FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

"Predicción de conversiones"

[75.06 / 95.58] Organización de Datos

Segundo Cuatrimestre de 2018

Alumno	Padrón	E-mail
Soro, Lucas Gustavo	95665	lugusor@gmail.com
~~Montes, Marcelo~~	~~81397~~	~~mdmontes@gmail.com~~
Risaro, Lucas	94335	lucasrisaro@gmail.com
~~Núñez Leyes, Javier Damián~~	~~94455~~	~~javier.nunezleyes@gmail.com~~

Repositorio:
https://github.com/Blaz77/DatosReloaded

Indice

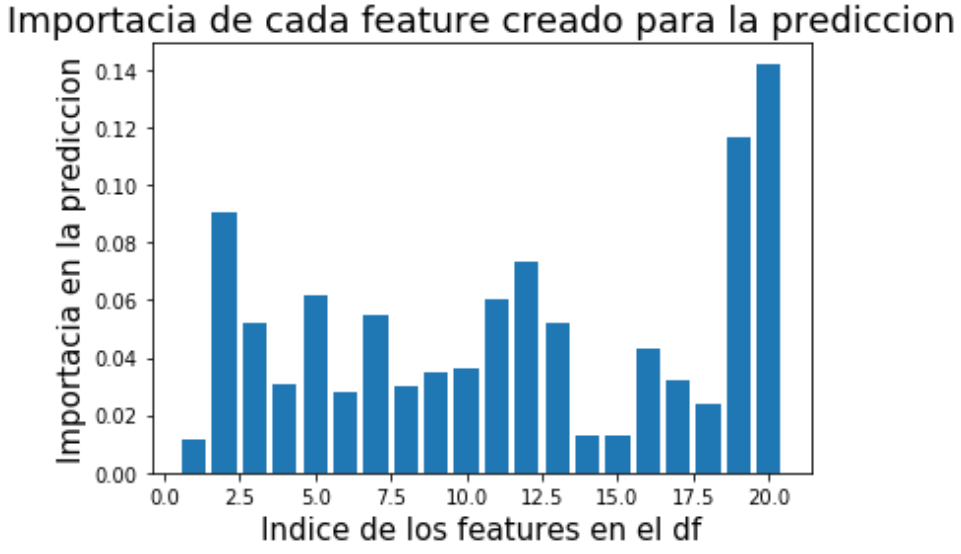
1. Creación de features:

Partimos de los resultados del análisis exploratorio realizado en el Trabajo Práctico 1 para comenzar con la creación de los features que nos permitieran realizar las predicciones.
Nos basamos en los datos relacionados directamente con las conversiones obtenidos en el análisis previo, como la cantidad de conversiones y de visitas al sitio de cada usuario, número de veces que cada uno vio los modelos más vendidos, analizamos los días y el horario donde se producían las mayores conversiones y analizamos la actividad de cada usuario en ese lapso de tiempo, entre otros más. Luego fuimos ampliando los datos de exploración y creando nuevos features. Creamos alrededor de 20 features.
Todo esto para intentar dilucidar un patrón de comportamiento de los usuarios que realizaron conversiones y los que no y en base a eso poder observar el comportamiento de los usuarios a predecir, compararlo con los datos que teníamos e intentar decidir si realizarian una conversión en el futuro o no.
A efectos prácticos, como solo teníamos datos de 3 meses para analizar, en los features donde medimos el tiempo desde el último “evento” pusimos como límite 180 días.

2. Análisis de los features:

Para esta tarea utilizamos Random Forest.
Random Forest tiene la particularidad de que el resultado de su herramienta para medir la importancia de los features a la hora de realizar una predicción se puede utilizar de forma general para cualquier algoritmo de clasificación. Es decir que los features que RF indica que son los más influyentes a la hora de realizar su predicción coinciden con los features que cualquier otro algoritmo de clasificación consideraria más importantes.

A continuación se muestran los resultados obtenidos:



Indice de los features:

1: cantidad de conversiones
2: cantidad de veces que vio un producto
3: cantidad de visitas a la página
4: cantidad de checkouts
5: visitaron el sitio entre las 13hs y 24hs
6: cantidad de vistas del J5
7: cantidad de vistas del iphone 5s
8: cantidad de vistas del samsung s6 flat
10: cantidad de vistas del iphone 6
11: cantidad de vistas del iphone 6s
12: cantidad de visitas entre semana
13: cantidad de veces que vio un producto el último mes
14: cantidad de veces que vio un producto la última semana
15: cantidad de checkouts en la última semana
16: cantidad de conversiones por visita
17: cantidad de returnings en el último mes
18: cantidad de returnings en la última semana
19: cantidad de días desde la última conversión
20: cantidad de días desde la última vista a producto
21: cantidad de días desde el último checkout

En base a estos resultados optimizamos nuestro algoritmo de predicción y nos concentramos en mejorar los features que más influencia tenían en las predicciones.

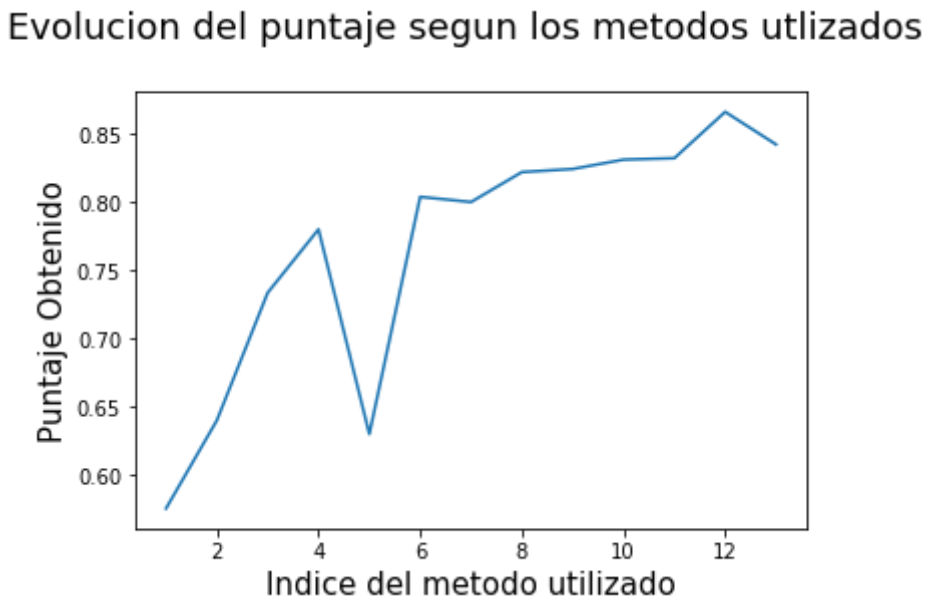
Al comienzo nuestros features se basaban principalmente en la cantidad de veces que cada usuario veía uno de los productos más vendidos, la cantidad previa de conversiones habían realizado, las veces que habían visitado la página en el último mes y semana.
Estos features daban buen resultado pero no era suficiente.
Usamos RF y nos dimos cuenta que el algoritmo le daba más importancia a la cantidad de veces que se veía un producto en general que a un producto en concreto y la cantidad de conversiones previas y nos concentramos en analizar eso.
De este análisis surgieron los últimos 3 features que fueron los que le dieron un gran impulso al resultado de las predicciones llegando a, hasta ahora, nuestro mejor resultado.

3. Algoritmos utilizados:

A lo largo del trabajo probamos 4 algoritmos para clasificar: KNN, Random Forest, Perceptron y XGBoost. Además de otros algoritmos para optimizar sus hiperparametros como Grid-Search y Cross Validation.
A lo largo del trabajo también probamos utilizar Factorización de Matrices para intentar predecir los datos que no teníamos en el data frame que nos fue proporcionado. Sin embargo los resultados que se obtuvieron utilizándolo no fueron buenos y decidimos no utilizarlo.

KNN fue el primer algoritmo que probamos y fue el que peor resultados devolvió lo cual era esperable ya que no era el algoritmo óptimo para esta tarea, en parte porque no detecta automáticamente el peso de cada feature en el modelo, y porque no se divide en etapa de entrenamiento y de predicción, con lo que hacer cada predicción toma bastante tiempo.
Luego pasamos a Random Forest.
Los resultados de RF superaron ampliamente a KNN y fue el que empezamos a usar de forma seria para encarar el trabajo. Lo combinamos con Cross Validation y Grid-Search para mejorar su performance, sin embargo aún optimizando sus hiperparametros los resultados que obteníamos no eran suficientemente buenos, en ese momento fue cuando pasamos a usar XGBoost. Este último fue el que mejor resultados nos dio. Sin embargo, luego lo seguimos usando para medir la importancia de los features, ya que xgboost no es conveniente para esa tarea.

A continuación se muestra de forma general la evolución del puntaje según los métodos que utilizamos:



Indice de métodos:

1: Baseline. KNN con K=4 (Tomado con pruebas simples).
2: KNN con K=11 (Tomado iterando K impares hasta 30 y haciendo cross-validation). 3: Random Forest con 33 árboles (Sin factorizar matrices).
4: Random Forest con 50 árboles para clasificar y 500 para predecir (sin factorizar).
5: Random Forest con 50 árboles para clasificar y 500 para predecir (factorizando la matriz).
6: Xgboost estándar sin modificar los atributos básicos del algoritmo.
7: Xgboost aplicando reg:logistic para clasificar.
8: Xgboost aplicando binary:logistic con features de visitas del último mes y semana y checkouts de la última semana.
9: Xgboost aplicando binary:logistic agregando feature de tasa de conversiones / visitas.

10: Xgboost aplicando binary:logistic modificando parámetros según cross-validation.
11: Xgboost con cv aplicando binary:logistic, agregó de features de cantidad de returnings a la página.
12: Xgboost con cv aplicando binary:logistic, agregó features de cantidad de días desde la última conversión, checkout y visita.
13: Perceptron

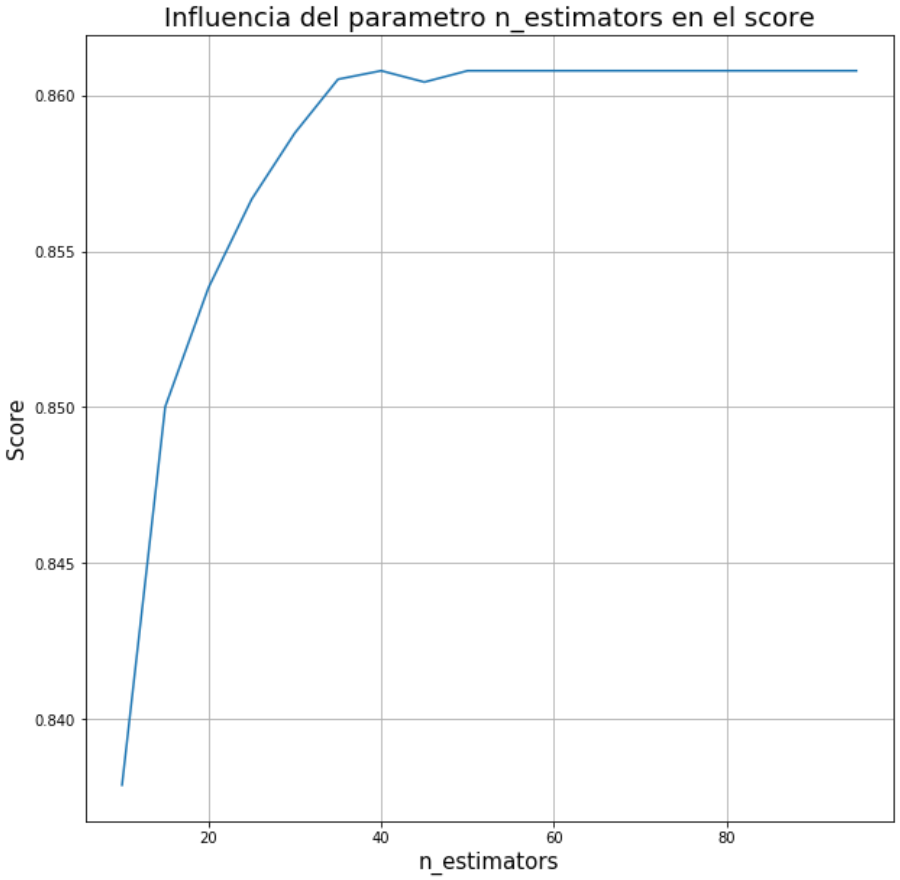
Aunque no están detalladas todas las variantes que utilizamos se puede ver como algunas de las variantes de cada algoritmo influyó en el puntaje de las predicciones finales. En especial se puede observar lo que comentamos anteriormente respecto al uso de la Factorización de Matrices, al tratar de predecir los datos NAN en la matriz de features utilizando el puntaje cae drásticamente.

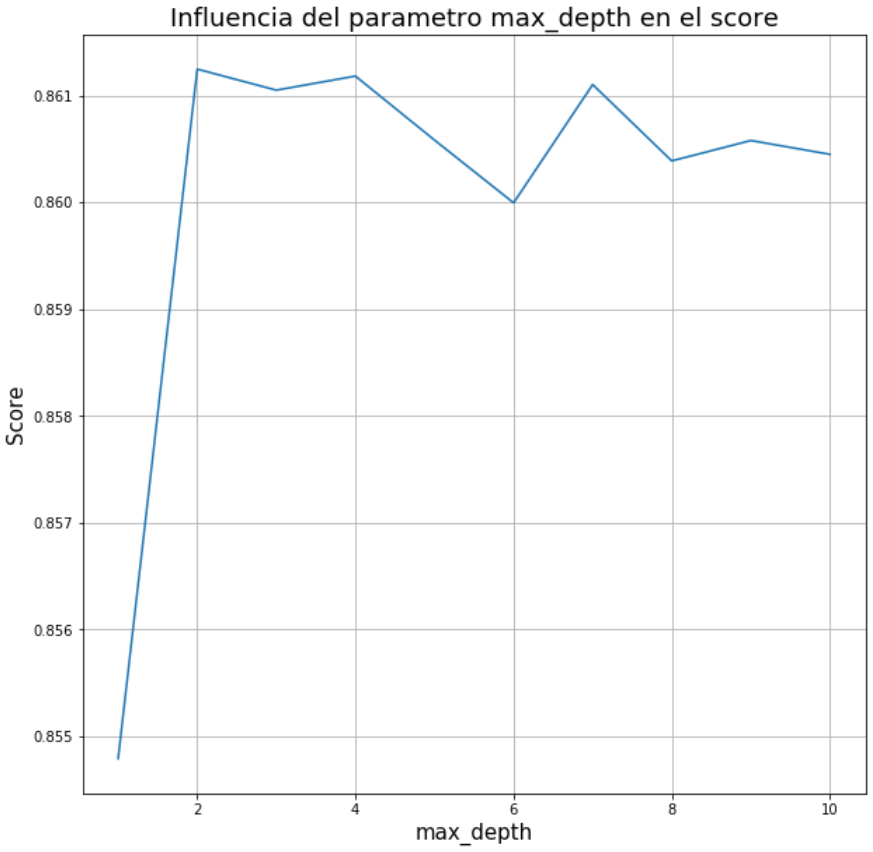
4. Algoritmo final Utilizado:

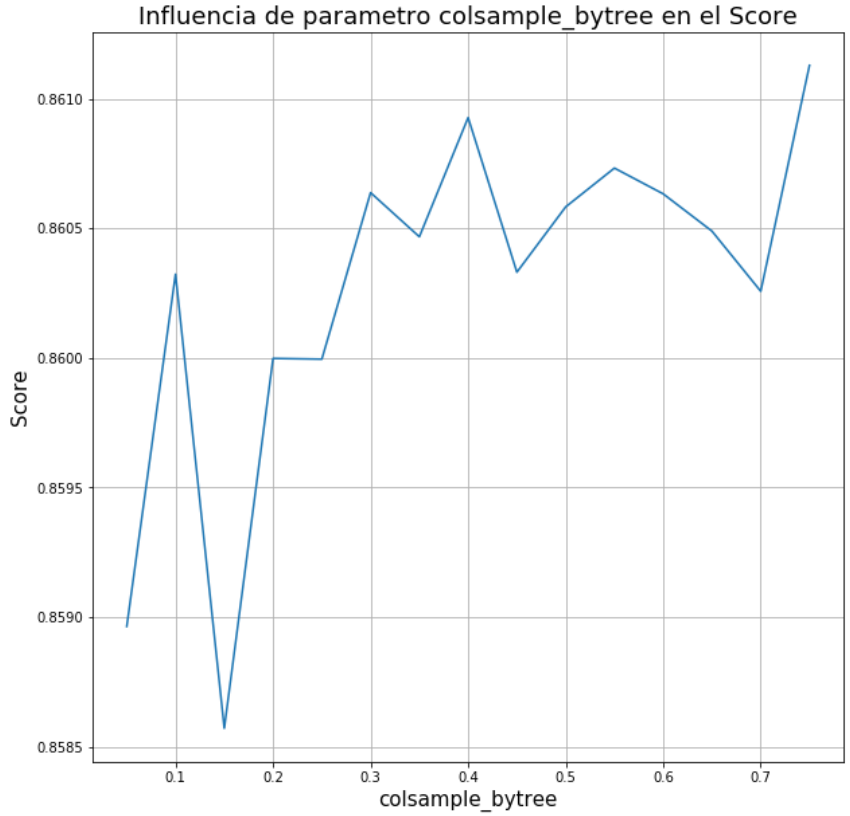
Nuestro algoritmo final fue una combinación de XGBoost para realizar las predicciones, Cross Validation para la optimización de los hiperparametros y Random Forest para la optimización de los features.

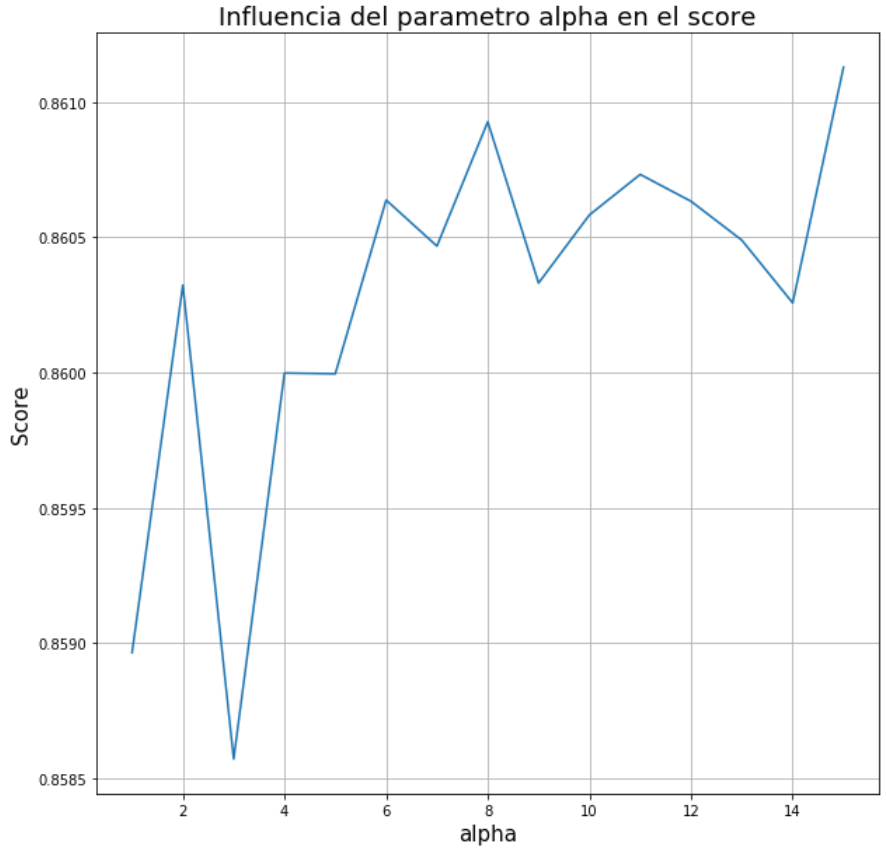
Primero ejecutamos RF para optimizar parámetros, luego CV para optimizar los hiperparametros ajustándolos a los nuevos features y finalmente ejecutamos la función XGBRegressor de XGBoost con esos parámetros para la predicción

Ya vimos un ejemplo del resultado del análisis de features de RF, a continuación mostramos un ejemplo de los resultados de CV al analizar los parámetros de XGBoost:


Influencia del parametro n_estimators en el score
Score
n_estimators
0.860
0.855
0.850
0.845
0.840
20
40
60
80


Influencia del parametro max_depth en el score
Score
max_depth
0.861
0.860
0.859
0.858
0.857
0.856
0.855
2
4
6
8
10


Influencia de parametro colsample_bytree en el Score
Score
colsample_bytree
0.8610
0.8605
0.8600
0.8595
0.8590
0.8585
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Influencia del parametro alpha en el score
Score
alpha
0.8610
0.8605
0.8600
0.8595
0.8590
0.8585
2
4
6
8
10
12
14

n_estimators: Cantidad de estimadores que va a usar XGBoost para realizar las predicciones.
max_depth: Indica la máxima profundidad de los árboles. Hay que tener cuidado con el valor que se usa ya que un valor muy alto hará que el algoritmo overfitee.
colsample_bytree: Indica la proporción de muestras de cada columna que usará el algoritmo para crear cada árbol.
alpha: regula los pesos de cada feature. Un valor alto hace que el algoritmo sea más conservador.

Utilizando estos datos optimizamos nuestro algoritmo y features y llegamos al mejor resultado hasta ahora que es de 0.86712

5. Conclusión:

A partir los resultados del trabajo pudimos obtener un acercamiento a lo que es el comportamiento de un potencial usuario interesado en realizar una conversión en las próximas semanas y el de uno que no lo hará.

Un usuario que hace mucho tiempo que no compra un dispositivo es más proclive a comprar uno en el corto o mediano plazo que uno que compró uno hace relativamente poco tiempo, esta idea se basa en la vida útil de los dispositivos, un movil con mas tiempo de uso es más probable que deje de funcionar y tenga que ser reemplazado que uno con poco tiempo de uso .
Luego un usuario interesado en comprar recorre mucho la pagina, tiene cantidad de visitas a la página y vistas a productos alto, en comparación a uno que no lo hará.
Del mismo modo un usuario que casi está decidido a comprar comienza a centrar su búsqueda en unos pocos modelos en concreto y deja de observar productos al azar.
El lugar de residencia también influye. Al ser una tienda brasileña si un usuario reside en ese país es más probable que compre allí algo que uno que vive en otro país.

Basados en nuestros resultados y contrariamente a lo que pensábamos al comienzo del trabajo, es más influyente a la hora de realizar una predicción la cantidad de visitas a productos en general que las visitas de cada usuario a los productos más vendidos. Del mismo modo es más importante la cantidad de tiempo desde la última conversión de un usuario que la cantidad de veces que haya comprado algo en la página.

En concreto estas observaciones sólo serán válidas para un usuario de Trocafone ya que solo trabajamos sobre sus datos. No podríamos hacer una observación detallada del comportamiento en concreto de un potencial comprador de tiendas online en general ya que para ello necesitaríamos mucho más datos de usuarios al azar. Estudios generales del comportamiento se basan en datos de búsquedas de internet de los usuarios en distintos motores de búsqueda, datos de distintas tiendas online, datos de avisos publicitarios clickeados en diferentes portales, datos económicos de diferentes regiones, datos de fechas festivas, etc.

Sin embargo creemos que con lo hecho pudimos acercarnos bastante a lo que sería un perfil aproximado de un potencial comprador.