75.06/95.58 Organización de Datos - 2C 2018

Trabajo Práctico 2

Competencia de Machine Learning

Grupo 17: "Barplotters"

Integrantes:

- Andrés Pablo Silvestri: 85881 (silvestri.andres@gmail.com)
- Axel Brian Erlich: 94847 (axel.erlich@yahoo.com.ar)
- Juan Manuel González: 79979 (juanmg0511@gmail.com)

Fecha de entrega: 03/12/18

Link a repositorio de GitHub: https://github.com/silvahlaravel/Organizacion_Datos_2C/tree/master/TP2/Entrega
* Vale aclarar que la URL es sobre la carpeta de entrega, pero en el repo en general está subido todo lo trabajado.

Contenido

1 - Introducción

En este trabajo se describe trabajo realizado para poder llegar a los distintos resultados que se fueron entregando a la competencia de kaggle trocafone.

La competencia consistía en predecir la probabilidad de que un usuario realice una conversión en un determinado periodo de tiempo, teniendo datos del comportamiento pasado del mismo en el sitio de trocafone, además de datos para otros usuarios para los cuales se conoce si realizaron o no una conversión.

A partir de estos datos, se fueron extrayendo distintos features y probando distintos algoritmos de machine learning para obtener las mejores predicciones posibles, el proceso para obtener este resultado se describirán a continuación.

Durante el trabajo, se utilizó el lenguaje de programación Python y algunas librerías extras como sklearn y pandas para facilitar el manejo de datos y no tener que implementar los algoritmos de ML.

2 - Transformación de datos y armado de features

Como primer medida se pasó a tipo 'category' todas las columnas que resultan acordes para ser categorizadas por tener una cantidad limitada de valores, lo que permite mejorar el rendimiento de las diferentes operaciones que involucren estas columnas. Además, se parseo la fecha del los eventos en tres columnas: hora, dia y mes para facilitar el manejo de los tiempos

Se describirán a continuación el armado de distintos features que se fueron probando, si bien no todos quedaron como definitivos para obtener el resultado final, fueron parte del proceso.

2.1 - Datos relacionados al tiempo

A continuación se describen los features que se armaron relacionados al tiempo de los eventos.

2.1.1 - Horas en el dia

Utilizando el dato de la hora de los eventos, se armó una división del día en madrugada, mañana, almuerzo, tarde y noche, donde en cada columna se indica con un 1 si el evento ocurrió en ese momento del día, ver el notebook para más detalles sobre los horarios que dividen cada grupo.

2.1.2 - Días de la semana

Utilizando el dato de la fecha y hora de los eventos, se armaron columnas con cada dia de la semana en la cual se indica con 1 si el evento ocurrió en ese dia. Además de una columna especial que indica si el evento ocurrió en el fin de semana, la cual toma valor 1 si el evento ocurrió en los días sábado o domingo.

2.1.3 - Meses del año

Utilizando el dato de la fecha y hora de los eventos, se armaron columnas con cada mes del año en la cual se indica con un 1 si el evento ocurrió en ese mes.

2.2 - Sistema operativo del dispositivo que originó el evento

Utilizando el dato de los sistemas operativos del usuario, se armaron las columnas correspondientes a los principales y donde se indica con un 1 en la columna correspondiente al sistema operativo desde el cual se genero el evento.

2.3 - Marcas de celulares

Utilizando el dato de la marca y modelo del celular a la venta, se armaron las columnas correspondientes a las principales marcas de celulares, donde se indica con un 1 en la columna correspondiente a la marca del celular que fue visitado en el evento.

2.4 - Datos relacionados a la locación del usuario

2.4.1 - País

Se tomaron los 5 países con mayor cantidad de eventos para armar sus correspondientes columnas y el resto de los países se agruparon en una columna “Otros Países”.

2.4.2 - Ciudad

Se tomaron las 200 ciudades con mayor cantidad de eventos para armar su correspondiente columna y el resto se agruparon en una columna “Otras Ciudades”

2.5 - Navegador desde el cual se genero el evento

Se tomaron los 50 navegadores más populares con sus respectivas versiones para armar sus correspondientes columnas y el resto de los navegadores/versiones e agruparon en una columna “Otros Navegadores”

2.6 - Capacidad de almacenamiento de los dispositivos buscados

Se tomaron las capacidades de almacenamiento más típicas de los dispositivos para armar una columna numérica que toma la capacidad de almacenamiento y pasa todo a GB como medida común, mientras que para los dispositivos que no se conoce la capacidad, se seteo un valor de 0, pudiendo así cualificar la cantidad de almacenamiento para un usuario.

2.7 - Estado del dispositivo

Se tomaron todos los valores posibles para la condición de un dispositivo y se agrego una columna con cada condición, indicando con un 1 si la condición es la correspondiente a la columna. Por otro lado se tiene una columna con el puntaje que se le asigna al estado de los dispositivos, es decir un valor numérico que nos indica por ejemplo que el nuevo tiene un puntaje de 5 y el bueno un puntaje de 2 a modo de ejemplo, pudiendo así cualificar el estado para un usuario.

2.8 - Usuario Nuevo o usuario recurrente

Usando el dato new_vs_returning, se agregaron dos columnas para indicar si el usuario que generó el evento es nuevo o es un usuario que ya había ingresado al sitio.

2.9 - Features utilizados

Luego de muchas pruebas, se fueron descartando/sumando features al modelo, los features que se utilizaron para la solución final son:

- Meses de los eventos
- Cantidad de eventos de cada tipo
- Almacenamiento de los dispositivos visitados
- Condición de los dispositivos visitados
- Si el usuario era nuevo o recurrente
- Sistema operativo del dispositivo que originó el evento
- Marca de los dispositivos visitados
- País desde el cual se originó el evento

- Tipo de dispositivo que originó el evento
- Navegador que originó el evento

Además, los eventos se agruparon por usuario, ya que no todos los eventos registran todos los datos, por lo tanto al hacer el agrupamiento por usuario, se sumaron todas las columnas con los distintos features numéricos que se explicaron anteriormente, por ejemplo en el caso de los eventos, se sumaron para todos los usuarios la cantidad de 'checkouts' y ese numero formó finalmente el feature.

3 - Algoritmos Utilizados

3.1 - Random Forest

Como primer medida se decidió iniciar el trabajo para la predicción usando Random Forest, entre otros motivos podemos decir que la elección está basada en el simple hecho de que cuando un atributo es un buen predictor suś arboles van a tener mejores resultados que aquellos que usan un conjunto de atributos que no son buenos predictores, como así también sabemos que son invariantes a la escala de los atributos es decir que no necesitamos normalizarlos, lo cual entre otras cosas nos facilita la operatoria, con todo esto partimos con la idea inicial de utilizarlo para ir descubriendo qué features pueden resultar positivos y cuáles no.

Para el primer submit lo que se hizo fue tomar como base todos las columnas numéricas y agregar muchas más columnas haciendo One Hot Encoding, para obtener la siguiente información como base, los días de la semana en que el usuario fue haciendo los diferentes eventos, el horario remarcando franjas horarias que nos permitieran distinguir si el evento lo hacía a la mañana, a la tarde, a la noche, etc. Por otro lado también incluimos los meses en los que se ejecutan los eventos como así también tenemos el listado y la distinción de los eventos generales.

Todos esos features fueron los primeros que incluimos y conseguimos un primer resultado de 0.71355 con lo cual a partir de se empezó a hacer un trabajo de ir probando los diferentes features que considerábamos podían o no sumar valor agregado, estos fueron Storage y Condition que nos llevaron a obtener cierta mejora con lo cual decidimos mantenerlos. Por otro lado tuvimos dos features Modelo y Color del dispositivo , que se agregaron y que a priori suponíamos que iban a sumar, pero tanto juntos como por separado nos generaron una reducción en el puntaje con lo cual decidimos no incluirlos.

Este fue nuestro primer acercamiento a Random Forest (luego lo volveremos a usar en una combinación de algoritmos para intentar mejorar el resultado) a este punto llegamos con el siguiente score 0.72024 y empezamos a notar que a pesar de probar diferentes hiper parámetros no conseguimos una mejora relevante, por lo cual se decidió empezar a probar otros algoritmos.

3.2 - KNN

Como alternativa a los dos algoritmos basados en árboles de decisión decidimos buscar algo diferente pero aprovechando el tipo de matriz y los features que teníamos armados, para esto decidimos usar KNN (K-Nearest Neighbors) y aunque a priori en el entrenamiento obtuvimos una precisión correcta los primeros resultados de predicciones contra Kaggle nos dieron como mucho 0.54550 lo cual es bastante bajo y no teníamos muchos features para trabajar o agregar, con lo cual nos quedaba el hiper parámetro que tampoco brindaba un cambio sustancial puesto que solo teníamos la cantidad de vecinos para cambiar, poniendo uno, tres, o diez no conseguimos en definitiva un cambio que nos hiciera optar por KNN como el algoritmo principal.

3.3 - XGBoost

Tomando como base los features que ya teníamos evaluados con Random Forest decidimos aplicar XGBoost como la primer alternativa a sabiendas que ya no podíamos encontrar muchas mejoras sobre lo que se obtenía con Random Forest, es por eso que aunque sabiendo que con este otro algoritmo no siempre es el mejor al menos casi siempre da buenos resultados y muchas competencias se ganan usándolo como base

En un primer momento no le pusimos mayor importancia a los hiper parámetros (no los trabajamos más que tomando algunos a base de pruebas) y probamos simplemente la ejecución del entrenamiento y la predicción, como si de saltar de Random Forest a XGBoost se tratara, y el resultado que obtuvimos en esta primer instancia fue el siguiente 0.82989 lo que a priori es un gran avance de casi un 0.1 en comparación al algoritmo anterior.

A partir de ese momento comenzamos a trabajar tanto con los hiper parámetros que se verá más en detalle más adelante como así también buscando otros features que nos pudiesen traer una mejora sustancial, como habíamos hecho en un principio con el algoritmo anterior. Tomando parte del análisis hecho para el primer trabajo práctico decidimos emplear ciertos features como ser los de Sistemas Operativos (agrupando entre los más usados) como así también las Marcas más relevantes (también en este caso agrupando entre las más usadas) y aunque no haya sido sustancial conseguimos pequeñas mejoras.

Por último comenzamos a buscar otros features que a priori no parecían tan importantes y por eso fueron relegados a una etapa tardía del análisis, como ser las Ciudades y los Países (sobre todo este último por saber que hay poca variedad en el set de datos), también se terminó agregando el tipo de dispositivo y los navegadores que utilizan para generar los eventos (aunque a priori esto no pareciera tener relación con la elección de los usuarios) y es en este punto donde se obtuvo el que hasta ahora es nuestro mejor rendimiento 0.85649

3.4 - Combinación de los algoritmos

Luego tener los tres algoritmos empleados por separado y a sabiendas que de momento teníamos a XGBoost como la alternativa más precisa decidimos emplear una combinatoria de los algoritmos para ver si esto nos generaba una mejora, con lo cual encaramos esto usando los mismos features que teníamos para XGBoost (los cuales hasta ese momento nos daban el mejor score) y es en base a estos features hicimos dos entrenamientos y dos predicciones una para Random Forest y otra para KNN, luego con este resultado decidimos agregar una columna por cada algoritmo para sumar como si de otro feature se tratara, esto no nos generó una mejora, pero como habíamos agregado los dos features en simultáneo decidimos hacer otras pruebas aplicándolos por separado, habiendo armado una secuencia de la siguiente manera:

(Features generales + RF + XGBoost) = 0.70416

(Features generales + KNN + XGBoost) = 0.68488

(Features generales + RF + KNN + XGBoost) = 0.72346

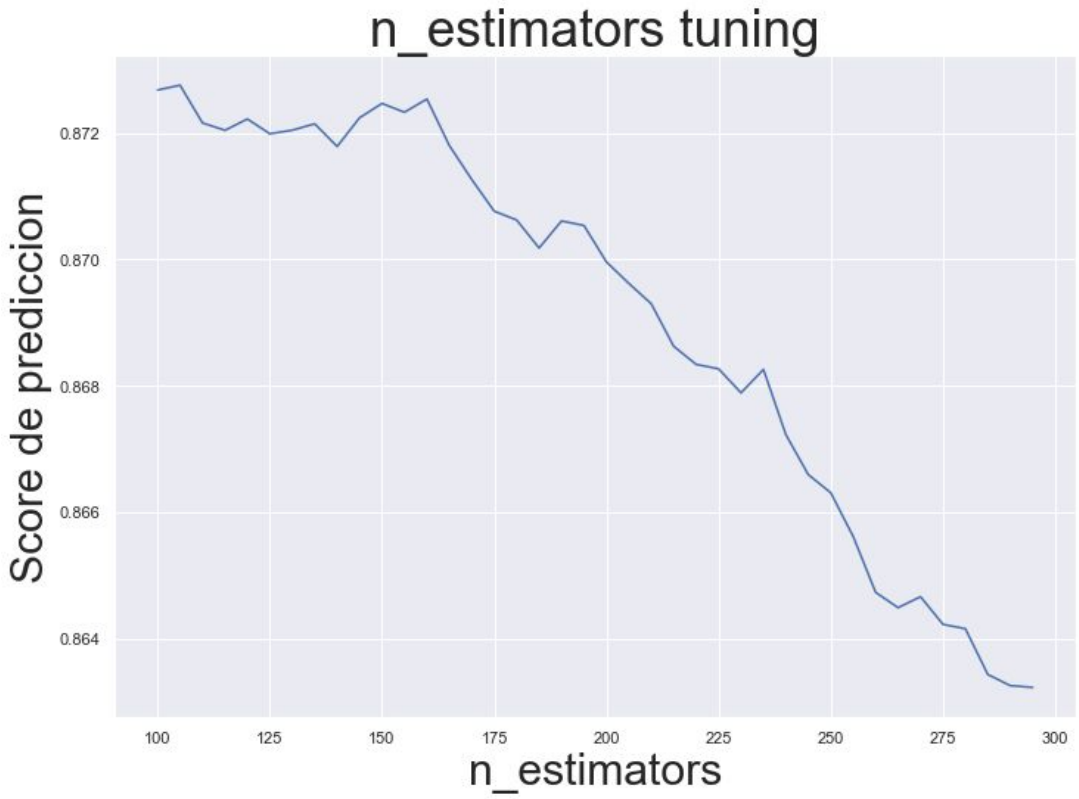
Teniendo los resultados antes expuestos decidimos no incluir estas alternativas puesto que todas terminaban empeorando lo que conseguimos con XGBoost reduciendo la performance en 0.13 como mínimo por lo cual se terminó optando por dejar XGBoost como clasificador principal y continuar mejorando el resultado que este nos diese en base a agregar o modificar features como así también modificar los hiper parámetros cosa que veremos a continuación.

4 - Hiperparametros

Una vez probados distintos algoritmos y combinaciones, se decidió usar XGBoost que fue el algoritmo con el cual se obtuvo mejor resultado, y luego ir tocando los hiper parámetros del algoritmo para ir mejorando el score. Para esto se usó el método de grid search, para el cual se determinaron un rango de valores para los parámetros y se recorrieron todas las combinaciones posibles hasta encontrar la óptima. Los valores de los rango se tomaron alrededor de los parámetros por defecto del algoritmo.

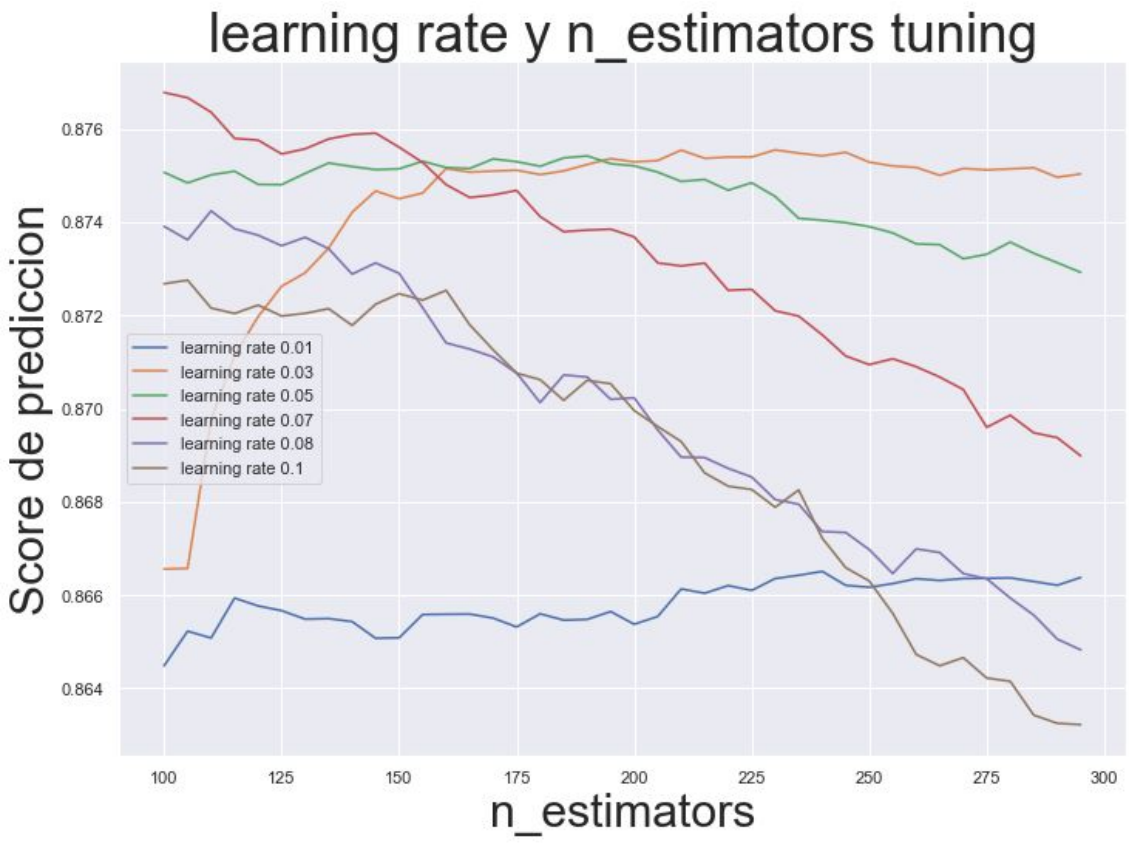
Para probar el rendimiento con diferentes hiperparametros, se dividió el set de entrenamiento en dos, una parte para entrenar y otra para testear, para evitar tener que submitear a kaggle. La división se hizo con una relación de 75% de los datos para entrenar y 25% para testear.

Lo primero que se tocó fue el valor de los n_estimators variando con pasos de 5 entre 100 y 300, el resultado se grafica a continuación:



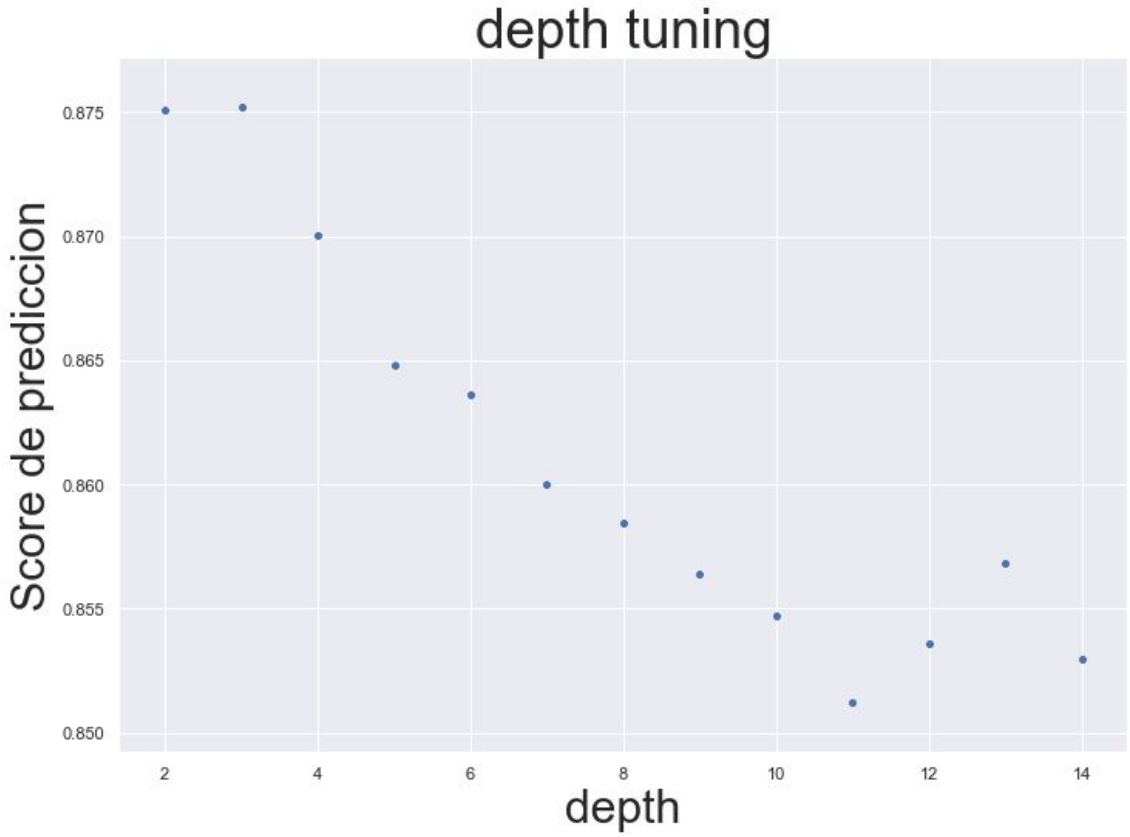
Se puede notar una tendencia clara en el score de la producción con relación al parámetro n_estimators.

El siguiente parámetro que se analizó fue learning_rate, a continuación se grafica el score en función del los n_estimators y el learning_rate.



En el gráfico podemos notar las distintas tendencias en función del learning rate, tenemos en marrón los resultados para un learning_rate = 0,1 que es el gráfico anterior, como se puede ver, el learning rate que parece tener mejor resultado es 0,07 con un rango de n_estimators desde 100 a 150 aproximadamente. Sin embargo, la diferencia entre el peor resultado y el mejor es de apenas 0,012 aproximadamente lo cual es insignificante, además estos números pueden variar según como se dividio el set de entrenamiento para probar. Lo que sí podemos concluir es que la línea correspondiente al learning rate = 0,5 es la más “estable”, por lo que se usó ese valor.

Luego determinar los mejores valores de n_estimators y learning rate, se analizo el parametro max_depth que determina la cantidad máxima de features a usar por árbol. Se grafica a continuación el resultado:



El mejor resultado para n_estimators=200 y learning_rate = 0.05 se obtuvo con un max_depth de 3, por lo que se usaron estos valores de hiperparametros.

Conclusiones

Durante el desarrollo de este trabajo, se utilizaron distintas herramientas de machine learning adquiridas durante la cursada, probando diferente algoritmos como así también la construcción de distintos features para ir mejorando el score de predicción. Si bien no se obtuvo un buen score en relación a los otros grupos, consideramos que es un resultado aceptable, considerando que estamos por debajo del mejor score por alrededor de un 3% y además teniendo en cuenta que el mejor score hasta el momento es de 0.88327, este problema se puede considerar como difícil.

Al probar distintos algoritmos de regresión, verificamos que en este problema de ML, XGBoost brinda una importante mejora en cuanto a score en relación a Random Forest, lo cual confirma el hecho de porqué este último se utiliza más para competencias de ML. Además notamos que si bien se obtenían mejoras al tunear los hiperparametros, las mejoras más significativas ocurrieron al agregar features que aportarán valor al modelo.