Organización de Datos
75.06

Trabajo Practico N°2

Repositorio:
https://github.com/hlabaisse/tp1_org_datos

Hugo LABAISSE N°103624

I/ Introducción

Realizaré un análisis de datos sobre un conjunto de eventos de web analytics de usuarios que visitaron www.trocafone.com, su plataforma de ecommerce en Brasil. Trocafone es un side to side Marketplace para la compra y venta de dispositivos electrónicos que se encuentra actualmente operando en Brasil y Argentina. Este set de datos posee información de alrededor de 1000000 de eventos de Trocafone. Nuestro objetivo será realizar un procesamiento de datos para predecir si una persona va a comprar o no un producto.

Como herramientas se usó Jupyterlab, un entorno de desarrollo para Data Science, particularmente en lenguaje Python.

II/ Objetivo y presentación de los datos

Tenemos tres Excel documentos:

- uno comporta todas las actividades sobre el sitio web de Trocafone del 01/01/2018 al 31/05/18.

- un otro tiene los ids de cada persona y un label asociado: 0 si la persona no compra entre el 01/06/2018 y el 15/06/2018 y 1 si la persona compra entre estas fechas.

- el ultimo es solo ids y debemos predecir si va a comprar o no y submit sobre la competición Kaggle.

Hay muchos eventos y datos en el set de entrenamiento y tenemos que filtrar y procesar los datos.

El objetivo es de predecir para cada persona si va a comprar o no un producto. Se mide con el AUC función (que es una función para tener en cuenta falsos negativos y positivos).

III/ Filtrar datos

Primero elijo de conservar solo algunos datos del Excel. Yo borro muchas columnas. Tengo que agrupar por persona después y pienso que hay muchas que no sean muy interesantes de guardar si tengo que hacer promedio o count para las columnas.

Primero, yo eliminé columnas relativas a color, storage, url. Pensaba que tenía que eliminar muchos datos por facilitar el procesamiento del algoritmo. Solo guardo el modelo del teléfono que me parece el mas importante. No es importante si la persona quiere comprar un smartphone azul o verde ni su almacenamiento.

La localización como ciudad, región, país no me parecen importante también. Tampoco el origen del evento: operating system, browser versión, screen resolution, search engine. New_vs_returning no es útil también porque al agrupar por persona, todos van a ser new y returning. No entendí bien channel entonces lo eliminé. Search term puede ser interesante, pero es demasiado complicado de analizar palabras.

Al fin me quedaba con timestamp, tipo de evento, person, sku, modelo, staticpage y campaign source.

IV/ Preprocesamiento de datos

Primero tengo que recordar que cada futura columna tiene que ser relacionar a una persona y no dos veces a la misma: no puedo tener mas filas que personas. Entonces tengo que agrupar por persona cada vez.

Lo que me parece el más interesante y fácil al primero estaba de agrupar por persona y de contar la cantidad de eventos.

Después quería hacer ratios. Me parece que es interesante de ver la correlación entre diferentes tipos de eventos. Por ejemplo, compra sobre vista. Efectivamente, si una persona mira muchos productos, pero nunca hace una conversión probablemente tenemos que predecir que no va a comprar en los próximos días tampoco. Después intento otras ratios: vista sobre visita de la página web y vista sobre brand listing para saber si la persona mira muchos productos cuando va en el sitio. También, hice vista sobre publicidad hit si la persona es sensible a publicidad. Conversion sobre checkout para saber si después de checkout una persona compra o no. En estas ratios hay muchas veces el evento visto porque vemos en el análisis exploratorio que es el evento mas frecuente (5 veces mas que el segundo: Brand listing). Si hay muchos datos me parece interesante de hacer más ratio con él.

Vimos también en el análisis exploratorio que había mucho más evento los últimos meses que en los primeros. Además, pienso que la información sobre el último mes es más útil para predecir si una persona va a comprar en los próximos 15 días o no. Efectivamente, si la persona hace todas sus búsquedas en enero y después nada, podemos pensar que no va a comprar en junio. Al contrario, una persona que hace todas sus búsquedas en mayo tiene mas probabilidad de comprar en junio. Entonces, en otro dataframe, hice todos los counts y ratios únicamente sobre el último mes.

Quería también utilizar la columna modelo. Efectivamente, mirar el mismo producto todo el tiempo o mirar a diferentes modelos no es el mismo. En el primer caso podemos pensar que la persona va a comprar y en el segundo no. Entonces agrego una columna que cuenta la cantidad de veces que la persona mira a su modelo favorito. Por favorito yo entiendo el modelo que la persona mira el mas. Lo hice solamente por el ultimo mes porque me parece que los datos del último mes son mas representativos de los próximos 15 días que todos los datos. Además, quería saber si la persona compra este modelo en el ultimo mes. En efecto, si la persona lo compro todavía hay menos suerte que lo compra una vez más. Sin embargo, cuando lo hice yo vi que solo se aplica a 4 personas … Entonces elije de no ponerlo en mi dataframe porque pienso que el costo de la columna sobre el procesamiento es mas grande que la ventaja de tener más información para 4 personas. Ademas yo hice la ratio del nombre de veces que la persona mira a su modelo favorito sobre su total de vista. En efecto, tenemos que comparar los 2 porque es normal que una persona que mira mucho más producto tiene un máximum de vista sobre un producto más alto que una persona que mira menos productos, pero no significa que la persona tiene una mas alta probabilidad de comprar el producto.

Me parece también interesante de mirar a la cantidad de modelos diferentes que una persona mira. No sé exactamente que se puede deducir de eso, pero pienso que puede ser útil para el algoritmo.

Quería analizar más el evento staticpage. Hay diferentes tipos de staticpage y pienso que algunos son interesantes. En efecto, si una persona consulta ‘how to buy’ en el último mes, se puede que va a comprar pronto. Si una persona consulta cosas relativas a e-commerce también. Yo agrego las tres cosas relativas a e-commerce juntas porque me parecen similares y una columna es bastante

para representarlos. Además yo pongo 'how to sell' porque pienso que si una persona quiere vendar su teléfono, es muy probable que va a comprar otro porque todo la población tiene un teléfono.

Ahora tengo mi dataframe y cada fila representa una única persona y cada persona se encuentra únicamente en una fila. Mis columnas son todos variables numéricas que pueden ser utilizados en algoritmos de machine learning.

V/ Machine learning: XGBoost

Queria empezar con XGBoost porque es un algoritmo sencillo, que no necesita normalizar los datos y que funciona en general muy bien. He convertido los datos con dbmatrix porque es más rápido por utilizar xgboost. Miré un sitio web para ver cómo implementarle y elijé los mismos parámetros que el para empezar. Me da casi mi mejor score con esos parámetros (tipo 0,845). Después implemento una cross validation. En efecto, para testar los diferentes parámetros es mejor de hacerlo sobre una cross validación porque es mas preciso: todos los valores son consideran como train y test (no al mismo tiempo). Entonces tengo mas de un set de test para evaluar los diferentes parámetros.

Después tenía que buscar los mejores parámetros. Primero yo aumento n_estimator porque yo vi que tenía una mejor precisión con 100. Parece lógico porque con más árbol puedo tener un mejor resuelto (pero no demasiado si no overfitting). No lo pongo más alto por un problema de tiempo de computación. Después yo sigo como se hace en el sitio web para mejorar los parámetros. Yo empiezo con la profundidad del árbol y la condición sobre tener un hijo o no. Después yo sigo con gamma y para terminar con colsample_by tree and subsample que son el porcentaje de columna y de fila que el algoritmo pone por árbol.

Además lo que intento es de ver como procede el algoritmo. Intento ver los diferentes nodos del árbol con el package Graphviz. Sin embargo, no entiendo bien porque solo tengo un nodo … Otra cosa muy interesante y que permite analizar la fase de feature engineering es de mostrar la importancia de los features. Es un grafo donde se cuenta cada vez que un feature es utilizada en el árbol.

Podemos ver que los features que sirven más son en majoridad en relación con la vista de producto. Me parece normal porque es el evento que hay mas en los datos. Hay algunas ratios en los primeras features entonces estaba una buena idea de hacer algunas. Sin embargo, las cosas en relación con e-commerce o how to buy son los últimos. Imagino que es porque solo se aplica a muy pocas personas.

Con la optimización de los parámetros gano 0,004 en el Kaggle. Es muy pequeño y muestre que el mas importante es el preprocesamiento de los datos y no la optimización de un algoritmo (si al principio el algoritmo funciona bien). Mi mejor score es 0.849 con XGBoost y parámetros optimizados.

Es muy raro de ver que la importancia de los features cambia después de la optimización de los parámetros. Los primeros se quedan los primeros y los últimos los últimos pero hay mucho cambio como para checkout_recent que es el primero y de largo ahora.

VI/ Machine learning: SVM

Quería probar otro algoritmo de machine learning para ver si puedo mejorar mis resueltos. Intento con SVM. Al principio no funciona. XGBoost procesa los datos muy rápido, pero con SVM espero una hora y no termina compilar. Entonces pienso a utilizar un PCA para reducir mis datos y después utilizar SVM.

Primero necesitaba normalizar mis datos porque si no lo hago, PCA no funciona. Después yo hice el PCA y me quedé con 7 dimensiones. Con una función podía ver que la primera columna representa 77% de la información en mis datos. Entonces si la reducción del PCA no es suficiente puedo solo utilizar los primeros columnas (las primeras tres son más de 90% de los datos). No fue útil porque esta vez el SVM compila. El segundo problema que encuentro es que mis datos son abalanzados. Con el XGBoost no pasa nada, imagino que el sabe como hacer con este tipo de datos, pero por SVM no funciona.

Entonces intenté de modificar el class_weight. Me permite de poner más peso sobre el 1 si el algoritmo hace un error del label 1. Yo vi que en internet debemos poner los class_weight sobre la distribución de los datos, entonces es lo que hice. Además, cambio el C. Cuando C aumenta, el mejor es mi algoritmo. C es el peso atributo a un error. Sin embargo, el resuelto de mi algoritmo no es bueno, alrededor de 0.65. Entonces pienso que tengo que mejorar el algoritmo. Quizas puede ser que este algoritmo no es bien para el tipo de problema que tengo (incluso si he visto que podemos resolvar problemas abalanzados tipo spam/no-spam con SVM). Talvez es un problema con mi PCA y que no hay suficiente información (yo puse 95% de los datos).

VI/ Conclusión

En este algoritmo la fase de preprocesamiento fue muy larga. En efecto, el data que teníamos estaba casi solo string y tenía que poner números para mis algoritmos de machine learning. Entonces intento de pensar a como modificar la data para que sea interesante por el algoritmo. Gracias a la importancia de features en XGBoost, yo vi que tenía buenas ideas de hacer cosas más recientes y también ratios en el feature engineering.

SVM no funciona bien con mis datos. Puede ser el PCA de antes o una otra cosa porque a veces algunos algoritmos no funcionan bien sobre algunos datos.

Yo sabia que el algoritmo XGBoost funciona bien en general y con mis datos se pasa bien. La optimización de los parámetros me ayuda a obtener un poco mejor score, pero no mucho. Si tenia una cosa a modificar sabría el procesamiento porque es que es el más importante.

VII/ Área de mejora

Sobre el SVM pienso que podemos mejorar la solución con optimización de los parámetros. Quizás puedo solo tomar algunas columnas del PCA para ver que pasa. Si tenia una computadora muy potente quizás hacer el SVM sin PCA. Pero antes tengo que eliminar columnas que no me parece importantes tipo el count de static page. Sin embargo, no pienso que es el mas importante, se pasa que algoritmo no funciona bien con datos y pienso que es mas interesante de poner esfuerzos a mejorar los algoritmos que funcionan bien desde el principio.

Sobre el XGboost puedo mejorar con optimización de otros parámetros que ahora no optimicé. También por cada parámetro puedo hacerlo mas preciso con una escalera mas pequeña. Sin embargo, no pienso que eso puede mejorar mucho la eficiencia del algoritmo.

Puedo también probar otros algoritmos. Algunos de tipo random forest para comparar con XGBoost. Puedo también probar Naive Bayes.

Lo que haré primero es el feature engineering. Al principio pensaba que 40 columnas pueden ser demasiado para el algoritmo. Me equivoco por XGBoost que lo hizo muy rápido. Entonces puedo agregar más información. Pensaba a agregar una columna que hace el count de publicidad de Google. En efecto en el análisis exploratorio vimos que la publicidad de Google es más eficiencia que los otros. Tambien, puedo hacer otras columnas en relación con el tiempo, esta vez solo los últimos 15 dias. Otra idea sería de sacar la marca del modelo y hacer un count de algunos eventos (viewed o conversión) por marca.

Al fin estoy contento de lo que hice. Aprendí muchas cosas. Estaba un proyecto muy interesante y si tenía más tiempo y gente conmigo quería pensar más al preprocesamiento para mejorar mi xgboost.