación de los algoritmos de Machine Learning para ver si ellos encuentran interesantes aquellas variables que pudieron llamar nuestra atención.

3. Metodología

3.1. Algoritmos basados en árboles de decisión

Los árboles de decisión son un tipo de algoritmo de aprendizaje automático supervisado (es decir, se explica cuál es la entrada y cuál es la salida correspondiente en los datos de entrenamiento) en el que los datos se dividen continuamente según un determinado parámetro. El árbol puede explicarse mediante dos entidades, los nodos de decisión y las hojas. Las hojas son las decisiones o los resultados finales. Y los nodos de decisión son donde se dividen los datos. (Witten, I. H., Frank, E., & Hall, M. A., 2016)

En este documento nos centraremos en los Árboles de Clasificación (de tipo Si-No), donde la salida de una variable es alguna de las categorías que la componen, es decir, es de tipo categórica (n cantidad de categorías). (Witten, I. H., Frank, E., & Hall, M. A., 2016) En nuestro caso, la variable es dicotómica, es decir, 2 categorías:



3.2. ¿Por qué árboles de decisión en este proyecto?

Dentro del mundo de árboles de decisión, existen modelos que utilizan a los mismos de una manera iterativa e incremental, de forma de que se arman familias de árboles, en donde la familia siguiente se enfocará en mejorar el error de la variable desarrollada por la familia actual de árboles y así ir reduciendo sistemáticamente el error total en la predicción a medida que se generan iteraciones de familias de árboles. (Witten, I. H., Frank, E., & Hall, M. A., 2016)

Estos modelos descritos, se llaman modelos Boosting (los cuales se desarrollarán en el siguiente punto), los cuales son muy populares por su relativamente sencilla implementación, gran adaptabilidad y buen desempeño en el mundo del machine learning. Estos pueden usarse tanto para variables categóricas como continuas y logran, en general, un excelente desempeño. Dentro de estos, podemos encontrar XGBoost, CatBoost, LightGBM, etc. En esta tesis, nos centraremos en los primeros dos.

Por estas razones comentadas anteriormente y por ser un problema de tipo de clasificación (decidir si un usuario de Ecobicis hará *churn* o no) se decidió utilizar árboles de decisión de tipo clasificación.

3.3. Introducción a modelos de boosting

El método de *boosting* es un algoritmo que se centra en la reducción del sesgo y la varianza en un contexto de **aprendizaje supervisado** (se utiliza data de entrenamiento *x* para poder predecir una variable *y*), basado en la idea de generar un clasificador robusto partiendo de múltiples clasificadores débiles, es decir, un clasificador robusto es aquel que sus clasificaciones se acercan más a las verdaderas clases.

Su funcionamiento se basa, principalmente, en la creación de modelos de clasificación (árboles de decisión de poca profundidad) que suelen ser lentos y simples, generados de manera secuencial y que van siendo ensamblados a medida que se generan, en donde la siguiente generación de predictores se centra en la reducción del error de sus predecesores, y así, poder ir mejorando el modelo total a medida que avanza. (Witten, I. H., Frank, E., & Hall, M. A., 2016)

En este trabajo, nos centraremos en los algoritmos de ***XGBoost*** *(eXtreme Gradient Boosting)* y ***CatBoost*** *(Categorical Boost)*.

3.4. XGBoost

XGBoost es un algoritmo de machine learning de tipo boosting, que mejora sus secuencias de árboles mediante el método de avance por gradiente extremo, es decir, va moviéndose en pos de la mayor pendiente de mejora de la variable seleccionada. A modo de ejemplo podríamos decir que si la variable fuera una montaña, el gradiente extremo nos daría como respuesta la ladera más empinada, que sería la más rápida para descender hasta la base. (Chen, T., & He, T., 2021)

En este caso haremos uso de un algoritmo de *XGBoost* de tipo clasificación para poder predecir de manera precisa a aquellos usuarios con mayor probabilidad de *churn*. Este tipo de modelos nos da la flexibilidad y adaptabilidad necesaria para poder generar las variables derivadas que necesitemos para poder hacer una predicción más precisa. Es de destacar también que este modelo tiene una gran ventaja para modelos de *churn*, ya que cuenta con

un parámetro para trabajar con datasets desbalanceados (la cantidad de 0s y 1s suele ser bastante desigual, generalmente el *churn* ocurre mucho menos que el *no-churn*), que es lo que sucede en general en estos tipos de problemas y lo cuál lo vuelve una mejor opción frente a otros modelos, como regresión logística, lineal, catboost, etc. Otra ventaja que podemos resaltar de este modelo es que tiene un rendimiento muy bueno para problemas de este tipo y a un costo de complejidad no tan alto, lo cual nos da como resultado una explicabilidad del mismo mucho más sencilla de la que podría ser, por ejemplo, en una red neuronal.

3.5. CatBoost

Categorical Boost o *CatBoost* es una librería de tipo open-source, lo cual es beneficioso en términos de no tener que contar con una licencia para su utilización y tampoco tiene un costo asociado.

Este modelo nos provee un framework de gradient boosting, también de tipo supervisado, que logra manejar todas aquellas features o campos que contengan, justamente, valores categóricos, sin la necesidad de transformarlos en numéricos para poder trabajar con los datos, es decir, no necesitamos aplicar one hot encoding para que el modelo pueda funcionar, como sí es necesario en *XGBoost*.

Una excelente feature con la que cuenta este algoritmo es que puede ser entrenado sobre múltiples GPU, esto hace que el rendimiento sea máximo, porque contamos con mucha más paralelización que por CPU, por ejemplo, en uno muy bueno, podríamos contar con unos 10 o quizás 20 hilos de ejecución en paralelo, mientras que con una GPU baja o mediana, podríamos contar con cientos o miles de hilos de ejecución, lo que le da una ventaja enorme por sobre los demás algoritmos que sólo corren sobre CPU, y también nos da la facilidad de implementar la ejecución sobre GPU sólo con indicarle un parámetro, sin tener que configurar cosas complejas en nuestra computadora u otro entorno en donde estemos ejecutándolo.

Otra de sus ventajas es que nos provee excelentes resultados trabajando sólo con parámetros por *Default*, por lo que nos ahorra bastante tiempo que destinaríamos al parameter tunning. (Dorogush, A. V., Ershov, V., & Gulin, A., 2020)

3.6. División del dataset en train y test

Una de las principales razones para prestar suma atención a este proceso es el *Data Leakage*, es decir, la filtración de datos, ya comentado en forma general en secciones anteriores.

El *Data Leakage* consiste en que, cuando estamos calculando campos agregados, dejemos pasar cierta información de la distribución de los mismos al dataset y luego dividamos en train y test. En este caso, dentro del set de train, tendríamos información relativa a data que todavía no vimos, la cual corresponde a test. Todo esto se traduce en un overtfitting o sobreajuste, porque sabemos, de alguna manera, qué datos van a venir. (Kelleher, J. D., & Tierney, B., 2018)

Para evitar esta filtración de información, lo que se procedió a realizar fue una división inicial del dataset inicial en datasets trimestrales y luego se realizó el cálculo de las métricas derivadas. Entonces, así, nos aseguramos que no haya disponible data que no podamos ver dentro de un cierto período de tiempo.

Por ejemplo:

> Si tomamos el período 01/01/2019 – 01/04/2019 (para realizar el entrenamiento del modelo), calculamos las métricas correspondientes a dichos meses y luego realizamos el test con el período 01/04/2019 – 01/07/2019, no tenemos forma de haber podido acceder a información "oculta" en las métricas calculadas desde nuestro dataset de train, ya que fueron divididos antes de realizar los cálculos.

3.7. Método de evaluación de modelos

En esta sección nos vamos a enfocar en las métricas utilizadas en el desarrollo del modelo para poder medir su desempeño y así poder establecer un estándar de comparación entre modelos, en el contexto de este problema.

Dentro de la vasta cantidad de maneras disponibles para la evaluación de modelos de machine learning, nos inclinamos por el uso de curva ROC, Accuracy, Precision, Recall (o sensibilidad) y F-Beta Score. Algunas de estas sirven como base para generar otras métricas derivadas, como por ejemplo, Precision y Recall son utilizadas para el cálculo de F-Beta Score, lo cual nos da un punto de vista diferente basado en cosas que ya conocemos, esto

puede ser clave a la hora de tener una forma diferente de mirar la misma información. (Kelleher, J. D., & Tierney, B., 2018)

Antes de continuar con la definición de cada uno de estas formas de medición, vamos a introducir el concepto de *Matriz de Confusión*, que es una matriz que nos permite visualizar las clasificaciones realizadas por nuestro modelo, éstas pueden ser de cuatro tipos diferentes, ***True Positives*** (**TP**), ***False Positives*** (**FP**), ***True Negatives*** (**TN**) y ***False Negatives*** (**FN**). (Kelleher, J. D., & Tierney, B., 2018)

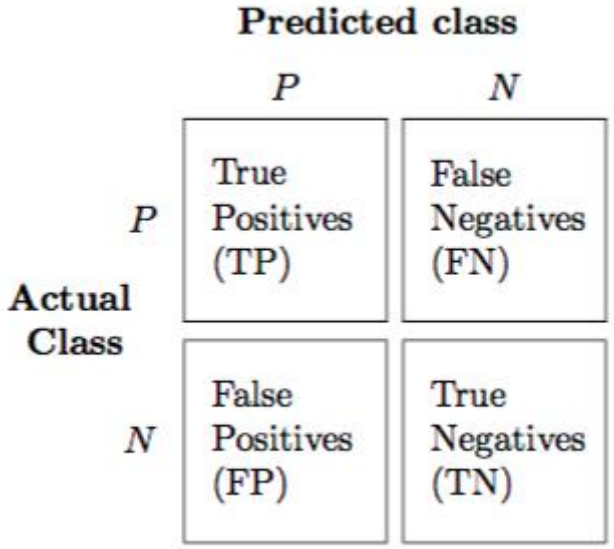


***Ilustración 19**. Matriz de confusión*

- True Positive (TP): fue clasificado como *churn* y fue *churn*.
- False Positive (FP): fue clasificado como *churn* y no fue *churn*.
- True Negative (TN): fue clasificado como no *churn* y fue no *churn*.
- False Negative (FN): fue clasificado como no *churn* y fue *churn*.

Estas cuatro formas de clasificación dan lugar a varias de las métricas que van a definirse a continuación.

La **Curva ROC** es la representación de la razón o proporción de verdaderos positivos (usuarios que se predijeron como *churn* e hicieron *churn*) frente a la razón o proporción de falsos positivos (usuarios que se predijeron como *churn* y no hicieron *churn*). (Kelleher, J. D., & Tierney, B., 2018)

El **Accuracy** mide la frecuencia con la que el clasificador hace la predicción correcta. Es la relación entre el número de predicciones correctas y el número total de predicciones

(cuantas predicciones acerté del total de predicciones hechas, tanto para *churn* como para no *churn*). (Kelleher, J. D., & Tierney, B., 2018)

Su fórmula de cálculo es la siguiente:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

La **Precision** nos dice qué proporción de eventos clasificamos como una cierta clase, en realidad era esa clase. Es una proporción de verdaderos positivos a todos los positivos. Es decir, nos da la cantidad de *churn* bien predichos, cuantos de todos los que dijimos que iban a hacer *churn* realmente hicieron *churn*. (Kelleher, J. D., & Tierney, B., 2018)

Su fórmula de cálculo es la siguiente:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

El **Recall** (o Sensibilidad) nos da la proporción de eventos que en realidad eran de cierta clase fueron clasificados por nosotros como esa clase. Es una proporción de verdaderos positivos a todos los positivos. Es decir, nos da visibilidad de la cantidad de churn bien predichos por sobre el total de *churns* que tenemos identificados. (Kelleher, J. D., & Tierney, B., 2018)

Su fórmula de cálculo es la siguiente:

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Para problemas de clasificación que están sesgados en sus distribuciones de clasificación, como en nuestro caso, el *Accuracy* por sí misma no es una métrica apropiada. En cambio, la *Precision* y el *Recall* son mucho más representativos.

Estas dos métricas pueden combinarse para obtener el F-Beta Score.

El **F-Beta Score** es la media ponderada (media armónica) de las puntuaciones de *Precision* y *Recall*. Esta puntuación es un número real que puede variar de 0 a 1, siendo 1 la mejor puntuación posible en F-Beta Score (tomamos la media armónica ya que estamos tratando con ratios). En nuestro caso tomamos $\beta = 1$, ya que estamos dando tanto a *Presicion* como a *Recall* el mismo peso. (Kelleher, J. D., & Tierney, B., 2018)

Esta medida nos dice qué tan bien nuestro modelo predice en términos de relación entre Precision y Recall.

La fórmula de cálculo es la siguiente:

$$F_\beta = \frac{(1+\beta^2)\cdot(Precision + Recall)}{(\beta^2 \cdot Precision + Recall)} \quad (8)$$

En el siguiente capítulo, veremos las performance y resultados de los modelos con las métricas recién descritas y haremos un análisis de los valores obtenidos.

4. Resultados

En esta sección, evaluaremos la performance de los modelos, procederemos con la importancia de las variables y realizaremos una breve introducción del método "*Feature Permutation*" para el análisis de las variables que son realmente importantes para el modelo, dejando de lado correlaciones entre variables y explicaremos la aplicación en este caso de estudio.

4.1. Performance de modelos

Para la evaluación de los dos modelos aplicados, *CatBoost* y *XGBoost*, utilizamos múltiples métricas para medir la performance de los mismos, ya comentadas en el punto 3.7, estas fueron elegidas por el tipo de problema que estamos encarando, es decir, en este caso, nos interesa más prevenir una baja que predecir mal una (alguien que predecimos como baja cuando no lo es, o sea, un falso positivo o error de tipo 1). Además, se hizo aplicación del método de optimización de parámetros *Grid Search*, para evaluar mejoras de performance mediante la combinación de múltiples valores de los mismos.

Para dar un poco más de contexto, *Grid Search* es una técnica para encontrar la combinación óptima de hiperparámetros de un modelo de aprendizaje automático. Se basa en la especificación de un conjunto de valores posibles para cada hiperparámetro que se desee ajustar, y luego se entrena y evalúa el modelo en cada combinación de valores posibles. Esto se realiza en una cuadrícula de posibles combinaciones de hiperparámetros, de ahí el nombre de la técnica. Una vez que se ha evaluado el modelo en cada combinación de valores posibles, se selecciona la combinación de hiperparámetros que ha producido el

mejor rendimiento en una métrica de evaluación determinada, como la precisión o el F1-score. Esta combinación de hiperparámetros se utiliza para el modelo final.

Es importante tener en cuenta que *Grid Search* puede ser un proceso computacionalmente costoso, ya que puede implicar entrenar y evaluar un gran número de modelos. (Kelleher, J. D., & Tierney, B., 2018)

Nuestra baseline a vencer por los algoritmos:

Validación	Test
0.626436	0.679725

Tabla 12. *Baseline a vencer por los algoritmos*

La *Baseline* nos indica la precisión (o *Accuracy*) que nos daría un modelo si en todas sus predicciones clasificaran a cada caso como 1 (*churn*), es decir, es nuestro modelo básico a vencer. Esto mismo es, la distribución de *churn* y *no-churn* de nuestra variable *target* en el dataset.

A continuación se presenta en la *Tabla 13* una comparativa de los modelos ejecutados en los datasets de train y test, separados por parámetros *Default* y *Grid Search*:

Modelo	**CatBoost**				**XGBoost**			
Conjunto	**Validación**		**Test**		**Validación**		**Test**	
Parámetros	**Default**	**Grid Search**	**Default**	**Grid Search**	**Default**	**Grid Search**	**Default**	**Grid Search**
ROC Score	0.7303	0.7300	0.7125	0.7100	0.7307	0.7289	0.7033	0.7057
Accuracy Score	0.7523	0.7518	0.7591	0.7575	0.7528	0.7517	0.7562	0.7620
F1 Score	0.8038	0.8032	0.8253	0.8244	0.8042	0.8037	0.8251	0.8306
Recall Score	0.8284	0.8273	0.8503	0.8505	0.8293	0.8305	0.8596	0.8722

Tabla 13. *Rendimientos modelos: CatBoost vs. XGBoost*

Como se puede apreciar, tal cual se comentó en la sección 3.5, *CatBoost* tiene un excelente desempeño con parámetros *Default*, e incluso mejor, en este caso, que mediante la optimización de parámetros, aunque en *XGBoost* vemos una diferencia dentro del set de validación y de test, en dónde vemos que performa de mejor manera con los valores obtenidos por *Grid Search* en el segundo, pero no así en el primero, el cual tiene valores mayores.

Analizando estos resultados que comentamos, no nos es difícil volcarnos hacia el modelo de *XGBoost*, el cual nos entrega un mejor desempeño en general, tanto en el dataset de validación como en el de test. Si bien las diferencias no son grandes (exceptuando el Recall), una vez que llegamos a esta altura de predicción, esas ínfimas diferencias suelen

ser muy difíciles de alcanzar. Por otra parte, enfocándonos en el indicador más relevante para este problema en particular, el ***Recall***, que nos dice cuántas de las bajas fueron correctamente identificadas (cuántos usuarios que iban a darse de baja o no utilizar el servicio por al menos 60 días en el próximo trimestre fueron bien clasificados), aquí contamos con un excelente puntaje de 0,8722 con *XGBoost*, es decir, casi 9 de cada 10 usuarios que van a hacer *churn* fueron identificados.

Modelo	**CatBoost**	**XGBoost**
Conjunto	**Test**	**Test**
Parámetros	**Default**	**Grid Search**
ROC Score	0.7125	0.7057
Accuracy Score	0.7591	0.7620
F1 Score	0.8253	0.8306
Recall Score	0.8503	0.8722

Tabla 14. *Comparativa rendimientos modelos en test: CatBoost vs. XGBoost*

En el siguiente punto nos centraremos en entender las variables en las que se basa el algoritmo para los valores que predice.

4.2. Importancia de variables

La explicabilidad de las variables es aquello que nos hace entender qué hace que un algoritmo de machine learning funcione tal cual lo hace, es romper con ese modelo de caja negra y meternos dentro de la formación de su "criterio".

Vamos a presentar dos casos para cada uno de los modelos, la importancia de variables con parámetros por *Default* así como también con parámetros obtenidos mediante el método de búsqueda y combinación *Grid Search*.

Comenzaremos con *CatBoost* con parámetros por *Default*. Veremos cuáles variables fueron las principales para generar las predicciones.

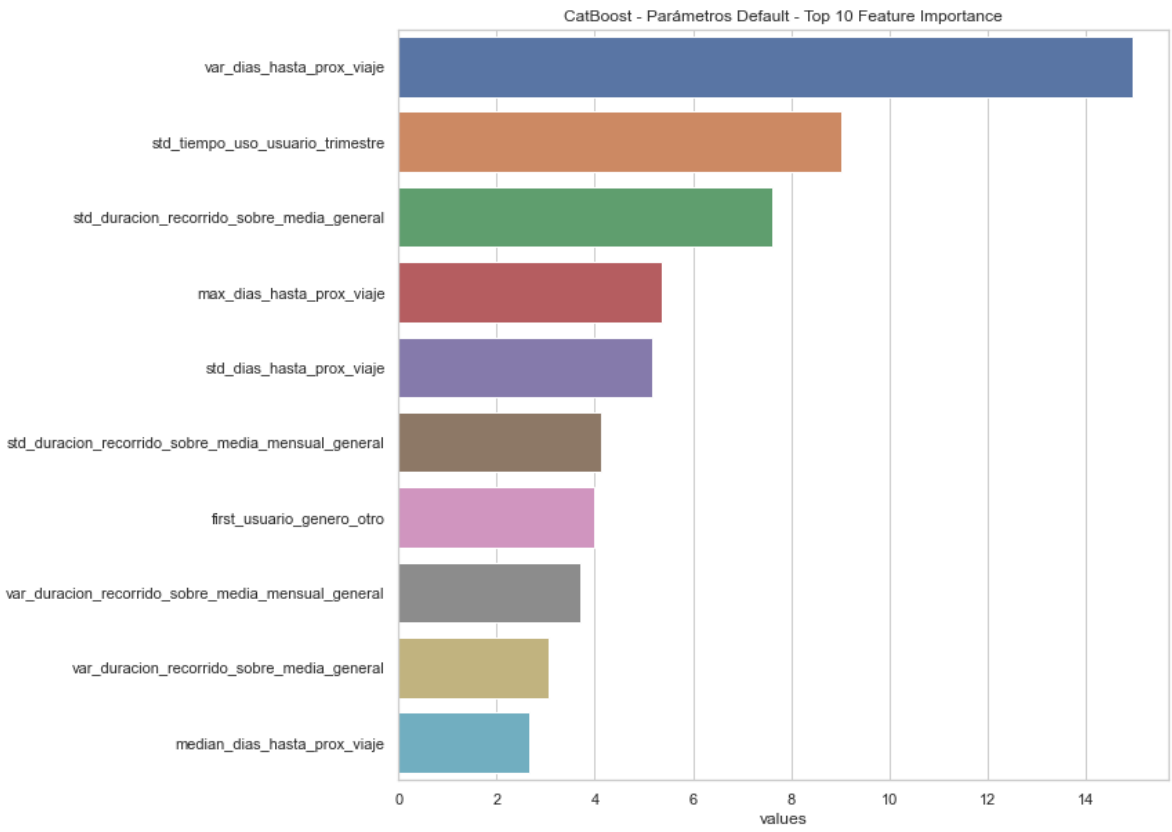


Ilustración 20. *Top 10 importancia de variables CatBoost parámetros Default*

En la ilustración anterior se ha hecho un ranking del top 10 de variables ordenadas por importancia. Aquí podemos ver que la que más destaca está relacionada al *target* (días hasta el próximo viaje), aunque no cuentan con información del trimestre siguiente (por lo comentado en la sección 2.4 sobre la separación de los datasets para evitar data leakage), pero parece que sí tiene algún tipo de correlación respecto del comportamiento del usuario, también vemos que otras de las variables son en su mayoría respecto al uso de las bicis por parte de los usuarios respecto del total, aquellas relacionadas a valores respecto de la media general, esto nos ayuda en la segmentación para poder discriminar mejor entre los segmentos de *churn* y *no churn*. Algo curioso que podemos ver, es que, los usuarios con género *otro* es una variable con peso a la hora de segmentarlos.

A continuación, seguiremos en la evaluación de las variables también para *CatBoost*, pero habiendo utilizado la optimización de parámetros *Grid Search*, veremos si hubo algún cambio respecto de las anteriores.

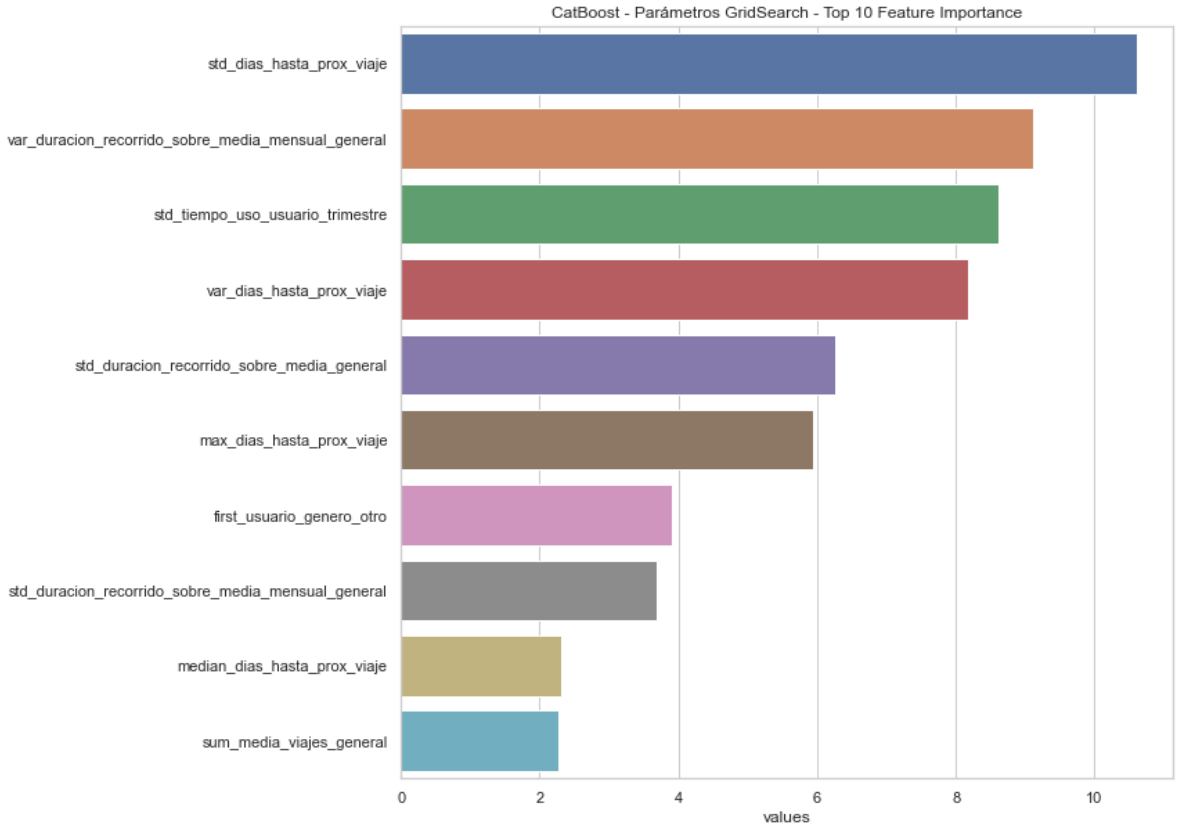


Ilustración 21. *Top 10 importancia de variables CatBoost parámetros Grid Search*

Como podemos ver, no se notan cambios significativos por sobre las variables respecto del modelo con parámetros por *Default*, esto también podemos notarlo o "preverlo", quizá, cuando vemos la diferencia en los rendimientos de ambos modelos (Tabla 13). Los cambios visibles son el ordenamiento en términos de importancia de las variables y el cambio de la variable *std_dias_hasta_prox_viaje* por *sum_media_viajes_general*.

Ahora vamos a ver el desempeño de ambos modelos de *XGBoost*, también, con parámetros por *Default* y por *Grid Search*.

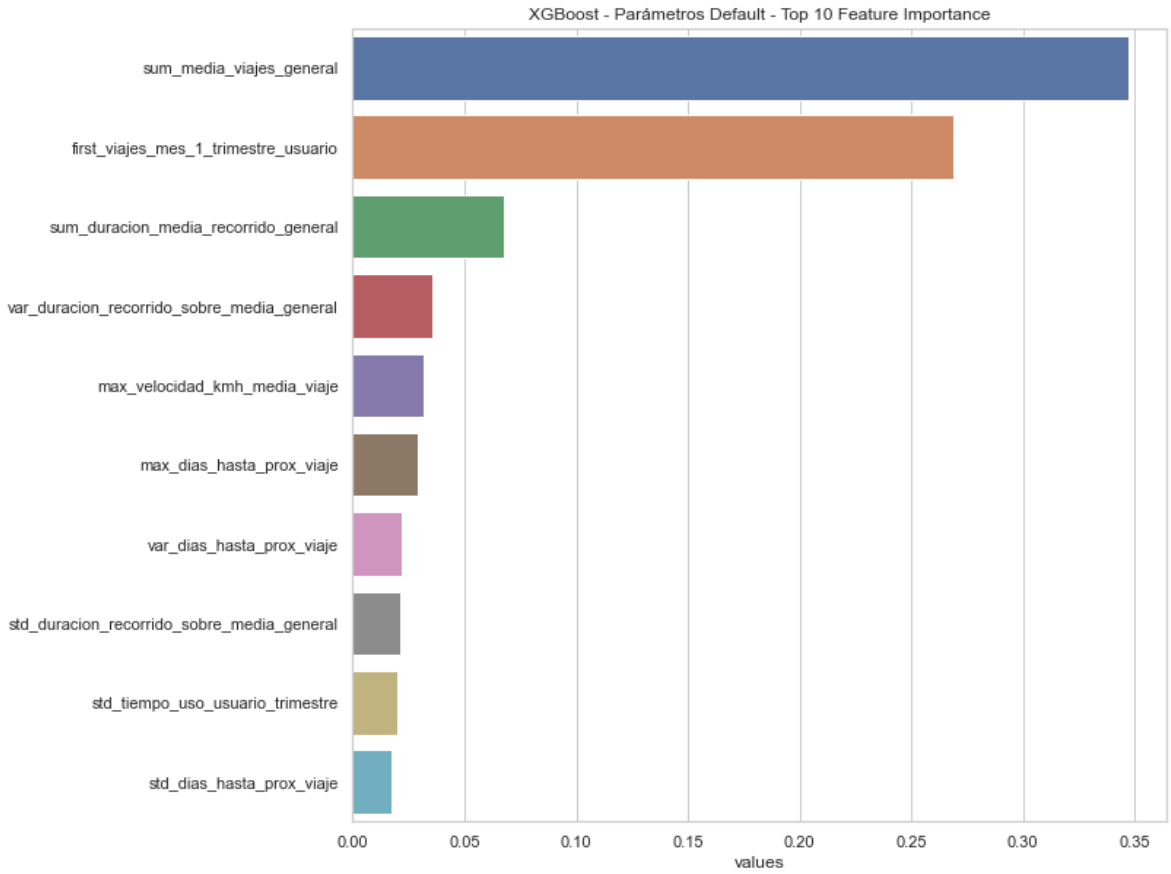


Ilustración 22. *Top 10 importancia de variables XGBoost parámetros Default*

La imagen anterior nos muestra un panorama muy diferente si lo comparamos contra ambos de los modelos de *CatBoost*, vemos una clara diferencia en términos de importancia, que en los anteriores no era tan abrupta. Aquí tenemos dos claros predictores por preferencia, la sumatoria de las medias generales de viajes de los usuarios y la cantidad de viajes realizados por el usuario en el primer mes del trimestre en cuestión. Luego, le siguen varias de las cuales podemos encontrar en los modelos anteriores, aquellas relacionadas a los valores respecto de la media general, las relacionadas al *target*, tiempo de uso del servicio en el trimestre, etc. Aunque, cabe destacar que ya no contamos con la variable género *otro* dentro de la gráfica.

Ahora veremos si al aplicar *Grid Search* cambian alguna de las importancias aquí descritas.

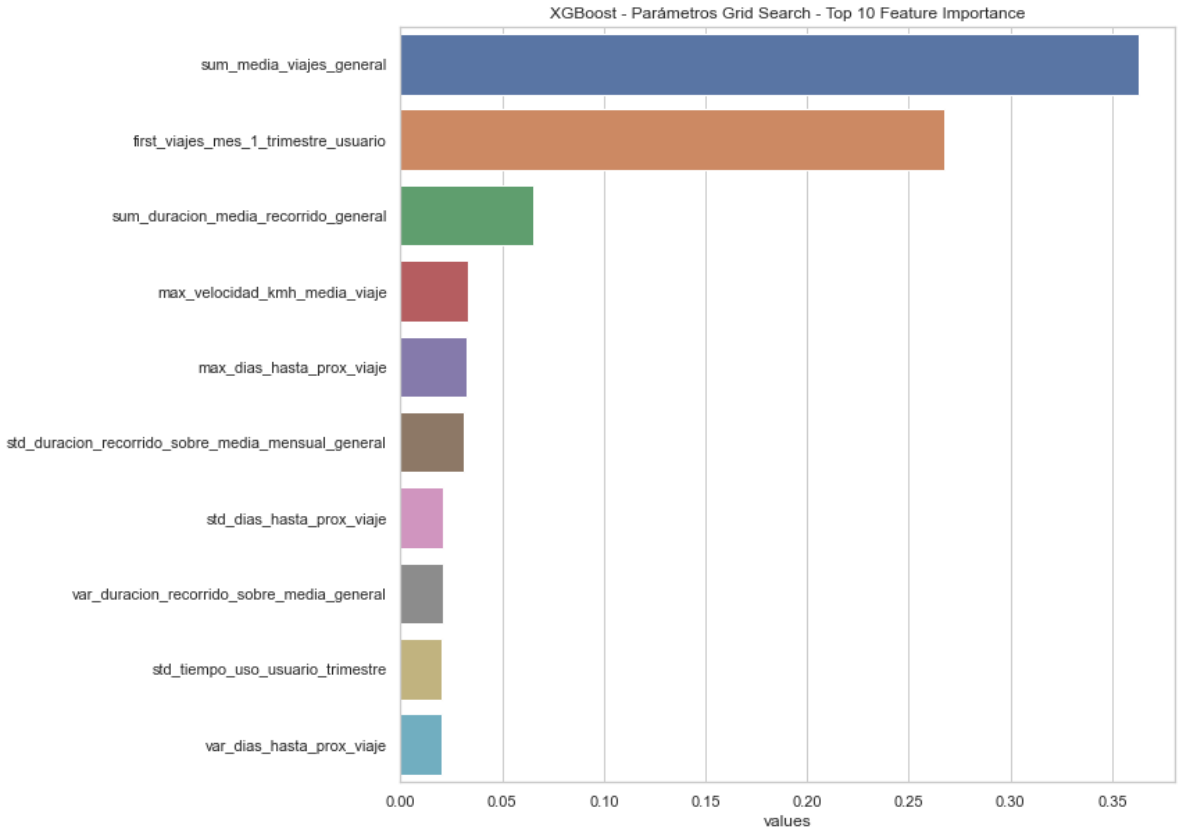


Ilustración 23. *Top 10 importancia de variables XGBoost parámetros Grid Search*

En este gráfico, podemos notar que hubo un cambio de variables en términos de importancia respecto del anterior, veámoslo en una tabla para poder evaluar de manera más sencilla las posiciones que ahora están tomando las variables.

N°	XGBoost parámetros Default	XGBoost parámetros Grid Search
1	sum_media_viajes_general	sum_media_viajes_general
2	first_viajes_mes_1_trimestre_usuario	first_viajes_mes_1_trimestre_usuario
3	sum_duracion_media_recorrido_general	sum_duracion_media_recorrido_general
4	var_duracion_recorrido_sobre_media_general	max_velocidad_kmh_media_viaje
5	max_velocidad_kmh_media_viaje	max_dias_hasta_prox_viaje
6	max_dias_hasta_prox_viaje	std_duracion_recorrido_sobre_media_mensual_general
7	var_dias_hasta_prox_viaje	std_dias_hasta_prox_viaje
8	std_duracion_recorrido_sobre_media_general	var_duracion_recorrido_sobre_media_general
9	std_tiempo_uso_usuario_trimestre	std_tiempo_uso_usuario_trimestre
10	std_dias_hasta_prox_viaje	var_dias_hasta_prox_viaje

Tabla 15. *Comparación variables XGBoost - Default vs. Grid Search*

Referencias:

- Subió posición respecto del otro modelo
- Mantuvo posición respecto del otro modelo
- Bajó posición respecto del otro modelo
- No se encuentra en ambos modelos (nuevo)

En la tabla previa, vemos que las primeras 3 variables a su vez que la número 9 conservaron sus respectivos lugares, es decir, parece que independientemente de los parámetros tomados en ambos modelos, mantuvieron su orden de pesos. Más no es así con las demás, las cuales, que fueron intercambiando lugares y hasta aparecieron nuevas entre los modelos. Tomando esta variación en cuenta, enfocándonos en la mejora del modelo con *Grid Search* respecto del por *Default*, probablemente el reemplazo de ésta variable nueva por la del otro parece adicionar valor en la mejora de las predicciones, que si las evaluamos, son referidas a las mismas variables "madre" pero derivando de otro tipo de cálculo respecto de la distribución de la misma, es decir, este nuevo enfoque parece agregar más valor que el anterior.

Lo analizado en los párrafos anteriores no es una verdad absoluta, los modelos pueden tener variabilidad en la importancia de las variables si cambiamos los datos sobre los cuales se entrena y predice, por lo que nos lleva a buscar asegurarnos de que esto sea así y la importancia que dicen que tienen sea tal. Para esto, vamos a introducir el concepto de *Feature Permutation*.

4.3. Feature permutation

El *Feature Permutation* mide el valor predictivo de una feature para cualquier estimador, clasificador o regresor de caja negra. Lo hace evaluando cómo aumenta el error de predicción cuando una característica no está disponible. Se puede utilizar cualquier métrica para medir el error de predicción (por ejemplo, F1 para la clasificación o R2 para la regresión). Para evitar la eliminación real de las características y el reentrenamiento del estimador para cada feature, el algoritmo mezcla aleatoriamente los valores de las características añadiéndoles ruido. A continuación, el error de predicción del nuevo conjunto de datos se compara con el error de predicción del conjunto de datos original. Si el modelo depende en gran medida de la columna que se baraja para predecir con precisión la variable objetivo, este reordenamiento aleatorio provoca predicciones menos precisas. Si

el modelo no depende de la característica para sus predicciones, el error de predicción no cambia. (Géron, A, 2019)

Las explicaciones del *Feature Permutation* generan una lista ordenada de características junto con sus valores de importancia. La interpretación de los resultados de este algoritmo es sencilla. Las características situadas en los rangos más altos tienen más impacto en las predicciones del modelo, aquellas en los rangos más bajos, menos. Además, los valores de importancia representan la importancia relativa de las características. (Géron, A, 2019)

A continuación, veremos las evaluaciones de los *Feature Permutation* de los modelos en cuestión, enfocándonos en aquel que mejor performance nos dio en las pruebas hechas y comentadas anteriormente e intentar evaluar el peso de cada variable.

Si analizamos las *Ilustraciones 26*, *27*, *28* y *29* (adjuntas en el Anexo), las importancias que aparecen son muy similares, es decir, la mayoría de las variables se repiten en todos, en diferentes posiciones y con leves variaciones, pero el foco que hacen los algoritmos, tanto con parámetros por Default como con Grid Search, es similar.

Yendo al listado de nuestro mejor modelo, XGBoost con *Grid Search*, vemos que la variable de la sumatoria de medias de viajes general de los usuarios es la que se lleva el protagonismo, como en el de parámetros por *Default*, pero también algunas relacionadas al *target* y demás variables comportamentales que hacen total sentido a la predicción, es decir, por ejemplo, la varianza del uso de las bicis, la velocidad máxima media de los viajes, duraciones de recorridos y demás, es lógico pensar que tengan un impacto grande en el modelo, ya que aquella persona más predispuesta al deporte y con mejor estado físico sea la que más regularmente pueda usar el servicio para poder obtener los beneficios que este pueda darle y usarlo de manera mucho más frecuente que alguien que no cumple con dichas condiciones, por ejemplo, no cansarse y optar por otro medio de transporte, pero sin segmentación en términos del sexo del usuario, como sí aparece en la importancia de variables de los que hablamos en la sección anterior.

Antes de concluir apartado actual, vamos a realizar un análisis de los modelos más performantes (*Tabla 14*) para contar con otra perspectiva del *Feature Permutation* usando la librería *Shapley (SHAP)* de Python.[4]

4 https://shap.readthedocs.io/en/latest/index.html

Comenzamos con el modelo *CatBoost* con parámetros por *Default*. El siguiente gráfico contiene las features del modelo que *SHAP(SHapley Additive exPlanations)* consideró más relevantes ordenadas de esta forma de mayor a menor según su *shap_value*, el cual es la contribución marginal media del valor de una feature (mayor rojo – menor azul) en todas sus combinaciones posibles.[5]

Esto, por ejemplo, podemos ver en la variable *max_dias_hasta_prox_viaje* en rojo tenemos aquellos con mayor impacto sobre el modelo, es decir, aquellos con mayor cantidad de días hasta su próximo viaje van a ser aquellos más probables de que dejen de usar el servicio o no hagan uso del mismo por un período prolongado. De igual forma, siguiendo esta lógica, veamos *first_viajes_mes_1_trimestre_usuario* tiene una distribución de colores al revés que la variables que comentamos anteriormente, pero tiene sentido pensar que aquellos con menos cantidad de viajes en el primer mes puedan seguir este comportamiento y tener más probabilidad de *churn*.

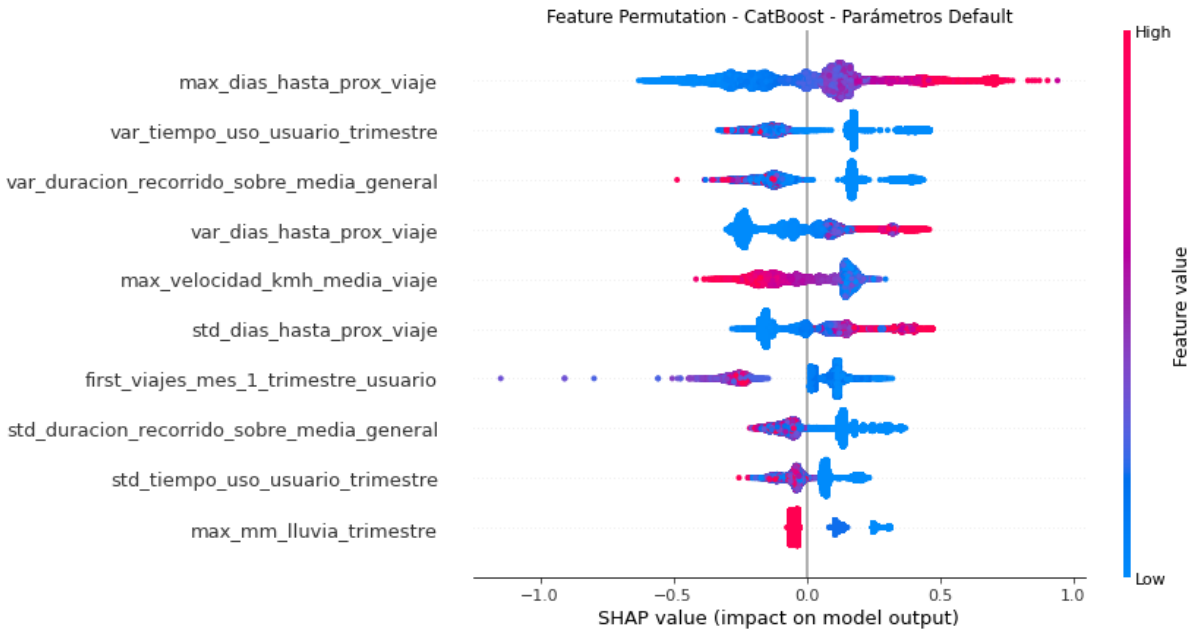


Ilustración 24. *Feature Permutation - CatBoost - Parámetros Default*

Podemos ver que las variables que aquí aparecen se comparten en general con las que analizamos en párrafos anteriores y detallada en la *Ilustración 29* en el Anexo, aunque muchas de ellas cambian de lugar y algunas nuevas aparecen, como var_duracion_recorrido_sobre_media_general y max_mm_lluvia_trimestre, por citar algunas, y otras como std_temp_media_trimestre ya no están, pero vemos que considera variables meteorológicas dentro del modelo total, aunque con bajo peso. Las más

5 Mukund Sundararajan, Amir Najmi Proceedings of the 37th International Conference on Machine Learning, PMLR 119:9269-9278, 2020

importantes siguen siendo asociadas a métricas de utilización de las bicicletas por parte de los usuarios.

Ahora, moviéndonos a *XGBoost* con parámetros optimizados por *Grid Search*, veremos si hubo cambios respecto de lo descrito en la *Ilustración 30* en el *Anexo*.

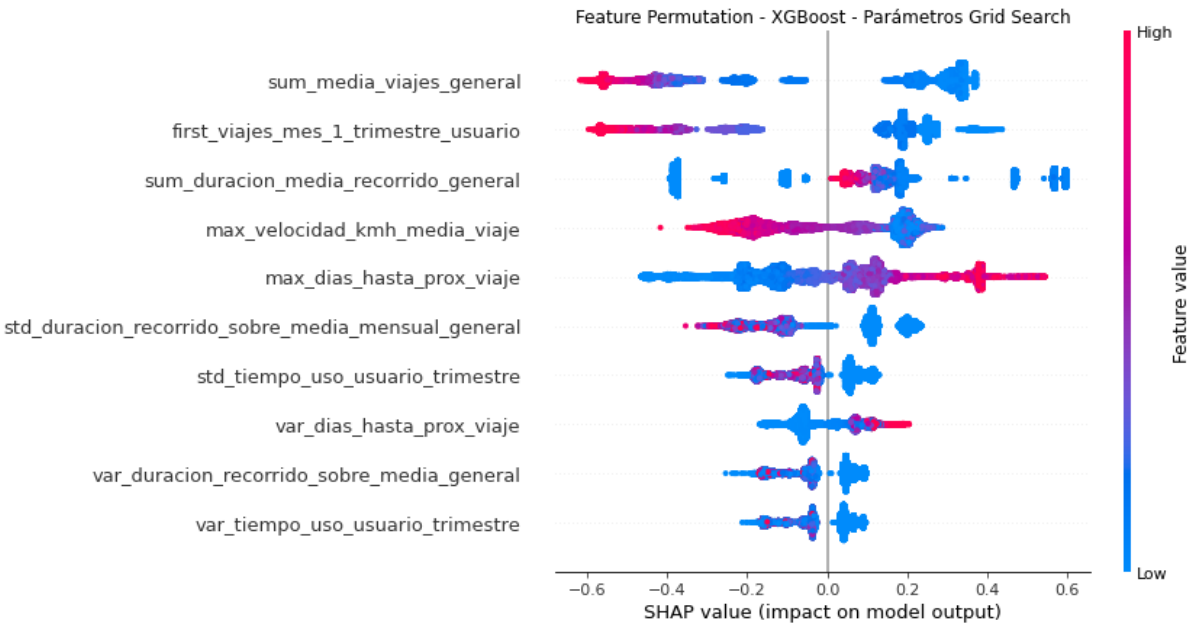


Ilustración 25. *Feature Permutation - XGBoost - Parámetros Grid Search*

Podemos ver que las variables se conservaron más, cambiando de nivel de importancia algunas, pero 8 de las 10 presentes se encuentran en ambos casos, y contrastando contra el modelo anterior, no vemos ninguna variable meteorológica dentro de la lista.

Para cerrar este punto, podemos decir que más allá del método que utilicemos para analizar las importancias relativas y contribuciones marginales de las variables, aquellas que encabezan las listas siguen manteniendo su lugar, lo que nos lleva a creer que realmente son imprescindibles para el buen desempeño de la predicción del modelo.

En la siguiente sección, vamos a finalizar hablando sobre las conclusiones que pudieron tomarse mirando lo comentado a lo largo de todos los puntos anteriores y los resultados empíricos obtenidos por los modelos evaluados.

5. Aplicaciones prácticas

Vamos a ver en términos monetarios cómo nos impacta la implementación de un modelo predictivo en un problema de ésta índole.

En un intento de retención de clientes, sin hacer uso de ninguna técnica ni algoritmo, tendríamos que darles a todos un beneficio por igual, porque no sabemos distinguir entre aquellos que van a dejar de hacer uso del servicio y los que no.

Analizando las distribuciones de nuestro dataset tenemos que las proporciones serían las siguientes (baseline promedio basado en la *Tabla 12*, sección 4.1), *churn* 65% y *no churn* 35%, aproximadamente. Digamos que el beneficio es regalar dos viajes en el trimestre, que cuesta $100 por viaje, y que el consumo promedio por usuario son $200, basándonos en una base de 10.000 usuarios tendríamos una erogación de $2.000.000 con la siguiente distribución:

Mal asignado	Bien asignado
$0{,}3500 * 10.000 * (-\$200 + \$200) = \$0$	$0{,}6500 * 10.000 * (-\$200 + \$200) = \$0$

Tabla 16. *Distribución del beneficio sin uso de predicciones*

Es decir, estamos gastando y ganando en partes iguales, nuestro resultado neto final es $0 por no contar con la información completa y/o estimada respecto de la distribución de usuarios.

Ahora, basándonos en la matriz de confusión de nuestro mejor modelo desarrollado aquí (*XGBoost*, ver métricas en la *Tabla 14*):

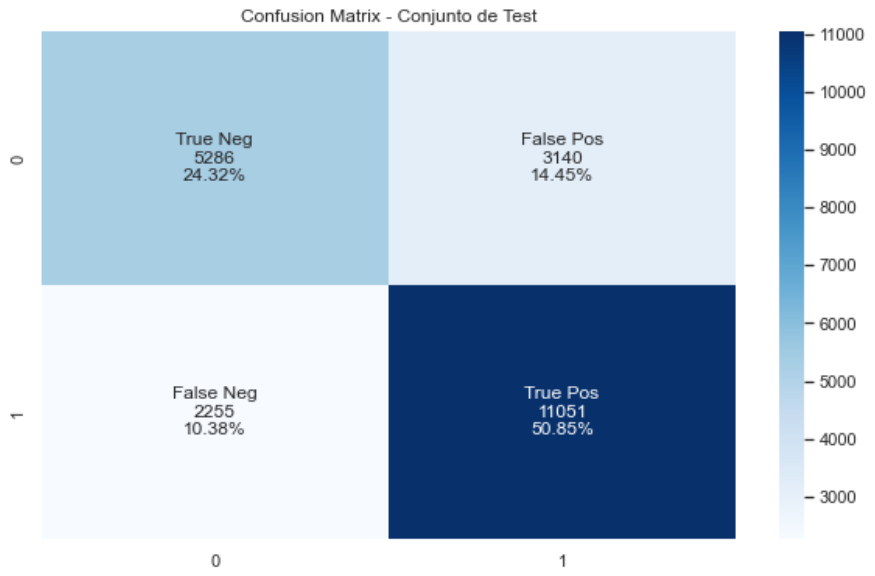


Ilustración 26. *Matriz de confusión XGBoost - Parámetros Grid Search - Conjunto Test*

Seguimos la distribución de los porcentajes obtenidos, interesándonos sólo la segunda columna, ya que corresponde a usuarios que harán *churn* (para el algoritmo) y por ende los que nos interesan retener, tendríamos:

Val. Real / Pred.	**0**	**1**
0	$0.2432 * 10.000 * (\$0 + \$200) = \$486.400$	$0{,}1445 * 10.000 * (-\$200 + \$200) = \$0$
1	$0.1038 * 10.000 * (\$0 - \$200) = -\$207.600$	$0{,}5085 * 10.000 * (-\$200 + \$200) = \$0$

Tabla 17. *Distribución del beneficio con uso de predicciones*

Con este enfoque estaríamos gastando la misma cantidad que en el anterior, pero tendríamos un rendimiento superior al gasto, es decir, nos queda un resultado final neto de:

$$\$486.400 - \$207.600 = \$278.800 \quad (9)$$

Es decir, un ROI del 27,88%, lo cual nos muestra que sin tener un modelo predictivo terminamos desperdiciando recursos que podrían ser mejor aprovechados y destinados a otras necesidades.

6. Limitaciones y posibles puntos de mejora

Durante el desarrollo del presente trabajo nos hemos encontrado con ciertos obstáculos que hubieran sido de gran utilidad para el desarrollo y mejora de la performance de los modelos llevados a cabo, como por ejemplo, poder contar con un dataset con información climatológica diaria, a diferencia de la que utilizamos, que contiene información mensual y métricas generales calculadas a ese nivel de granularidad temporal. También podría de hacerse uso de información relacionada análisis de la distribución ciclovías, seguridad y facilidad de accesos para los ciclistas. En ésta misma línea, contar con encuestas y respuestas reales de los usuarios, que puedan escapar a los datos crudos que podemos obtener sería de extrema importancia.

Un punto muy importante a destacar es que, dentro de la información analizada en el presente documento se encuentra el período de COVID, lo cual supone un shock externo en términos de los cambios de patrones de consumos y muchas costumbres que la gente solía tener antes del mismo. Valdría la pena analizar, como continuación, el impacto que esto tuvo sobre las personas y sus acciones en el día a día.

Por otra parte, para poder tener un mejor contraste en términos económicos, podría extenderse el actual estudio mediante el análisis de impacto de la implementación del corriente sistema de pago para este servicio y comparar si se encuentran cambios en los patrones de uso de las Ecobicis y determinar si el factor económico es una variable decisiva en la términos de la baja de usuarios.

Por último, agregar que en la variable dependiente, es decir el *target*, podría hacerse un *Survival Analysis* para la optimización del punto de corte para la misma, ya que es un parámetro extra que puede tunearse fácilmente cambiándolo en la generación de los datasets, volviendo a ejecutar el código que genera las demás variables y corriendo los modelos nuevamente. Esto puede ser de gran impacto en el rendimiento de los modelos, aunque escapa al alcance del presente documento.

Todo lo anterior comentado aportaría mucho a la mejora en el entendimiento del caso de estudio y en la afinación y generalización de las predicciones de los modelos aquí descritos.

7. Conclusiones

Como conclusión final, luego de todo el análisis y los enfoques tenidos en cuenta, podemos ver que el modelo *XGBoost* nos provee un mejor rendimiento por sobre *CatBoost* aunque nos encontramos con un trade-off de tiempos de entrenamiento y parameter tunning mayor en el primero que en segundo. Más allá de esto y considerando que los tiempos de entrenamiento no son demasiado extensos, nos interesa optar por el modelo con mayor rendimiento, que en este caso es *XGBoost*.

Haciendo referencia al estudio de variables, vemos que aquellas de mayor relevancia fueron las relacionadas al comportamiento del usuario en el uso de las bicicletas, como podemos ver en la *Ilustración 25*, de SHAP values, en donde las primeras 3 incluyen la media de viajes en el trimestre, la cantidad de viajes del usuario en el primer mes del trimestre y la duración media de los recorridos que hagan. No vemos que se incluyan variables relacionadas al clima, pero esto puede ser debido a lo comentado en el punto 6 de que no contamos con información con mayor granularidad.

En términos económicos, vemos que la aplicación de un modelo de predicción nos estaría dando un ROI de 27,88% superior a si no tuviéramos ningún tipo de algoritmo, descrito en el punto 5, que nos ayudara a poder reducir la incertidumbre futura en lo que respecta a la baja de

usuarios de este servicio. Esto es de alta relevancia, ya que es dinero que podría ahorrarse y destinarse a impulsar otros proyectos.

8. Trabajos citados

BA Data - Ecobicis. (s.f.). *Bicicletas Públicas*. Obtenido de https://data.buenosaires.gob.ar/dataset/bicicletas-publicas

BA Data - Precipitaciones. (s.f.). *Registro de Precipitaciones*. Obtenido de https://data.buenosaires.gob.ar/dataset/registro-precipitaciones-ciudad

BA Data - Temperaturas. (s.f.). *Registro de Temperatura*. Obtenido de https://data.buenosaires.gob.ar/dataset/registro-temperatura-ciudad

Bigazzi, A. S. (2017). *Bike Share Usage Patterns and User Demographics in a Mid-Sized US City.* Transportation Research Record.

Chen, T., & He, T. (2021). *XGBoost: Scalable and Flexible Gradient Boosting.* Taylor & Francis Group.

Dirección de Investigación Accidentológica & Dirección Nacional de Observatorio Vial. (2021). *¿Hacia una movilidad sustentable y segura? Una mirada global y local sobre el uso de la bicicleta como modo de transporte.*

Dorogush, A. V., Ershov, V., & Gulin, A. (2020). *CatBoost: Gradient Boosting with Categorical Features Support.* Springer.

Ente de la Movilidad de Rosario. (2018). *EMR*. Obtenido de http://www.emr.gov.ar/info_tnm.php

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc.

Gobierno de la Ciudad de Buenos Aires. (s.f.). *buenosaires.gob.ar*. Obtenido de https://buenosaires.gob.ar/ecobici/historia-de-la-bici#:~:text=Ecobici%3A%20Programa%20de%20bicicletas&text=Antes%20de%20lanzar%20el%20programa,infancia%20compartido%20con%20seres%20queridos.

Gobierno de la República Argentina. (s.f.). *Plan Nacional de Transporte Sostenible*. Obtenido de https://www.argentina.gob.ar/transporte/transporte-sostenible

González, D., Llorca, J., & Coque, J. J. (2018). *Assessing the Value of Social Networks to Predict Bike Sharing Churn.* Transportation Research Part C: Emerging Technologies.

Graciela Español, A. B. (2014). *Córdoba - Camino a una Movilidad más Amigable.* Córdoba. Obtenido de http://congresodevialidad.org.ar/congreso2016/TRA/TRA-168.pdf

Han-Soon Sin, Woojin Paik. (2019). *Predicting Churn Rate Of The Massively Multiplayer Online Role-Playing Game (Mmorpg) Users By Analyzing Playing Behavior.* International Journal of Scientific & Technology Research.

Kelleher, J. D., & Tierney, B. (2018). *Data science: An introduction (2nd ed.).* CRC Press.

Kumar, V., & Reinartz, W. (2012). *Customer Relationship Management: Concept, Strategy, and Tools.* Springer Science & Business Media.

P. Brighten Godfrey, Scott Shenker, and Ion Stoica. (s.f.). *Minimizing Churn in Distributed Systems.* Berkeley: UC Berkeley.

Phillippa Lally, Cornelia H. M. Van Jaarsveld. (2009). *How are Habits Formed: Modelling Habit Formation in the Real World.* London: European Journal of Social Psychology.

Resolución 173, Subsecretaría de Transporte. (20 de 12 de 2010). *Boletín Oficial*. Obtenido de https://boletinoficial.buenosaires.gob.ar/normativaba/norma/162092

Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. (2004). *Handling Churn in a DHT.* Berkeley and Intel Research, Berkeley: University of California.

Witten, I. H., Frank, E., & Hall, M. A. (2016). *Data Mining: Practical Machine Learning Tools and Techniques (4th ed.).* Morgan Kaufmann.

Yu, Q., Liu, Y., & Gong, Y. (2019). *What Influences Bike-Sharing Churn? A Case Study of Citi Bike.* Sustainability.

Anexo

Parámetro	Valores
Iterations	[1000, 2000, 4000]
Learning_rate	[0.01, 0.02, 0.05, None]
Max_depth	[3, 5, None]
L2_leaf_reg	[0.5, 1, None]

Tabla 18. *Parámetros de Grid Search para CatBoost*

Parámetro	Valores
N_estimators	[100, 200, 500]
Learning_rate	[0.01, 0.05, 0.10, 0.30]
Max_depth	[3, 5]

Tabla 19. *Parámetros de Grid Search para XGBoost*

Modelos	Parámetro	Mejor valor encontrado
CatBoost	iterations	2000
	learning_rate	0,02
	max_depth	5
	l2_leaf_reg	1
XGBoost	n_estimators	500
	learning_rate	0,01
	max_depth	3

Tabla 20. *Mejores parámetros para CatBoost y XGBoost encontrados por Grid Search*

```
max_dias_hasta_prox_viaje 0.007 +/- 0.000
first_viajes_mes_1_trimestre_usuario 0.003 +/- 0.001
sum_media_viajes_general 0.003 +/- 0.001
mean_dias_hasta_prox_viaje 0.003 +/- 0.000
var_dias_hasta_prox_viaje 0.002 +/- 0.000
std_tiempo_uso_usuario_trimestre 0.002 +/- 0.000
max_velocidad_kmh_media_viaje 0.002 +/- 0.000
std_dias_hasta_prox_viaje 0.002 +/- 0.000
median_dias_hasta_prox_viaje 0.001 +/- 0.000
std_temp_media_trimestre 0.001 +/- 0.000
```

Ilustración 27. *Feature Permutation Importance - CatBoost Parámetros Default*

```
max_dias_hasta_prox_viaje 0.011 +/- 0.001
sum_media_viajes_general 0.003 +/- 0.000
std_dias_hasta_prox_viaje 0.003 +/- 0.000
var_dias_hasta_prox_viaje 0.003 +/- 0.000
first_viajes_mes_1_trimestre_usuario 0.002 +/- 0.001
max_velocidad_kmh_media_viaje 0.002 +/- 0.000
std_tiempo_uso_usuario_trimestre 0.001 +/- 0.000
sum_duracion_media_recorrido_general 0.001 +/- 0.000
var_duracion_recorrido_sobre_media_mensual_general 0.001 +/- 0.000
sum_cantidad_recorridos_sobre_media_mensual_general 0.001 +/- 0.000
```

Ilustración 28. *Feature Permutation Importance - CatBoost Parámetros Grid Search*

```
sum_media_viajes_general 0.008 +/- 0.001
max_dias_hasta_prox_viaje 0.008 +/- 0.001
max_velocidad_kmh_media_viaje 0.005 +/- 0.000
first_viajes_mes_1_trimestre_usuario 0.004 +/- 0.001
var_tiempo_uso_usuario_trimestre 0.004 +/- 0.000
var_dias_hasta_prox_viaje 0.003 +/- 0.000
var_duracion_recorrido_sobre_media_general 0.003 +/- 0.000
median_dias_hasta_prox_viaje 0.003 +/- 0.000
std_duracion_recorrido_sobre_media_general 0.002 +/- 0.000
std_tiempo_uso_usuario_trimestre 0.002 +/- 0.000
```

Ilustración 29*. Feature Permutation Importance - XGBoost Parámetros Default*

```
sum_media_viajes_general 0.006 +/- 0.000

max_velocidad_kmh_media_viaje 0.005 +/- 0.000

max_dias_hasta_prox_viaje 0.005 +/- 0.000

std_dias_hasta_prox_viaje 0.005 +/- 0.000

var_dias_hasta_prox_viaje 0.004 +/- 0.000

std_duracion_recorrido_sobre_media_mensual_general 0.004 +/- 0.000

first_viajes_mes_1_trimestre_usuario 0.003 +/- 0.000

var_duracion_recorrido_sobre_media_general 0.002 +/- 0.000

var_tiempo_uso_usuario_trimestre 0.002 +/- 0.000

std_duracion_recorrido_sobre_media_general 0.002 +/- 0.000
```

Ilustración 30. *Feature Permutation Importance - XGBoost Parámetros Grid Search*