Trabajo Practico 2: Organización De Datos

Grupo 31: Datalia Datalia

2do Cuatrimestre 2018

Link al GitHub:
https://github.com/MartinCohen98/OrganizacionDeDatosTP2

1 Introducción

Este trabajo practico consiste en el armado de un algoritmo de Machine Learning capaz de predecir qué usuarios de la página de internet Trocafone comprarán un producto en la página desde el 01/06/2018 hasta el 15/06/2018, dados los datos de este año hasta el 31/05/2018 de estos mismos usuarios. Este informe detalla el proceso de armado de ese algoritmo, y los resultados finales obtenidos. En función de los datos en los eventos, y una serie de labels que tenemos como dato, hay que generar un set de entrenamiento que permita al algoritmo generar predicciones correctamente.

Esta predicción es un problema de Machine Learning de clasificación. Eso significa que con los datos, tenemos que decidir en qué clase tenemos que incluir a cada usuario. En este problema en particular, la clasificación está dividida en 1 si realiza una compra, y 0 si no lo hace. En nuestra solución, utilizamos el lenguaje Python en su versión 3.6.

El proceso de creación del algoritmo incluye varios tipos de preparación diferente, explicitados en la siguiente sección:

2 Preparación del Algoritmo

2.1 Transformaciones a los Datos

Antes de comenzar a trabajar con los algoritmos de Machine Learning, es necesario familiarizarse con los datos para saber que son los datos importantes, y que transformaciones hay que efectuar en los datos antes de comenzar a trabajar con ellos.

En este caso, debido a que los datos utilizados son los mismos que estudiamos durante el primer trabajo práctico, en el que hicimos un análisis exploratorio de los mismos, el proceso de familiarización con los datos ya estaba casi completo desde un principio. Los únicos datos agregados en esta instancia con los que no habíamos trabajado antes son los labels de el set de entrenamiento que tuvimos que crear. La particularidad más importante de estos es que estaban desbalanceados. Esto significa que hay una porción de los labels sustancialmente más grande de una clase que de la otra. En nuestro set de entrenamiento, teníamos una proporción de un 1 cada veinte 0. Esto es un problema ya que tiene diferentes implicaciones en los diferentes algoritmos que utilizamos (ver sección 2.3).

En principio, lo que hicimos fue generar un set de datos reducido, en el que la cantidad de 1 y la cantidad de 0 fueran iguales. Esto fue necesario para un algoritmo probado en particular, pero finalmente para el algoritmo final utilizado no fue necesario ya que presentaba una pérdida importante de cantidad de datos y solucionamos el problema con mejores resultados agregándole más peso a los 1, y penalizando más al algoritmo cuando no acertaba la predicción de estos. Esto evitaba que en entrenamiento los algoritmos intentaran tener un score muy grande con lo que en la práctica funciona como un 'return 0' (con los datos desbalanceados, eso funciona un 0.95 de las veces).

2.2 Feature Engineering

Gran parte del trabajo fue la creación de los diferentes features que utilizó el algoritmo para realizar sus predicciones. Estas features están definidas como una propiedad o característica mensurable de un fenómeno observado, y fueron extraídas del set de datos a nuestra discreción, tomando en cuenta qué información podía ser importante para la predicción.

La extracción de features del set de datos fue realizada a través de la librería Pandas. Este trabajo consistió generalmente en, hacer un filtrado de los datos y luego aplicar una función a los mismos luego de un GroupBy.

La importancia de los features depende de cuánto sirve para diferenciar en qué clase el algoritmo tiene que ubicar a cada usuario. Algunos de los algoritmos probados nos permitieron ver cuáles fueron los más importantes; esta información se encuentra en la sección de conclusiones.

A continuación se encuentra la lista de features usada, y la lógica detrás de la inclusión (no necesariamente terminó funcionando como en un principio creíamos):

Cantidad De Eventos: Usuarios con más eventos, son más activos en la página. Por lo tanto tienen más probabilidad de comprar un producto.

Cantidad De Compras: Un usuario que ya realizó una compra en la página ya demostró interés en utilizar la misma para comprar. Mientras más compras tenga, más probable es que vuelva a comprar.

Cantidad De Leads: Debido al análisis exploratorio que realizamos anteriormente sabemos que la utilización de leads es muy poca, entonces es probable que la gente que la usa tenga interés en comprar productos. Mientras más productos pida leads, más probabilidad de comprar alguno tiene.

Compras En Cada Mes: La cantidad de compras que realizó cada usuario por mes puede ayudar a determinar si un usuario compró más o menos en los últimos meses en comparación con los primeros. Eso puede ayudar a determinar el interés del usuario con la página en el tiempo.

Leads Hace Un Mes: Si un usuario realizó una lead en el mes anterior, debería ser probable que compre un producto en el próximo mes.

Cantidad De Checkouts: Un usuario que realiza muchos checkouts muestra interés en comprar a través de la página.

Promedio De Ventas Por Mes: Un usuario que en promedio realiza más compras por mes debería tener más probabilidad de comprar un producto.

Dias Desde El Último Evento: Si un usuario no realiza un evento en la página desde hace mucho, esto significa que no es activo en la página, y por ende es poco probable que realice una compra.

Dias Desde El Primer Evento: Un usuario que está hace más tiempo en Trocafone debería tener más confianza en la página que un usuario nuevo, entonces es más probable que la use para comprar productos.

Días Desde Última Compra: Hay dos lógicas posibles para este feature. Puede que un usuario que realizó una compra hace poco tiempo esté interesado en la página y por ende compre de nuevo, o puede que como Trocafone trabaja principalmente con teléfonos celulares, una persona que acaba de comprar un teléfono no necesariamente quiere comprar otro al poco tiempo. Decidimos incluir la feature y dejar que el algoritmo decida.

Cantidad De Search Engine Hits: Si buscan mucho a la página en internet, es probable que un usuario muestra interés en la pagina.

Entraron Más De Una Vez: Este feature muestra si un usuario volvió a entrar a la página luego de la primera vez o no. Si el usuario volvió a entrar a la página, es más probable que realice una compra que si nunca volvió.

Dias Desde El Último Checkout: Si un usuario realizó un checkout hace poco tiempo, es más probable que quiera comprar un producto en el futuro cercano.

Dispositivo De Entrada: Este es un grupo de features que tiene un 1 si el usuario utilizó determinado dispositivo para entrar a la página, y un 0 si no. En la práctica funcionó casi como un One Hot Encoding de cada tipo de dispositivo, ya que hay muy pocos usuarios que utilizaron más de un tipo. Pensamos que era probable que usuarios que compran podrían llegar a tender a utilizar un dispositivo más que otro.

Cantidad De Ad Hits: Si una persona entra a la página a través de un ad es probable que quiera utilizarla para comprar; mientras más hits, más probable que compre.

Días Desde Último Ad Hit: Si un usuario entró a la página por un ad hace poco tiempo, es probable que le interese comprar un dispositivo.

Días Desde Último Search Engine Hit: Un usuario que buscó la página por un buscador de internet hace poco tiempo es alguien que muestra interés en la página.

Días Desde Última Búsqueda: Esta feature muestra cuántos días pasaron desde la ultima vez que un usuario utilizó el buscador de la página. Si un usuario buscó algo hace poco, significa que está interesado en un producto en específico, y quizás quiera comprarlo.

Cantidad De Colores Vistos: Si un usuario vio dispositivos de muchos colores es probable que haya mirado una gran cantidad de productos, o que haya mirado muchos colores del mismo producto decidiendo cuál comprar. Ambas hacen probable que el usuario compre algún producto.

Cantidad De Días Diferentes De Visita: Si un usuario visitó la página en muchos días diferentes, es más probable que esté interesado en la página que uno que visitó en menos instancias.

Días De Actividad De Usuario: La diferencia entre la fecha del primer evento del usuario con la fecha del último. Es más probable que un usuario con un mayor número esté más familiarizado con la página que alguien con un número menor.

Cantidad De Modelos Vistos: Una persona que miró muchos modelos probablemente buscó entre varios para decidir cual comprar.

Usuario Es De Brasil: En el análisis exploratorio anterior, descubrimos que la mayor parte de los usuarios son de Brasil. Esto significa que le página es más conocida ahí, luego un usuario de ese país debería tener una probabilidad mayor de efectuar una compra que un usuario de cualquier otra parte del mundo.

Usuario Es De São Paulo: Al ser los usuarios de São Paulo quienes realizan más conversiones, también decidimos incluirlo en la lista de features.

Página estáticas accedidas: Si un usuario accedió a la página 'how-to-buy' es posible que realice una compra en el futuro.

Compras de cada condición: Puede que un usuario que compre productos de mayor calidad y pague más sea un buen candidato a realizar otra compra; por otro lado, uno que compre productos de menor calidad y más baratos no.

Cantidad De Búsquedas Por Marca: Un usuario que busca una marca o modelo específica debe tener interés en comprarlo.

Eventos Realizados Luego De Una Búsqueda: Si un usuario realiza un evento específico luego de una búsqueda más que otro, el algoritmo puede llegar a encontrar tendencias y poder clasificar mejor.

- **Visitas a marca (vam)** Cantidad de visitas a un producto de la marca buscada.
- **Visitas a otra marca (vao)** Cantidad de visitas a un producto de una marca diferente a la buscada.
- **Checkouts a marca (coam)** Cantidad de checkouts con un producto de la marca buscada.
- **Checkouts a otra marca (coao)** Cantidad de checkouts con un producto de una marca diferente a la buscada.
- **Conversiones a marca (cam)** Cantidad de compras de un producto de la marca buscada.
- **Conversiones a otra marca (cao)** Cantidad de compras de un producto de una marca diferente a la buscada.

- **Leads a marca (lam)** Cantidad de leads con un producto de la marca buscada.
- **Leads a otra marca (lao)** Cantidad de leads con un producto de una marca diferente a la buscada.

Algunos de estos features no terminaron siendo utilizados ya que no mejoraban los resultados. Esto también será desarrollado en más profundidad en la sección de conclusiones.

2.3 Selección del Algoritmo

Antes de decidir qué algoritmo utilizaríamos, probamos con tres opciones diferentes. Los algoritmos que probamos fueron KNN, Random Forests, y XGBoost. Aunque todos tuvieron una ventaja sobre los otros, los resultados fueron más favorables para uno de los algoritmos en específico. A continuación se describe la experiencia con cada uno.

2.3.1 KNN

Desde un principio no tuvimos muchas esperanzas de que KNN fuera el algoritmo que terminaríamos usando, ya que supusimos que el problema era demasiado complejo para que fuera efectivo. Sin embargo, como podíamos estar equivocados, decidimos probarlo debido a que es un algoritmo con muy pocos hiperparámetros y podíamos utilizar las mismas features que los otros algoritmos.

En este algoritmo no pudimos evitar reducir el set de entrenamiento, porque con una relación de veinte a uno para las clases el algoritmo tendía a dar muchos más predicciones 0 que 1. Esto es lógicamente una desventaja, ya que utilizar más datos en un algoritmo de Machine Learning suele dar mejores resultados, y luego de la reducción el set de entrenamiento pasaba de tener más de 19000 usuarios a menos de 2000.

Los resultados del algoritmo no fueron muy exitosos comparado con su competencia, llevándonos a dejar a KNN de lado. Tenemos una fuerte sospecha de que una de las razones por las que que no funcionó tan bien es que algunas de las features tienen una varianza mucho más grande que otras y el algoritmo no toma en cuenta la importancia de las features para determinar qué puntos son los más cercanos, además de tener un modelo muy simple.

2.3.2 Random Forests

En principio este algoritmo pareció muy llamativo: no tiene muchos hiperparámetros, y es resistente al overfitting dado que los anteriores sean correctos. El algoritmo también tiene potencial de predicción de modelos muy complejos.

Tiene un hiperparámetro que permite indicar si el set de datos esta desbalanceado, y el mismo algoritmo se ocupa de regular cómo resolver el problema ajustando el peso de las clasificaciones 1, y penalizando al algoritmo si el mismo

se equivoca en ellas en el entrenamiento. Esto nos dejó utilizar el set de datos completo para entrenalo.

A pesar de haber sido fácil de utilizar, el algoritmo dio resultados muy buenos, acercándose mucho a los mejores que tuvimos. Sin embargo, los resultados fueron un poco peores que los de XGBoost, que nos llevó a optar por este último hasta el final. Nunca dejamos de utilizar Random Forests como referencia por la ventaja de tener una Cross Validation mucho más simple que el XGBoost, y que los resultados de los dos algoritmos eran muy similares.

2.3.3 XGBoost

Este es el algoritmo que mejores resultados nos dio, y por ende el que decidimos terminar utilizando. Presenta una gran variedad de hiperparámetros que permiten resolver problemas muy complejos ajustándolos.

A pesar de esto, este algoritmo es el que mayor tiempo nos llevó utilizar correctamente, ya que el proceso de Cross Validation es muy lento. Al tener muchos hiperparámetros, fue necesario hacer un Randomized Search para aproximar los parámetros buscados, y luego ajustarlos con un Grid Search para obtener resultados óptimos. El proceso pudo llegar a tardar aproximadamente una hora, pero una vez llegado a eso, el entrenamiento del algoritmo es rápido.

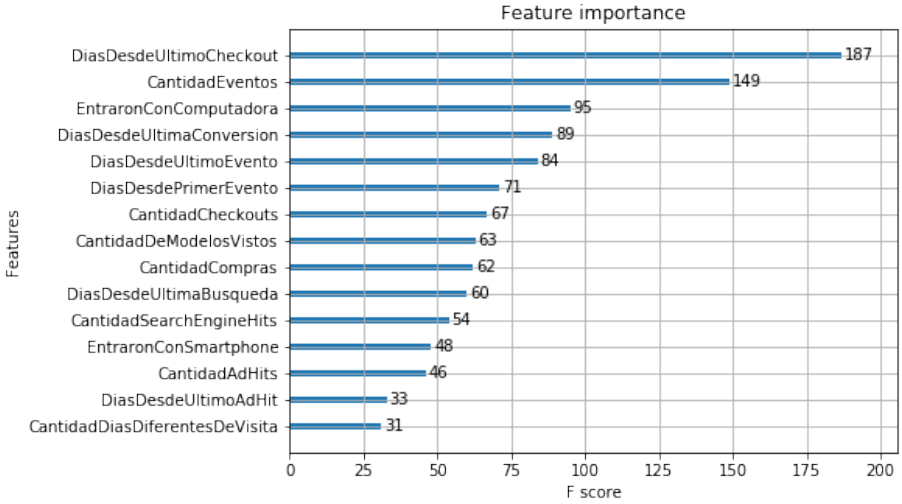
De este algoritmo también obtuvimos las features más importantes para decidir la predicción a través de un gráfico propio de la librería de Python de XGBoost. Con este plot podemos ver de qué features podemos extraer más información sobre lo que queremos predecir.

3 Conclusiones

Como expresamos antes, el algoritmo XGBoost nos permite ver un gráfico que nos muestra si la feature fue importante para realizar las predicciones o no. Durante la realización del trabajo la importancia de las features se mantuvo estable, con respecto a lo que terminó siendo la mejor de nuestras predicciones.

Algunas features, al agregarlas, no mejoraban el nivel de predicción de los algoritmos (como por ejemplo, los accesos a las páginas estáticas y las compras de productos de cada calidad). Estas no fueron incluidas para la predicción final.

El siguiente gráfico muestra las importancias de las features para XGBoost:



Estos resultados muestran que las features más importantes fueron los días desde el ultimo checkout, la cantidad de eventos totales del usuario, y si alguna vez entraron al sitio de Trocafone desde una computadora (en ese orden). También se le da bastante importancia a la cantidad de días desde la última compra y desde la última vez que buscaron a la página en un buscador.

Los días desde el primer y desde el último evento del usuario también fueron sustancialmente importantes, tanto como la cantidad de compras checkouts y ad hits.

Los resultados de la feature importance de Random Forest tiene algunas similitudes y diferencias con las de XGBoost. La cantidad de días desde el ultimo checkout sigue siendo la más importante. Esto parece en un principio lógico: si un usuario realizó un checkout días antes de la fecha en la que se quiere predecir, significa que definitivamente está interesado en comprar un producto.

En cambio, Random Forest le da mucha más importancia a la cantidad de checkouts que XGBoost, estando no muy por debajo de la feature más importante. La relación con la predicción no es tan evidente como la anteriormente mencionada, pero en el algoritmo mantiene una importancia bastante más alta que el resto.

Este algoritmo también posiciona a los días desde el primer y desde el último evento del usuario por encima de la cantidad total de eventos del mismo.

Ambos algoritmos coinciden en que el dispositivo que más ayuda a diferenciar entre los usuarios que compran y los que no es la computadora. Se llega a esta conclusión viendo que ambos algoritmos rankean a la feature que indica los usuarios que la utilizan para entrar a la página mucho más alto que las demás.

Durante el primer trabajo práctico, encontramos una posible proporcionalidad entre la cantidad de ventas de la página y la cantidad de ad hits. Es importante notar que versiones anteriores de los algoritmos mostraban tanto cantidad de ad hits como días desde ultimo ad hit como más importantes. Sin embargo, en las últimas predicciones de XGBoost su importancia bajo enorme-

mente. En Random Forest sigue siendo una feature importante, aunque bajó su ranking entre las otras features asimismo.

Estas son algunas features que imaginamos inicialmente como importantes pero no lo fueron:

- Si entraron más de una vez a la página.
- El promedio de compras por mes del usuario. Esto parece ser porque la mayoría de las compras están concentradas en los últimos meses.
- La utilización de leads. Creemos que se debe a que una porción muy pequeña de los usuarios usa esta función de la página.

Un comentario importante es que ambos algoritmos, dándole importancia diferente a la mayoría de las features, llegaron a un nivel de predicción parecido (con una diferencia cercana al 1%). Esto significa que no podemos asegurar que las features más importantes que utilizan los diferentes algoritmos para predecir son necesariamente las features más importantes en la realidad.

En términos de algoritmos, llegamos a la conclusión de que para un problema de esta complejidad el algoritmo de KNN no es suficiente para lograr una buena predicción, y que Random Forest y XGBoost tienen capacidad de atacar un problema complejo como este y dar buenos resultados. Sin embargo, XGBoost demostró poder dar un resultado mejor, con la desventaja de que es necesario un trabajo de ajuste de hiperparámetros para dar una buena predicción que puede tomar mucho tiempo.

También vale notar que el ajuste de hiperparámetros solo puede mejorar la eficacia de una predicción hasta un determinado punto. Una vez llegada esa instancia, la forma más sustancial de subir la precisión es a través del Feature Engineering, o el cambio total del algoritmo utilizado. En caso del Feature Engineering, esto no asegura el aumento del score, ya que las features agregadas pueden no ser significativas para el problema. Al cambiar el algoritmo, tampoco se asegura una mejora, ya que hay algoritmos que funcionan mejor para algunos problemas que para otros. La forma definitiva de aumentar las predicciones del algoritmo es a través de prueba y error.