Comencé el trabajo que originó este post tratando de demostrar, con un ejemplo simple, la importancia de disponer de una capa de monitoreo del ***Drifteo*** de los datos. En el camino entendí que también se podía abordar otro tema relevante: el **escalado** de los datos y su importancia.

Cabe aclarar que el caso abordado es extremadamente simple y, sin lugar a dudas, van a existir variantes mucho más complejas. El propósito es comenzar discutir aquellos temas que son **poco frecuentes** en el mundo del Machine Learning. Por ello, tanto el drift como el escalado serán el centro de la escena de esta publicación.

Estan disponibles los notebooks de cada caso que se expondrá al final de esta publicación.

Entiéndase *Drift* como los cambios que ocurren sobre la distribución de una variable específica. En el momento que desarrollamos un modelo, fijamos las condiciones a los parámetros de la distribución en ese momento. Que estos cambien puede representar un problema.

Respecto al *Escalado* se puede decir que es una transformación sobre una variable. Tiene como objetivo poder llevarla a un rango comparable con cualquier otra variable (que tome valores entre 0 y 1 o que tenga los limites de una distribución normal, etc).

**El Caso**

**Datos**

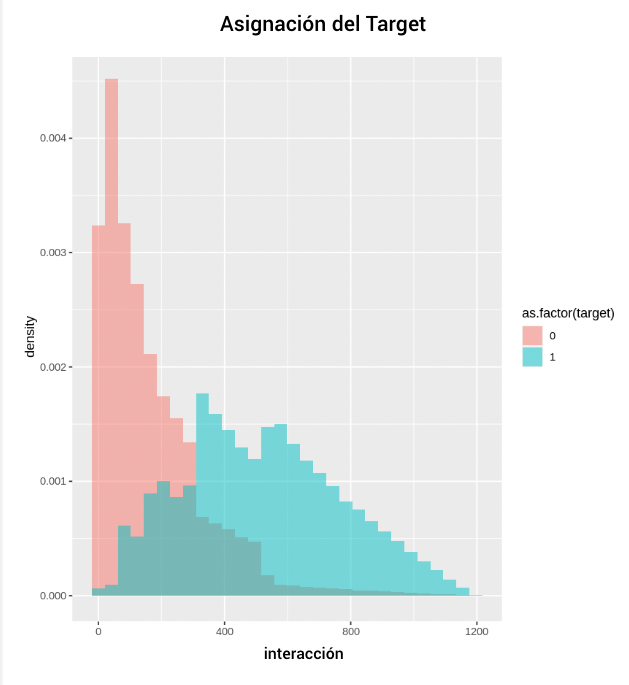
Se generó un dataset que contiene 3 variables.

La primera y la segunda son generadas al azar, basadas en una distribución uniforme. El cambio, entre una y otra, es en el rango posible de valores que puede tomar cada una.

La primera puede adoptar valores de 0 a 4, mientras que la segunda lo puede hacer entre 1 y 300.

¿Es la mejor forma de asignación al azar de una variable? Posiblemente no (podría haberse usado alguna distribución parecida a las de las variables de ingresos, por ejemplo).

La tercer variable representa el Target. El mismo fue asignado de manera "injusta" cargando más de *"unos"* a los valores más altos del producto entre la primer y segunda variable (denominado como interacción). Y más de *"ceros"* a los valores bajos.



**Alteración**

Para poder esbozar algunas conclusiones generales se *"alteró"* la fracción seleccionada para el **Test** del algoritmo. Conformando con ella un **tercer dataset** de trabajo.

El proceso constó en tomar entre un 20% y un 40% del valor de variable original de la primer variable y para la segunda se optó por hacerla crecer entre un 150% y 190%.

Para tratar de emular la realidad, se aisló un 5% de los casos y no se alteró su valor original.

El **sentido** de generar una alteración, es replicar un fenómeno que en Argentina suele afectar bastante a los modelos y que se denomina **inflación**. Y es un efecto que denominaré como variables que "se mueven solas", ya que no hay una acción que hagamos que las altre, sino que simplemente el contexto o paso del tiempo las afecta.

Entonces queda fuera de análisis un corrimiento en la distribución provocado por un cambio de comportamiento y muchos otros posibles problemas sobre los datos.

**El Algoritmo**

Imaginaran que este punto es lo de menos. La disponibilidad de dos variables independientes y una dependiente no nos dejan muchas posibilidades.

Se usó un ***XGBoost*** con pocas iteraciones y, obviamente, con un crecimiento en sus ramas altamente limitado.

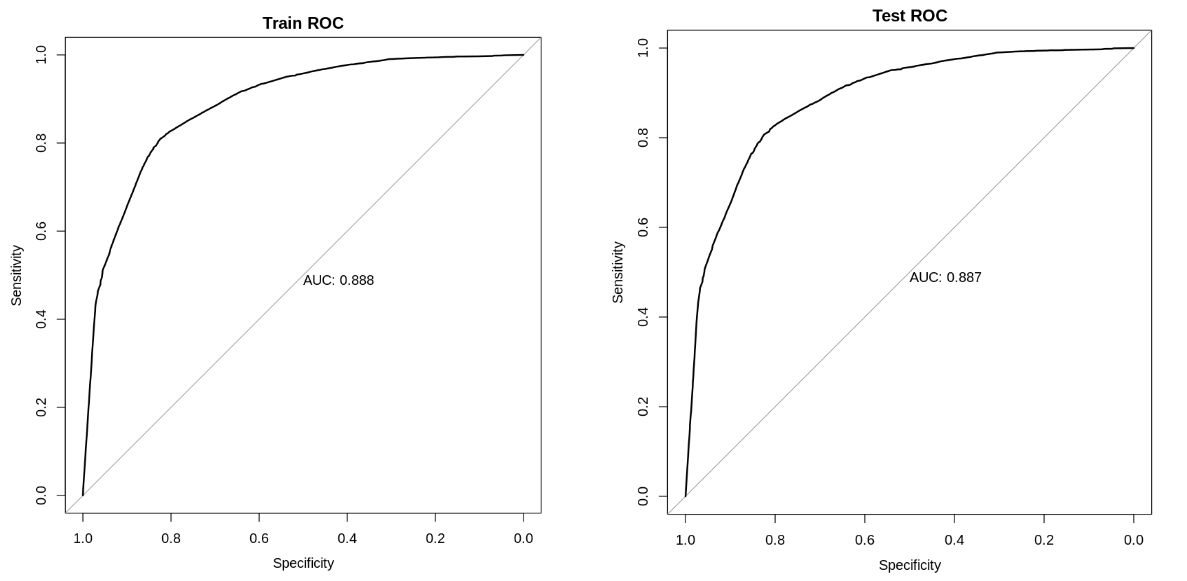
**Hallazgos**

**El proceso**

Se dividió el dataset original en:

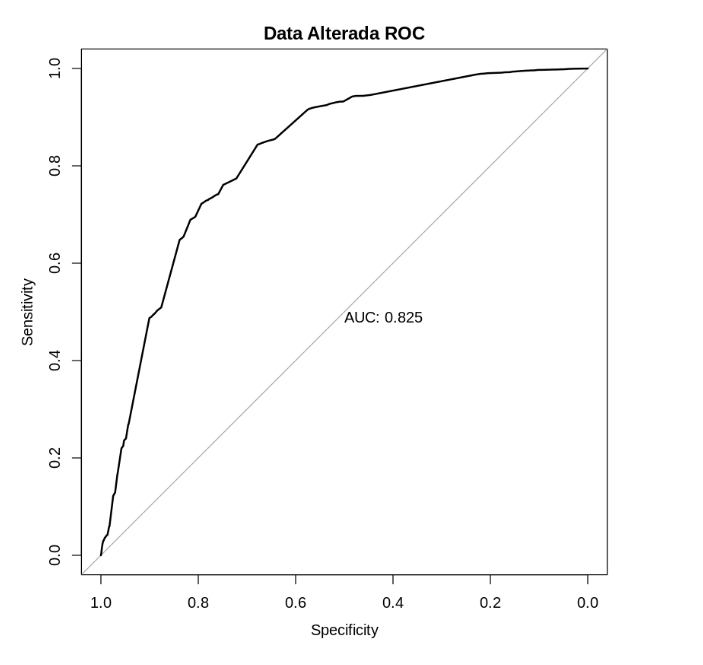
* 70% para Train
* 30% para Test
* El 30% de Test, además, se **alteró** (según lo ya mencionado)

Se entrenó el modelo obteniendo, como era de esperarse, una buena performance en Train y en Test:



**Hallazgos [Parte 1]**

Ahora, el objetivo radica en ver como la ***alteración*** afecta la performance. Para ello nos sirve el tercer dataset, que además tiene los mismos "unos y ceros" que el set de Test, en pos de generar igualdad de condiciones en la comparación. Revisemos su curva y su área:



Es evidente que los corrimientos en las distribuciones afectan de manera significativa el desempeño del modelo. Pero cabe destacar, que la métrica sigue estando en valores considerablemente altos.

**Primer Aprendizaje**

Para profundizar el efecto del Drift o intentar detectarlo se hicieron algunas pruebas adicionales.

La primera trataba de responder la pregunta "¿es posible con un test estadístico alertar el Drift?". La idea de este punto es PREVENIR y buscar una forma que simplifique o nos permita automatizar alguna acción correctiva o warning a los afectados por una mala predicción. Básicamente porque la perfomance sólo la podemos medir cuando los eventos ocurrieron, y no nos queda alternativa que pensar en acciones anteriores a la ocurrencia del evento (ya que sería tarde).

Lo primero que se me vino a la mente fue **Kolmogorov Smirnov**, un test estadístico de bondad de ajuste. Se puede aplicara las variables dependientes de Test y Test Alterado, y verificar si hay diferencias. En efecto, se obtuvo un valor considerablemente alto del estadístico (que **marca que las distribuciones cambiaron**) para cada una. Por lo que constituye un método efectivo para hacer la detección.

**Segundo Aprendizaje**

Seguimos sin saber cuán fuerte fue el efecto del cambio, porque si nos guiamos por el AUC (área bajo la curva) parece que tenemos un modelo que empeoró, pero que es bueno.

Pero, ¿realmente es bueno? . Para responder esa pregunta trabajaré sobre un supuesto, imaginemos que el modelo es usado en un proceso con un ***cut-off* fijo**. Es decir, alguien analizó los resultados y dijo que los que "me interesan" son aquellos que tienen más de 0.75 de probabilidad predicha.

Ese punto de corte nos devolvía 4500 usuarios en el set de Test (original), con una tasa de acierto del 90% (es decir 9 de cada 10 eran un "uno"). Pero si revisamos en la distribución alterada nos daremos cuenta que ahora nos devuelve 658 usuarios, con una tasa de acierto del 75%.

No solo empeoró la capacidad predictiva, sino que me da muchos menos usuarios/clientes candidatos.

**Tercer Aprendizaje**

Leyendo el punto anterior seguramente alguno pensó "nunca mas voy a usar un cut-off", y dijo "voy a hacer una acción para el 10% más propenso a ser uno" (el famoso primer decil). Bueno, malas noticias, en Test el primer 10% tiene una tasa de acierto del 90%, pero en Test Alterado esa **tasa cae a 75%**. Son 15 puntos de descenso.

Esta también es una situación alarmante.

**Conclusión Parcial**

Es importante hacer un checkeo previo. El KS o cualquier otro método menos sofisticado (control de medias, máximos o mínimos, etc) pueden ser muy útiles para verificar la salud de las variables y en consecuencia la salud del modelo.

Por otro lado, las variables que se ***"mueven solas"*** siempre van a representar un problema en nuestros modelos. Por ello hay que **MONITOREARLAS** para reaccionar de manera temprana. Para darte una idea se me ocurren la edad, la cantidad de dias que pasan desde, variables que contengan saldos en una moneda volátil, etc, todas ellas son variables que sin hacer nada van a crecer y no representan un cambio de comportamiento en el usuario/cliente/fila del dataset.

**Escalado**

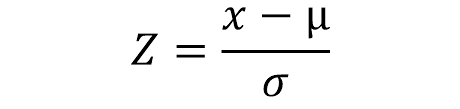
En este proceso de muchos aprendizajes me auto-impuse el desafío de ver si además de prevenir un "mal output", podía intervenir de manera activa y generar un modelo que no reaccione tanto a las variables que se *"mueven solas".* ¿Existe eso? Bueno la respuesta siempre va a ser "depende" o "probalo". Pero sobre este caso ensayé algunas pruebas y te cuento los hallazgos.

**El proceso**

Tanto Train, como Test, como Test Alterado fueron pasados cada uno en individual por un proceso de ***escalado***. Esto consta en alterar los valores, transformarlos, de forma tal de llevarlos a un rango de valores un poco más controlables.

En definitiva el aporte más valorable tiene que ver con que se transforma el sentido de la variable y deja de ser un "cuanto gastó el cliente" para pasar a hacer "en qué posición se ubica entre los que gastan, ¿gasta mucho o poco en mi negocio?". Ese pequeño cambio tiene sus virtudes y tiene un bajísimo costo de implementación.

Este es el cálculo sobre el que se basa uno de los metodos:

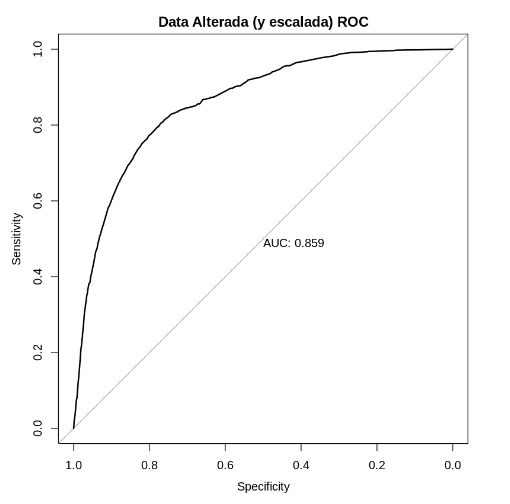


Simplemente consta de pre-calcular la media y el desvío standard de la variable, e ir visitando cada valor particular, restarle la media y dividirlo por el desvío.

Ese simple proceso devuelve resultados muy interesantes.

**Hallazgos [Parte 2]**

Si revisamos la performance de Test Alterado, sin haber hecho más que calcular medias y desvíos y aplicar la fórmula anterior, el resultado es el siguiente:



Es increíble! tan solo ese cambio **cuido nuestra AUC**. Pero como ya revisamos más arriba, esto no previene nada (solo nos muestra una realidad que ya ocurrió).

Así que revisé el **KS para las variables**: 0,14 y 0,02 son los estadísticos para las dependientes. Son considerablemente bajos y considerablemente más bajos que el de los datos sin tratar. Lo que quiere decir que la alteración y las diferencias entre las distribuciones que ocurren, **generan efectos despreciables cuando tratamos la variable con escalado**.

Eso representa el plano PREVENTIVO, pero para ser justo revisaré lo mismo que en el caso sin el escalado.

Validemos el **cut-off**. Definido un valor de corte (0,75 por ejemplo) para tomar casos de interés en base al output vemos lo siguiente:

* En Test (original) tenemos 4500 casos que entran, con una tasa de acierto del 90%.
* En Test Alterado tenemos 2400 casos que entran, con una tasa de acierto del 85%

Solo perdimos 5 puntos en aciertos (contra 15 del caso sin escalado) y caímos bastante en casos, pero no tan fuertemente como en el caso sin escalado (de 4500 a 658).

Por último, si accionamos sobre el primer 10% en base al output:

* Test (original) tiene una tasa de aciertos del 90%
* Test Alterado tiene una tasa de aciertos del 86%

Recordemos que en el caso sin escalado la caída era de 15 puntos, mientras que en el caso escalado la caída es de 4 puntos.

También el escalado soluciona un gran problema, la distribución del output. Revisando de manera rápida con un promedio se observa que:

* La alteración en el caso sin tratamiento provoca un drift de la media del score desde 0.37 a 0.19. Esto explica en gran medida porque no funciona el cut-off.
* La alteración en el caso con tratamiento provoca un drift de la media del score desde 0.37 a 0.36. Aunque evidentemente cambia la distribución ya que el cut-off muestra una caída, en términos muy generales la situación tiene mayor estabilidad que en el caso anterior.

**Conclusiones**

1. El control/monitoreo de drift (en este caso usamos solo KS) es esencial para poder entender el impacto en performance futura que pueden tener nuestros modelos. Es un check totalmente necesario para nuestros modelos.
2. Ese tipo de control simplifica la automatización alertarnos. Por ende, nos permite escalar en la cantidad de modelos productivos sin tener que estar controlando todos a mano.
3. Por otro lado, el escalado no es una solución "mágica". Si existiera un movimiento en el valor de variable, lo que va a prevenir es que los cortes seleccionados en los Árboles de *XGBoost* no caduquen y sigan sirviendo por un tiempo más prolongado.
4. En relación con lo anterior, el escalado no es una solución de largo plazo. Esos modelos en algún momento van a tener que ser re-entrenados. Pero sí le damos mas vida util.
5. El escalado soluciona los problemas para las variables que se "mueven solas", no nos da cobertura para los casos en los que hay un cambio de comportamiento voluntario y masivo de los usuarios/clientes.
6. El escalado es un camino costo efectivo, ya que es simple de aplicar y cuida nuestros artefactos.