Trabajo Práctico 2

[7506/9558] Organización de Datos
Primer cuatrimestre de 2019

Grupo 27

Alumno	Número de Padrón	Email
Minino, Nahuel	99599	nahuel.minino@gmail.com
Alvarez, Gonzalo A.	98359	gonza2703@gmail.com
Aparicio, Axel	96283	axelaparicior@gmail.com
Peña, Alejandro	98529	alee.pena.94@gmail.com

Repositorio GitHub: https://github.com/alepenaa94/Datos_2019_tp2

Índice

1. Glosario

- *Convertir*: el objetivo de mostrar publicidad es que un dispositivo instale una aplicación, a ese evento se le llama **conversión**.
- *Retargeting*: es una estrategia de marketing que vuelve a atraer a los usuarios, que han mostrado interés en sus productos o servicios, mediante la publicación de anuncios personalizados en sus dispositivos móviles.
- *Instalación implícita*: Instalaciones atribuidas a jampp pero que no fueron contadas como conversión mediante el flujo habitual (Ej: Usuario instala la aplicación publicitada buscándola en el application store).
- *Dispositivo/Device*: entidad con un id de publicidad asociado. Por ejemplo: un celular Samsung J6 con Android tiene un id único, un Apple iPhone tiene un identificador único.
- *Evento*: cualquier tipo de interacción categorizada dentro de una aplicación.
- *Subasta/Auction*: en el momento que una aplicación quiere mostrar una publicidad, ese espacio se vende en una subasta (generalmente de segundo precio) donde todos los interesados en mostrar una publicidad ofertan un precio y gana quién más ofrece.
- *Agente de usuario/User Agent*: un agente de usuario es un software que actúa en nombre de un usuario. Un uso común del término se refiere a un navegador web que "recupera, procesa y facilita la interacción del usuario final con el contenido web".
- *St*: El tiempo hasta que un dispositivo aparezca de vuelta en una subasta RTB.
- *Sc*: El tiempo hasta que un dispositivo convierta.

2. Introducción

El objetivo de este Trabajo Práctico fue intentar predecir, para determinados dispositivos target presentados por la empresa Jampp, el tiempo que transcurrirá hasta que los mismos aparezcan nuevamente en una subasta, y el tiempo hasta que el usuario decida instalar una nueva aplicación. Esto puede ser de suma importancia ya que, teniendo una estimación de cuándo será la próxima vez que se vea un dispositivo y una estimación de la próxima vez que el dispositivo convierta, se pueden elaborar estrategias de apuestas, con mayor énfasis en aquellos dispositivos cercanos a convertir y menor en aquellos que les falta mucho tiempo para convertir.

Para realizar esto se utilizaron distintos algoritmos de Machine Learning, principalmente de la librería sklearn de Python, con los cuales se realizaron predicciones en base a un entrenamiento sobre la información, y una posterior validación de dichas predicciones. Además, se utilizó como herramienta Python3 con las librerías de Pandas y Numpy.

Como contábamos con información del 18/04 al 26/04 inclusive, la estrategia utilizada fue dividir los datasets en ventanas de 3 días entrenando cada una y validándola contra los 3 días siguientes.

El trabajo fue dividido en 3 partes:

- Limpieza de los datos y creación de los dataframes a utilizar.
- Creación de features por cada dataset.
- Entrenamiento de los algoritmos y validación de las predicciones.

Se utilizó un repositorio en GitHub (https://github.com/alepenaa94/Datos_2019_tp2) como herramienta de integración, donde se pueden encontrar los notebooks utilizados para la limpieza de los datos, para la creación de los features y para la propia predicción con los algoritmos.

3. Limpieza de los datos y creación de dataframes

En primer lugar nos encargamos de analizar la información recibida y realizar una limpieza de los datos cambiando los tipos de los datos para que ocupen menos espacio y ejecutar más rápido las sentencias. Además, redefinimos ciertas columnas de información para mejorar su legibilidad durante el análisis de los datos. Luego, armamos 4 dataframes con la información del período a analizar: auctions, events, clicks e installs.

Para este paso aprovechamos el trabajo previo realizado en el TP1 con sets de datos similares a los de este trabajo. Todo este proceso lo separamos del resto en Notebooks de Limpieza para facilitar la división de las tareas y reducir el acoplamiento.

4. Cálculo del St y Sc

Para calcular el St tomamos el dataframe de auctions, el cual contiene todos los registros de las subastas realizadas por dispositivos en el período de interés y los dividimos en ventanas de 3 días, luego por cada una de estas ventanas tomamos como St la cantidad de segundos en la que se produce la primer aparición del dispositivo contando a partir del inicio de la ventana.

Para el cálculo del Sc hicimos el mismo procedimiento que en el cálculo del St, solo que esta vez usamos el dataframe de installs, el cual contiene los registros de las instalaciones realizadas por los usuarios.

Los resultados de estos cálculos los guardamos como labels para utilizarlos durante la validación de las predicciones.

5. Features

5.1. Creación de features

La creación de los features la hicimos en cada ventana por separado y para cada uno de los distintos dataframes con los que contamos. Cabe destacar que nos enfocamos en features que describan el comportamiento por dispositivo y no en features por registro. Esto para facilitar el posterior mergeo de los features de todos los datasets en un único dataframe de features por cada ventana.

En un principio comenzamos con features sencillos para poder comprobar el correcto funcionamiento del algoritmo. A medida que los fuimos desarrollando, buscamos alternativas en los algoritmos en los que los probábamos. Algunos de estos features contenían valores numéricos, pero en cambio otros eran categóricos o de texto que se lo decidió considerar como categórico dado su importancia. Debido a esto se buscó métodos de conversión de variables categóricas a numéricas representativas.

5.2. Métodos utilizados

El primer método utilizado fue OneHotEncoder de la librería de sksurv parte de pre-procesamiento, este recibe un dataframe y convierte las M variables categóricas en M-1 columnas representando dichas variables.

En un principio con los primeros features y con datos acotados de prueba era viable el entrenamiento, cuando comenzamos a agregar cada vez mas features y mas datos de entrenamiento observamos dificultades para entrenar. Esto llevo a realizar ”feature importance” y un análisis extenso de los datos que se estaban entrenando, donde encontramos que una columna categórica como por ejemplo el tipo de evento, poseía gran variedad de categorías haciendo que, luego de aplicar OneHotEncoder, la cantidad de columnas crezca notablemente.

Debido a ello se decidió utilizar el método ”categorical_to_numeric” el cual en vez de generar columnas por cada categoría posible, usaba un número representativo para la categoría, sin alterar la cantidad de columnas del set de datos.

5.3. Features elaborados

Detallamos los features que probamos por cada dataframe:

5.3.1. Cantidad de subastas en la ventana

Del dataframe de auctions, contamos la cantidad de subastas en las que participa un dispositivo en la ventana.

5.3.2. Cantidad de subastas por día

Además, para profundizar más en la información, probamos con features que mostraban la cantidad de subastas en las que participaba cada dispositivo por día. Estos features no resultaron ser muy útiles ya que notamos que, al no participar en muchas subastas, estos features daban muchos ceros y tenían mucho ruido empeorando la predicción. Por lo tanto, decidimos no incluírlos.

5.3.3. Cantidad de clicks en la ventana

De la misma forma que hicimos con el dataframe de auctions, contamos la cantidad de clicks que realizaba el dispositivo en la ventana para tratar de hallar algún comportamiento que permita mejorar la predicción.

5.3.4. Cantidad de clicks por día

Al igual que con auctions, agregamos features que calculaban los clicks por día, estos features no aportaron mucho a la predicción por lo que opotamos por sacarlos.

5.3.5. Tiempo promedio en hacer click

En este feature agrupamos los registros de la ventana por dispositivo y calculamos el promedio del tiempo que se tarda en hacer click. Consideramos que este feature puede aportar mucha información para predecir el tiempo de conversión.

5.3.6. Región de pantalla más clickeado

En este feature se divide la pantalla en 9 partes iguales y se calcula la región de la pantalla en la que el dispositivo tiende a clickear más seguido. Esto lo hacemos con el objetivo de hallar algún comportamiento que común entre los usuarios.

5.3.7. Cantidad de instalaciones en la ventana

En este feature calculamos la cantidad de instalaciones que realiza el dispositivo en la ventana.

5.3.8. Cantidad de instalaciones por día

De la misma manera que hicimos con los otros dataframes calculamos las instalaciones por día.

5.3.9. Cantidad de clicks previo a instalar

Se calcula la cantidad de clicks previos a realizar una instalación por ventana, contando los de la ventana actual y previas.

5.3.10. Relación cantidad de instalaciones / cantidad de eventos

Para intentar obtener más información sobre el comportamiento del usuario, calculamos la cantidad de instalaciones que el dispositivo realiza en relación a la cantidad de eventos que realiza.

5.3.11. Tipo de instalación típico

Notamos que las intalaciones de tipo .ºpen.º semejantes son los que más predominan de entre todos los tipos de instalaciones. En este features determinamos si el dispositivo realizó alguna instalación de dicho tipo en la ventana.

5.3.12. Cantidad de eventos en la ventana

Al igual que con los demás dataframes, calculamos los eventos por ventana por cada dispositivo.

5.3.13. Cantidad de eventos por día

Se calculo la cantidad de eventos por cada día de la semana pero, al notar que empeoraban la predicción, optamos por no incluir estos features.

5.3.14. Cantidad de instalaciones por hora

Para el caso de los eventos, además, calculamos los eventos que ocurren a cada hora del día por cada dispositivo de la ventana.

5.3.15. Evento más típico

Se buscó cual fue el evento más típico registrado en la ventana usando la función mode() por cada dispositivo.

5.3.16. application_id típico

En este feature se buscó cuál fue la aplicación más usada en esa ventana de tiempo por cada dispositivo.

5.3.17. Tipo de evento típico

Se buscó cual fue el tipo de evento que más ocurrió por cada dispositivo.

6. Algoritmos y Ensambles probados

En general, lo que hicimos fue ir probando distintos algoritmos y comparar tres medidas para darnos una idea de como se comportaba cada uno y en donde erraban. Las medidas que utilizamos fueron: el score predefinido del algoritmo, prediciendo con el dataset de test, el score predefinido del algoritmo con el dataset de training y por ultimo, el error cuadrático comparado con el set de test.

Primero probamos con una de las ventanas, particionando en set de test y de training para evaluaciones preliminares de los algoritmos. Luego, pasamos a realizar el training con todas las ventanas. El primer testeo de los agoritmos los hicimos con hiperparametros por default.

6.1. Gradient Boosting Survival Analysis

Este algoritmo se diferenciaba un poco del resto en cuanto al formato del set de training. No logramos adaptarlo al problema que tratábamos de resolver, debido a que la función predict() nos devolvía un "risk error" y no la predicción en sí misma, por lo que lo descartamos.

6.2. KNN

Fue uno de los que menos rendimiento nos dio. No performo de manera esperada, por lo que decidimos no darle demasiada relevancia.

6.3. Random Forest

Nos dio bueno resultados. Fue uno de los scores mas altos. Las métricas que usábamos eran todas buenas. Este junto con gradient boosting fueron los candidatos a tener en cuenta para la resolucion de problema planteado.

6.4. Ada Boost

Otro de los algoritmos que mejor performance tuvo. No fue el mas alto pero podria haber sido el elegido para entrenar el modelo final.

6.4.1. Decision Tree y Extra Tree

Los algoritmos de arboles en general, no daban mal score pero no se acercaban al de los ensambles. No eran malos, pero nos resultaron mas beneficiosos los otros. Las ventajas de estos es que podiamos ver la importancia de los features que pusimos.

6.5. XGBoost

Ensamble que no era parte de la libreria de sklearn. Dio buenos resultados, pero no los mejores.

6.6. Majority Voting

De todos los algoritmos que implementamos, probamos hacer majority voting variando los modelos, pero no daban nada superador. El score era bueno, pero era mejor usarlos por separado.

6.7. Bagging

Probamos en BaggingRegressor de sklearn con hiperparametros por default. Score bueno pero no se acercaba al de los otros algoritmos.

6.8. HistGradientBoosting

Version de gradient boosting para grandes cantidades de datos. Lo probamos, pero tardaba demasiado y la cantidad de parametros no se ajustaba a lo que se recomendaba como minimo para utilizarlo.

6.9. Gradient Boosting

Fue el mejor algoritmo que probamos junto con RandomForest, por lo que decidimos dedicarle tiempo para realizar cross validation.

7. Optimización de Hiper parámetros

Se optó por realizar cross validation en los algoritmos que mejor performaban. Gran parte del tiempo se lo dedicamos a XGBoost inicialmente, utilizando el método GridSearch. Sin embargo, como vimos que nos convenía utilizar GradientBoosting. Dedicamos gran parte del tiempo a la optimización de hiper-parámetros de este con GridSearch también.