Detección de potenciales infractores en sistemas de bicicletas públicas en Ciudad de Buenos Aires

- Diego Castro -

Índice:

---

1. - Introducción:

1.1 - Contexto:

El gobierno de la ciudad de Buenos Aires brinda hace varios años un servicio de bicicletas de uso público y gratuito. Los usuarios primero deben darse de alta en el sistema completando información referente al nombre y apellido, sexo, edad, y DNI -en caso de poseer uno-, y una vez cumplido con esto quedan habilitados a retirar una bicileta de alguna de las multiples estaciones, la cuál debe ser devuelta (ya sea a la misma estación o a alguna otra) en un plazo que no debería superar los 30 minutos.

1.2 - Problema:

Puesto que las bicicletas no cuentan con un dispositivo GPS que permita determinar su posición a cada instante, una vez que el usuario retira una bicicleta del sistema, no se tiene más información de la misma hasta que no es devuelta.

Si bien la violación al límite de tiempo de 30 minutos es algo que puede ocurrir incluso a usuarios que actuando de buena fé se atrasan en la devolución o calculan mal el tiempo de viaje entre estaciones, el foco está puesto en equellos usuarios que no tienen ninguna intención de devolver la bici en el tiempo estipulado, con tiempos de viaje que superan las 24 horas.

1.3 - Objetivo:

El objetivo es por lo tanto predecir qué usuarios demorarán más de un día en regresar la bicicleta, de modo tal que el sistema bloquee al usuario momentáneamente e impida el retiro de la unidad.

2. - Metadata

2.1 - Contexto Analítico:

Para intentar cumplir el objetivo contaba con 4 conjuntos de tablas:

- Una conjunto tiene información referida a los viajes realizados en cada año calendario, donde bajo un ID del viaje queda registrada la información del usuario que retira la bicicleta, la estación de origen y la estación de destino (nombre, id, y posición gps de ambas), el horario en el que comienza el viaje y la duración del mismo en segundos.
- Una segundo conjunto de tablas presenta la información que cargaron los usuarios al momento de darse de alta en el sistema bajo un id_usuario asignado: nombre y apellido, edad, sexo, fecha y hora de alta y si cuenta o no con un DNI
- La tercera tabla proporciona información de las estaciones: Id, nombre, dirección, barrio, comuna, tipo de emplazamiento, cantidad de anclajes y posición gps.
- Y por último, la cuarta tabla con la que cuento contiene la información de cuántos habitantes viven en cada una de las comunas, sólo en caso de que pueda serme útil después.

Se compilaron estos datasets en uno único y se incorporaron columnas calculadas que me serían útiles. Hecho eso, para cada viaje efectuado se contaba con la siguiente información:

- Información referente al recorrido:
    - Id_recorrido: Identificación única para cada viaje que se realizó
    - duracion_min: Duración de cada viaje en minutos
- Información referente al origen del viaje:
    - fecha_origen_recorrido: fecha de comienzo del viaje en formato YYYY-MM-DD HH:mm:ss
    - id_estacion_origen: Identificación única para cada estación
    - nombre_estacion_origen: Nombre de la estación
    - direccion_estacion_origen: Dirección en CABA
    - long_estacion_origen: Longitud en grados
    - lat_estacion_origen: Latitud en grados
- Información referida al destino del viaje:
    - fecha_destino_recorrido: fecha de finalización del viaje en formato YYYY-MM-DD HH:mm:ss
    - id_destino_origen: Identificación única para cada estación
    - nombre_estacion_destino: Nombre de la estación
    - direccion_estacion_destino: Dirección en CABA
    - long_estacion_destino: Longitud en grados
    - lat_estacion_destino: Latitud en grados
- Información referida al usuario:
    - id_usuario: Identificación única para cada usuario
    - género: género del usuario que realiza el viaje
    - genero_usuario: género del usuario que realiza el viaje proveniente de la tabla origen de altas en el sistema
    - edad_usuario: edad del usuario que realiza el viaje en años
    - fecha_alta: fecha en la que el usuario se dio de alta en el sistema en formato YYYY-MM-DD
    - hora_alta: hora en la que el usuario se dio de alta en el sistema en formato HH:mm:ss
    - Customer.Has.Dni..Yes...No.: Si la persona aportó su DNI a la hora de darse de alta en el sistema
- Información referida a la estación de retiro de bicicleta:
    - barrio_origen: Barrio donde se encuentra la estación donde se retira la bicicleta
    - comuna_origen: Comuna a la que pertenece el barrio
    - emplazamiento_origen: Ubicación de la estación en la vía pública: Vereda, parque, parque cerrado, etc.
    - poblacion_comuna_origen: Cantidad de habitantes de la comuna donde se origina el viaje
    - anclajes_origen: Cantidad de anclajes -posiciones para bicicletas- en la estación dónde se retira la bici
    - modelo_bicicleta: Modelo de la bicicleta
- Etiqueta:
    - infractor: Categorización de si el usuario cometió una infracción de tiempo en su viaje. 1: Sí 0: No

Una vez finalizado el trabajo con todas las tablas mencionadas el DataFrame resultante contaba con 30 columnas y 1.399.818 registros de viajes distintos realizados durante 2023, pero debido al bajo número que representaban los infractores (menos del 1%) me vi obligado a rastrear e incorporar también infractores de 2021 y 2022. El objetivo de engrosar un poco este subgrupo era que me permitiese obtener características más marcadas de sus integrantes, ya que para que una muestra se considere balanceada debería representar al menos el 20 o 30% del total.

3. - Preguntas de interés:

3.1 - Preguntas principales:

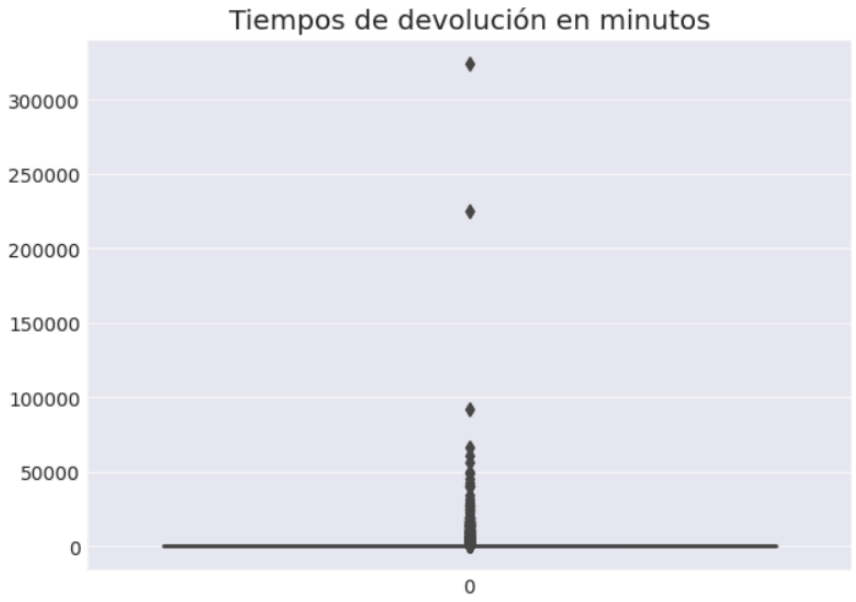
¿Existen usuarios que hacen uso/abuso del sistema de bicicletas e incumplen con los tiempos de devolución pautados?
¿Hay alguna forma de lograr una detección temprana de estos usuarios de modo tal de bloquearles el acceso a las bicicletas?

3.2 - Preguntas secundarias:

De toda la información con la que cuento: ¿Cuál puede serme de utilidad para detectar a los infractores? ¿Existe un patrón?
¿Qué tan efectiva es esta detección? ¿Acaso genera inconvenientes a usuarios que utilizan el servicio debidamente?

4. - Análisis exploratorio:

Lamentablemente a nadie extraña que en un servicio gratuito, en el que uno obtiene un bien material con valor comercial sin una supervisión constante, existan infracciones y hurtos. Las bicicletas de la ciudad no están exentas de esto y podemos verlo reflejado si graficamos los tiempos de devolución en un boxplot



La existencia de outliers tan extremos genera que su gráfico se distorsione al punto de perder claridad, pero de todas formas sirve para probar el punto de que existen tiempos muy por encima de los 30/60 minutos establecidos por el prestador del servicio.

A partir de lo antepuesto, la idea del siguiente análisis es intentar averiguar cuáles de estos features pueden ser útiles a la hora de predecir nuevas infracciones. Para esto se graficaron las columnas de interés tanto para el grupo de infractores como para el grupo que cumplió con la devolución a tiempo y se compararon las distribuciones en orden de intentar percibir diferencias considerables entre ambos grupos como una primera aproximación al tema.

Las columnas de interés en un principio eran:

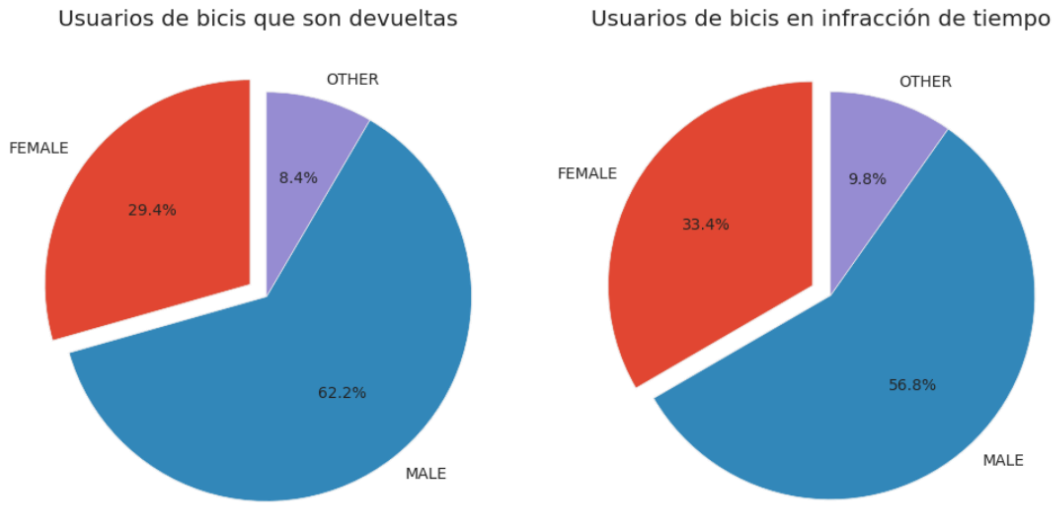
- Género del usuario
- Edad del usuario
- Estación de origen del viaje (o bien barrio de origen)
- Horario de origen del viaje
- Tiempo transcurrido desde el alta en el sistema hasta el origen del viaje

Estoy dejando fuera del análisis toda la información correspondiente a la estación de destino porque si la idea es predecir la posibilidad de que se cometa una infracción en un viaje que todavía está por hacerse, no voy a contar con la información de hacia dónde se dirige, ya que cuando uno retira una bici no proporciona este dato.

4.1 - Caracterización de usuarios por género

¿Acaso existe un sesgo en el sexo de los usuarios infractores?

Para responder la pregunta se graficó la distribución por género de los viajes realizados, comparando no-infractores vs infractores:



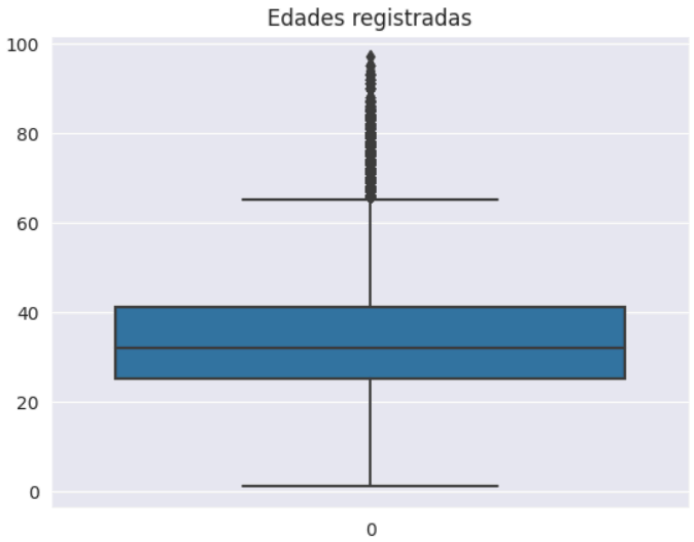
4.1.1 - Insights:

- Puede apreciarse fácilmente que hay el doble de hombres que de mujeres haciendo uso del servicio de bicicletas del gobierno de la ciudad
- Vemos también que los infractores mantienen aproximadamente las mismas proporciones que los usuarios convencionales. La mayor diferencia (porcentual) se da en lo que respecta a "OTHER" que sube un 16.6% (de 8.4 a 9.8%). ¿Acaso los infractores intentan brindar la menor cantidad de información posible sobre su persona? No parecen ser el mejor indicador de si se cometerá una infracción o no.

4.2 - Caracterización de usuarios por edad

¿Hay mucha dispersión en las edades que declaran los usuarios del sistema? ¿Hay valores que carezcan de sentido biológicamente hablando?

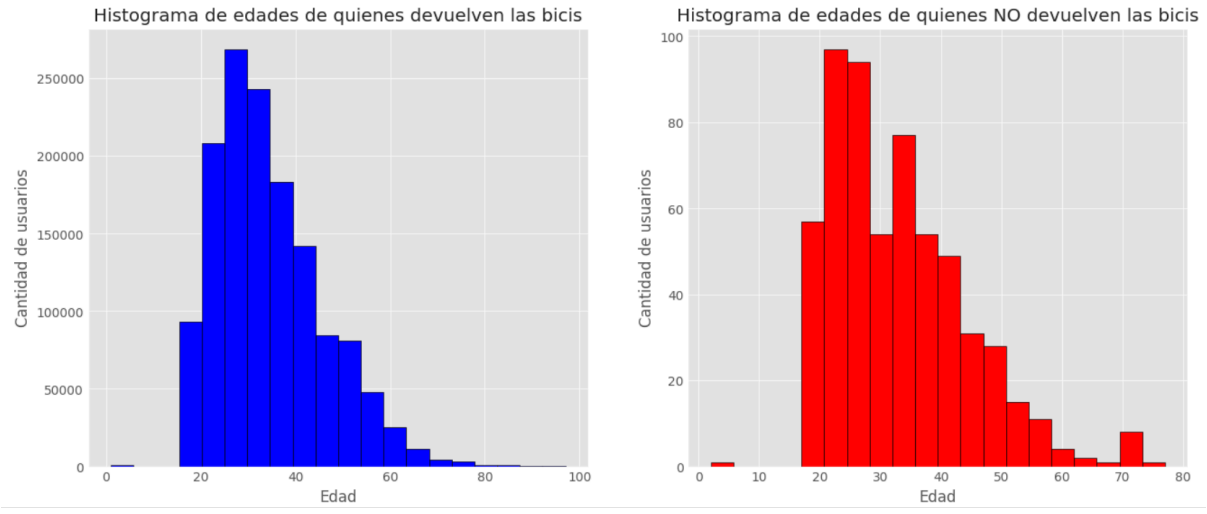
Trabajé un poco con el campo *edad_usuario* para responder estas preguntas, truncándolo a valores menores a 100 para una representación gráfica más clara:



Y así truncado y todo, existen todavía una gran cantidad de outliers, pero ya la representación es un poco más clara.

Ahora bien, ¿la edad que declaran los usuarios al darse de alta podrá ser un factor que me ayude a predecir potenciales infractores?

Para responder procedí a graficar las distribuciones de ambos subgrupos (infractor vs no infractor) para compararlas:



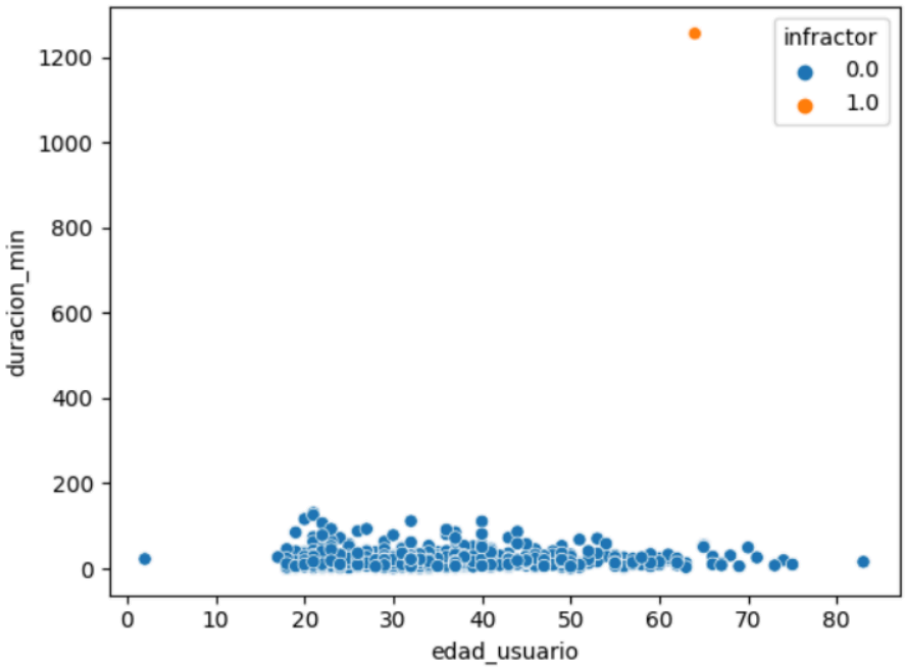
4.2.1 - Insights:

Acá sí se aprecian algunas diferencias en la forma de la distribución: Nótese por ejemplo el pico por encima de los 70 años que presenta el histograma de infractores. Si bien debe tenerse presente que la cantidad de datos mucho menor puede influir en la gráfica de la distribución, igual podría tratarse de un factor a considerar a la hora de hacer predicciones.

4.2.2 - Para pensar:

¿Debo reemplazar los datos mal colocados? ¿O un dato intencionalmente mal puesto puede ser un factor útil para la detección de posibles infracciones y hurtos? En ese caso los outliers también me están brindando información útil que no debo descartar, por lo que decidí dejarlos tal y como estaban.

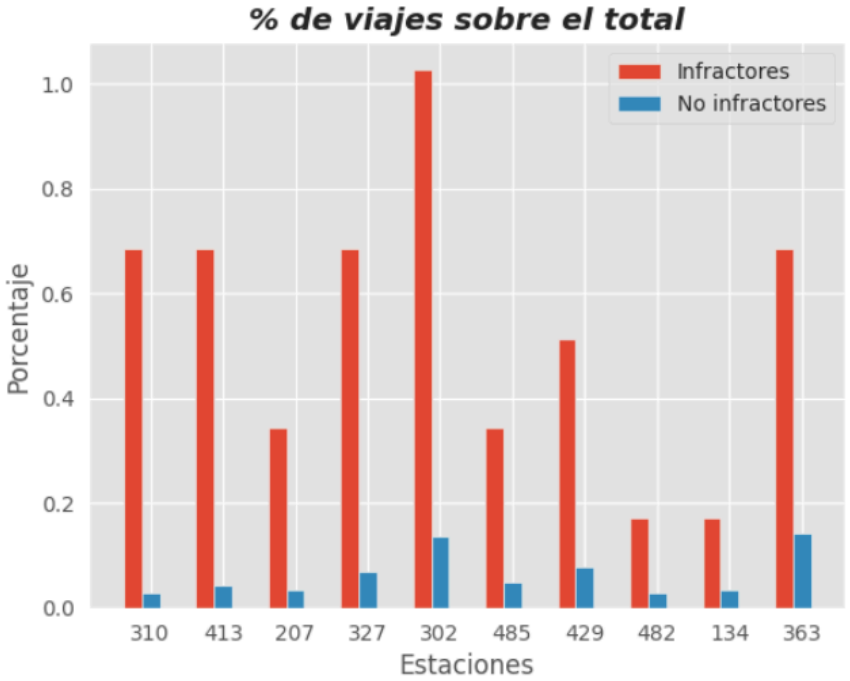
De todas maneras, a través de un scatter plot se logra apreciar que no siempre las edades extremas son un predictor de infracciones.



Tal como se ve en el gráfico, se detectan casos de usuarios de casi 0 años y más de 80 que realizan la devolución de la bicicleta en tiempo y forma, pero no debe pasarse por alto que el único infractor de la muestra también declaró en el momento de darse de alta tener casi 70 años, lo cual es por lo menos sospechoso y podría funcionar como una red-flag.

4.3 - Análisis de estación de origen

¿Y si existe una relación entre infractores y la estación de la que retiran la bicicleta? ¿Hay estaciones donde se concentren los hurtos? Se realizó un gráfico de barras para responder.



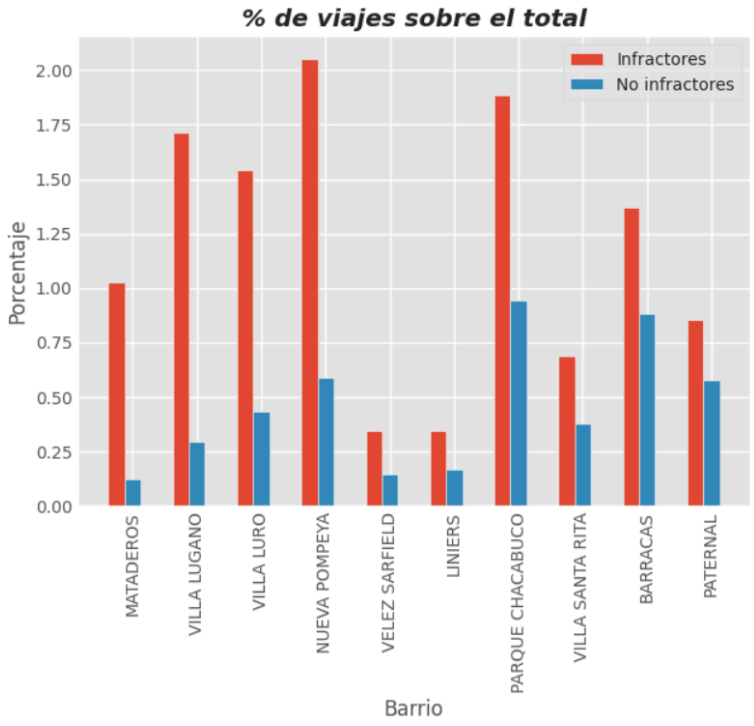
4.3.1 - Insights:

Lo que interpreto sobre el gráfico es que algunas estaciones aparecen representando un porcentaje mayor de viajes en infracción de lo que representan en general como % de viajes sobre el total.¿Entonces puede decirse que hay estaciones apuntadas? En principio parecería que sí, o al menos más vulnerables.

Por ejemplo la estación 310, que en promedio representa el 0.028% de todos los viajes realizados, reunió el 0.68% de las infracciones → Es un incremento del 2269%. Y casos semejantes se repiten en otras estaciones. ¡Es un resultado cuanto menos interesante!

4.4 - Por barrio de origen

¿Le cabe un análisis similar a los barrios desde donde inician los viajes? Veamos.



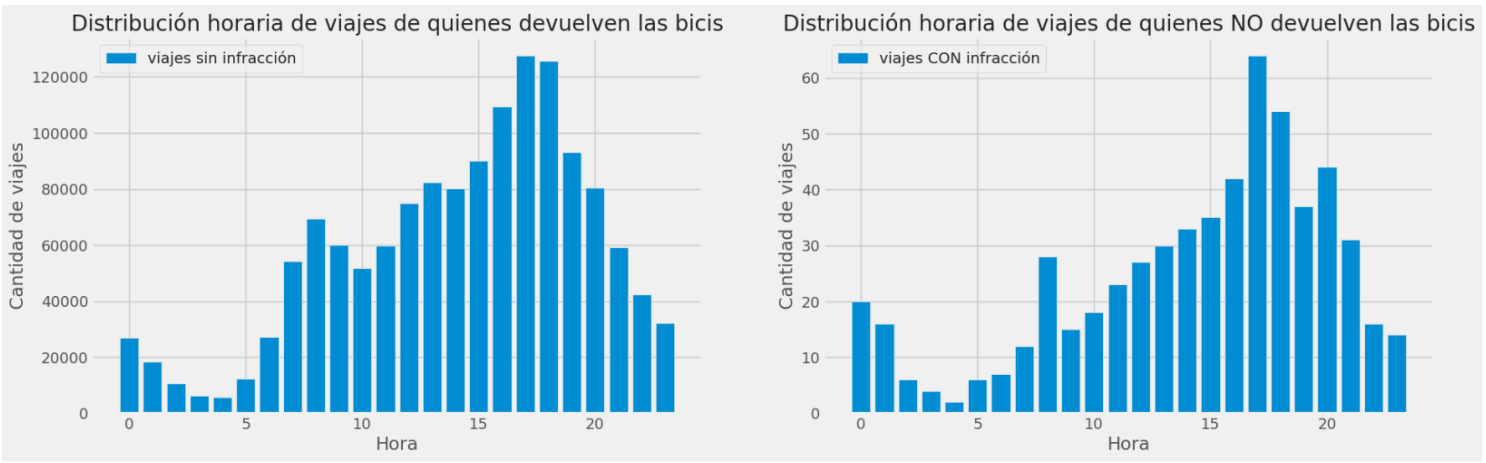
4.4.1 - Insights:

Si bien los incrementos máximos no son tan grandes como en el caso de las estaciones puntuales, hay barrios donde el % de infractores es marcadamente mayor que el % de viajes sin infracción

Por ejemplo Mataderos, que representa el 0.076% de los viajes sin infracción se lleva más del 1% de las infracciones, lo que significa un 1243% de aumento.

4.5 - Horario de origen de viaje

¿Y la hora en la que se retira una bicicleta de una estación me da alguna información adicional? ¿Hay horarios que sean potencialmente más riesgosos?

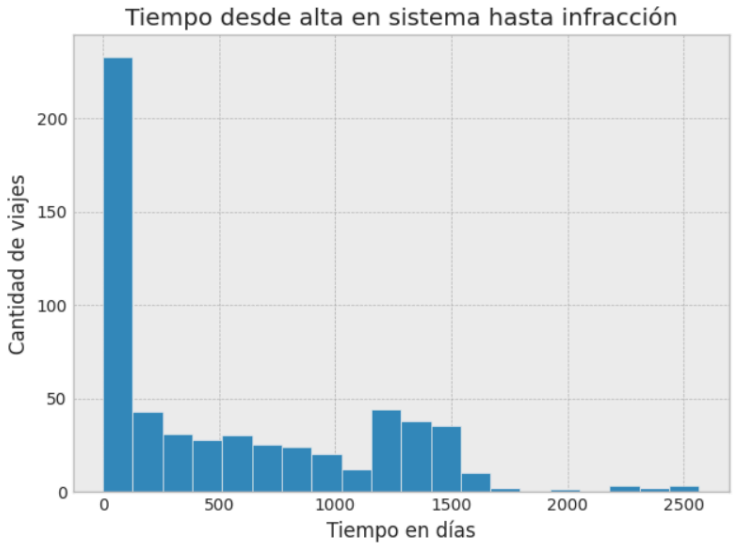


4.5.1 - Insights:

A primer golpe de vista (y para mi desaliento) da la sensación de que las distribuciones son bastante semejantes, mostrando picos de viajes lícitos e ilícitos entre las 15 y las 20 horas. Por lo tanto este no sería un factor determinante a la hora de predecir infracciones, pero antes de descartarlo por completo habría que mirar con más atención los picos cerca de medianoche

4.6 - Análisis del tiempo transcurrido desde alta en sistema

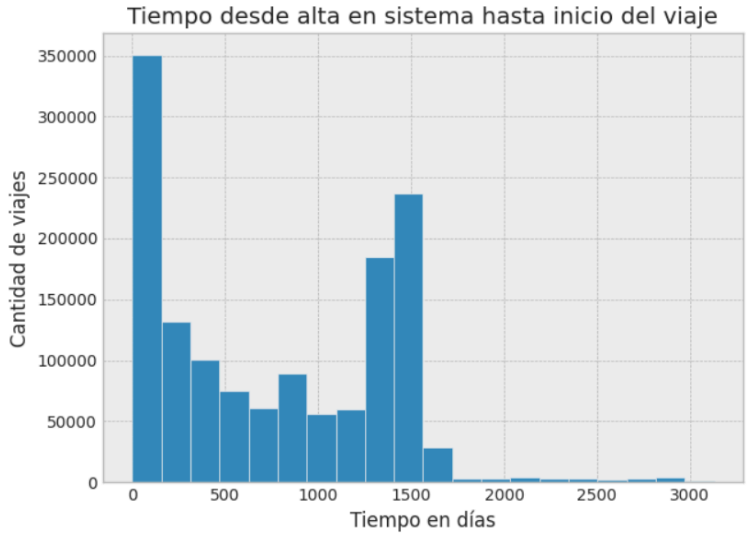
Por último traté de determinar si había algún patrón en el tiempo que transcurría desde el alta en el sistema hasta que se producía la infracción (con la idea intuitiva de que si la infracción era premeditada, la persona generaba el usuario ya con el objetivo de no devolver la bici). Lo que esperaba ver acá era una gran cantidad de infractores con usuarios recién dados de alta en el sistema, así que para confirmarlo generé un histograma:



4.6.1 - Insights:

En principio da la sensación de que el presentimiento de que hay gente que crea usuarios con la intención de no devolver las bicis parece andar en lo cierto porque la mayoría de las infracciones las lleva adelante gente con usuarios nuevos, con un día o menos de creados.

El problema es que esto resulta engañoso y hay que tener cuidado cuando uno saca conclusiones apresuradamente, porque lo que en realidad ocurre es que esa es la distribución habitual de todos los usuarios de bicis: muchos viajes luego de las primeras horas de darse de alta y un rápido decrecimiento del factor de uso después.



De todas maneras, las formas de las distribuciones no son completamente iguales, ya que por ejemplo aquellos usuarios que se han dado de alta hace 3 ó más años y que utilizan mucho el servicio (el pico de los 1500 días) es realmente poco probable que incumplan con el tiempo de devolución y deberíamos estar relativamente tranquilos con ese grupo. Entonces el tiempo transcurrido desde el alta en el sistema sí me está aportando información útil y debo tenerlo en consideración.

5. - Modificaciones al df - Columnas calculadas:

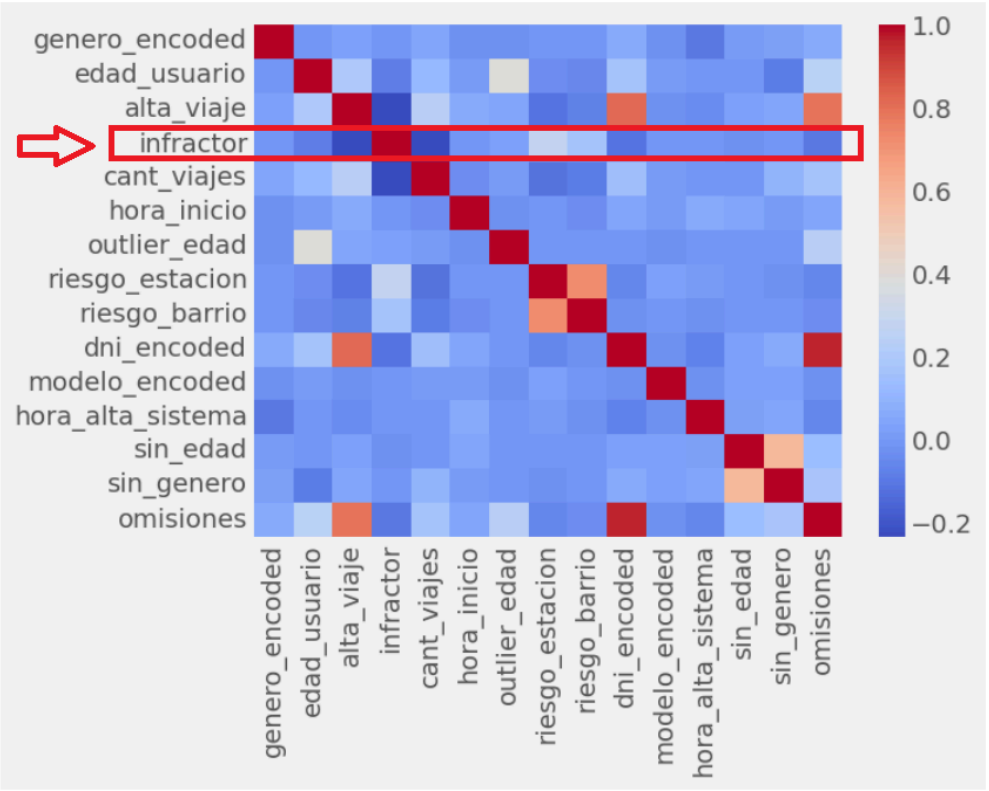
A partir de las visualizaciones generadas durante el análisis exploratorio de datos concluí que lo mejor era modificar el df de trabajo incorporando columnas que me permitieran explotar los hallazgos:

- Cantidad de viajes por usuario
- Hora de inicio de viaje
- Hora de alta en el sistema
- Identificación de edades fuera de rango
- Identificación de estaciones con mayor probabilidad de infracción
- Omisiones: Que totaliza la cantidad de datos faltantes o mal colocados de cada usuario

Finalmente, a partir de lo hasta aquí analizado y descartando por ejemplo columnas referentes a la información de la estación de destino (porque como ya se mencionó si la idea es predecir infracciones no voy a contar con la información de hacia dónde se dirige el usuario) y otras que tenían información redundante o derivativa (LD), escogí los siguientes features como primera selección por considerarlos los más prometedores:

```
'genero_encoded','edad_usuario','alta_viaje','infractor','cant_viajes',
'hora_inicio','outlier_edad','riesgo_estacion','dni_encoded','modelo_en
coded','hora_alta_sistema','sin_edad','sin_genero','omisiones'
```

A través de un heatmap vemos que ninguno de ellos tiene una correlación fuerte con la variable objetivo *"infractor"*, por lo que hubo que trabajar horas extra.



Particularmente en este caso, por tratarse de un problema de clasificación binaria (infractor - no infractor) debió implementarse un algoritmo de aprendizaje supervisado para tal fin.

6. - Breve resumen técnico:

No entraré en detalles sobre los distintos métodos usados a la hora de buscar una solución al problema planteado puesto que ese no es el objetivo de esta presentación ejecutiva, pero sí haré una breve descripción del proceso.
Se pusieron a prueba cuatro métodos de aprendizaje supervisado:

- *Decision Tree Classifier*
- *Random Forest Classifier*
- *K-Nearest-Neighbor*
- *Logistic Regression*

Y a la luz de los resultados obtenidos se decidió descartar los últimos dos, para seguir adelante con Decision Tree y Random Forest.

Debido a que todavía nos encontrábamos trabajando con 14 features, el paso siguiente fue intentar reducir la dimensionalidad a través del análisis de los componentes principales (*PCA: Principal Component Analysis*). El método PCA tiene en cuenta el problema de la multicolinealidad e intenta reducirla a través de la utilización de combinaciones lineales de los features existentes, pero puesto que los resultados obtenidos fueron inferiores a aquellos con los que ya contaba se decidió realizar primero una selección de features.

Se pusieron en práctica 3 métodos de selección de features:

- *Forward selection*
- *Backward elimination*
- *Bi-directional Elimination (Stepwise)*

Y tras correrlos todos tanto para Decision Tree como para Random Forest, se optó por el método de Backward elimination, por ser el que mejores resultados arrojaba para el conjunto de datos de trabajo.

Luego de un nuevo e infructuoso intento de reducción de dimensionalidad con PCA, sólo restaba la etapa de ajustes y mejoras, que incluyó *Tradeoff Bias-Variance, Cross Validation y modelos de ensamble.*

El error debido al Bias de un modelo es la diferencia entre el valor esperado del estimador (es decir, la predicción media del modelo) y el valor real. Cuando se dice que un modelo tiene un bias muy alto quiere decir que el modelo es muy simple y no se ha ajustado a los datos de entrenamiento, por lo que produce un error alto en todas las muestras: entrenamiento, validación y test. Por el contrario, los modelos demasiado complejos se acomodan demasiado bien a los datos de entrenamiento, y justamente por eso pequeños cambios en el dataset conlleva a grandes cambios en el output, no generando buenas predicciones sobre el conjunto de prueba. Estos modelos de alta varianza están asociados generalmente al overfitting. Lo óptimo que queremos obtener es un modelo robusto, es decir un modelo que tenga poco bias y poca varianza. El problema es que un modelo con bias es un modelo sencillo, mientras que uno con varianza es un modelo complejo, y es imposible que un modelo sea sencillo y complejo a la vez. De ahí surge el tradeoff.

Para intentar tener mayor precisión sobre si estamos ante un caso de under u overfitting lo que podemos hacer es, luego de ajustar el modelo, realizar las predicciones sobre el

conjunto prueba y compararlas con las predicciones obtenidas sobre el conjunto entrenamiento. En cierta medida es esperable que los resultados obtenidos para el subconjunto Train sean superiores a los logrados con el subconjunto Test, puesto que los datos de Train son justamente los que se usaron para calibrar el modelo, pero lo que no queremos encontrar son resultados muy disímiles, con valores muy cercanos a 1 para Train, y muy lejanos para el subconjunto Test, porque sería evidencia de un modelo demasiado bien ajustado a los datos de entrenamiento, pero poco flexible para datos nuevos.

Una alternativa para intentar manejar esta situación consiste en la validación cruzada, que difiere de la validación simple en que ya no se hace una única partición del conjunto de datos en train y test para entrenar el modelo sino que esta partición se hace múltiples veces y todas se utilizan para ir mejorando el ajuste. En el límite *LeaveOneOut cross-validation* (LOOT) -que fue la utilizada- realiza n-1 particiones en train-test (donde n es el número de registros con los que cuento), logrando así entrenar al modelo con la mayor cantidad de datos disponibles.

Por último, nos quedaba probar refinar la solución obtenida a través de los modelos de ensamble, ya sea de bagging o de boosting. Los algoritmos de bagging entrenan los modelos en paralelo y deciden por "votación", por lo que al haber escogido Random Forest como algoritmo de aprendizaje supervisado ya estuvimos haciendo uso de este tipo de modelos de ensamble (RandomForest es el algoritmo típico de del tipo bagging).

El Boosting en cambio, en lugar de entrenar los modelos en paralelo, entrena los modelos en forma secuencial. De esta forma el rendimiento general puede ser mejorado haciendo que un modelo posterior le dé más importancia a los errores cometidos por los modelos previos. El algoritmo utilizado fue el de AdaBoost y los resultados obtenidos (junto a los obtenidos por Cross Validation) se presentan y se comparan en la siguiente sección.

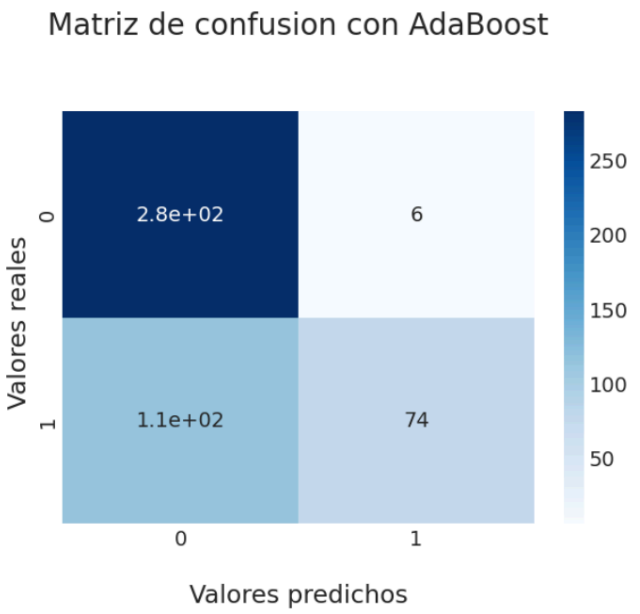
7.- Resultados y conclusiones:

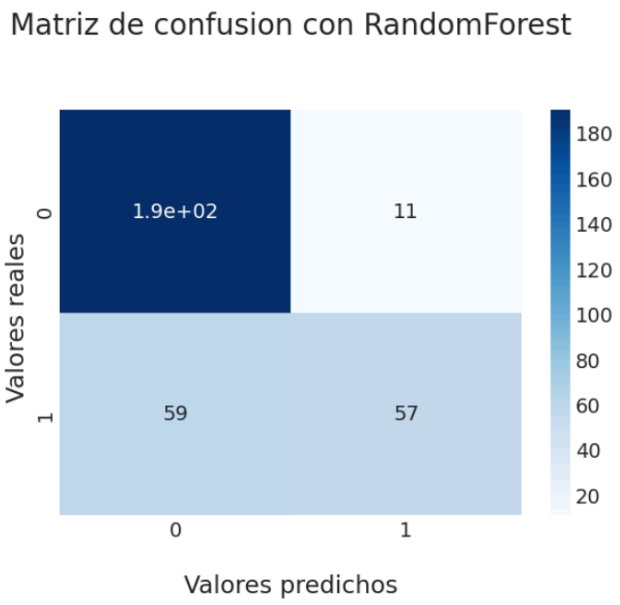
Se muestran a continuación las 3 mejores soluciones obtenidas entre las exploradas.

	Resultados obtenidos utilizando **RandomForest**			Resultados obtenidos utilizando **CV-LOOT**			Resultados obtenidos utilizando **AdaBoost**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
0.0 (No infractor)	0.76	0.95	0.84	0.74	0.97	0.84	0.71	0.98	0.82
1.0 (infractor)	0.84	0.49	0.62	0.87	0.41	0.55	0.93	0.39	0.55
Accuracy			0.78			0.76			0.75
Macro avg	0.80	0.72	0.73	0.80	0.69	0.69	0.82	0.69	0.69
Weighted avg	0.79	0.78	0.76	0.79	0.76	0.73	0.80	0.75	0.72

La diferencia entre ellas está en que a medida que “me voy poniendo más estricto” a la hora de determinar quienes cometerán una infracción, empiezo a cometer cada vez más errores al categorizar de infractores a usuarios que no lo son (error del tipo I: falsos positivos). Si bien este error no representa un costo "material", continuos errores de este tipo resultarán en inconvenientes repetidos para usuarios cumplidores, afectando la credibilidad en el sistema y generando un descontento generalizado. Por tanto, a costa de perder algunas bicis, considero que este es el error más importante a minimizar.

A mi entender todo esto se visualiza más fácil a través de las matrices de confusión:

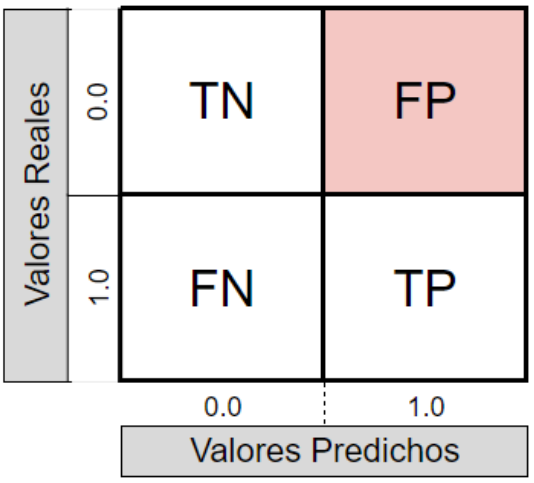




Y considerando:

- True Positive (TP) aquel usuario que el modelo predice que incumplirá y no devolverá la bicicleta a tiempo y efectivamente así sucede
- True Negative (TN) aquel usuario que el modelo predice que NO incumplirá y por tanto devolverá la bicicleta a tiempo y efectivamente así sucede
- False Negative (FN) son aquellos casos en los que el sistema determina un negativo (es decir que predice que NO se incumplirá), cuando en la práctica resulta en el robo de la bicicleta -con el correspondiente costo de reponerla-
- False Positive (FP) son los casos opuestos, en los que el sistema predice que el usuario cometerá una infracción y por lo tanto procede a bloquearlo, pero que en la práctica se comprueba que se hizo un uso legítimo de la bicicleta.

En este caso la matriz quedaría como se muestra a continuación:



Como decía anteriormente, siendo que se trata de un servicio que brinda el gobierno, considero que su principal objetivo va a ser no incomodar a los ciudadanos, es decir que el error más grave sería que un usuario que iba a devolver la bici en tiempo y forma sea bloqueado por el sistema por predecir que cometerá una infracción: Un False Positive.

Con esto en mente, además de la efectividad a la hora de detectar infracciones (True Positive), no debemos perder de vista la especificidad:

$$\text{Specifity} = \frac{\text{True Negative}}{\text{True Negative + False Positive}}$$

Vemos que utilizando AdaBoost, la especificidad es de casi el 98%, ya que solo 6 usuarios de los 288 no_infractores terminaron resultado bloqueados. A cambio de este 2% de usuarios con inconvenientes, se consigue evitar casi el 40% de los robos de bicicleta -lo que queda reflejado en el recall del informe de Clasificación de la página 11 ó que puede calcularse mediante la siguiente expresión:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

En el otro extremo tenemos los resultados conseguidos a través del RandomForest. Siguiendo esos lineamientos conseguimos predecir casi el 50% de los robos de bicicleta, pero a cambio la cantidad de usuarios "buenos" que registrarán problemas pasa del 2 al 5,5%.

ó siempre está la alternativa de ir por la opción que queda en el medio de ambas, que en este caso es CV-LOOT, que predice el 41% de los robos a cambio de que sea el 3% de usuarios "buenos" los que tengan problemas.

Incomodar a más usuarios es un costo político que se está dispuesto a pagar? Cuesta más eso ó la reposición de las bicicletas? Respondiendo a estas preguntas es que podremos elegir uno u otro modelo.

En mi opinión, hay que prestar un servicio que le brinde a los usuarios la menor cantidad de problemas posibles. Duplicando la cantidad de usuarios con problemas solo rescatamos un 10% más de bicicletas. Yo me quedaría con las predicciones que realiza AdaBoost: previniendo el 40% de los robos sin afectar prácticamente a nadie.