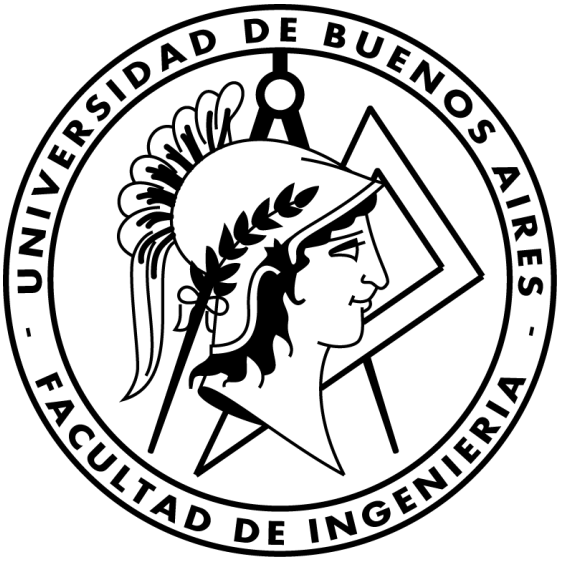

Universidad de Buenos Aires
Facultad de Ingeniería
Año 2018 - 2º Cuatrimestre

Organización de datos - Trabajo Práctico 2

https://github.com/ConstanzaGonzalez/DatataFrita

Integrantes

Nombre	Padrón
Gonzalez, Constanza	98031
Marino, Gonzalo	97794

Índice

Introducción

El objetivo de este informe es poder determinar, a partir de distintos algoritmos de Machine Learning, cuál es la probabilidad de que cada usuario realice una conversion (compra) en Trocafone en un periodo de tiempo determinado.
La idea es probar distintos algoritmos e, incluso, la combinación de ellos para poder encontrar la mejor predicción.

El informe consta de dos secciones: por un lado tendremos el preprocesamiento de datos y los features buscados para poder aplicar los algoritmos de Machine Learning. Por otro lado, tendremos los algoritmos probados y detallaremos si éstos nos fueron de utilidad o no y el por qué. Mostraremos el desarrollo y los resultados que obtuvimos.

Preprocesamiento de datos

El preprocesamiento de datos consta en la modificación de los datos provistos por la cátedra sobre Trocafone, con el fin de adaptarlos a nuestra conveniencia para el desarrollo del trabajo.

Como sólo obtuvimos un dataset con información que abarca desde el 1 de enero del 2018 hasta el 31 de mayo del 2018, procedimos a cambiar el tipo de dato de las distintas columnas del mismo. Pero, como los datos más relevantes para las distintas features fueron la fecha y hora del evento que realizó el usuario, solo nos enfocamos en la columna aplicada para cada caso.

Features

Lo más importante a la hora de buscar features es encontrar un comportamiento similar entre los usuarios, es decir, si estos cumplen o no un cierto patrón a la hora de realizar una compra. Lo cual no es tarea fácil, debido a que el comportamiento de los usuarios pocas veces resulta similar, porque en ocasiones un usuario puede realizar muchos más eventos que otros antes de comprar.

Es por está razón que, para determinar cuáles son aquellos usuarios que realizaron tantos o cuantos eventos, decidimos implementar las siguientes features para el desarrollo del trabajo.

Features Básicas

Para empezar, las features más básicas que pudimos obtener del dataset se basan en la cantidad de eventos realizados por los usuarios, especificandolos en sus distintas categorías. Se optó por utilizar la mayoría de los eventos, reconociendo a **"cantidad de conversions, checkouts y viewed products"** como los que le son más importantes a los algoritmos.

Luego, otro feature que obtuvimos fue el cual indica el promedio de eventos por sesión y la cantidad de sesiones para los distintos usuarios.

Eventos

- Obtuvimos la cantidad total de eventos, la cual está determinada por la suma de todos los eventos realizados por usuario.
- Primero observamos la cantidad de vistas que tuvo cada producto y, en base a los datos obtenidos, verificamos si esa cantidad total supera el promedio entre todos los usuarios.
- Luego, se separaron en distintas columnas las diferentes StaticPage, determinando así, cuál fue la última StaticPage que el usuario visitó.
- Algo parecido pasa en la siguiente feature, en la cual aparecen columnas que separan los distintos eventos existentes, para luego determinar cuál de éstos fue el último que el usuario realizó.
- La feature Channel separa los distintos tipos de canales y marca el último canal que el usuario utilizó según el último evento.
- Por último, la feature Device refiere al dispositivo que el usuario usó al momento de realizar un evento, separándolos en distintas columnas según el tipo de Device.

Compras

Para el evento compras realizamos una feature tomando los datos de la última conversion de cada usuario e identificando cuál fue la marca elegida por los mismos. Esto resultaría en columnas designadas para cada marca (Apple, Samsung y Motorola -Las más populares-), en la que se detallan cuáles usuarios compraron productos de cada una.

Tiempo

- Primero determinamos una feature que permite reconocer la cantidad de eventos realizados según las horas activas del usuario, es decir, cuántos eventos se realizaron en cada hora que el usuario interactúa. Dicha feature nos permite identificar cuál es la hora del día en la cual los usuarios están más activos, por eso se omite el tipo de evento y el día. Además podremos observar si hay un patrón constante en el cual se repite un horario particular.
- Luego, para establecer la segunda feature, tomamos los datos en base a la última sesión del usuario, esto es: la última fecha, el último horario y el último evento realizado.
- Así también como la feature anterior se realizó otra similar, pero ésta vez tomando los datos de la primer sesión del usuario (día, hora y evento).
- Después se realizaron features que suman la cantidad de horas (en segundos) y días del usuario. En una feature solo termina en la suma, y en otra se realiza un promedio de todo el comportamiento del usuario.
- La siguiente feature funciona según la última conexión, de la cual sólo se elige la hora en la que el usuario realizó su última interacción con el sitio.
- Por último, desarrollamos una feature relacionada a la última vez que el usuario interactuó con los siguientes eventos: check, search, conversion y view.

Modelos

Para la siguiente feature, verificamos cuál fue la marca que más visitas tuvo (usando el evento Viewed Product) por parte del usuario. Determinando así, si el usuario tiene preferencias sobre una marca en particular para su compra. De la misma manera, se realizó una feature para el evento checkout.

Para ambos casos nos centramos en tres marcas en particular: Apple, Samsung y Motorola, debido a que en el TP1 resultaron las más populares elegidas por los usuarios.

Países y Regiones

En esta sección se realizó un feature, usando **One Hot Encoding,** que permite dividir en columnas los distintos países, dentro de las cuales identificamos si el usuario pertenece o no a tal país. Luego, dado que se registraron 37042 usuarios, nos quedamos con la columna que contiene los datos de Brasil.
Del mismo modo, se realizó otra feature que divide en columnas las distintas regiones, considerando a Sao Paulo como la más frecuentada.

Ambas features fueron desarrolladas en base a los datos obtenidos en el TP1.

Sku

En relación al SKU, buscamos el código del último producto que el usuario visitó y nos quedamos con el valor obtenido. Lo mismo realizamos para el evento checkout.

A modo de prueba, realizamos el mínimo, el promedio y la suma de todos los valores del SKU por usuario. A pesar de que dichos valores no provienen de un feature engineering, vimos que son importantes para el algoritmo a la hora de clasificar.

Algoritmos Utilizados

Decision Tree

Este algoritmo fue uno de los últimos probados, por lo que se probó con casi la mayoría de las features creadas. Sabíamos que para este caso, Decision Tree permitía obtener buenos resultados, y por dicha razón empezamos a experimentar en su uso.

Sorprendentemente al probar este algoritmo, empezamos a notar un alto valor en los resultados obtenidos, incluso similares a los puntajes obtenidos anteriormente con los otros algoritmos probados hasta el momento. Sin embargo, no hicimos un desarrollo amplio de este algoritmo porque XGBoost (algoritmo de cabecera y el más utilizado), progresaba de mejor manera.

Aunque no hubo un gran desarrollo del Decision Tree, al momento de probarlo usamos un grid search para saber el mejor valor para el hyper-parámetro “max-depth”, y así obtener un mejor resultado. Dicho puntaje fue de 0.81.

Creemos que si se hubiese realizado una búsqueda más profunda con la búsqueda de otros valores para el grid search, obtendríamos mejores resultados.

KNN

El algoritmo consiste en buscar los k vecinos más cercanos, siendo k un hyper-parámetro que se pasa cuando se instancia el clasificador. Al encontrarlos, es posible predecir la probabilidad para clasificar una clase.

Debido a que KNN es uno de los algoritmos más simples y suele arrojar buenos resultados, decidimos intentar utilizarlo. Aunque al usar el set de entrenamiento nos arrojó un score del 94%, al momento de predecir obtuvimos un puntaje del 70%, el cual es relativamente bueno, pero no es lo suficientemente adecuado respecto a otros algoritmos que daban mejores resultados, incluso con la misma cantidad de features.

Creemos que para mejorar el valor del puntaje, debíamos realizar una búsqueda más exhaustiva para encontrar el mejor valor de k.

Random Forest

Este fue el primer algoritmo que probamos que nos arrojaba buenos resultados sólo con el uso de las features más básicas que pudimos obtener.
Para el set de entrenamiento obtuvimos un score de entre el 80-90%, y para predecir arrojaba entre el 75-80%. Luego de haber introducido una mayor cantidad de features, obtuvimos valores entre el 80-85%.

Creemos que al buscar los mejores valores para los hyper-parámetros: cantidad de estimadores y “min sample split”, hubiéramos obtenido mejores resultados. Descartamos el uso de éste algoritmo al probar otros y ver que nos daba mejores o iguales resultados.

SVM

Debido a que este algoritmo es uno de los más populares y suele arrojar buenos resultados, decidimos probarlo para mejorar el rendimiento. Sin embargo, fue uno de los últimos en ser probados

Al probar SVM, se utilizaron todas las features creadas hasta el momento, las cuales estaban lo suficientemente desarrolladas como para que nos devuelvan un puntaje similar o mejor que el obtenido en ese entonces.
Al principio, los resultados en el entrenamiento fueron buenos y suficientes, pero al momento de hacer la predicción, nos devolvió un set lleno de ceros. Esto resultó en un valor de 0.5 en kaggle, porque no estaba realizando una predicción coherente, por lo que fue descartado debido al bajo rendimiento obtenido.

Creemos que hubiese resultado mejor la implementación de este algoritmo si lo hubiéramos desarrollado de manera más profunda.

Vowpal Wabbit

Al descubrir este algoritmo intentamos utilizarlo para el desarrollo del trabajo, pero no pudimos hacerlo funcionar con un ejemplo, debido a que hubieron problemas con la instalación de la librería necesaria para su funcionamiento. Estando casi en la etapa final, se decidió descartar rápidamente el uso del mismo.

XGBoost

La decisión de incluir este algoritmo fue tomada en base al hecho de que es uno de los más recomendados para clasificación, además de ser uno de los más populares en las competiciones de kaggle, ya que se arrojan los mejores resultados en ensambles donde éste es el clasificador base.

En los comienzos de su utilización, en paralelo a Random Forest, notamos que XG Boost arrojaba menores valores, por lo que fue dejado de lado. También, otra de las razones por la cual omitimos su uso fue la demandante manipulación de las features para que el algoritmo funcionara. Es decir, que las features debían tener demasiadas modificaciones con algún encoding (One Hot Encoding) para que pueda entrenar el algoritmo.

Al principio, el set de entrenamiento nos daba un resultado del 95%, pero al subir el primer submit a la página de kaggle, obtuvimos un valor de 0.72. Dichos valores resultaban buenos, pero no lograban superar los resultados de Random Forest.

Sin embargo, al realizar un grid search para los hyper-parámetros: Max Depth, N Estimators y Scale Pos Weight obtuvimos mejores resultados, obteniendo un score de 0.87[1]. Por lo que se decidió continuar con su uso, e incluso, implementarlo como el algoritmo principal para el desarrollo del trabajo.

Ensambles

Los ensambles consisten en combinar los resultados de dos o más algoritmos para obtener una clasificación distinta.
Las combinaciones que se probaron fueron:

- **XGBoost con Decision Tree:** aunque estos fueron los que mejor resultados nos dieron, el valor del ensamble obtenido se encontró dentro del promedio entre valor de ambos. Es decir, que logró una puntuación de 0.86.
- **XGBoost con Random Forest:** El ensamble del mismo no superó al anterior, ya que su resultado fue de 0.85.
- **XGBoost con Random Forest y Decision Tree:** Aunque estos fueron los algoritmos que mejores resultados arrojaron. creímos que el ensamble de los mismos iba a superar al resultado obtenido con XGBoost (sin ensamble), pero el valor del mismo se encontraba entre el puntaje del primer ensamble mencionado y el de XGBoost.

[1] Último valor obtenido en la fecha 3 de Diciembre de 2018.

Conclusión

La mejor Solución

Desde un principio sabíamos que los mejores algoritmos que iban a funcionar para el desarrollo del trabajo eran Decision Tree, Random Forest y XGBoost, debido a que son los algoritmos más populares en este tipo de competencias. Es por eso que se llevó a cabo una búsqueda más exhaustiva respecto a su rendimiento de éstos que de los otros.

Si bien los resultados de los distintos algoritmos arrojaron valores bastantes similares y buenos (superando el 0.80, casi llegando al 0.88), estos no son los suficientes buenos como para ganar la competencia y estar dentro de los 5 mejores.
Nuestro mejor puntaje se logró al usar el algoritmo de XGBoost, el cual nos dio como resultado de score 0.87187[2].

Un gran impedimento a la hora de predecir la compra, ocurre por el hecho de que la cantidad de conversiones (compras) son demasiado pocas con respecto a la cantidad de usuarios que utilizan el sitio, por lo que es difícil llegar a una buena precisión y obtener buenos resultados. No obstante, la precisión que se obtuvo es aceptable, ya que los algoritmos de Machine Learning pocas veces obtienen un score mayor al 0.92 sin llegar a la situación de overfitting.

[2] Último valor obtenido en la fecha 3 de Diciembre de 2018.