75.06/95.58 Organización de Datos - 1C 2019

Trabajo Práctico 2

Competencia de Machine Learning

Grupo 34: "DataTravellers"

Integrantes:

- Andrés Pablo Silvestri: 85881 (silvestri.andres@gmail.com)
- Juan Manuel González: 79979 (juanmg0511@gmail.com)
- Patricio Pizzini: 97524 (pizzinipatricio@yahoo.com.ar)

Link a repositorio de GitHub:
https://github.com/silvahlaravel/Organizacion_Datos_1C2019/tree/master/TP2

Fecha de entrega: 24/06/2019

Contenido

1 - Introducción

En este informe se describe el trabajo realizado para poder llegar a los distintos resultados que se fueron entregando a la competencia de Kaggle: jampp-at-fiuba-competition.

La competencia consistía en estimar cúando será la próxima vez que un dispositivo convierta orgánicamente, a fin de saber cuándo y si conviene apostar a una posible subasta RTB dada. El problema involucra dos partes:

- En un instante dado, estimar $S_t(d)$: el tiempo hasta que un dispositivo d aparezca de vuelta en una subasta RTB.
- En un instante dado, estimar $S_c(d)$: el tiempo hasta que un dispositivo d convierta.

Cada una de estas partes puede verse como un problema de survival analysis . Es importante notar que ambos valores deben ser no negativos.

A partir de los datos proporcionados, 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 describe 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 parseó la fecha del los eventos en tres columnas: hora, dia y mes para facilitar el manejo de los tiempos, a sabiendas que tanto el año y el mes eran irrelevantes puesto que ambos siempre iban a ser los mismo, de todas maneras decidimos quitar el año y dejar el mes.

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, coimo así también el formatear cierta información de modo que nos quede todo numérico y poder utilizar algoritmos de tipo árbol que son los que elegimos para plantear la solución de este problema.

Finalmente, se aclara que los archivos 'Events.csv' y 'Clicks.csv' no fueron utilizados en la solución final, dado que los features configurados no aportaron al resultado obtenido, si bien a priori nos parecía lógico que podrían brindar una información más detallada o enriquecida, pero también cabe mencionar que nos abocamos en resolver el problema en base a los datos provistos por los installs y los Auctions que son los más relevantes y relacionados para el resultado buscado.

2.1 - Datos relacionados al tiempo

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

2.1.1 - Días del mes

Utilizando el dato de la fecha, se armaron rangos de tiempo, a fin de tener los datos repartidos en "ventanas", en las que se dividieron los dataframes. Los valores utilizados son:

- Día 21 a 23
- Día 24 a 26

Estas ventanas fueron usadas en primer instancia para entrenar y probar los algoritmos haciendo el cálculo de tiempos en base a la diferencia en segundos entre un auction y otro, como así también entre un install y otro, luego solo se utilizó la ventana del 24 al 26 para la predicción final.

2.1.2 - 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. Esto es algo que ya habíamos utilizado en el trabajo práctico número 1 y que decidimos incluirlo en este puesto que nos brindaba un dato más acabado sobre el tiempo, puesto que por ejemplo para los auctions solo teníamos como dato primario la fecha y fue ahí donde con esta maniobra pudimos enriquecer un poco más la información brindada.

2.2 - Features utilizados

Luego de muchas pruebas, se fueron descartando/sumando features al modelo, con lo cual vamos a dividir en dos secciones esta parte, por un lado los features que finalmente quedaron en la solución final brindada y por otro lado los features que fueron descartados por no brindarnos valor a la hora del análisis, vale aclarar que estos siendo bastantes y quedando algunos en el camino, solo vamos a mencionar aquellos que a priori nos parecían más relevantes y que para nuestra sorpresa no nos dieron un valor significativo o positivo.

Features utilizados en la solución final:

- Pertenece a la franja de la Madrugada.
- Pertenece a la franja de la Mañana.
- Pertenece a la franja de la Almuerzo.
- Pertenece a la franja de la Tarde.
- Pertenece a la franja de la Noche.
- Cantidad de auctions para cada ID.
- Cantidad de installs para cada ID.

- Installs según ref types.
- Tipo de conexión (Wifi).
- Si la instalación es atribuída o no.
- Si la instalación es implícita o no.
- Agrupando por el user agent.
- El tipo (kind) de instalación.

Para todos los features mencionados fueron creadas columnas en forma análoga a lo ya explicado para las bandas horarias, con valores posibles 0 y 1 lo que nos permitió luego agrupar y sumar estos valores para tener las cantidades según el agrupamiento que necesario para la distinción buscada. También vale aclarar que para cuestiones de tiempo se decidió utilizar un agrupamiento con el Mínimo valor y no con el Máximo o con un promedio, esto se debe a que en varios pruebas nos dió mejor resultado.

Features que fueron dejados de lado:

- Cantidad de eventos para cada ID.
- Cantidad de clicks para cada ID.
- El user agent de la sesión.
- Cantidad de apariciones en la franja Mañana
- Cantidad de apariciones en la franja Tarde.
- Cantidad de apariciones en la franja Noche.
- Cantidad de apariciones en la franja Almuerzo.
- Cantidad de apariciones en la franja Madrugada

Lo que se intentó hacer en parte fue que agrupando por cada ID (ref_hash), ver que tantas de las apariciones son vinculadas a tal o cual franja horaria: teniendo esta cantidad asumimos que podría haber dado un valor positivo al cálculo, cosa que no resultó así, por

lo que además de sumarle mucha complejidad y trabajo nos resultaba perjudicial por lo que decidimos quitarlo. Lo mismo ha pasado para los eventos y los clicks, si bien entendemos que se podría haber trabajado muchísimo más sobre estos dos data frames y obtener un resultado enriquecedor, en las pruebas básicas que nosotros hicimos que fue relacionar los ref_hash y calcular las cantidades de eventos y clicks, notamos que el resultado también fue perjudicial por lo que decidimos quitarlos.

Como mejora, podríamos pensar y agregar algunos otros features que fuimos probando en un comienzo pero que quedaron perdidos puesto que fue en una primera prueba inicial que quizás con el correr del tiempo si los hubiésemos retomado podrían haber sumado valor.

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 sus árboles 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 producir St se tomaron los features del dataframe de 'Auctions', mientras que para el de Sc se utilizó 'Installs', para luego combinarlos al realizar el entrenamiento. Un primer intento de submit a Kaggle arrojó un resultado de 117434.39887.

3.2 - 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: el resultado obtenido en este caso fue de 115909.20864.

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. Corriendo toda la solución nuevamente logramos nuestro mejor score hasta el momento: 110407.89172.

3.4 - Combinación de los algoritmos

Luego tener los dos 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 entrenamiento. Esto no nos generó una mejora, por lo que decidimos no incluir estas alternativas puesto que todas terminaban empeorando lo que conseguimos con XGBoost.

RF + XGBoost: 125884.56353.

Por lo tanto, 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.

3.5 - Hiperparámetros

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 random search.

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 innecesariamente. La división se hizo con una relación de 75% de los datos para entrenar y 25% para testear. Siendo que utilizamos las ventanas de tiempo que se habían pautado en el enunciado del trabajo práctico, tratando de emplear la información del 18 al 21 y del 21 al 23, luego lo mismo pero del 21 al 23 y del 24 al 26, esto es importante de remarcar puesto que para este tipo de problema es muy importante tener bien definidas las ventanas de tiempo.

Luego volviendo puntualmente sobre los hiper parámetros podemos explicar que se trabajó inicialmente variando n_estimators y learning_rate. Luego de determinar los mejores valores, de 10 y 0.1 respectivamente, se analizó el parametro max_depth que determina la cantidad máxima de features a usar por el árbol.

El mejor resultado para n_estimators = 10 y learning_rate = 0.1 se obtuvo con un max_depth de 5, por lo que se usaron estos valores de hiperparametros.

8 - 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 (al momento de confeccionar el informe la posición en la 'Leaderboard' era 15), consideramos que es un resultado aceptable. Asimismo, agregamos que teniendo en cuenta que el mejor score hasta el momento es del orden de 70.000, este problema se puede considerar como difícil.

Al probar distintos algoritmos de regresión, verificamos que en este problema de ML, XGBoost brinda una 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.

Finalmente, notamos que si bien se obtenían mejoras al tunear los hiperparametros, las mejoras más significativas ocurrieron al agregar features que aportaron valor al modelo.